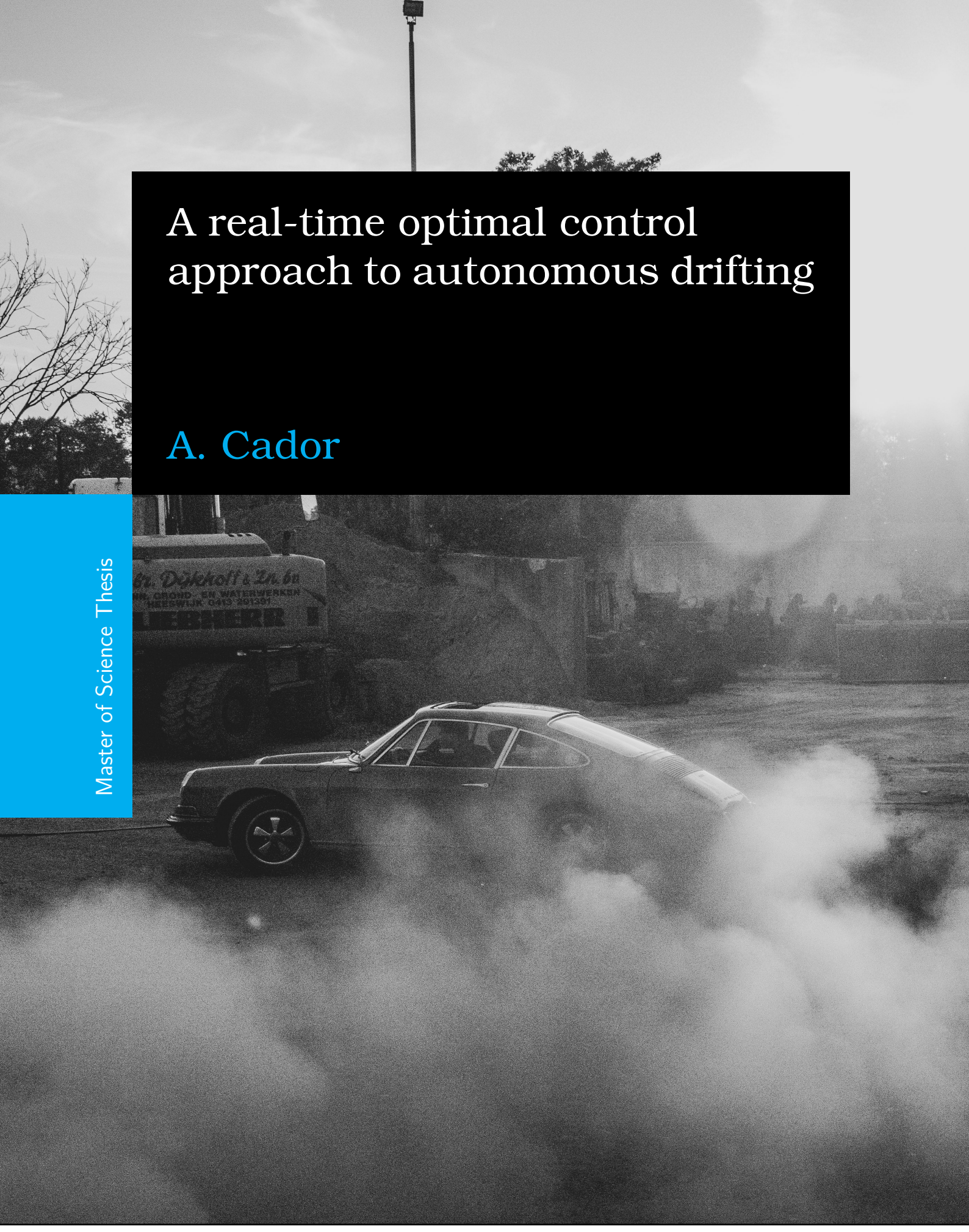


A real-time optimal control approach to autonomous drifting

A. Cador

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

A real-time optimal control approach to autonomous drifting

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

A. Cador

December 1, 2022

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

Full vehicle automation requires complete control over all driving scenarios that can be encountered on roads in order to ensure passenger safety at all times. This extends to dangerous situations such as losing control on slippery surfaces, commonly known as drifting. This work aims to make a step toward ensuring passenger safety in these situations by gaining control over vehicle behavior in high side slip regions. As these regions tend to be highly nonlinear and drifting dynamics happen at a millisecond scale, the time availability for generating proper commands is highly restricted. This work further proposes a method based on Nonlinear Model Predictive Controller (NMPC) that conveniently splits and reformulates the drifting problem to reduce the computational burden of the control structure and make it suitable for real-time implementations. The overall performance and robustness will be evaluated in obstacle-avoidance scenarios and in more general driving situations.

Table of Contents

List of Figures

List of Tables

"In the future, airplanes will be flown by a dog and a pilot. And the dog's job will be to make sure that if the pilot tries to touch any of the buttons, the dog bites him."

— *Scott Adams*

Chapter 1

Introduction

1-1 Motivation

The ground transportation services of the future are frequently described as environments majorly populated by autonomous vehicles. The artificial drivers that control these vehicles would successfully handle every driving scenario, ensuring passenger comfort and safety throughout the route. Autonomous cars would also bring performance and economic gains such as an increase in lane capacity, a decrease in fuel waste, a lower travel time, and most importantly a significant reduction in traffic deaths [6]. However, this future is made possible only after the vehicles reach their highest level of autonomy.

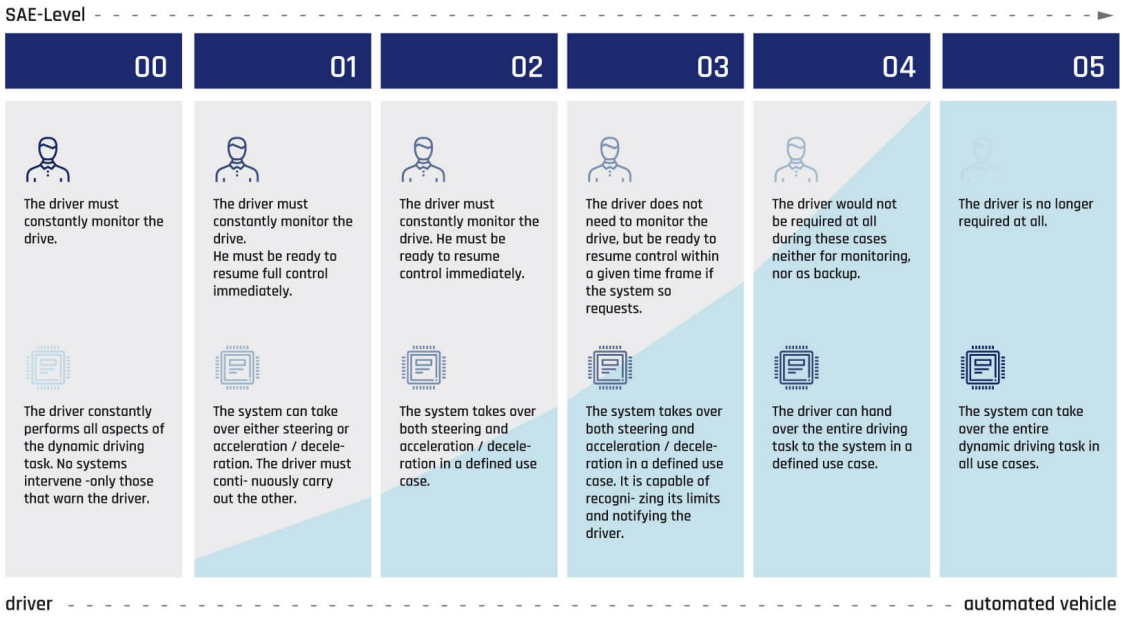


Figure 1-1: SAE levels visualized [1].

According to the new SAE International Standard J3016 which describes the levels of driving automation, there are six levels of automation [1]. The first level, frequently called *No automation* or Level 0, ensures that warnings are issued, but provides no sustained vehicle control. At the other end of the spectrum, *Full Autonomy* or Level 5 guarantees that the steering wheel is optional regardless of the vehicle location and weather conditions. These levels are depicted in Figure 1-1.

Importantly, a fully autonomous vehicle should also handle extreme scenarios while keeping passengers unharmed. This capability includes navigating corners on low-friction surfaces which potentially lead to unwanted high side slips maneuvers, commonly referred to as drifting.

Drifting is mostly remembered as a spectacular maneuver done by expert rally drivers and professional racers in tight corners or in various motor sport exhibitions for the amusement of the general public. It can provide a competitive advantage in off-road conditions when cornering on uncertain terrain, being mostly used on dirt or gravel to maximize exit velocity. Drifting is caused by rear tire saturation, a phenomenon that can be experienced by both rally drivers and, at a lower scale, regular drivers on slippery surfaces. These driving scenarios are dangerous as they are not easily manageable as the typical cornering experience.

These aggressive maneuvers are difficult to control, demanding constant and knowledgeable involvement from the driver to steer the car to safety. They require a sequence of unnatural reactions that are hardly expected in such a limited time from the general driving population. As the amateur driving experience rarely exhibits forces that get close to the maximum available friction, the typical driver does not gather relevant experience for controlling such maneuvers. In contrast, professional pilots frequently drive at the limits of friction to gain competitive advantages and are better prepared to respond to drifting as a result. This can be better understood from Figure 1-2 .

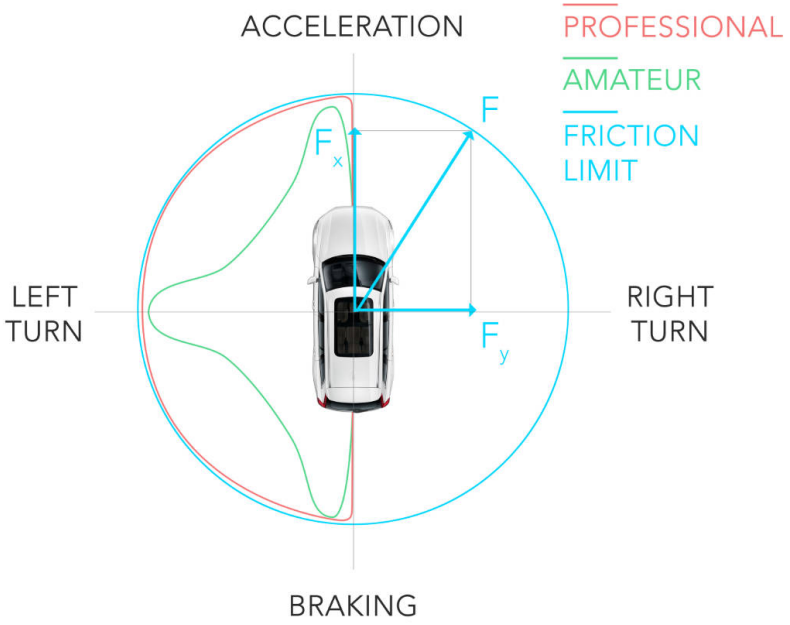


Figure 1-2: Professional and amateur driving visualized [2].

Currently, the effort of many engineers and scientist contributed to modern Electronic Stability Control (ESC) that successfully improve the vehicle stability and avoid the tire saturation. However, the possibility of drift occurrence on slippery surfaces cannot be completely dismissed. Regardless of the performance of these stabilizing mechanisms, human fault can still destabilize the vehicle on low friction surfaces, possibly causing harm not only to the passengers inside the vehicle.

Therefore, the possibility of occurrence of such maneuvers must be considered. As a result, it is valuable to develop a control framework that allows full control of drifting maneuvers, having the capability to safely navigate the environment even in these scenarios. As full autonomy requires operating in all regimes, extreme driving conditions must be included and completely exploitable. If successful, the same framework can be applied for both autonomous racing and passenger safety purposes. For these reasons, autonomous drifting can be a great step toward Full Autonomy.

1-2 State-of-the-art

The literature offers generous ideas on autonomous drifting, but also on adjacent subjects such as autonomous racing and the execution of aggressive maneuvers like the double lane change. This section will be largely organized by the research groups or universities to which the studies are attributed to.

First, the authors in [7] construct a drift controller for an experimental test bed using only the lateral dynamics. Despite its success, the study concludes with the need to incorporate longitudinal dynamics to increase both performance and robustness. A control framework is later proposed in [8] to address these limitations by creating separate steering and longitudinal controllers, advising in the end building a control structure that fully coordinates steering and longitudinal inputs beyond the proposed simple feedback. This suggestion is similar to what is observed in the behavior of expert rally drivers. This motivates the work from [9], which further validates the coordinated control structure on a modified 1981 DMC DeLorean for drifting along circular trajectories. A Nonlinear Model Inversion scheme is proposed in [10] to drift along circles of changing radii and validated using the same DMC DeLorean.

An in-depth study on drift cornering conditions and stabilization using only longitudinal control is contained in [11]. Motivated by this, a further work is presented in [12] aimed at controlling drifting maneuvers for Rear-Wheel Drive (RWD) vehicles using an LQR and a low-level wheel speed controller. The authors use a more accurate dynamical model and coordinate both longitudinal and steering inputs inspired by observed expert driver actions. The authors in [13] build and compare three strategies based on Nonlinear Model Predictive Controller (NMPC) aimed at stabilizing the vehicle at the limits of handling in cornering conditions. It is worth noting that this study also analyzes these aggressive maneuvers from a computational perspective to ensure the feasibility of a real-time implementation.

Acosta proposes an Advanced Driver-Assistance Systems (ADAS) in his work from [14] based on a drift initiation and control framework for various cornering scenarios. Similarly, the authors in [15] propose a hybrid hierarchical structure with proportional controllers and gain-scheduled Linear Quadratic Regulator (LQR). The chosen strategy is capable of drifting along rally-like segments, including the ability to switch drifting sides.

Lastly, a relevant work is presented in [16] for autonomous racing of miniature cars. The authors propose a framework that achieves millisecond-level solving times for the implemented contouring controller, forming the basis for the solver choice later described.

The body of knowledge includes various other approaches, however, the consensus is that exploiting drift maneuvers requires coordination of throttle and steering inputs. This is congruent with what rally drivers perform in competitions. As it will be discussed in Chapter 3, this coupling and the associated nonlinearity can be handled well by a Nonlinear Model Predictive Controller, which will form the basis for the proposed strategy. This choice is further motivated by the observation that vehicle control applications, and by extent obstacle-avoidance scenarios, are inherently constrained problems.

As drifting constitutes a highly nonlinear, fast dynamical process, significant efforts need to be made to allow a real-time implementation of the NMPC as it tends to be computationally demanding. Regardless, a thorough search of the relevant literature yielded no strategy that exploits drifting particularities to minimize the computational burden of the control structure. Conceptually, the closest work toward a computationally feasible, optimal control structure for drifting is in [13]. However, the authors study low side slip regimes, and as the next section shows, using a more generous sampling time of 50 milliseconds. As a result, there are no direct design suggestions towards reducing the computational demand for drift control purposes.

Relevant works such as [16] showed that millisecond-range solving times are possible for aggressive driving, suggesting the possibility of extending this to autonomous drifting. If successful, the proposed solution would enhance the performance of drift-exploiting strategies and the overall safety of passengers on slippery surfaces.

1-3 Problem statement

The current work aims to make a step toward ensuring passenger safety in drifting scenarios. The problem of controlling such maneuvers can be decomposed into three challenges:

1. Drifting occurs when the rear tires reach their nonlinear operating regime (saturation), making the high side slip regions correspond to highly nonlinear dynamics. Moreover, as it will be described in Chapter 3, these maneuvers are associated with unstable saddle points, making this an overall difficult control problem.

2. Drifting is a relatively fast dynamical process, a characteristic that can be fundamentally traced back to the tire-road interaction. This severely limits the available time to compute proper control actions.

3. The usual drifting scenario describes a heavily constrained environment: the vehicle needs to remain withing road limits and to avoid possible obstacles along the way, while being affected by state and inputs constraints.

These challenges can be used to formulate a set of actionable requirements that can be further used in the controller design phase:

1. Proper chassis and tire models need to be chosen that strike the perfect balance between simplicity and accuracy. While a simple model would conveniently lower the computational burden, a sufficiently high complexity is required to track vehicle behavior in nonlinear regimes,
2. A suitable control structure needs to be designed to fully exploit the model capabilities to generate suitable commands in a feasible amount of time. Moreover, it should be able to (efficiently) incorporate problem constraints.

Regarding the first requirement, the current work proposes to build on top of the already existent knowledge. Several experts have validated the use of various dynamical models for similar goals, therefore the current project does not intend to make any contributions in this matter. However, an educated model choice needs to be made to ensure a convenient trade-off.

The contribution of the current work lies in proposing a strategy that exploits drifting characteristics and achieves computation times that permit real-time implementation. Naturally, the naive approach does not directly allow this, as it will be presented in Chapter 5. Numerically, the computational availability corresponds to the sampling time T_s, which, inspired by works such as [14], will be set to 20 milliseconds.

Concisely, this strategy will be primarily based on the Nonlinear Model Predictive Controller (NMPC), which provides enough control authority for the previously mentioned challenges. This will be further motivated in Chapter 3. As previously discussed, this choice is associated with a considerable computational demand in its naive formulation. Therefore, the challenges of autonomous drifting shifts to a computational perspective, reformulating the problem statement as following:

How to adapt the naive Optimal Control Problem formulation to guarantee a sufficiently short solving time, while still granting full control over drifting maneuvers?

This will be answered by a convenient design of the control loop.

Fundamentally, this work assumes the vehicle starts in the vicinity of a drift equilibrium. As further Chapters will describe the proposed solution in greater detail, the split in dynamics does not allow the Navigation layers to access knowledge on drift equilibria. Therefore, the proposed framework is intended to function only after starting from a pair of states that are close to an equilibrium that is relevant for the given track in terms of curvature.

Lastly, the central problem in this work will deviate slightly from the typical purpose in autonomous racing, namely minimizing lap time. Similar to what is presented in [9], there will be a dual objective embedded in the proposed solution. First, there is a tracking objective that aims to keep the vehicle as close as possible to the road central path while avoiding obstacles. Second, the vehicle should exhibit a side slip angle that is as close as possible to a pre-defined, desired side slip angle. Throughout this work, the side slip setpoint will be $\beta^{\text{sp}} = 40°$, similar to other works in the field.
Because of this choice, the current work can be viewed as a limit-testing scenario on drift control, primarily viewed from a computational perspective. There will be no incentive for lap-time optimization as in racing scenarios because the vehicle velocity will be used to maintain the desired side slip magnitude.

1-4 Thesis outline

The rest of the document is organized as follows: **Chapter 2** will include information about vehicle and tire modeling choices. **Chapter 3** represents a short discussion on the stability of drifting maneuvers, and continues by describing the chosen control structure from a high-level perspective. **Chapter 4** introduces necessary optimization techniques and concepts intended at solving in real-time the Nonlinear Model Predictive Controller (NMPC) problem. This chapter continues by introducing all the proposed improvements that are targeted at consistently achieving reduced solving times in comparison with the straightforward approach. **Chapter 5** is the broadest chapter of the work and it is built as an extensive analysis on the performance and limitations of the proposed control structure. **Chapter 6** completes this work with a conclusion on the overall contribution, and further proposes possible research avenues.

Chapter 2

Vehicle Modeling

At the very core of the autonomous drifting problem lies the complexity of the tire-road interaction at the limits of handling. Different models with varying degrees of complexity can be found in the literature, however the options are restricted to dynamical models. As drifting is fundamentally a dynamic process, kinematic models cannot provide enough modeling power for the purpose of this work.
The vehicle model is split into a chassis model, including states such as velocity and yaw rate, and tire models, used for estimating the tire forces that affect the chassis dynamics.
As drifting is a fast dynamical process, the central problem is striking the perfect balance between simplicity and modeling accuracy. Highly-accurate models will be computationally expensive, while a simplistic model might not provide the necessary control authority during high side slip maneuvers.

2-1 Generic notions

First, a few generic notions need to be introduced for later reference.
The following relationship holds for a vehicle traveling with a velocity V, in a turn of radius R with a yaw rate r:

$$R = \frac{V}{r} \tag{2-1}$$

Generally, in vehicle dynamic applications the curvature of the road κ is more common than the turn radius R. It offers certain advantages such as a numerically stable way to express straights ($\kappa = 0$), and a continuous transition for alternating turns (R is discontinuous).

$$\kappa = \frac{r}{V} \tag{2-2}$$

2-2 Chassis model

2-2-1 Drifting dynamics

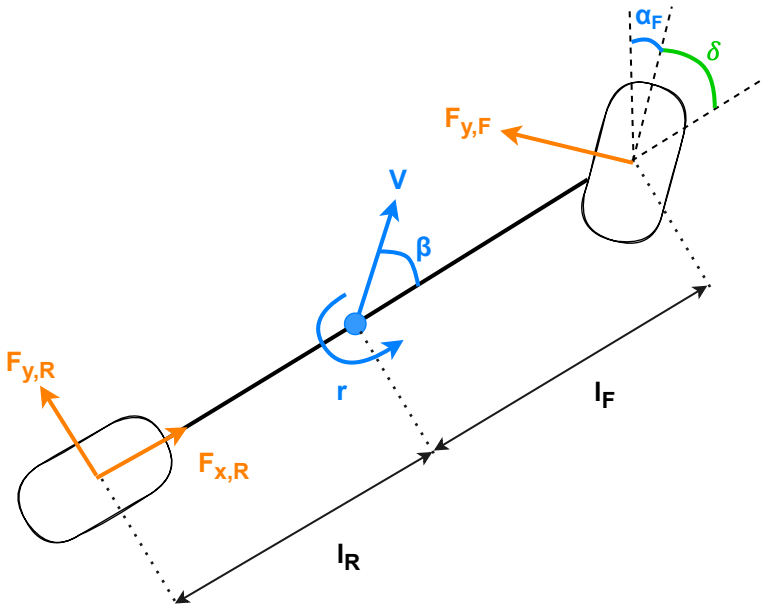


Figure 2-1: Dynamic Bicycle Model

Also called single track model, the Dynamic Bicycle Model (DBM) simplifies the chassis dynamics by lumping the tires on each axle into a single tire, leading to a reduced state which is beneficial for real-time implementations.
The literature offers results in autonomous drifting using a two-state version of then DBM. This version only includes lateral dynamics, i.e. the side slip angle β and the yaw rate r, measured around the vehicle Center of Gravity (CoG). However, the experimentally-validated analysis presented in [7] suggests the coordination of lateral and longitudinal dynamics to control drifting maneuvers. This is also analogous to the expert rally drivers behavior in cornering on low friction surfaces.

The use of the three-state DBM, which includes both longitudinal and lateral dynamics has been validated in numerous works on autonomous drifting such as [11, 10]. Precisely, the authors in [10] tested its suitability for drifting maneuvers using a real vehicle to track complex track segments.

Consequently, this work utilizes the three-state DBM displayed in Figure 2-1 and described in (2-3) to model drifting dynamics.

$$\dot{V} = \frac{-F_{\mathrm{y,F}} \cdot \sin(\delta - \beta) + F_{\mathrm{y,R}} \cdot \sin\beta + F_{\mathrm{x,R}} \cdot \cos\beta}{m} \tag{2-3a}$$

$$\dot{\beta} = \frac{F_{\mathrm{y,F}} \cdot \cos(\delta - \beta) + F_{\mathrm{y,R}} \cdot \cos\beta - F_{\mathrm{x,R}} \cdot \sin\beta}{m \cdot V} - r \tag{2-3b}$$

$$\dot{r} = \frac{l_{\mathrm{F}} \cdot F_{\mathrm{y,F}} \cdot \cos\delta - l_{\mathrm{R}} \cdot F_{\mathrm{y,R}}}{I_{\mathrm{z}}}, \tag{2-3c}$$

where V is the Total Velocity Magnitude, β is the side slip angle, and r is the yaw rate. The vehicle mass is denoted by m, the vehicle's yaw inertia by I_z, the steering angle by δ, and l_F and l_R represent the distance between the CoG and the front and rear axles, respectively. Moreover, the tire forces for each wheel are denoted by $F_{i,j}$, having $i = \{\mathrm{x}, \mathrm{y}\}, j = \{\mathrm{F}, \mathrm{R}\}$. Throughout this document, in the case of tire forces the first subscript i will indicate the direction of the force, 'x' for longitudinal, 'y' for lateral, and 'z' for vertical, and the second subscript j will indicate the wheel, 'F' for front, and 'R' for rear wheel. Therefore, $F_{\mathrm{y,F}}$ denotes the front tire lateral force, $F_{\mathrm{y,R}}$ represents the rear tire lateral force, and $F_{\mathrm{x,R}}$ is the rear tire longitudinal force.

Similar to what is found in the relevant literature, the longitudinal input is assumed to be the torque applied to the rear wheels, as in the case of Rear-Wheel Drive (RWD) vehicles. As the DBM lumps the rear wheels together, there will be only one longitudinal input into the wheel speed dynamics, the torque τ:

$$\dot{\omega} = \frac{\tau - R_\mathrm{tire} F_\mathrm{x,R}}{I_\omega}, \tag{2-4}$$

where ω is the wheel speed, I_ω is the moment of inertia about the rear wheel's axis of rotation, and R_tire is the radius of the rear wheel. The rolling resistance was neglected. The wheel speed ω will be necessary to describe the tire forces as presented in the following section, linking the throttle input τ and the chassis dynamics.

The vertical load on each tire is assumed constant and calculated using the static weight distribution:

$$F_\mathrm{z,F} = \frac{l_\mathrm{R}}{l_\mathrm{F} + l_\mathrm{R}} mg \tag{2-5a}$$

$$F_\mathrm{z,R} = \frac{l_\mathrm{F}}{l_\mathrm{F} + l_\mathrm{R}} mg, \tag{2-5b}$$

where g is the gravitational acceleration.

The modeling choices presented until this point sit at the simple end of the model fidelity spectrum used for similar tasks. The aim is to minimize the computational burden while still being able to track drifting nonlinearities, and accepting this trade-off brings certain limitations.
One such limitation is that drifting maneuvers excite roll and pitch dynamics which are neglected in the classical formulation of the three-state Dynamic Bicycle Model. First, the roll dynamics are implicitly eliminated by lumping tires together, ignoring the change in vertical load on right and left wheels depending on the turn side. Second, the pitch dynamics are neglected by assuming a static distribution of vertical load.

For the latter, the performance can be improved by fitting the weight distribution on measurements taken during drifting maneuvers. This will likely result in a better performance during maneuvers similar to the fitting procedure, however this is still scenario-specific and can be hardly extended to more generic conditions. A better solution is including load transfer dynamics between the front and rear axles. The literature offers such examples related to autonomous drifting in works such as [12, 13]. Moreover, these two works also approach the problem using a more complex Two-Track Model which considers all four wheels.
Regardless, this work will be focused on validating the use of the simple DBM as it is the

computationally cheapest, yet proven way in dealing with drifting maneuvers. The remaining loop time can be used for additional dynamic layers to further push the performance, but this will be proposed as a further research avenue.
A proper evaluation of the effects of load transfer will be done in Chapter 5.

2-2-2 Motion planning

The Dynamic Bicycle Model is central for drift stabilization, however the motion planning task needs to be solved in order to navigate the environment. In the current work, this layer is expected to generate state references for the drift stabilization layer to track in order to guarantee collision-free trajectories.
Typically, the path planning step requires longer horizons in order to generate proper references. Consequently, the models used for the motion planning layer should be computationally simpler than the lower, dynamical layer to ensure the computational feasibility of the loop.

A popular choice in relevant vehicle control applications is the Kinematic Bicycle Model (KBM) [17, 16], which is described in (2-6).

$$\dot{X} = V \cdot \cos(\phi + \beta) \tag{2-6a}$$

$$\dot{Y} = V \cdot \sin(\phi + \beta) \tag{2-6b}$$

$$\dot{\phi} = r, \tag{2-6c}$$

with the global coordinate system (X, Y, ϕ), where X and Y illustrate the position in a Cartesian space and ϕ reflects the vehicle orientation.
The KBM will be used in the baseline model for comparison purposes.

As it will be further detailed in Chapter 4, the KBM brings computational disadvantages and lacks useful features found in other models for maneuvers such as drifting. For these reasons, the curvilinear dynamics of the vehicle will be introduced further.

The perspective shifts from conventional global Cartesian coordinates into the Curvilinear space where the coordinates are relative to track. Instead of (X, Y, ϕ) pairs, the position and heading of the vehicle are expressed w.r.t to the central path of the track using the following three components:

1. The progress along the path S. This implies having the arc-length parametrization of the road. Importantly, the arc-length parametrization is not a trivial task for generic curves, while for the circular tracks considered in this work this parametrization is straightforward.

2. The lateral (orthogonal) deviation from the central path n. In this work, the following convention applies: *a bigger turn radius (relative to central path) while moving in the trigonometric sense ($\kappa > 0$) corresponds to a positive lateral deviation n ($n > 0$).*

3. The local heading θ which determines the orientation in the curvilinear space (in Figure 2-2, the green line $\theta = 0$ represents the road tangent).

The conversion from global coordinates to curvilinear coordinates is achieved by projection, while the reverse is ensured by the arc-length parametrization of the track.

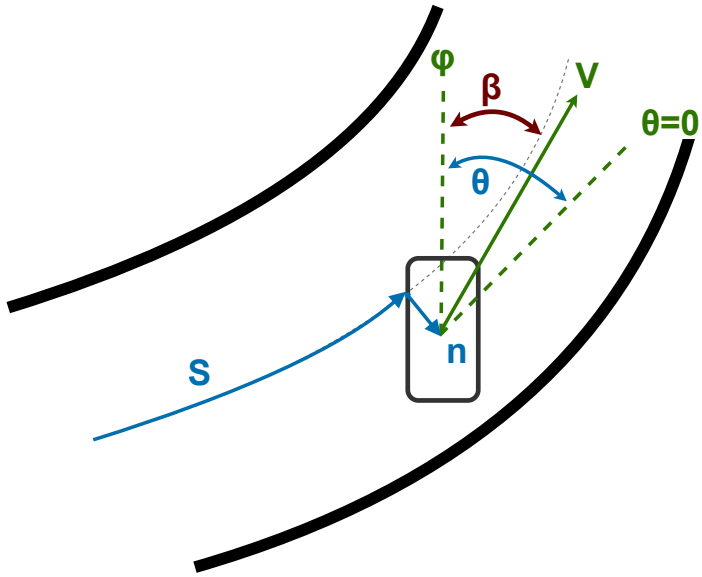


Figure 2-2: Curvilinear dynamics

Mathematically, the curvilinear dynamics used in this work are described in (2-7).

$$\dot{S} = \frac{V\cos(\theta - \beta)}{1 + n\kappa} \tag{2-7a}$$

$$\dot{n} = V\sin(\theta - \beta) \tag{2-7b}$$

$$\dot{\theta} = r - \kappa\frac{V\cos(\theta - \beta^{\text{ref}})}{1 + n\kappa}, \tag{2-7c}$$

where κ is the road curvature.

Note that this is slightly different from what can be found in similar works such as [15]. There are fundamentally two reasons for this variation:

1. Assuming the sign of the angles introduced so far, the minus sign inside the trigonometric functions is introduced to align the signs of the dynamical β and the local heading θ. In similar works [15], tracking is achieved when $(\beta + \theta = 0)$ since they have opposite signs. This is detrimental to the purpose of the planner in this work, namely state reference generation, hence the introduction of the minus sign.

2. The convention on the sign of the lateral deviation n requires having the denominator in the progress dynamics as $(1 + n\kappa)$. This can be easily proven by investigating the identities related to the Instant Center of Rotation (ICR).
Following the convention that a bigger turn radius for a trigonometric movement $(\kappa > 0)$ leads to positive lateral deviation $(n > 0)$, the vehicle velocity projected on the path V^S is:

$$V^S = \underbrace{V\cos(\beta - \theta)}_{\text{projection}} = \underbrace{\left(\frac{1}{\kappa} + n\right)\dot{\psi}}_{\text{radius} \times \text{angular velocity}}, \qquad n > 0 \tag{2-8}$$

where ψ is the angular velocity around the ICR. However, the time-derivative of progress can be derived similarly w.r.t to ψ:

$$\dot{S} = \frac{1}{\kappa}\dot{\psi} \tag{2-9}$$

Consequently, by imposing that ψ is equal in both (2-8) and (2-9), we arrive at (2-7a):

$$\dot{S} = \frac{1}{\kappa}\frac{V\cos(\beta-\theta)}{\left(\frac{1}{\kappa}+n\right)} = \frac{V\cos(\theta-\beta)}{1+n\kappa} \tag{2-10}$$

Notice that this work creates a curvilinear space targeted specifically at drifting. The alignment between β and θ, and the definition of n allows the use of the trajectories generated in curvilinear space as references.
Moreover, given the above expressions for the angles, the vehicle in curvilinear space in travels in a mirrored version of the Cartesian space, or colloquially expressed as underground. This is a result of how the θ dynamics are expressed.

2-3 Tire model

As it is often mentioned across literature, the tires represent the most important part of the vehicle, especially when the focus is on controlling aggressive maneuvers. The model presented above describes the vehicle as being directed by three forces, all being the result of the tire-road interaction. The tire forces are the medium through which the control structure can drive the vehicle in to the desired state using throttle input (τ) and steering (δ).

Drifting occurs when the rear tires enter their nonlinear operating regime or tire saturation. In this state, the rear tire longitudinal and lateral forces become coupled since there is only a limited amount of friction to be used. Along similar lines, the linearization around a drifting equilibrium reveals that in these regions steering and throttle inputs have similar contributions to lateral dynamics [4].

The studied literature seems to agree on the usage of a few tire models, showing great success in experimental drifting tests using real vehicles [10]. In other words, despite the actual complexity of the tire-road interaction, some interactions can be neglected without compromising the ability to drift.

In terms of tire models, the state-of-the-art offers examples of autonomous drifting using the physics-based Brush Tire Model such as [10]. This usually combined with a derating factor to account for the force coupling in the rear tire. However, its formulation is non-smooth and slightly more computationally expensive than the alternatives. Another option found in the literature is the semi-empirical Pacejka's Magic Formula (MF), which has an elegant way to introduce rear tire forces coupling as seen in [11, 12]. Moreover, it is smooth and this makes it friendlier for optimization problems, which will become relevant in Chapter 3. In its simplified and computationally lightweight variation, the model computes tire forces using the stiffness, shape, and peak factors, denoted by B, C, D, respectively.

In this work, the rear tire forces ($F_{\mathrm{x,R}}$ and $F_{\mathrm{y,R}}$) will be determined using Magic Formula as a function of tire slip. Tire slip refers to the non-dimensional relative velocity of the tire

w.r.t to the road [11]. The front tire lateral force $F_{\mathrm{y,F}}$ will be computed using the lateral formulation of MF as a function of slip angle. This choice is based on the observation that the front tire does not exhibit longitudinal forces (RWD), and the lateral formulation of MF easily accounts for this without additional assumptions.

First, the velocities at the center of each wheel can be expressed as functions vehicle states using geometrical identities as in [11]. The resulting expressions can be found in (2-11).

$$V_{\mathrm{x,F}} = V\cos(\beta-\delta) + rl_{\mathrm{F}}\sin\delta \tag{2-11a}$$
$$V_{\mathrm{y,F}} = V\sin(\beta-\delta) + rl_{\mathrm{F}}\cos\delta \tag{2-11b}$$
$$V_{\mathrm{x,R}} = V\cos\beta \tag{2-11c}$$
$$V_{\mathrm{y,R}} = V\cos\beta - rl_{\mathrm{R}} \tag{2-11d}$$

Similarly, we can define tire slip angles α by projecting V to obtain the longitudinal and lateral velocities in the body frame for each tire using (2-12).

$$\alpha_{\mathrm{F}} = \delta - \arctan\left(\frac{V\sin\beta + rl_{\mathrm{F}}}{V\cos\beta}\right) \tag{2-12a}$$
$$\alpha_{\mathrm{R}} = \arctan\left(\frac{-V\sin\beta + rl_{\mathrm{F}}}{V\cos\beta}\right) \tag{2-12b}$$

For the front tire, α_{F} will be sufficient to express the resultant force coefficient.

In contrast with the front tire, the rear tire will need to account also for the force coupling specific to drifting, as it generates both longitudinal and lateral force (RWD vehicle). This requires formulating the tire model w.r.t to theoretical slip quantities s [11], whose definitions can be found in (2-13).

$$s_{\mathrm{x,R}} = \frac{V_{\mathrm{x,R}} - \omega R_{\mathrm{tire}}}{\omega R_{\mathrm{tire}}} \tag{2-13a}$$
$$s_{\mathrm{y,R}} = \frac{V_{\mathrm{y,R}}}{\omega R_{\mathrm{tire}}} \tag{2-13b}$$

The resultant slip for the rear tire is therefore expressed in (2-14).

$$s_{\mathrm{R}} = \sqrt{s_{\mathrm{x,R}}^2 + s_{\mathrm{y,R}}^2} \tag{2-14}$$

Assuming a linear dependence of tire forces on the vertical load, we obtain the friction coefficients μ for each tire using both MF formulations as in (2-15).

$$\mu_{\mathrm{F}}(\alpha_{\mathrm{F}}) = \frac{F_{\mathrm{y,F}}}{F_{\mathrm{z,F}}} = D_{\mathrm{F}}\sin\left(C_{\mathrm{F}}\arctan\left(B_{\mathrm{F}}\alpha_{\mathrm{F}}\right)\right) \tag{2-15a}$$
$$\mu_{\mathrm{R}}(s_{\mathrm{R}}) = \frac{F_{\mathrm{R}}}{F_{\mathrm{z,R}}} = D_{\mathrm{R}}\sin\left(C_{\mathrm{R}}\arctan\left(B_{\mathrm{R}}s_{\mathrm{R}}\right)\right), \tag{2-15b}$$

where F_{R} is the resultant tire force at the rear tire, $F_{\mathrm{R}} = \sqrt{F_{\mathrm{x,R}}^2 + F_{\mathrm{y,R}}^2}$.

Further assuming symmetric tire characteristics in longitudinal and lateral directions, the resultant friction force for the rear tire lies within the friction circle presented in Figure 2-3.

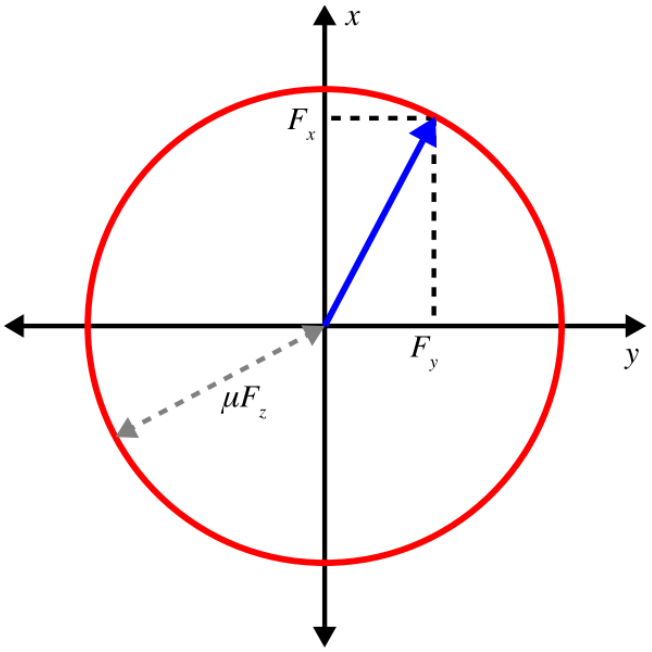


Figure 2-3: Friction circle

This allows the decomposition of the total friction coefficient into longitudinal and lateral components:

$$\mu_{x,R} = -\frac{s_{\mathrm{x,R}}}{s_{\mathrm{R}}}\mu_{\mathrm{R}}(s_{\mathrm{R}}) \tag{2-16a}$$

$$\mu_{y,R} = -\frac{s_{\mathrm{y,R}}}{s_{\mathrm{R}}}\mu_{\mathrm{R}}(s_{\mathrm{R}}) \tag{2-16b}$$

Using the expression from (2-15) and (2-16), all tire forces that appear in the Dynamic Bicycle Model (2-3) can be computed using (2-17).

$$F_{\mathrm{y,F}} = D_{\mathrm{F}} \sin\left(C_{\mathrm{F}} \arctan\left(B_{\mathrm{F}}\alpha_{\mathrm{F}}\right)\right) F_{\mathrm{z,F}} \tag{2-17a}$$

$$F_{\mathrm{x,R}} = -\frac{s_{\mathrm{x,R}}}{s_{\mathrm{R}}} D_{\mathrm{R}} \sin\left(C_{\mathrm{R}} \arctan\left(B_{\mathrm{R}}s_{\mathrm{R}}\right)\right) F_{\mathrm{z,R}} \tag{2-17b}$$

$$F_{\mathrm{y,R}} = -\frac{s_{\mathrm{y,R}}}{s_{\mathrm{R}}} D_{\mathrm{R}} \sin\left(C_{\mathrm{R}} \arctan\left(B_{\mathrm{R}}s_{\mathrm{R}}\right)\right) F_{\mathrm{z,R}} \tag{2-17c}$$

It is worth noting that the MF has a wide range of variations depending on the application. In its original formulation [18], the semi-empirical model has more terms which make the overall tire force generation more computationally heavy. This is the reason for utilizing a simplified version which offers a lighter computational burden at the cost of minor accuracy losses.

2-4 Summary

This chapter introduced the modeling choices made for describing the vehicle behavior at the limits of handling. For the chassis dynamics, the Dynamic Bicycle Model was chosen in its simplest form for its computational advantages. Regardless, the literature contains works that have validated its use for similar purposes [11, 10]. Further, the tire forces will be computed using Pacejka's Magic Formula, which represents a computationally lightweight and elegant way to model tire behavior in the nonlinear regime. These two choices will be central for the control strategy introduced in the next chapter.

Chapter 3

Controller Development

This chapter will introduce the proposed control scheme. Fundamentally, the control strategy should provide enough control authority over the fast, highly nonlinear drifting maneuvers. Moreover, this should be done in a computationally feasible way, in the presence of state and input constraints. This implies building a structure that is highly specific to drifting, and the structural decision that will make this possible will be detailed further. Before introducing the Nonlinear Model Predictive Controller (NMPC), a short section is dedicated to the analysis of drifting equilibria. Lastly, this chapter will also propose a simple friction update strategy since a reliable friction estimate is necessary for drift control.

It is worth mentioning that this chapter serves as a high-level, structural overview on the proposed control strategy. As the central actor in the scheme is the NMPC, most of the control logic happens inside an optimization problem. As a result, as autonomous drifting is viewed fundamentally from a computational perspective, the contribution of this work will be primarily understood in Chapter 4 where the Optimal Control Problem in each layer will be thoroughly presented.

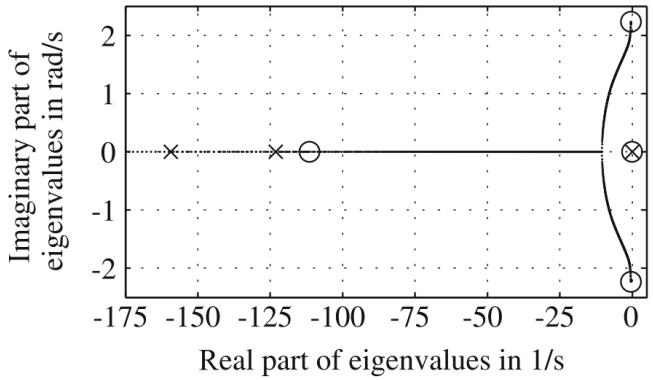
3-1 Stability analysis of drifting maneuvers

Several investigations on the stability of drift equilibria were undergone in the literature, and this work proposes to build on top of the existent knowledge. This section does not by any means provide a contribution in this sense and will provide lesser personal insights. However, this section is included for completeness reasons as throughout this document, a significant part of the observations are based on the instability of drift equilibria.

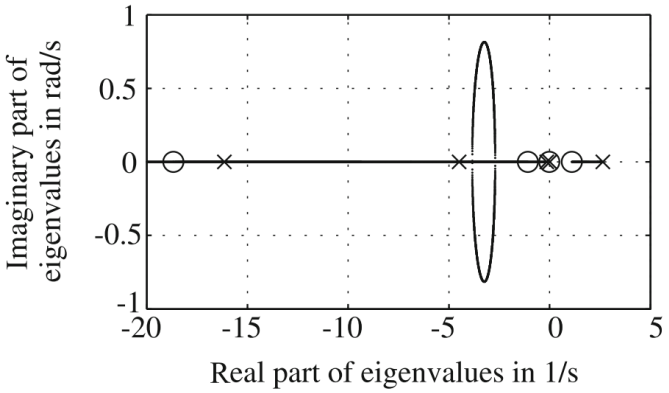
First, a generic nonlinear dynamical system is described by the following state dynamics:

$$\dot{x} = f(x, u), \tag{3-1}$$

having the system dynamics f, the state vector x and the input vector u. In the context of this work, x denotes the vehicle states and u represents the throttle and steering commands.



(a) Root locus for regular cornering [3].



(b) Root locus for drift cornering [3].

Figure 3-1: Root locus for regular and drift cornering based on velocity V [3].

The equilibria of the system denoted by the pair (x^{eq}, u^{eq}) satisfy the following relationship:

$$f(x^{eq}, u^{eq}) = 0 \tag{3-2}$$

For the purpose of autonomous drifting, a thorough investigation was done in [3] using a complex four-wheel model. In this article, the authors introduced the popular term 'powerslide' to refer to drifting equilibria and to drifting in general. Initially, the work in [3] linearizes a 10-state complex vehicle model with two inputs (throttle and steering) around steady-state solutions that were computed for various velocities V and constant turn radius ($R = 50m$). The analytical findings were related to these states only through the normal acceleration $\frac{V^2}{R}$. The results were compared to experimental data gathered using a real-sized vehicle and despite minor deviations due to model simplifications, the theoretical characteristics match the measurements.

Next, the root locus is presented to further investigate the differences between casual (or regular) driving and drifting. First, the root locus is presented for regular cornering in Figure 3-1a, where the crosses (X) represent the lowest velocities and the circles (O) represent the maximum velocities. Essentially, there are no eigenvalues with positive real part for regular cornering. There is only one eigenvalue on the real axis that approaches the origin,

but it remains on the left side of the axis. On closer inspection, it is revealed that its eigenvector is strongly driven by velocity V. These characteristics clearly indicate a stable equilibrium, corresponding to comfortable and trustful handling characteristics.

These results are put in contrast with Figure 3-1b. For drifting scenarios, the starting velocity is $V_0 = 58\frac{km}{h}$. Concisely, there are two eigenvalues with positive real part: one that is close to the origin and another one that starts at 2.5 and move towards the origin with increasing velocities. Altogether, this root locus shows that the drifting maneuver has positive real eigenvalues regardless of the velocity, making the associated equilibria unstable - given their null imaginary part, there is no oscillatory behavior. Concisely, the study concludes that drifting corresponds to unstable equilibria.

A similar conclusion is reached using a simpler two-state Dynamic Bicycle Model in [4], having only the side slip angle β and the yaw rate r in the state vector. The author assumes constant longitudinal velocity $V_x = V_x^{eq}$ and constant steering angle $\delta = \delta^{eq}$. The equilibrium pairs (β^{eq}, r^{eq}) were obtained numerically by solving $f(x^{eq}, u^{eq}) = 0$, and are denoted with red dots in Figure 3-2. This procedure is also repeated for the three-state DBM, and the author's remarks are similar with what was observed for a more complex model in [3]: drifting maneuvers correspond to unstable saddle points.

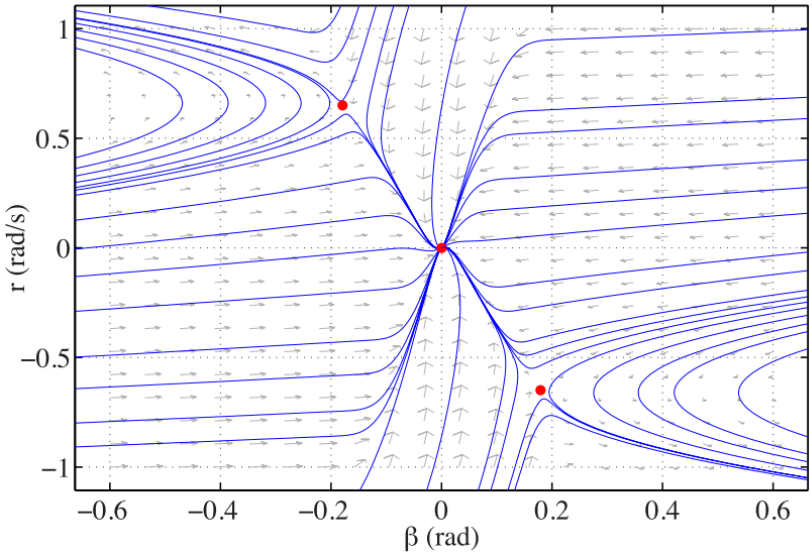


Figure 3-2: Phase portrait analysis of drifting equilibria [4].

Regardless of the model fidelity used in the analysis, the conclusion that drifting corresponds to unstable equilibria remains. As a result, the vehicle constantly needs educated control actions in order to be stabilized around drifting equilibria. More precisely, drift stabilization needs the coordination of steering and throttle inputs as remarked in [7], similar to what expert rally drivers perform on slippery terrains.

3-2 Proposed control structure

From a control perspective, the system requirements differ from the usual drifting scenarios seen in motor sport competitions as explained in Section 1-3. Typically, expert rally drivers approach a corner with a certain velocity V and negotiate the turn by indirectly adjusting the lateral states (β, r) to maximize exit velocity. The purpose of this control scheme is qualitatively different as there is no incentive to maximize lateral acceleration or velocity. The goal is on one hand to track a desired side slip angle setpoint β^{sp} by adjusting V and r, and to ensure a collision-free trajectory on the other hand. Moreover, the vehicle should accomplish these objectives while not violating track limits and state and input constraints.

3-2-1 Splitting navigation and drift stabilization

The proposed control structure splits the autonomous drifting problem into two components:

1. **Navigation (Planner).** This represents the high-level reference generator that is used for obstacle avoidance and maintaining the vehicle within track limits. It also needs a more generous look ahead to generate smoother trajectories that are suitable drifting.
2. **Drift stabilization (Controller).** This is the low-level controller aimed at driving the vehicle states toward the generated reference. This layer is responsible with generating vehicle inputs that maintain the side slip tracking error $\|\beta - \beta^{\text{sp}}\|$ small, while still ensuring the tracking of the planned path.

This decomposition is depicted in Figure 3-3.

The proposed control strategy relies on building a Nonlinear Model Predictive Controller (NMPC) for each layer. For drift stabilization purposes, this choice can be easily motivated:

1. As previous analysis has shown, drifting requires the coordination of steering and throttle inputs. NMPC can innately handle Multiple-Input Multiple-Output (MIMO) systems and the coupling between inputs without any simplifying assumptions. The NMPC will exploit as much modeling power as the model can provide, which is a major benefit since drifting is a highly nonlinear process.
2. The obstacle-avoidance scenario and vehicle control applications in general compose a heavily constrained environment. This includes spatial limitations, but also state and input constraints. NMPC not only is able to incorporate these constraints, but it can also efficiently work on them to generate proper commands.
3. The use of NMPC has been validated in similar scenarios at the limits of handling such as the double lane change maneuver [13].

From a navigation perspective, this controller choice is also beneficial:

1. The usage of NMPC allows to generate state references rather than planar references, lowering the overall computational load of each individual problem. This will be detailed in Chapter 4.

2. The Planner will be able to work efficiently on environment constraints given by track boundaries and the presence of obstacles.

3. It provide an effortless way to suggest generating circular trajectories that are suitable for drifting.

All of these reasons make the Nonlinear Model Predictive Controller a great choice for handling the problem of drift stabilization and reference tracking concurrently.

However, the use of NMPC brings a significant disadvantage. Compared to other strategies, it has a significant computational cost. As the drifting dynamics are a fast process, this can potentially lead to problems if the controller cannot not come up with adequate commands in the available time.

Consequently, the focus is shifted to reformulating the Optimal Control Problem (OCP) that sits at the core of NMPC to ensure the feasibility of a real-time implementation. In other words, the aim is to reduce the computational burden as much as possible by making the problem specific to drifting. The solution to this problem represents the central contribution of this work and will be further discussed in Chapter 4.

Apart from the additional increase in computational burden, having two stacked NMPC generally has one more drawback. Fundamentally, this structure can lead to a spiraling effect: a planner re-computation will alter the state references, the change the low-level controller receives in reference creates a different vehicle response, causing the subsequent planner re-computation to deviate further, and this gets propagated until failure.
However, this problem is scenario-dependent and it will be addressed by properly tuning the weights to penalize harshly the change in references.

3-2-2 Decoupling the wheel dynamics

Lastly, as inspired by works such as [10, 12], the wheel dynamics are removed from the system dynamics and regulated in an inner loop. This is also suggested by its simple SISO dynamics. In comparison with having a mapping for the generation of longitudinal forces at the rear tire, it has been shown to increase tracking performance [10]. Moreover, it also lowers the overall complexity of the drift-stabilization layer by shortening the state vector and by decreasing the overall nonlinearity. This will be further discussed in Chapter 4.

The wheel speed dynamics will be controlled using a simple parallel-form PI Controller with anti-windup. The wheel speed reference is regarded as an input in the drift stabilization layer and it is fed into the PI Controller, the latter generating the torque command that is sent to the vehicle.

The proposed control structure can be viewed schematically in Figure 3-3.

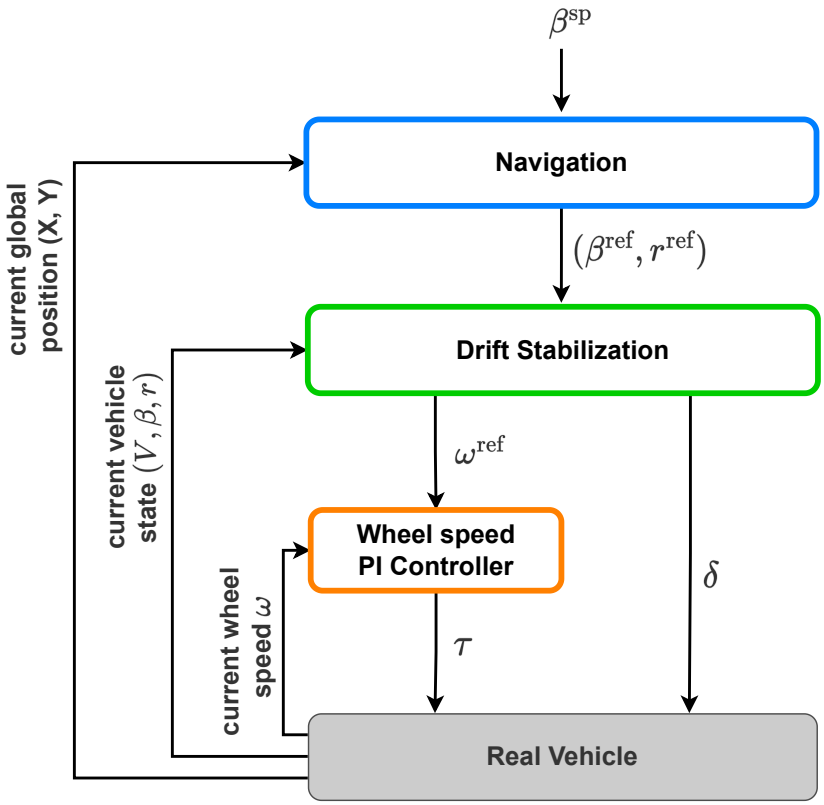


Figure 3-3: Proposed hierarchical decomposition.

The sequence of steps at each simulation step k can be found below:

1. The desired side slip is given β^{sp}. The Cartesian coordinates of the vehicle are measured.

2. The Planner projects the current position in the global cartesian frame to the curvilinear space. Using the resulting (S_k, n_k), the NMPC solution is recomputed for the new initial condition. A set of N_P pairs of (β^{ref}, r^{ref}) is generated and is further sent to the Drift stabilization layer, where N_P is the prediction horizon length of the Planner.

3. The current vehicle states are estimated. The Controller solution is recomputed, using (β^{ref}, r^{ref}) as state references. Notice that the total velocity magnitude V is free, without any reference to track. The purpose is to relax the tracking goals and sacrifice V (which can be maximized in a racing scenario) to provide more freedom for drift stabilization. Moreover, this can be understood also from an obstacle-avoidance perspective: on short-term, the lateral dynamics of the vehicle are essential for avoiding collisions, while the velocity becomes more important in the long run. By leaving V independent, the controller can freely choose it in order to both better track β^{sp} and avoid near obstacles. The resulting Controller output consists of N_C pairs of (ω^{ref}, δ), N_C being the prediction horizon length of the Controller. The wheel speed reference is an artificial control input that replaces throttle input in the Controller model. The first steering angle input δ_k can be directly sent to the vehicle, while the first wheel speed reference ω_k^{ref} is sent to the PI Controller.

4. The current wheel speed ω_k is measured, and the current tracking error e and error integral e^I up to step k are computed. The throttle input is calculated using the proportional gain K_P and the integral gain K_I:

$$e_k = \omega_k^{\text{ref}} - \omega_k \tag{3-3a}$$

$$\tau_k = K_{\text{P}} e_k + K_{\text{I}} \left(e_{j-1}^{\text{I}} + e_k T_s \right), \tag{3-3b}$$

where T_s is the sampling time. Consequently, τ_k and δ_k are sent to the vehicle and the loop iteration finishes.

PI tuning

The tuning of the PI controller was done experimentally in closed-loop. The wheel speed dynamics have a significant contribution in the tire force generation: a small change in the controller gains leads to different wheel speed response, which further create different solutions in the Navigation and Drift Stabilization layers. This difficulty of separating the PI controller from the rest of the control loop restricts the range of available tuning methods.
Furthermore, the tuning process is also affected by the force coupling at the rear tire which makes the experimental tuning less intuitive.
In the proposed approach, the PI Controller was tuned in a similar fashion with the Ziegler-Nichols method [19]. Fundamentally, a P controller was first created that brought the drifting (through wheel speed dynamics) maneuvers close to instability. Afterwards, the proportional gain was significantly lowered, and the integral component was introduced. The integral gain has been tuned to favor a fast response with minor oscillations. There is no derivative gain as experiments have shown that is detrimental for drift stabilization.

3-3 Reference disturbance estimator

The control structure presented above also includes a reference disturbance estimator that has not been placed in the figure for clarity purposes. In model mismatch scenarios, this can improve the overall robustness of the strategy.

Precisely, the disturbance estimator is built as a simple integrator based on the difference between the measured state at step k and the prediction of state k made at step $k-1$:

$$\hat{d} = \begin{bmatrix} \beta_k - \beta(k|k-1) \\ r_k - r(k|k-1) \end{bmatrix}, \tag{3-4}$$

where $\hat{d}$ is the estimated disturbance, β_k and r_k are the actual states at step k. $\beta(k|k-1)$ and $\beta(k|k-1)$ are the one-step ahead predictions at step $k-1$. These predictions are made using the Controller Model with nominal parameters.
Precisely, $\hat{d}$ can be added to the state references generated by the Navigation NMPC to lower the effect of model mismatch.

3-4 Friction update

As mentioned at the beginning of the chapter, the proposed solution introduces a simple method to account for friction uncertainty.

Inspired by works such as [20, 21], this work will propose a Recurise Least Squares (RLS) algorithm to update an artificial friction coefficient scaler $\hat{\mu}$. Precisely, the algorithm assumes the following tire friction model:

$$F_{\mathrm{y,F}} = \hat{\mu} D_{\mathrm{F}} \sin\left(C_{\mathrm{F}} \arctan\left(B_{\mathrm{F}} \alpha_{\mathrm{F}}\right)\right) F_{\mathrm{z,F}} \bar{\mu} \tag{3-5a}$$

$$F_{\mathrm{x,R}} = -\hat{\mu} \frac{s_{\mathrm{x,R}}}{s_{\mathrm{R}}} D_{\mathrm{R}} \sin\left(C_{\mathrm{R}} \arctan\left(B_{\mathrm{R}} s_{\mathrm{R}}\right)\right) F_{\mathrm{z,R}} \tag{3-5b}$$

$$F_{\mathrm{y,R}} = -\hat{\mu} \frac{s_{\mathrm{y,R}}}{s_{\mathrm{R}}} D_{\mathrm{R}} \sin\left(C_{\mathrm{R}} \arctan\left(B_{\mathrm{R}} s_{\mathrm{R}}\right)\right) F_{\mathrm{z,R}} \tag{3-5c}$$

Notice that updating $\hat{\mu}$ has the same effect as updating the D_F and D_R parameters in the Magic Formula. However, the intent was to preserve the original tire parameters and to insert an artificial scaler that tracks the departure of the amplitude from the nominal parameter.

The RLS algorithm with forgetting factor Γ can be found below:

1. Compute the state derivatives w.r.t to $\hat{\mu}$ (as the regression vector $\phi(k)$) and the linear approximation of the time-derivative of β and r:

$$\phi(k) = \left[\frac{l_{\mathrm{F}} F_{\mathrm{y,F}} \cos\delta - l_{\mathrm{R}} F_{\mathrm{y,R}}}{I_z}, \quad \frac{F_{\mathrm{y,F}} \cos(\delta - \beta) + F_{\mathrm{y,R}} \cos\beta - F_{\mathrm{x,R}} \sin\beta}{mV}\right]^T \tag{3-6}$$

$$\hat{\dot{x}} = \left[\frac{r(k) - r(k-1)}{T_s}, \quad \frac{\beta(k) - \beta(k-1)}{T_s} + r(k-1)\right]^T \tag{3-7}$$

2. Compute the current identification error $e(k)$:

$$e(k) = \hat{\dot{x}} - \phi(k)\hat{\mu}(k) \tag{3-8}$$

3. Compute the update gain K and update the covariance matrix P:

$$K(k) = P(k-1)\frac{\phi(k)}{\Gamma + \phi(k)^T P(k-1)\phi(k)} \tag{3-9}$$

$$P(k) = \frac{1}{\Gamma}\left(P(k-1) - \frac{P(k-1)\phi(k)\phi(k)^T P(k-1)}{\Gamma + \phi(k)^T P(k-1)\phi(k)}\right) \tag{3-10}$$

4. Update the current $\hat{\mu}$:

$$\hat{\mu}(k+1) = \hat{\mu}(k) + K^T e(k) \tag{3-11}$$

As (3-7) suggests, the algorithm is based on a linear approximation of $\dot{\beta}$ and $\dot{r}$. The upside is that the friction update is bringing minimal computational load, however abrupt changes in β and r might be a source of disturbance for its converge. Its suitability for realistic model mismatch scenarios will be further tested in Chapter 5.

3-5 Block diagram of proposed control structure

The proposed control structure, including the friction update and the reference disturbance estimator, can be visualized in Figure 3-4.

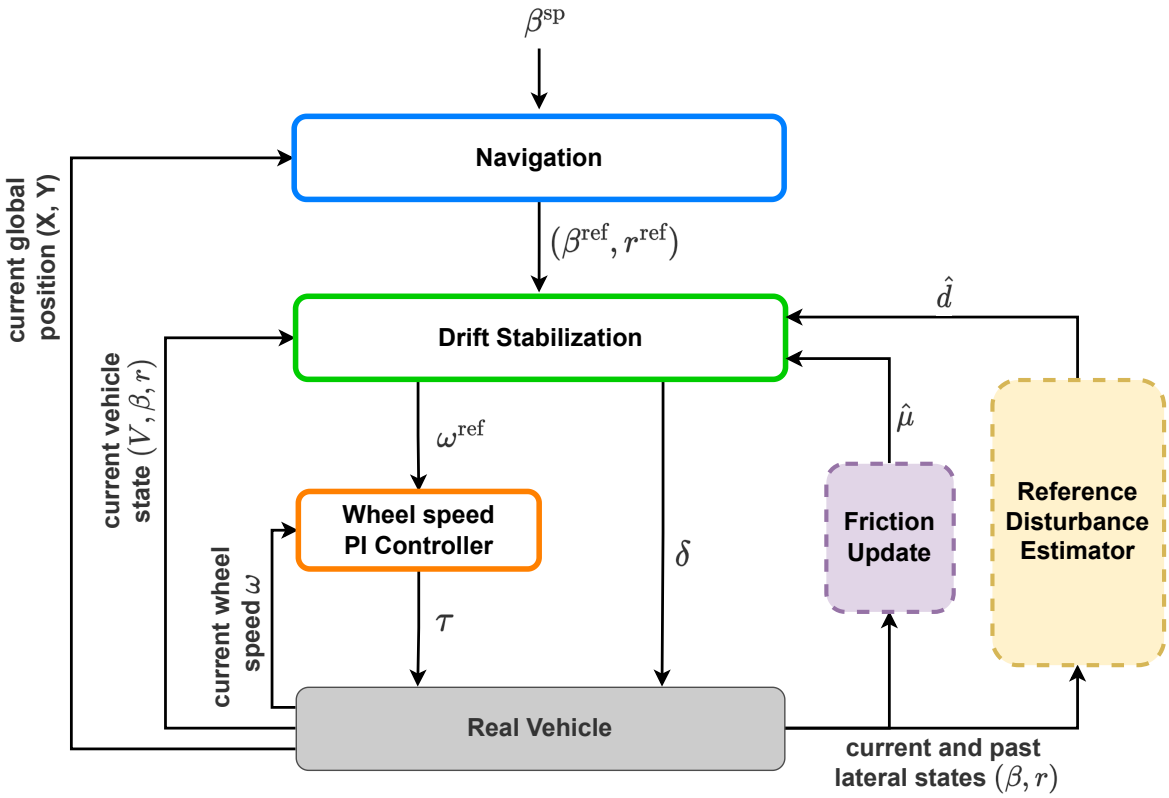


Figure 3-4: Block diagram of the proposed control structure.

3-6 Summary

This chapter has presented the proposed drifting controller from a high-level perspective. Its purpose is to motivate the choice of a hierarchical NMPC-based structure and the overall benefits it brings. Moreover, a reference disturbance estimator has been introduced and a simple friction update algorithm has been proposed to increase the robustness of the proposed control strategy to parametric uncertainty with minimal computational cost.
A detailed description of the Navigation and Drift Stabilization layers from an optimization point of view will be further presented in the next chapter.

Chapter 4

Optimization

Having chosen the Nonlinear Model Predictive Controller as the main tool in building the drift controller, the next step is defining the relevant options to solve the Optimal Control Problem (OCP) in real-time. First, it is necessary to define the shift from a control problem to an optimization problem using shooting methods, and solve options for the resulting problem. Laying this foundation ensures that the numerical improvements are well motivated. Second, the naive All-in-one approach will be described to construct a baseline for comparison. Lastly, this chapter will accurately describe the individual components of thw two OCP, one for Planner and one for Controller.

It is worth mentioning that the setup discussed so far can also be viewed as the basis for a computational problem with autonomous drifting particularities. The obstacles can be perceived as events that make the planner recompute its trajectory. The friction update imposes working with a varying parameter, further requiring to depart from previous Controller solutions. This should motivate the inclusion of both obstacles and friction update in the problem.

4-1 Optimization background

4-1-1 Direct Multiple Shooting

As noted in [5], direct shooting methods tend to deliver the best results in practice. On one hand, methods based on solving the Hamilton-Jacobi-Bellman equation suffer from the curse of dimensionality (because of the state-space tabulation). On the other hand, Indirect Methods, which require solving a Boundary Value Problem, provide better accuracy but require precise initial guess for convergence. This is hardly available in drifting scenarios.

Direct methods transform the initial infinite OCP into a finite-dimensional Nonlinear Programming (NLP) problem which is later solved by structure-exploiting numerical optimization methods, functioning on the premise 'first discretize, then optimize'. They can use state-of-the-art solving methods for the NLP solution and they are easily applicable (especially in

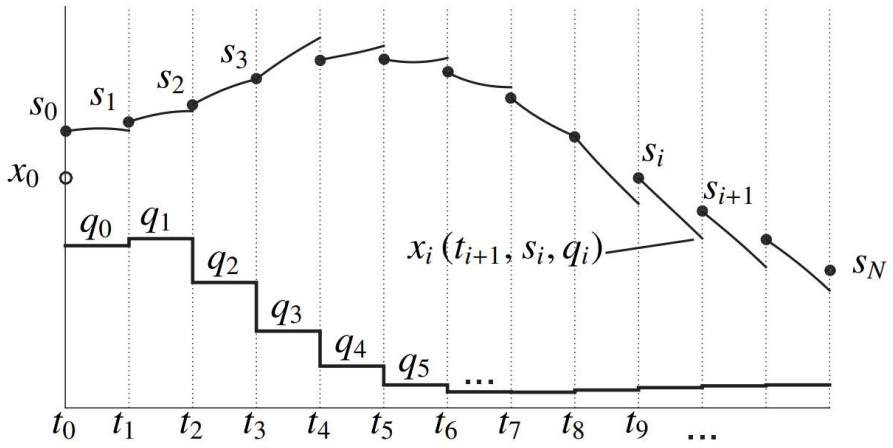


Figure 4-1: Direct Multiple Shooting [5].

treating inequality constraints) and robust, yet they obtain only suboptimal solutions (finite-time).

Direct Multiple Shooting [5, 22] is a simultaneous simulation-based approach which relies on numerical integration methods (typically a Runge-Kutta method) to obtain the system dynamics in discrete time. Using this shooting method, the resulting in better convergence properties, dealing with a bigger, but less nonlinear problem than Single Shooting Methods. Moreover, in comparison with Direct Collocation Methods, which construct a sparser, larger, even less nonlinear problem at the end, the Direct Multiple Shooting method considers only the starting state and control input [2]. In each control interval, there are no intermediary collocation points, lowering the computational complexity.

Based on the discretized states s and control inputs q from (4-1)-(4-3), the procedure can be visualized in Figure 4-1. In few words, the method divides the horizon in N segments, N representing the prediction horizon length. Further, it computes a piece-wise discretization of the control action for each resulting time interval $[t_i, t_{i+1}]$, each interval starting with the artificial initial state s_i [5]:

$$u(t) = q_i, \quad t \in [t_i, t_{i+1}] \tag{4-1}$$

$$x_i(t_i, s_i, q_i) = s_i \quad \text{(initial condition)}, \tag{4-2}$$

where x_i is the state trajectory computed using RK4.

In contrast with Single Shooting, the method will solve a separate Ordinary Differential Equation (ODE) for each interval $[t_i, t_{i+1}]$ [5]:

$$\dot{x}_i(t, s_i, q_i) = f(x_i(t, s_i, q_i), q_i), \quad t \in [t_i, t_{i+1}] \tag{4-3}$$

To avoid the mismatches between the end of a state simulation $(x_i(t_{i+1}); s_i, q_i)$ and the following initial state s_{i+1}) (from the next control interval), as seen in Figure 4-1, continuity constraints are imposed:

$$x_i(t_{i+1}, s_i, q_i) - s_{i+1} = 0 \tag{4-4}$$

Likewise, we can also compute the stage-cost component l_i associated with the choice on the interval $[t_i, t_{i+1}]$:

$$l_i(s_i, q_i) = \int_{t_i}^{t_{i+1}} L(x_i(t, s_i, q_i), q_i)dt \tag{4-5}$$

The resulting NLP problem can be described by (4-6), according to the notation from [5]:

$$\begin{aligned} \underset{s,q}{\text{minimize}} \quad & \sum_{i=0}^{N-1} l_i(s_i, q_i) + E(s_N) \\ \text{subject to:} \quad & s_0 - x_0 = 0 \qquad \text{(initial condition)} \\ & x_i(t_{i+1}, s_i, q_i) - s_{i+1} = 0, \qquad i = 0, ..., N-1 \\ & h(s_i, q_i) \leq 0, \qquad i = 0, ..., N, \end{aligned} \tag{4-6}$$

with h denoting generic (nonlinear) inequality constraints.

Since the optimization variables are organized in consecutive state-input pairs $z := (s_0, q_0, ..., s_N)$, the Jacobian and the Hessian are block sparse, a structure that can be further exploited by solvers such as FORCES Pro [23, 24].

For simplicity, the continuity and initial condition constraints will not be shown in future OCP formulations for clarity purposes. The purpose will be to transmit the suggested contribution, and adding additional constraints to drifting constraints might hinder the reader from putting them into context.

Concretely, the Direct Multiple Shooting offers a great trade-off between nonlinearity and computational complexity, and will be further considered in this work using the 4th-order Runge-Kutta (RK4) method for integration.

4-1-2 Solving the NLP

Lagrange Functions and Duality

In essence, the duality principle allows us to associate a *dual problem* with the optimal value d^*, to the original minimization problem (*primal problem*, with optimal value p^*), the former acting like a lower bound to the solution of the latter.

In the interest of avoiding conflicts with previous notations, a generic, possibly nonconvex, constrained optimization problem is described in (4-7), following the ideas from [25]:

$$\begin{array}{cc} \textbf{Primal} & \textbf{Dual} \\ f : \mathbb{R}^n \to \mathbb{R}, g : \mathbb{R}^n \to \mathbb{R} & G : \mathbb{R}^m \to \mathbb{R} \\ \underset{x}{\text{minimize}} \quad f(x) & \underset{\lambda}{\text{maximize}} \quad G(\lambda) \\ \text{subject to} \quad g_i(x) \leq 0, i = 1, ..., m & \text{subject to} \quad \lambda \geq 0 \end{array} \tag{4-7}$$

The basic idea in Lagrangian duality is to take the constraints into account by augmenting the objective function with a weighted sum of the constraint [25]. The Lagrangian $L : \mathbb{R}^{n+m} \to \mathbb{R}$

associated with the Primal problem is described below:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \tag{4-8}$$

In the above equation, λ_i are called Lagrange multipliers or dual variables. The Lagrange dual function G (which is concave regardless of the convexity of f_i) is defined as:

$$G(\lambda) = \inf_x L(x, \lambda) \tag{4-9a}$$

$$= \inf_x \left(f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) \right) \tag{4-9b}$$

The advantage in this reformulation is that the dual problem, if transformed to a minimization problem, is always a convex problem, even if the primal problem is not.

In general, the two optimal values do not to coincide, a condition called *weak duality* $d^* \leq p^*$ [25]. The difference between the two solutions $(p^* - d^*)$ is called *optimal duality gap*. Usually, for convex problems we deal with *strong duality* (verified using Slater's Condition [25]), which guarantees that the two optimal values coincide: $p^* = d^*$.

Karush-Kuhn-Tucker (KKT) conditions

The KKT conditions are a set of sufficient conditions for optimality. In other words, if x^*, λ^*, and μ^* satisfy the KKT conditions, it is sufficient to conclude they are the optimal solutions to the primal and dual problems. For a constrained minimization problem, with f the objective function, g representing the inequality constraints and h the equality constraints

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \qquad i = 1, ..., M \\ & h_j(x) = 0, \qquad j = 1, ..., L \end{aligned}$$

the Karush-Kuhn-Tucker conditions are the following:

$$\textbf{Stationarity}: \ \nabla f(x) + \sum_{i=1}^{m} \mu_i \nabla g_i(x) + \sum_{j=1}^{l} \lambda_j \nabla h_j(x) = 0 \tag{4-10a}$$

$$\textbf{Primal feasibility}: \ g_i(x) \leq 0, \tag{4-10b}$$

$$h_j(x) = 0, \tag{4-10c}$$

$$\textbf{Dual feasibility}: \ \mu_i \geq 0, \tag{4-10d}$$

$$\textbf{Complementary slackness}: \ \sum_{i=1}^{m} \mu_i g_i(x) = 0 \tag{4-10e}$$

where μ_i and λ_j are the associated dual variables.

A relevant observation is that for a convex problem with nonaffine constraints, if there is a feasible x satisfying the strict inequalities, then strong duality holds (in this case, KKT becomes necessary conditions for optimality).

Primal-Dual Interior Point Method

The sparse problem from (4-6) can be solved directly using a Primal-Dual Interior Point (PDIP). As discussed in [5], the Primal-Dual method holds several advantages over the Barrier Interior Point method, providing superlinear convergence, allowing the start from an infeasible point, and offering a higher efficiency in practice.
For the purpose of illustrating the steps in PDIP, the following constrainted convex optimization problem is introduced:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \\ & Cx + c = 0 \end{aligned}$$

The nonnegative slack variables s are introduced to relax the original KKT conditions (4-10):

$$\nabla f(x) + \nabla g(x)^T \lambda + C^T \mu = 0 \tag{4-11a}$$
$$Cx + c = 0 \tag{4-11b}$$
$$g(x) + s = 0 \tag{4-11c}$$
$$\lambda^T s = K \tag{4-11d}$$
$$\lambda, s \geq 0, \tag{4-11e}$$

where K is the barrier function multiplier ($K = 0$ corresponds to the primal solution).

The current document will summarize the steps in PDIP as presented in [5, 24].
The central idea of PDIP is to keep the dual multipliers for inequalities λ as optimization variables, which are implicitly defined by the primal barrier function in the Barrier Interior Point method in contrast. The next step is to solve both primal and dual problems using the above relaxed KKT conditions. If the KKT conditions hold, the iterate is on the central path, and by further reducing K to 0, the algorithm follows this central path to solution.

The conditions from (4-10) are considered a root finding problem and are further solved using Newton's method in order to find the optimal $(\Delta x, \Delta\mu, \Delta\lambda, \Delta s)$ directions. At each step, the KKT conditions are linearized and the resulting linear system from (4-12) is solved.

$$\underbrace{\begin{bmatrix} H & C^T & J^T & 0 \\ C & 0 & 0 & 0 \\ J & 0 & 0 & 0 \\ 0 & 0 & S & \Lambda \end{bmatrix}}_{\frac{\partial r}{\partial z}} \underbrace{\begin{bmatrix} \Delta x \\ \Delta\mu \\ \Delta\lambda \\ \Delta s \end{bmatrix}}_{\Delta z = z_{k+1} - z_k} = - \underbrace{\begin{bmatrix} r_C \\ r_E \\ r_I \\ r_s \end{bmatrix}}_{r} \tag{4-12}$$

The steps of the algorithm can be found below:

1. Initialize $z_0 = (x_0, \mu_0, \lambda_0, s_0)$
2. Newton's root finding problem (Linearized KKT) from (4-12). Obtain Δz_i.
3. Determine step size α (Backtracking line search - Armijo-Goldstein condition [5])

4. $z_{i+1} = z_i + \alpha \Delta z_i$. Return to step 2.

As noted in [5], $\approx 95\%$ of the computation time is solving the linear system from (4-12).

Ferreau mentions in [5] that Interior Point Method can be formulated to give a runtime guarantee and is relatively easy to parallelize, but it has limited warm-starting capabilities. Consequently, applications usually implement a practically-proven version of IPM with no runtime guarantee.

Furthermore, as stated in [26], the complexity is linear in terms of horizon length ($\mathcal{O}(N)$), but cubic in terms of the number of differential states ($\mathcal{O}(n_x^3)$). This suggests lowering the state vector of both Planner and Controller OCP as much as possible to provide the biggest decrease in computational burden.

PDIP vs SQP

Lastly, several works in the literature rely on a Sequential Quadratic Programming (SQP) method to solve the original NLP, which tends to be faster and provide less accurate solutions [13, 27]. The accuracy is less valuable for the purpose of this work as there is greater emphasis on reducing the computational complexity. In other words, sufficiently good solutions are preferred instead of highly-accurate, more computationally demanding ones.

However, experimental tests have shown that the local successive convex approximations that are part of SQP are not suitable for drifting. Several tests have shown that controlling drifting maneuvers with SQP required precise initial guesses to arrive at an obstacle-avoidance trajectory. Moreover, despite encountering numerical errors also while using a PDIP method, the empirical observations suggest that this happened more frequently while using SQP. Lastly, the use of the PDIP for similar scenarios has been validated in [13].

FORCES Pro

FORCES Pro [23, 24] introduces an efficient method for solving the OCP, providing significant reduction in computation time. As the authors note, for typical MPC this improvement reaches $\approx -75\%$ reduction [5, 24]. Moreover, the use of FORCES Pro for autonomous racing of model cars with obstacle-avoidance features has been validated in [16]. The solving time for these experiments sits in the millisecond range, making it highly-valuable for the purpose of this work. The included example set also provides guidelines for path-tracking and obstacle-avoidance features.

Altogether, the FORCES Pro environment is a great candidate for solving the resulting NLP from (4-6) and will be the basis for implementing the Nonlinear Model Predictive Controller in this work.

4-2 Baseline approach (All-in-one formulation)

As stated at the beginning of the chapter, it is necessary to construct a baseline model to measure the comparative performance of the proposed approach.

In this sense, the All-in-one formulation was created as a single NMPC that simultaneously achieves both objectives:

1. Obstacle-avoidance and central path following. Simply put, this requires the vehicle's global position to be inside the vehicle track and outside any obstacle by a predefined margin. As it will be explained further in the chapter, this margin is equal to the vehicle's half-diagonal.
2. Drift Stabilization. The control structure is tasked with tracking the desired side slip setpoint β^{sp}. The wheel dynamics are not incorporated into the vehicle model, meaning that the NMPC controls the wheel speed reference ω^{ref} and steering δ inputs. A low-level PI Controller is used to compute throttle commands τ.

The central path of the track is given as N pairs of global waypoints $(X^{\text{wp}}, Y^{\text{wp}})$. These waypoints are selected by minimizing the Euclidian distance between vehicle position and track, and taking the next $N-1$ points after the closest point.
The optimization variable vector Z can be found below:

$$Z = \begin{bmatrix} \epsilon & d\omega^{\text{ref}} & d\delta & \omega^{\text{ref}} & \delta & V & \beta & r & X & Y & \phi \end{bmatrix}^T, \tag{4-13}$$

where ϵ represents a nonnegative slack variable, and $d\omega^{\text{ref}}$ and $d\delta$ are the differences in wheel speed reference and steering angle, respectively. The rate constraints are necessary to ensure physically achievable commands and to comply with the mechanical limitations of a vehicle.

The Optimal Control Problem for the All-in-one formulation can be found below:

$$\min_{Z} \quad \sum_{k=1}^{N} \left[(X - X^{\text{wp}})^2 + (Y - Y^{\text{wp}})^2 + (\beta - \beta^{\text{sp}})^2 + (d\omega^{\text{ref}})^2 + (d\delta)^2 + \epsilon \right] \tag{4-14a}$$

$$\text{s.t.} \quad Z_{k+1} = f_{\text{vehicle}}(Z_k) \qquad \text{[Vehicle dynamics]} \tag{4-14b}$$

$$\underline{Z} \leq Z_k \leq \overline{Z} \qquad \text{[Hard constraints (bounds)]} \tag{4-14c}$$

$$X^2 + Y^2 + \epsilon \leq R_{\text{road}} - d_{\text{car}} \qquad \text{[Road outer boundary]} \tag{4-14d}$$

$$X^2 + Y^2 - \epsilon \geq r_{\text{road}} + d_{\text{car}} \qquad \text{[Road inner boundary]} \tag{4-14e}$$

$$\frac{(X - X_{\text{O1}})^2}{(R_{\text{O1}} + \text{c}\, d_{\text{car}})^2} + \frac{(Y - Y_{\text{O1}})^2}{(R_{\text{O1}} + \text{c}\, d_{\text{car}})^2} + \epsilon \geq 1 \qquad \text{[Obstacle O}_1\text{]} \tag{4-14f}$$

$$\frac{(X - X_{\text{O2}})^2}{(b_{\text{O2}} + \text{c}\, d_{\text{car}})^2} + \frac{(Y - Y_{\text{O2}})^2}{(a_{\text{O2}} + \text{c}\, d_{\text{car}})^2} + \epsilon \geq 1 \qquad \text{[Obstacle O}_2\text{]} \tag{4-14g}$$

In the above equations, the vehicle dynamics f_{vehicle} are created by stacking the Kinematic Bicycle Model, Dynamic Bicycle Model, and Pacejka's Magic Formula, while $\underline{Z}$ and $\overline{Z}$ are the lower and upper bounds on optimization variables, respectively. R_{road} and r_{road} denote

the outer and inner radii of the track, respectively, and d_{car} is the vehicle's half-diagonal. Note that the obstacle O_1 is modeled as an ellipse centered in (X_{O1}, Y_{O1}) of radii (a_{O1}, b_{O1}), while obstacle O_2 is modeled as a circle centered in (X_{O2}, Y_{O2}) of radius R_{O2}. The obstacle dimensions are enlarged by the scalar called clearance, denoted by c.

As the simulations account only for circular tracks and U-turns, the track limits are modeled using only two concentric circles, the outer road boundary being a circle of radius R_{road} and the inner one a circle of radius r_{road}.

Note that the obstacle-avoidance constraints are softly constrained to avoid controller infeasibility. This choice enforces the generation of a computational baseline despite violating the enlarged obstacle constraint. However, coupled with this added clearance, this approach should ensure a collision-free trajectory with the real-sized obstacle as tests have shown.

Lastly, all options that the FORCES Pro package makes available to improve the solving time are active for this formulation. This will become relevant in Chapter 5.

4-3 Proposed approach

This section is split into three subsection, each describing an improvement that successfully reduces the overall complexity of the drift control problem:

1. A hierarchical decomposition of the original problem into Navigation and Drift Stabilization layers as seen in Chapter 3. In comparison with the previous chapter, this subsection will focus on improvements from a computational perspective.
2. The reformulation of the Navigation layer into curvilinear coordinates to exploit the properties of circular tracks. Moreover, this reformulation also innately allows to incentivize circular trajectories that are friendly for drifting.
3. The assumption of constant velocity over the planning horizon which further reduces problem size without major compromises.
4. Generic simplifications that are not targeted at any drifting particularity. These improvements can be applied outside the area of autonomous drifting with minor modifications, but in some aspects they still can be regarded as problem-specific.

4-3-1 Hierarchical decomposition

First, as explained also in Chapter 3, the original formulation can be split into two objectives: navigation and drift stabilization. From a computational perspective, this brings two benefits:

1. First, splitting vehicle dynamics into motion planning and drift dynamics lowers the nonlinearity in each problem. This favors faster solving times at the expense of simplifying the interactions between vehicle states and global coordinates. The overall performance loss will be equal to the tracking error between the two layers. However, as it will be also validated experimentally, this will not be problematic, while the computational benefits are considerable.

2. Second, investigating the OCP of the All-in-one formulation reveals that both kinematic and dynamic states are propagated for the same horizon length N. This brings marginal benefits for the overall performance, and is highly detrimental computation-wise. While the generous look ahead given by propagating kinematics further is important for generating proper, smooth trajectories, the propagation of dynamics is computationally expensive and hardly needed in the long-term.

Based on these drawbacks of the original formulation, the proposed control structure splits the problem into a Navigation layer and a Drift Stabilization layer. The interplay between the two can be visualized in Figure 3-4. For compactness reasons, only the improvements in the OCP will be shown further.

Planner OCP

Starting with the Navigation layer, the Planner decision variables are contained in Z^P:

$$Z^P = \begin{bmatrix} \epsilon & dV^{\text{ref}} & d\beta^{\text{ref}} & dr^{\text{ref}} & V^{\text{ref}} & \beta^{\text{ref}} & r^{\text{ref}} & X & Y & \phi \end{bmatrix}^T, \tag{4-15}$$

where dV^{ref}, $d\beta^{\text{ref}}$, and dr^{ref} represent the difference in vehicle state references. Notice the superscript $*^{\text{ref}}$ that is used to indicate that the velocity V, the side slip angle β, and the yaw rate r are no longer states, but artificial inputs.
As discussed in Chapter 3, stacking two NMPC can lead to stability problems. To combat this, rate constraints are introduced to limit the changes in state references and sizable variations are further discouraged in the cost function.

The Planner OCP formulation can be found below:

$$\min_{Z^{\text{P}}} \quad \sum_{k=1}^{N_{\text{P}}} \Big[(X - X^{\text{wp}})^2 + (Y - Y^{\text{wp}})^2 + (\beta^{\text{ref}} - \beta^{\text{sp}})^2 + (dV^{\text{ref}})^2 + (d\beta^{\text{ref}})^2 + (dr^{\text{ref}})^2 + \frac{1}{\epsilon} \Big] \tag{4-16a}$$

$$\text{s.t.} \quad Z^{\text{P}}_{k+1} = f_{\text{planner}}(Z^{\text{P}}_k) \qquad \text{[Planner dynamics]} \tag{4-16b}$$

$$\underline{Z}^{\text{P}} \leq Z^{\text{P}}_k \leq \overline{Z}^{\text{P}} \qquad \text{[Hard constraints (bounds)]} \tag{4-16c}$$

$$X^2 + Y^2 + \epsilon \leq R_{\text{road}} - d_{\text{car}} \qquad \text{[Road outer boundary]} \tag{4-16d}$$

$$X^2 + Y^2 - \epsilon \geq r_{\text{road}} + d_{\text{car}} \qquad \text{[Road inner boundary]} \tag{4-16e}$$

$$\frac{(X - X_{\text{O1}})^2}{(R_{\text{O1}} + \text{c}\ d_{\text{car}})^2} + \frac{(Y - Y_{\text{O1}})^2}{(R_{\text{O1}} + \text{c}\ d_{\text{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_1\text{]} \tag{4-16f}$$

$$\frac{(X - X_{\text{O2}})^2}{(b_{\text{O2}} + \text{c}\ d_{\text{car}})^2} + \frac{(Y - Y_{\text{O2}})^2}{(a_{\text{O2}} + \text{c}\ d_{\text{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_2\text{]} \tag{4-16g}$$

The planner dynamics f_{planner} are based on the Kinematic Bicycle Model, while $\underline{Z}^{\text{P}}$ and $\overline{Z}^{\text{P}}$ are the lower and upper bounds on planner optimization variables, respectively.

The planners outputs N_{P} sets of state references $\left\{V^{\mathrm{ref}}, \beta^{\mathrm{ref}}, r^{\mathrm{ref}}\right\}_{N_{\mathrm{P}}}$ that ensure a collision-free path. The controller will use these pairs as state references.

Hereafter, the obstacle-avoidance constraints are modeled as hard constraints, in comparison with the softly-constrained avoidance from the All-in-one formulation. This is congruent with the expectations in the typical collision avoidance scenarios. The Planner is encouraged to increase the distance between the vehicle and obstacles by penalizing the inverse of slack variable $\frac{1}{\epsilon}$ in the cost function. Note that this slack variable has a negative sign in (4-16f) and (4-16g), suggesting that a positive ϵ indicates a positive distance between vehicle extremities and the enlarged obstacle boundary. This scheme will be used further in this work.

Controller OCP

The Controller decision variables are contained in Z^C:

$$Z^{\mathrm{C}} = \begin{bmatrix} d\omega^{\mathrm{ref}} & d\delta & \omega^{\mathrm{ref}} & \delta & V & \beta & r \end{bmatrix}^T, \tag{4-17}$$

with the notation introduced before.

The Controller OCP formulation can be found below:

$$\min_{Z^{\mathrm{C}}} \quad \sum_{k=1}^{N_{\mathrm{C}}} (V - V^{\mathrm{ref}})^2 + (\beta - \beta^{\mathrm{ref}})^2 + (r - r^{\mathrm{ref}})^2 + (d\omega^{\mathrm{ref}})^2 + (d\delta)^2 \tag{4-18a}$$

$$\text{s.t.} \quad Z^{\mathrm{C}}_{k+1} = f_{\mathrm{controller}}(Z^{\mathrm{C}}_k) \qquad \text{[controller dynamics]} \tag{4-18b}$$

$$\underline{Z}^{\mathrm{C}} \leq Z^{\mathrm{C}}_k \leq \overline{Z}^{C} \qquad \text{[hard constraints (bounds)]} \tag{4-18c}$$

The controller dynamics $f_{\mathrm{controller}}$ are based on the Dynamic Bicycle Model, having the Magic Formula embedded inside to model tire forces. Overall, the low-level drift stabilization layers is a straightforward reference-tracking NMPC. Inspired by works such as [10, 12], the fundamental improvement in computation time at this level comes from detaching the wheel speed dynamics from the controller dynamics. This has two significant advantages:

1. The state vector is shorter, and as noticed in [26], this has a cubic benefit from a computational perspective.
2. The nonlinearity around the generation of tire forces is considerably reduced as the wheel speed (now ω^{ref}) is implemented as an artificial control input rather than a state.

The Drift Stabilization layer generates N_{C} sets of wheel speed references $\left\{\omega^{\mathrm{ref}}, \delta\right\}$. The first steering angle δ_k can be dirrectly applied to the vehicle, while the first wheel speed reference ω^{ref}_k is sent to the low-level PI Controller to generate the throttle input τ.

Drawbacks

The proposed decomposition is successful in reducing the nonlinearity of the problem, making the problem more numerically stable and bringing considerably faster solving times. However, there are three fundamental drawbacks, all of them strictly related to the Navigation layer:

1. The track limits introduce a nonlinear, nonconvex constraint which is detrimental for computation times.
2. The waypoint-tracking strategy creates an implicit velocity reference since providing positional references at certain samples includes an inherent temporal dimension. This can be tackled by contracting or dilating the samples of the central path that is being tracked, but this might make the initial search for the closest point on track computationally expensive.
3. As drifting is hardly suitable for planar trajectories with reduced curvature such as straights, it is hard to incentivize the planner to generate circular trajectories that are friendlier for drifting maneuvers.

These drawbacks will be resolved by reformulating the Planner in curvilinear coordinates.

4-3-2 Reformulation in curvilinear coordinates

The curvilinear dynamics will be reminded for convenience purposes in (4-19), with the suitable notations for the Navigation layer instead of the generic ones.

$$\dot{S} = \frac{V^{\text{ref}} \cos(\theta - \beta^{\text{ref}})}{1 + n\kappa} \tag{4-19a}$$

$$\dot{n} = V^{\text{ref}} \sin(\theta - \beta^{\text{ref}}) \tag{4-19b}$$

$$\dot{\theta} = r^{\text{ref}} - \kappa \frac{V^{\text{ref}} \cos(\theta - \beta^{\text{ref}})}{1 + n\kappa}, \tag{4-19c}$$

where the subscripts $*^{\text{ref}}$ were added for consistency with previous subsections.

By reformulating the Navigation layer in curvilinear coordinates, all the previously mentioned drawbacks will be tackled:

1. The track boundaries no longer constitute a nonconvex constraint since in the curvilinear model this can be encoded as a simple bound on the lateral deviation n.
2. There is no need for a waypoint-tracking scheme since tracking the central path can be stated converted to minimizing lateral deviation from the central path.
3. The planner can incentivize circular trajectories by penalizing the lateral deviation time-derivative $\dot{n}$.

For compactness reasons, since the reformulation provides no alterations in the Drift Stabilization layer, only the description of the Planner OCP is given. Having the optimization variables stored in Z^P, this OCP is described below:

$$Z^P = \begin{bmatrix} \epsilon & dV^{\text{ref}} & d\beta^{\text{ref}} & dr^{\text{ref}} & V^{\text{ref}} & \beta^{\text{ref}} & r^{\text{ref}} & S & n & \theta \end{bmatrix}^T \tag{4-20}$$

$$\min_{Z^{\mathrm{P}}} \quad \sum_{k=1}^{N_{\mathrm{P}}} \left[(\theta - \beta^{\mathrm{sp}})^2 + \sin(\theta - \beta^{\mathrm{ref}})^2 + n^2 + (d\beta^{\mathrm{ref}})^2 + (dr^{\mathrm{ref}})^2 + \frac{1}{\epsilon} \right] \tag{4-21a}$$

$$\text{s.t.} \quad Z^{\mathrm{P}}_{k+1} = f_{\mathrm{planner}}(Z^{\mathrm{P}}_k) \qquad \text{[Planner dynamics]} \tag{4-21b}$$

$$\underline{Z}^{\mathrm{P}} \leq Z^{\mathrm{P}}_k \leq \overline{Z}^{\mathrm{P}} \qquad \text{[Hard constraints (bounds)]} \tag{4-21c}$$

$$\frac{(S - S_{\mathrm{O1}})^2}{(R_{\mathrm{O1}} + \mathrm{c}\, d_{\mathrm{car}})^2} + \frac{(n - n_{\mathrm{O1}})^2}{(R_{\mathrm{O1}} + \mathrm{c}\, d_{\mathrm{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_1\text{]} \tag{4-21d}$$

$$\frac{(S - S_{\mathrm{O2}})^2}{(b_{\mathrm{O2}} + \mathrm{c}\, d_{\mathrm{car}})^2} + \frac{(n - n_{\mathrm{O2}})^2}{(a_{\mathrm{O2}} + \mathrm{c}\, d_{\mathrm{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_2\text{]} \tag{4-21e}$$

Note that obstacle positions are expressed in curvilinear coordinates $(S_{\mathrm{O1}}, n_{\mathrm{O1}})$ and $(S_{\mathrm{O2}}, n_{\mathrm{O2}})$, based on the track progress S and the lateral deviation n.
The track boundaries are now formulated as bounds on the lateral deviation n in (4-21c). This is a significant computational gain since the former constraints were nonconvex.
Moreover, the side slip tracking problem is formulated by penalizing the misalignment between the local heading θ and the desired side slip setpoint β^{sp}. In addition to this, the change in lateral deviation is penalized by the $\sin(\theta - \beta^{\mathrm{ref}})^2$ term, and the vehicle is incentivized to approach the central path by n^2.

Drawbacks

The reformulation in curvilinear coordinates greatly reduced the computation time of the control structure. Regardless, the previously described Planner OCP still propagates dynamics that are unnecessary for the control objective. As a result, there is one more assumption that can further reduce the computational load without compromising obstacle avoidance and drift stabilization.

4-3-3 Assumption of constant velocity over planning horizon

Taking this one step further, an assumption on obstacle-avoidance scenarios can be made:

> Considering the usual obstacle-avoidance scenario in a receding horizon fashion, the short-term ability to avoid collisions is heavily influenced by the lateral states in DBM (side slip angle β and yaw rate r), while the effect of velocity V is negligible until further ahead in the future.

This allows the assumption of a constant velocity $V^{\mathrm{ref}} = V^{\mathrm{rp}}$ throughout the planned motion, where the superscript $*^{\mathrm{rp}}$ indicates that the velocity is now a runtime parameter. This would further reduce the Planner decision variables vector Z^{P} and make the overall problem faster.

Precisely, the Planner velocity from the curvilinear dynamics in (4-19) can be given as a runtime parameter that is equal to the current vehicle velocity, hence motivating the notation.

As a result, the size of the Planner artificial inputs is reduced now, containing only the side slip reference β^{ref} and yaw rate reference r^{ref}.

Naturally, if the Planner velocity can be set constant for the whole prediction horizon, it can be set to a higher velocity to travel a longer distance in the same number of samples. This would be a significant increase in the look ahead distance with no additional computational cost as the prediction horizon N^{P} will be remain constant. Consequently, the Planner velocity can be chosen as following:

$$V^{\text{rp}} = \eta V_k, \tag{4-22}$$

where η is called haste factor, and V_k is the vehicle velocity at step k.

However, given that the planned motion is simulated for a higher velocity, in order to obtain the intended turn curvature, which is defined as:

$$\kappa = \frac{r}{V}, \tag{4-23}$$

the yaw rate reference r^{ref} the Controller receives needs to be adjusted accordingly:

$$r^{\text{ref, C}} = \frac{r^{\text{ref}}}{\eta}, \tag{4-24}$$

where the additional superscript $*^{\text{ref, C}}$ denotes the reference the Controller is receiving.

Proposed Planner OCP

Combining the previously mentioned features, the Planner decision variables are contained in Z^{P}:

$$Z^P = \begin{bmatrix} \epsilon & d\beta^{\text{ref}} & dr^{\text{ref}} & \beta^{\text{ref}} & r^{\text{ref}} & s & n & \theta \end{bmatrix}^T, \tag{4-25}$$

The reformulated Planner OCP is described below:

$$\min_{Z^{\text{P}}} \quad \sum_{k=1}^{N_{\text{P}}} \left[(\theta - \beta^{\text{sp}})^2 + \sin(\theta - \beta^{\text{ref}})^2 + n^2 + (d\beta^{\text{ref}})^2 + (dr^{\text{ref}})^2 + \frac{1}{\epsilon} \right] \tag{4-26a}$$

$$\text{s.t.} \quad Z^{\text{P}}_{k+1} = f_{\text{planner}}(Z^{\text{P}}_k) \qquad \text{[Planner dynamics]} \tag{4-26b}$$

$$\underline{Z}^{\text{P}} \leq Z^{\text{P}}_k \leq \overline{Z}^{\text{P}} \qquad \text{[Hard constraints (bounds)]} \tag{4-26c}$$

$$\frac{(S - S_{\text{O1}})^2}{(R_{\text{O1}} + \text{c}\, d_{\text{car}})^2} + \frac{(n - n_{\text{O1}})^2}{(R_{\text{O1}} + \text{c}\, d_{\text{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_1\text{]} \tag{4-26d}$$

$$\frac{(S - S_{\text{O2}})^2}{(b_{\text{O2}} + \text{c}\, d_{\text{car}})^2} + \frac{(n - n_{\text{O2}})^2}{(a_{\text{O2}} + \text{c}\, d_{\text{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_2\text{]} \tag{4-26e}$$

Following the same hierarchical structure, the planners outputs N_{P} sets of state references $\left\{\beta^{\text{ref}}, r^{\text{ref}}\right\}_{N_{\text{P}}}$ that ensure a collision-free path. The controller no longer receives a velocity reference, which has an overall positive effect on drift stabilization: the total velocity V is now freely used to stabilize drifting, which makes achieving both tracking objectives easier.

Proposed Controller OCP

For completeness, the Controller OCP is reintroduced despite that the only modification in its formulation is the lack of a velocity reference V^{ref} in the cost function. Having the decision variables stored in Z^{C}, the OCP is defined as:

$$Z^{\text{C}} = \begin{bmatrix} d\omega^{\text{ref}} & d\delta & \omega^{\text{ref}} & \delta & V & \beta & r \end{bmatrix}^T, \tag{4-27}$$

$$\min_{Z^{\text{C}}} \quad \sum_{k=1}^{N_{\text{C}}} (\beta - \beta^{\text{ref}})^2 + (r - r^{\text{ref}})^2 + (d\omega^{\text{ref}})^2 + (d\delta)^2 \tag{4-28a}$$

$$\text{s.t.} \quad Z^{\text{C}}_{k+1} = f_{\text{controller}}(Z^{\text{C}}_k) \qquad \text{[controller dynamics]} \tag{4-28b}$$

$$\underline{Z}^{\text{C}} \leq Z^{\text{C}}_k \leq \overline{Z}^{C} \qquad \text{[hard constraints (bounds)]} \tag{4-28c}$$

The low-level interface with the vehicle through the PI Controller remains unchanged.

The final control structure is implemented using the Optimal Control Problem formulations detailed in this last section. The performance and its sensitivity to parameter uncertainty will be further evaluated in Chapter 5.

4-3-4 Generic simplifications

Warm Starting. Successive NLP problems solved by an NMPC are very similar in nature, the only differences being the initial state x_0 and the change in the environment. Naturally, in our case this represents an obstacle found in the N-th sample of the prediction horizon. As a result, an intuitive improvement is providing the previous optimal solution as the initial guess for the next optimization problem. As there is no information about the last sample, the guess for the $(N-1)th$ sample will be utilized also for the last step in the horizon.

Curvature as runtime parameter. As the curvilinear dynamics presented in (4-19) require the road curvature κ to be given, the straightforward approach is to provide the Planner the arc-length parametrization of curvature $\kappa(s)$. However, this increases the nonlinearity of the problem and the solving time as a result. A solution is proposed to overcome this: instead of relying on the precise curvature function $\kappa(s)$, the curvature at step k will be given as a runtime parameter for each of the N_P steps. The curvature for each step in the horizon will be computed based on the progress made in the solution of the previous step $k-1$:

$$\kappa^{\text{rp}}_{N_P,k} = \kappa\left(s_{N_P,k-1}\right), \tag{4-29}$$

where $s_{N,k-1}$ indicates the N_P samples of track progress s from the Planner solution at step $k-1$, and $(\kappa^{\text{rp}}_k)_{N_P}$ represents the N_P samples of curvature runtime parameters to be used at step k.
For the first step, as there is no previous solution available, it is easy to assume a linear increase in progress based on the initial velocity V and to estimate the corresponding curvature.

Obstacle and vehicle dimensions. As hinted in the previous section, all the actors are modeled using simple shapes. First, the obstacles O_1 and O_2 are approximated by a circle and an ellipse, respectively. They are further enlarged to provide an additional safety margin to guarantee collision-free trajectories.
Second, the vehicle dimensions in the Planner are exaggerated and approximated by a circle with the radius equal to the vehicle's half-diagonal. This further represents an additional step to increase the distance between the real vehicle and obstacles.

Relaxed tolerances. As the purpose of the drift controller is fundamentally centered on ensuring collision-free trajectory with restrictive time availability, the final residuals in 4-12 are of lesser importance. The goal is to ensure the computational feasibility, and this can be further ensured by relaxing the tolerances for the solutions of the NLP. As suggested also in [16], the optimality criteria are aimed at finding sufficiently good solutions, increasing the magnitude of the admissible residuals for equality and inequality constraints to 10^{-5}, and for the duality gap to 10^{-4}.
Naturally, the custom FORCES Pro solver options are set to optimize the generated code toward minimizing solving times.

4-4 Summary

Since the central actor in the proposed control structure is a Nonlinear Model Predictive Controller, a great emphasis needs to be placed on Optimal Control Problem formulation to ensure real-time feasibility. This chapter first introduced the necessary optimization concepts that are required to understand the motivation behind each numerical choice. Afterwards, the original optimization problem was split and conveniently reformulated with the aim to reduce the computational demand of the drifting controller. The last proposed improvement is the assumption of constant velocity over the Planner prediction horizon, which is then followed by generic computational improvements.
Putting the last two chapters together, the resulting control structure is ready to be tested in various scenarios that aim to measure its obstacle-avoidance capabilities, tracking performance, and sensitivity to model mismatch and imperfect estimations.

Chapter 5

Controller evaluation

This chapter is dedicated to validating the proposed control structure in various scenarios to determine its benefit and limitations. However, it is difficult to make definitive statements since the simulations still happen in a pristine environment, in comparison with an implementation on a physical medium.
Therefore, the purpose of this chapter is to shed light into the possible improvement this structure can bring into handling dangerous driving scenarios. The results represent an approximate bound on the maximum benefit it can bring since a more accurate model or interfacing it with a real vehicle will likely further decrease its performance.

First, as the autonomous drifting problem is regarded in this work primarily from a computational perspective, it is necessary to state the baseline against which the solving time of the proposed approach will be compared. Nevertheless, the only legitimate comparison will be with the experiment found in Section 5-4 because of the similarity of experiments.

Further, this chapter will demonstrate the convergence of the control scheme toward a drift equilibrium of the three-state bicycle model after starting in its vicinity. This chapter will then progress to scenarios that evaluate the nominal performance of the proposed approach, in the absence of parameter uncertainty and measurement noise.

As the proposed structure is fundamentally model-based, the last step is to do a more in-depth investigation on the robustness of the solution by constructing several metric-based simulations in harsher testing conditions. These simulations are dedicated to analyzing the sensitivity of the control structure to uncertainty in the following ares:

1. The tire-road friction model.
2. Initial lateral deviation w.r.t to central path.
3. Total mass and weight distribution between front and rear axles.
4. Side slip angle estimation error (both bias and variance).

The selected scenarios include both obstacle courses and obstacle-free circular tracks and U-turns. The evaluation metrics encompass the capability to complete the track without collisions, tracking errors w.r.t to both central path and side slip setpoint, and the spread of vehicle states and trajectories during the experiment.

It is worth noting that the obstacle avoidance scenarios stem from the central motivation of this work, being derived from the need to ensure passenger safety after losing vehicle control on slippery surfaces. Despite its relevance, it can be seen as a limited scenario due to the arbitrariness of the obstacle placement and size, which cannot be directly extended to real-life segments. This limitation is compensated by later test-cases which are more generic and can form a less biased opinion on the overall performance.

A table containing all the numerical values of vehicle parameters and details about the testing environment can be inspected in Appendix A.

5-1 Computation time baseline (All-in-one formulation)

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The goal is to navigate the track while avoiding two obstacles that are approximated by an ellipse and a circle, respectively.

This scenario will represent the baseline model for this work. A straightforward all-in-one NMPC formulation was created, containing both obstacle avoidance and drift stabilization goals in the same OCP, i.e. the two problems were not split and the planning dynamics (in this case, the Kinematic Bicycle Model) are formulated in Cartesian coordinates.

Notably, as all the vehicle states and details are part of the same OCP, the autonomous drifting task become significantly more difficult from an optimization perspective as the problem's nonlinearity has increased. This requires a good initial guess in order to find a drifting trajectory that can complete the obstacle course, and this scenario is a good example of this. Regardless, the solving time can still serve as the baseline for this work and can be observed in Figure 5-1.

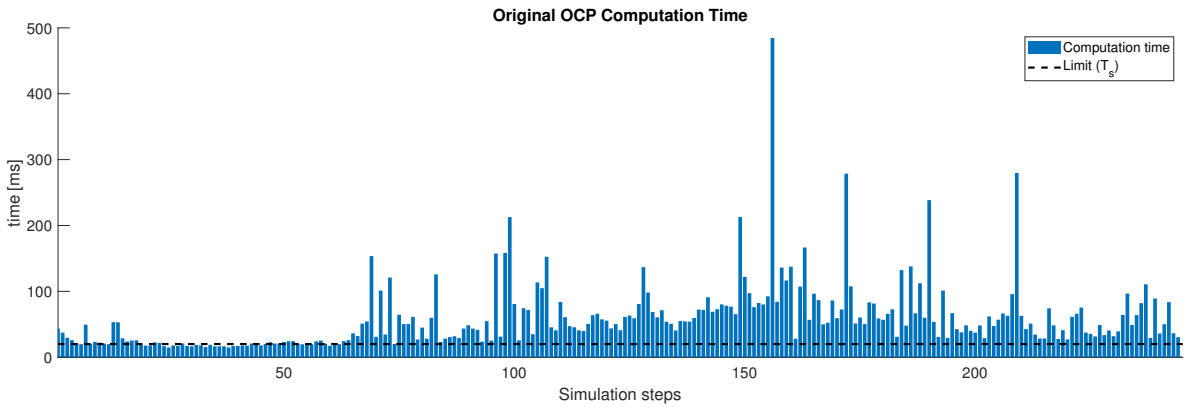


Figure 5-1: Computation time with the All-in-one formulation (Baseline).

	Total Time
Maximum [ms]	484.5774
90th Percentile [ms]	96.9145
Average [ms]	47.3654

Table 5-1: Baseline computation time.

The solving times exceeds the availability (T_s) in the vast majority of simulation iterations, making it unfeasible for a real-time implementation. As outlined in Table 5-1, in the worst-case scenario this availability is exceeded by approximately 25 times.

The sizable increase in solving time in comparison with the proposed solution can also be motivated by the lengthy prediction horizon (in this case, $N = 150$), which is required to plan collision-free paths. As both navigation and drift stabilization are propagated for the same number of steps, there is a significant burden in propagating dynamical states (V, β, r) that is both unnecessary and computationally expensive.

5-2 Convergence to equilibrium

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamicsg. The goal is to reach steady-state drifting in the absence of obstacles, with a minor incentive to stay close to the center path.

The motivation for this trial stems from splitting the vehicle dynamics between the planning and tracking layers. Since the Planner does not include information about dynamics, it naturally does not contain knowledge about drift equilibria. As a result, it is valuable to observe if the interplay between the two layers preserves the ability to convergence to a drift equilibrium while still being able to follow a desired path.

Naturally, this is fundamentally rooted in the assumption that the vehicle starts in the vicinity of a drifting equilibrium. Starting further away decreases the chance the Planner will generate references that can correspond to a vicinity of a drift equilibrium and the control structure will fail in accomplishing both drift stabilization and path tracking.

Figure 5-2 shows the evolution of vehicle states over time in this scenario. After the transient regime during which the planner guides the vehicle towards a (β^{ref}, r^{ref}) that matches the central path of the track, the controller starts converging towards the corresponding drift equilibrium.

A more insightful perspective can be obtained using the phase portrait presented in Figure 5-3. The bar presented on the right confirms that the first few samples are used for guiding the vehicle toward the path, while the vast majority of simulation iterations are spent on getting closer to the drift equilibrium. The phase portrait only describes the trajectories of the (β, r) pairs since these are at the core of the planner-controller interaction, however the equilibrium corresponds to the three-state Dynamic Bicycle Model (V, β, r).

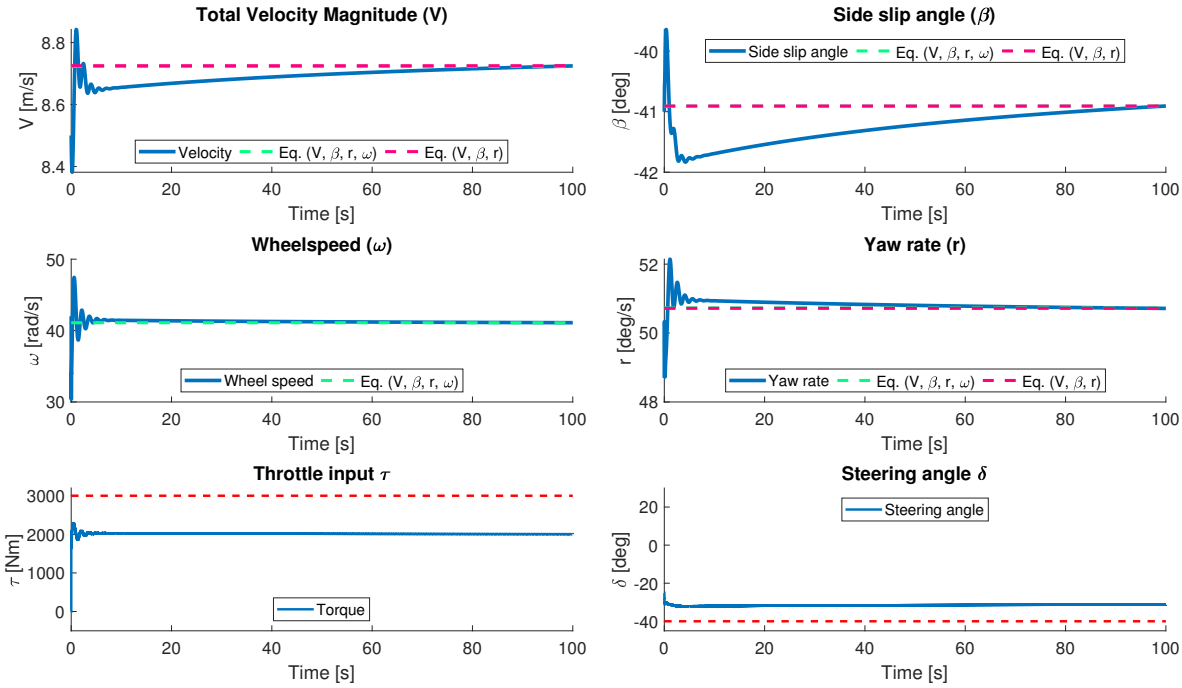


Figure 5-2: Convergence of vehicle states toward drift equilibrium.

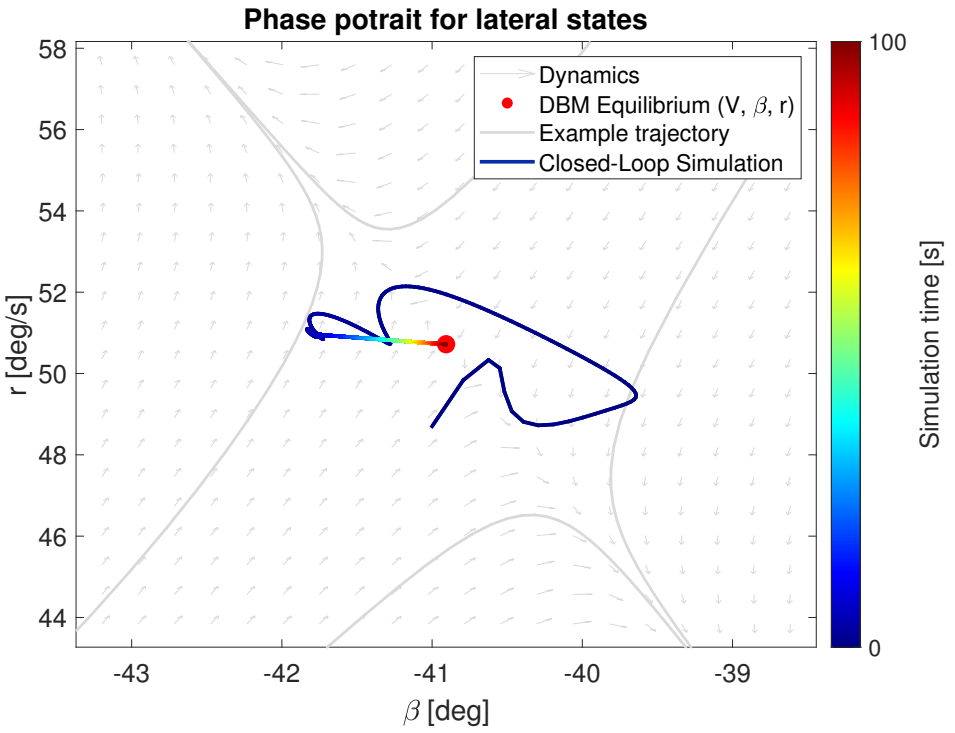


Figure 5-3: Phase portrait centered on a drift equilibrium.

5-3 Friction update

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The goal is to test the closed-loop response after a sudden road surface change and the effectiveness of the RLS friction estimation.

Considering that the availability of a precise friction model is essential for drift stabilization and control, one of the natural questions is regarding the availability of a precise friction model for the given tire-road interaction. This is especially valuable after a sudden change in friction that can be encountered for example beneath an overpass or in wilder ares where the tree coverage is non-uniform. In this sense, the friction coefficient update can be useful to retain control over drifting maneuvers.

Unlike a realistic scenario, this experiments only assumes that the new road surface produces only a change in amplitude, precisely changing only the D parameter in the Magic Formula and leaving C and B unaltered. The motivation behind this choice is the need to first test the friction update algorithm in a nominal case, in which the real friction change perfectly resembles how it is modeled in the algorithm. Naturally, this is over-simplistic since also the shape is altered during the transition to a new surface, and the assumption of a perfect initial fitting is naive (without which there would be a mismatch for both B and C coefficients). The sensitivity to these variations will be more extensively investigated in Section 5-8.

Vehicle trajectory

Figure 5-4 shows the vehicle behavior after a sudden friction change. Notice that the simulation allows the vehicle to make multiple passes. The control structure successfully reaches steady-state both before and after the friction coefficient change. Regardless, the second and third passes closely resemble the first path in terms of planar trajectory, making global coordinates samples indistinguishable at the end.
Moreover, the vehicle total velocity V, the wheel speed ω, and the throttle input τ are lower in comparison with the first 8 seconds of the simulation as the available friction diminishes.

Friction coefficient estimation

The evolution of the friction coefficient estimate ($\hat{\mu}$) over time can be analyzed in Figure 5-5.

The RLS friction update algorithm converges to the real friction coefficient both before and after the sudden friction change. However, the transient response is heavily influenced by the nonlinearity of the regime. This can be noticed in the transient of the friction coefficient estimate. In order to understand the influence of the dynamical nonlinearities on the friction update algorithm, the following scenario is described: the sudden change in road friction makes the vehicle deviate from the central path, making the planner recompute its trajectory to decrease the lateral deviation, changing the state references in the process. This change

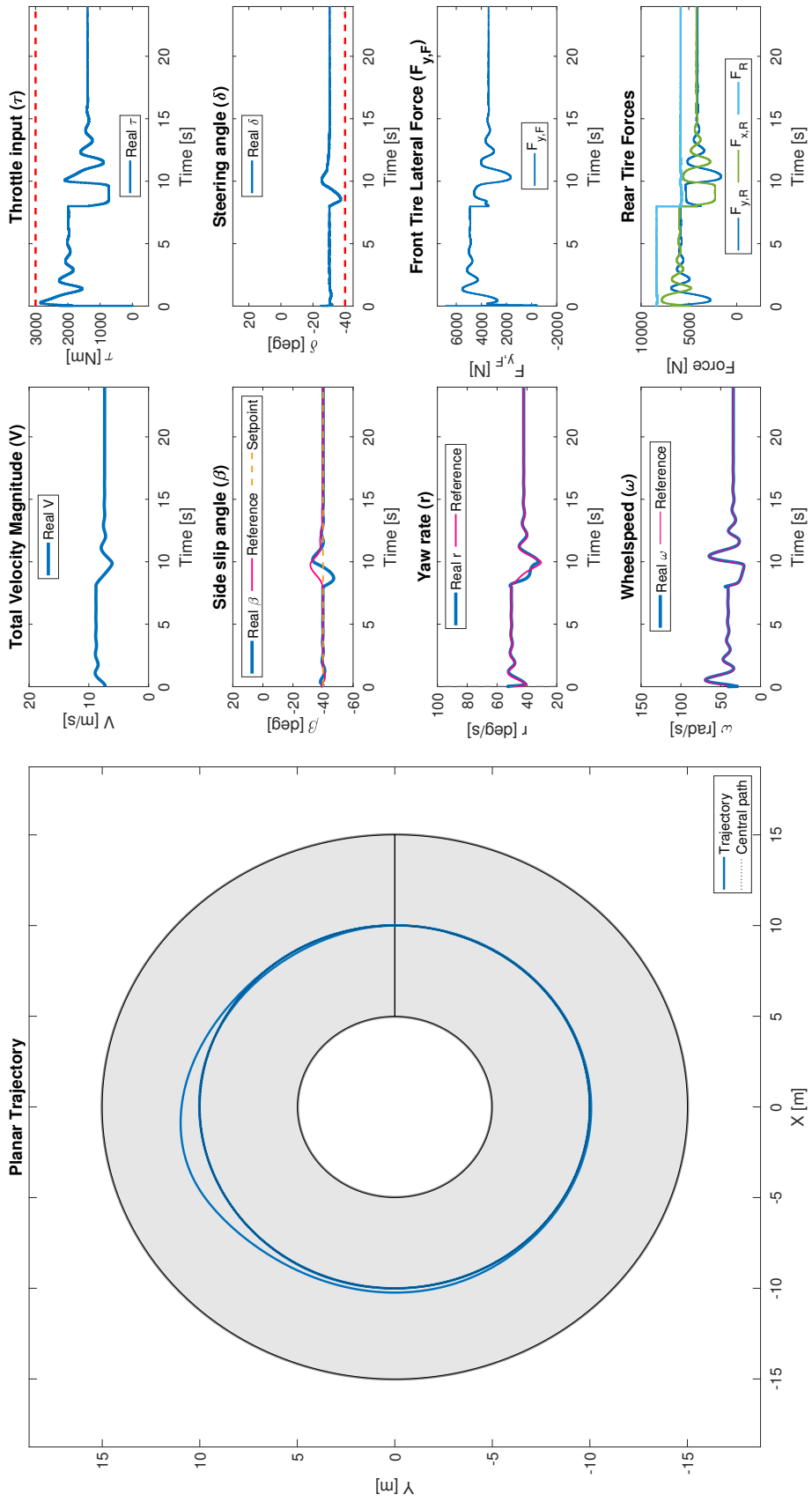


Figure 5-4: Obstacle-free trajectory with friction update.

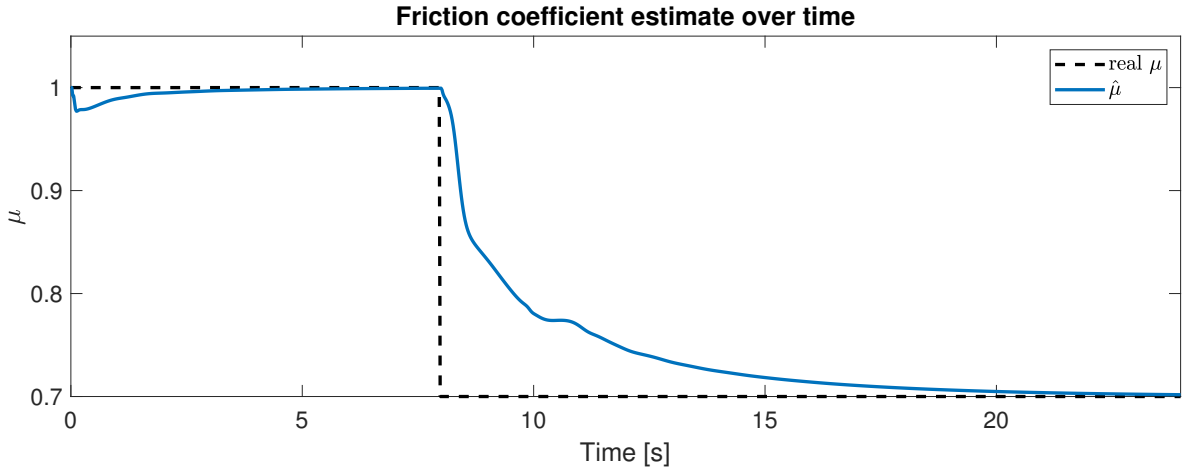


Figure 5-5: Friction coefficient estimate over time with no obstacles.

in reference causes the controller to execute more aggressive maneuvers to stabilize drifting along the desired path, exciting the nonlinear dynamics of the regime.
Since the RLS method relies on linearly approximating the state derivatives, there is a significant approximation error which decreases both the accuracy and speed of convergence to the actual friction coefficient. However, as it can be observed in Figure 5-5, once the vehicle approaches steady-state drifting and lowers the lateral deviation, the linear approximation becomes better and the algorithm converges to the real value.

This is the most significant drawback of this friction update method. Despite its effectiveness, a nonlinear estimation method would greatly improve both the accuracy and the speed of the convergence for the friction estimation problem. As the drifting regime is highly nonlinear and aggressive maneuvers are expected, this improvement will be valuable for real-life scenarios.

Moreover, as noted in Section 3-4, the proposed RLS method estimates only the scaling of the total available friction, rather than the actual friction model. This corresponds to estimating the D coefficient in the Magic Formula, or the peak value. However, it is likely that this friction uncertainty manifests also in the stiffness and shape parameters, or B and C. Having a nonlinear estimation method would allow one to directly adjust the B, C, D parameters in the Magic Formula, empowering the algorithm to adjust the entire friction model and leading to a more accurate model overall.

A short overview of possible solutions to these problems will be presented in Chapter 6.

Computation time

Figure 5-6 shows the solving times for both Planner and Controller problems, along with the computation time of the friction update function (mostly consisting of the computation of the gain and covariance matrices part of RLS). A numerical overview can be found in Table 5-2. The friction update computation time is hardly noticeable since it consists of only matrix multiplications which are considerably faster than NMPC iterations.

First, the computation times presented in Table 5-2 suggests a generous availability to include more complex friction update techniques. Even worst-case scenario computation instances

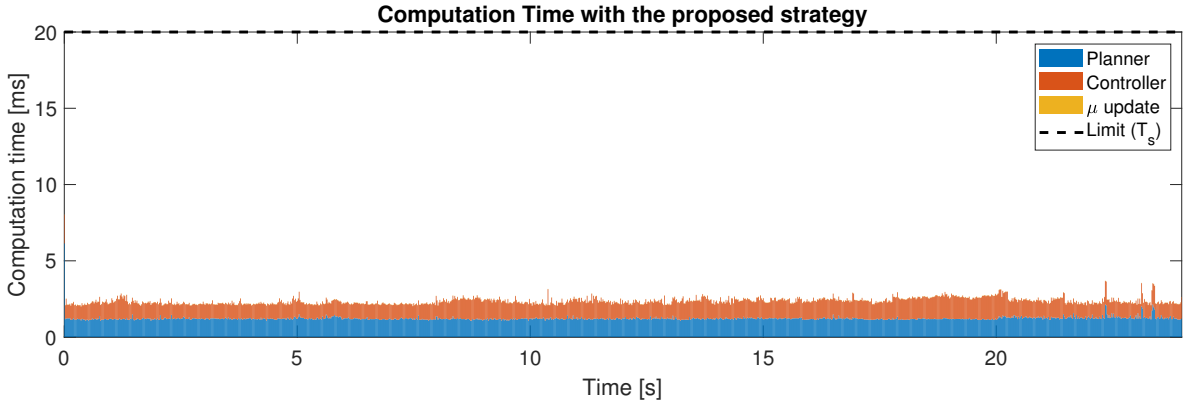


Figure 5-6: Computation time for obstacle-free track with friction update.

	Planner	Controller	Friction Update	Total Time
Maximum [ms]	5.9132	1.998	0.0145	7.9257
90th Percentile [ms]	1.2542	1.4083	0.0145	2.677
Average [ms]	1.2049	1.1144	0.0145	2.3338

Table 5-2: Numerical solving times with friction update.

could allow more computationally demanding update methods such as NMHE, validating this possible topic for future work. Moreover, it is important to note that controller solve times hardly vary for constantly updating friction coefficients. There is a minor increase around the 8th simulation second, however, it quickly diminishes and the solve time returns toward the mean.

Interestingly, the sudden friction change can be traced down in Figure 5-6 due to the gradual increase in Controller solving time. This is because the OCP demands more resources to come up with a solution for the vehicle that is departing the state reference as previous solutions becomes less relevant for a different surface.

5-4 Obstacle avoidance

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The goal is to navigate the track while avoiding two obstacles that are approximated by an ellipse and a circle, respectively.

Vehicle trajectory

The vehicle behavior can be observed in Figure 5-7.

First, the vehicle successfully avoids the obstacles by managing steering and throttle inputs while maintaining high side slip angles. Similar to expert rally drivers, the controller governs drifting by using high throttle inputs (acceleration) and by counter-steering (steering away from the center of the turn).

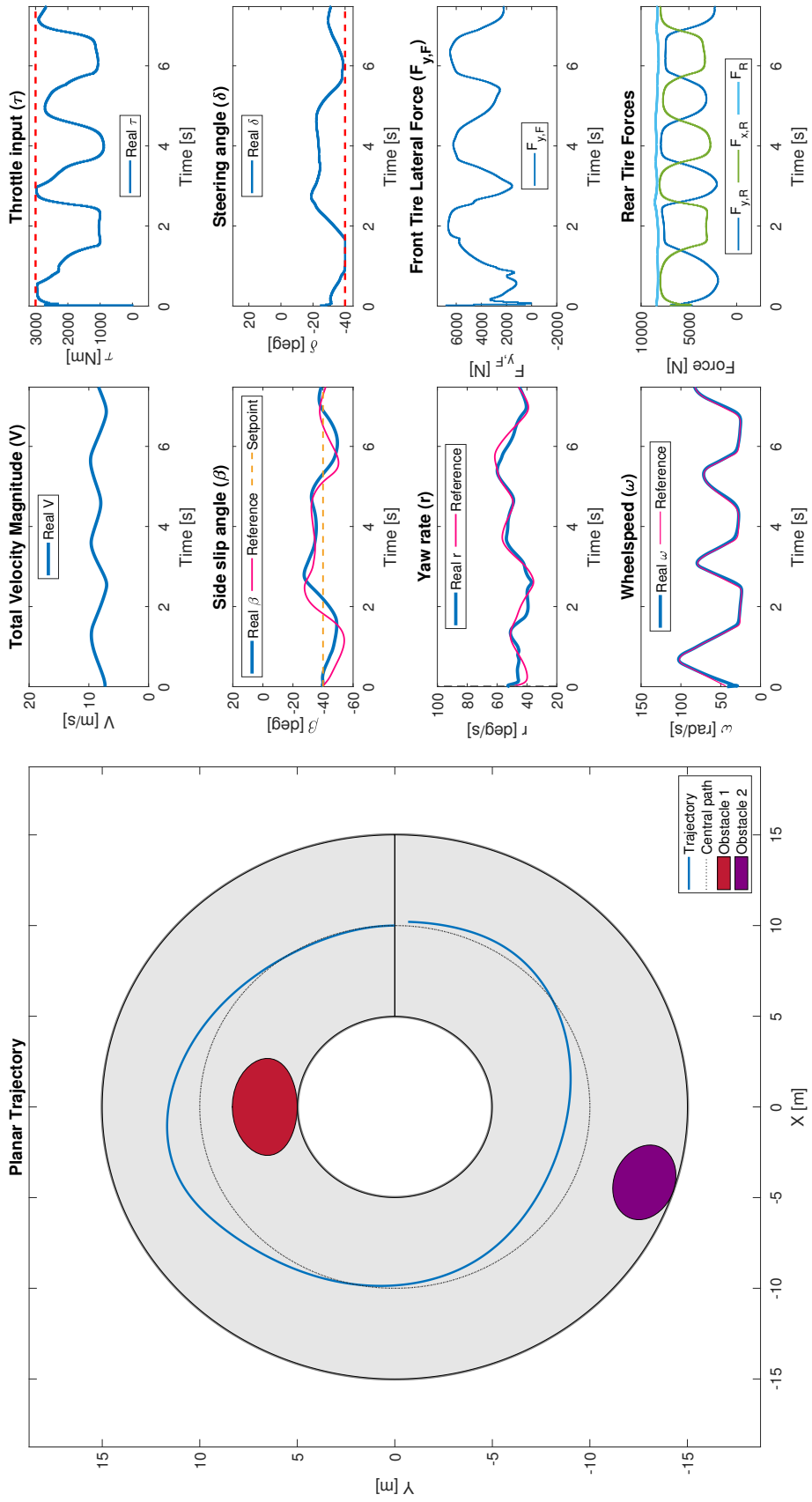


Figure 5-7: Obstacle course trajectory.

Second, notice that the Controller increases total velocity (V) in order to track the desired side slip setpoint. This is in contrast with the typical intent of expert drivers that negotiate the turn in the opposite manner, i.e. the entry velocity is fixed and the side slip magnitude is adjusted to maximize exit velocity.

The samples when planner first detects the second obstacle or when it reaches the first obstacle while approaching the end of the experiment can be noticed in the state and input trajectories.

Computation time

Figure 5-8 depicts the solving time for both Planner and Controller OCPs for each sample. The numerical findings are summarized in Table 5-3, which also contains a column for the speedup factor in comparison with the All-in-one approach highlighted in Section 5-1.

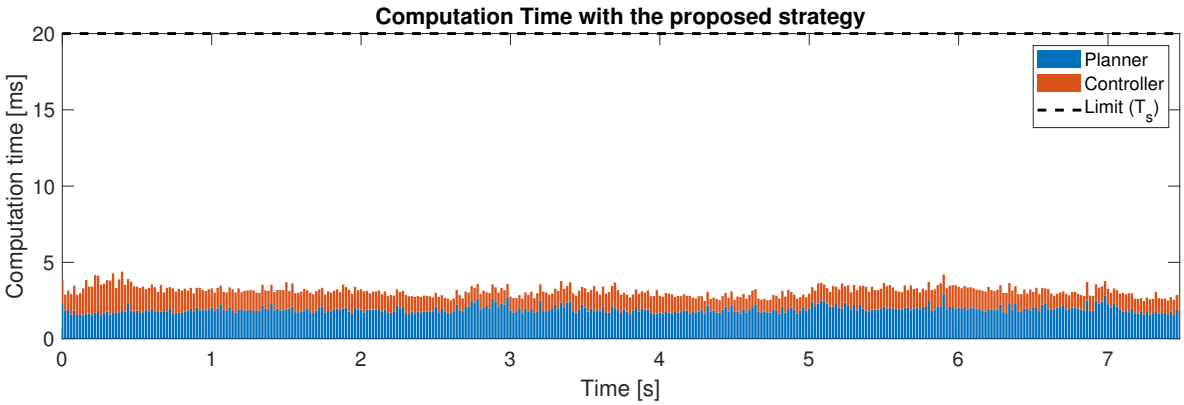


Figure 5-8: Computation time for obstacle course.

	Planner	Controller	Total Time	Speedup
Maximum [ms]	3.1644	2.5173	5.6817	85x
90th Percentile [ms]	2.1787	1.4818	3.6605	26x
Average [ms]	1.903	1.1992	3.1022	15x

Table 5-3: Numerical solving times for obstacle course (Benchmark).

First, it is worth noticing that the solving times are considerably smaller than the computational availability (T_s). This suggests the feasibility of real-time implementation.

Moreover, the effort of recomputing the path to avoid a newly detected obstacle and leaving the previous guess (warm-starting) is reduced as the total solving time is approximately constant throughout the experiment.

Importantly, this is the scenario that can be considered the computational benchmark of the proposed strategy. The obstacle avoidance goal and drifting conditions are identical with the naive, All-in-one formulation approach. Therefore, we can reliably state that the computational benefits are the result of the proposed improvements.

The average solving time has ben significantly decreased in comparison with the All-in-one formulation, showing that the OCP improvements are meaningful. However, arguably the

most valuable contribution lies in the major decrease in maximum solving time. This metric can decide whether passenger safety can be ensured or not in the highlighted scenario. Precisely, the considerable speedup factor of 85 shows that the proposed structure is able to handle well even worst-case scenarios from a computational perspective.

5-5 Obstacle avoidance and friction update

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The goal is to navigate the track while avoiding two obstacles that are approximated by an ellipse and a circle, respectively. After one full pass of the circle, the friction coefficient abruptly changes and the vehicle is supposed to accommodate this and still follow a collision-free drifting trajectory.

This scenario combines the features of the previous two scenarios. The purpose is to see whether the update rule is accurate and fast enough to allow the vehicle to still avoid both obstacles. Moreover, this scenario sparks interest also from a computational perspective since the abrupt change in friction causes the Controller to abandon its previous guesses and converge to a new one that corresponds to the lower friction surface.

Vehicle trajectory

The vehicle trajectory can be observed in Figure 5-9.

First, it can be noticed that the friction coefficient update rule allows the vehicle to safely navigate the track for both before and after the friction change. Interestingly, the planar trajectories after passing Obstacle 1 for both surfaces are highly similar despite the abrupt change in friction. This suggests that the RLS algorithms adjusts the friction model in due time and further allows the controller to exploit the drifting maneuver as intended.

As seen in Section 5-3, since the second surface is characterized by a smaller friction coefficient, the velocity V, the wheel speed ω, and the throttle input τ are on average lower after switching to the second surface.

Friction coefficient estimation

The evolution of friction coefficient estimate $\hat{\mu}$ over time can be seen in Figure 5-10. The observations made in Section 5-3 are more noticeable in this simulation. Despite the update algorithm allowing the vehicle to adapt the drifting maneuver to the new friction coefficient, the estimate is noticeably disturbed by the errors caused by linearly approximating derivatives. These errors contribute to changes in friction estimate despite interacting with the same surface. However, the amplitude of and the recovery after these disturbances suggest these errors are comparable to what measurement noise would cause in a realistic scenario.

These estimation errors can be traced back to how the scenario was constructed: in comparison with the obstacle-free friction update experiment, the planner generates more aggressive

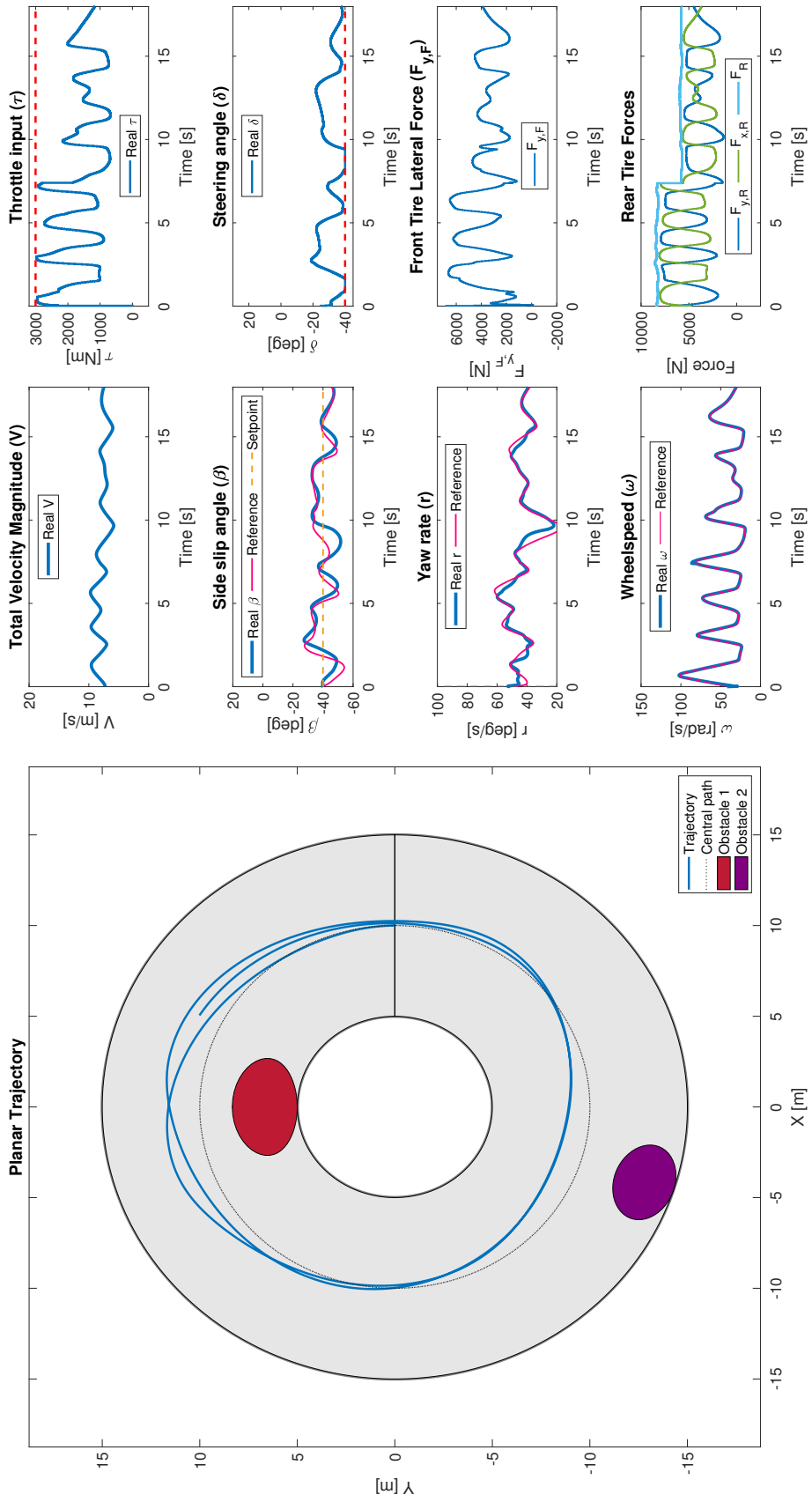


Figure 5-9: Obstacle course trajectory with friction update.

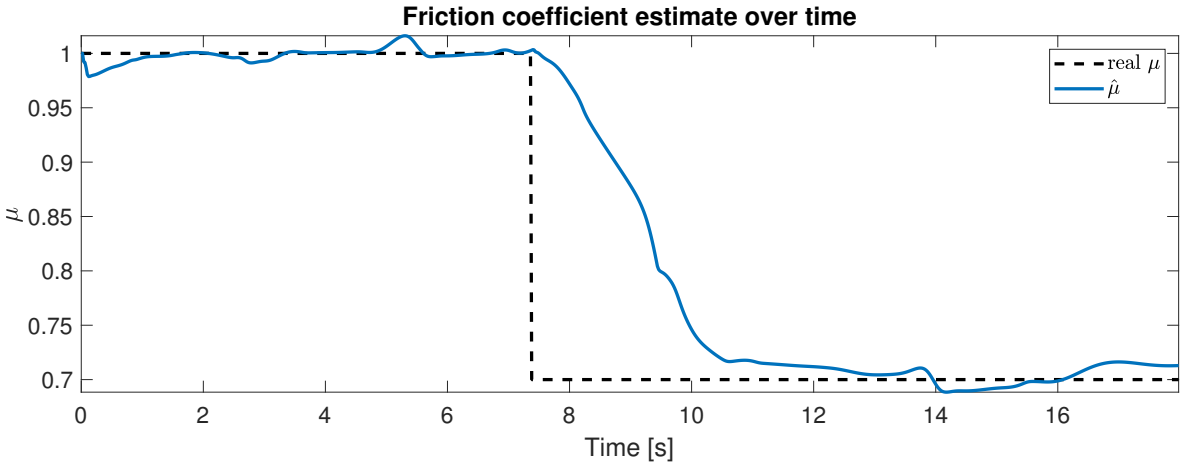


Figure 5-10: Friction coefficient update with two obstacles.

references to avoid collisions. This causes more aggressive changes in vehicle side slip and yaw rate, both in references (Navigation layer) and states (Drift Stabilization layer), which are less accurately approximated by the backward difference (3-7).

Overall, the simple, linear RLS algorithm is an effective method to preserve the control authority over drifting maneuvers in the presented scenario. This is even more suggestive if the minimal computational overhead is considered. However, this scenario further adds to the suggestion of incorporating a nonlinear friction update method that can improve the transient response and distinguish nonlinearities from friction changes. This is similar to the observations made in Section 5-8.

Computation time

From a computational perspective, this is the most difficult testing case. The experiment requires adapting to a new friction surface (which affects Controller solving time) and to upcoming obstacles (increasing the Planner solving time). These events make the second pass of the circle significantly more challenging than the first. This can be observed in Figure 5-11 and Table 5-4.

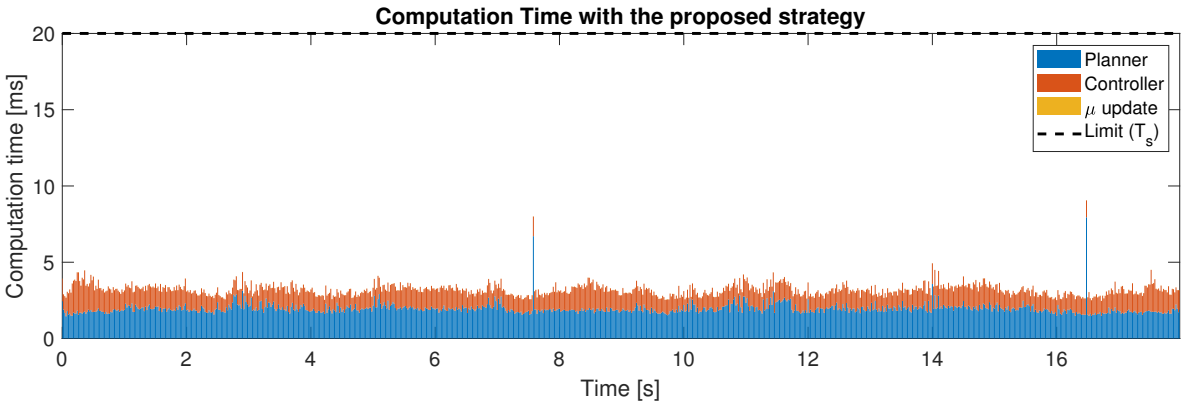


Figure 5-11: Computation time for obstacle course with friction update.

	Planner	Controller	Friction Update	Total Time
Maximum [ms]	7.6957	2.6292	0.009	10.3339
90th Percentile [ms]	2.1881	1.4515	0.009	3.6485
Average [ms]	1.8903	1.1824	0.009	3.0817

Table 5-4: Numerical solving times for obstacle course with friction update

The artificial sudden increases in Planner solving time are caused by how the maps are generated and embedded in the simulation. This will be explained further. Precisely, as this work takes into account only circular maps, the simulation relies on projecting the cartesian position of the car to a circle in the curvilinear space. As a result, there will be a minor mismatch between projections when crossing the zero-progress line. This in turn lowers the effectiveness of warm-starting for the first sample after crossing.

In contrast with one-pass obstacle courses and multi-pass obstacle-free tracks, where either obstacles or start lines are encountered, this scenario contains both events concurrently. Therefore, in comparison with the obstacle-free, friction update scenario from Section 5-3, this time there is an additional event: the first, elliptical obstacle needs to be avoided during the second pass. As a result, the departure from the previous guess is even more sizable, leading to a noticeable increase in planner solving time.

Regardless, these artifacts do not extend to real-life scenarios when the projection of the vehicle position relative to the road in the curvilinear space will be less artificial.

Lastly, despite the departure of both Planner and Controller problems from previous solutions, the proposed strategy does not violate its time availability. During the second pass, it can be noticed that although the total solve time is increased, it is well withing computational availability (T_s). This confirms the suitability for a real-time implementation.

The subsequent scenarios are dedicated to more elaborate, aggregate simulations that fundamentally study the sensitivity of the proposed control structure to initial condition variation and model mismatch.

5-6 Obstacle avoidance with varying initial lateral deviation

Scenario: The vehicle starting position is varied by changing its initial lateral deviation w.r.t to the central path according to a predefined range. Regardless, the initial vehicle states are kept in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The goal is to navigate the track while avoiding two obstacles that are approximated by an ellipse and a circle, respectively.

The current scenario evaluated the success rate of trajectories that start with different initial positions. Precisely, the initial lateral deviation n_0 is varied from -1.8 meters to 1.8 meters, with a step of 0.1 meters, with a total road width of 10 meters. It is expected that trials starting with sizable, negative n_0 will be prone to colliding with the first obstacle, while big, positive n_0 will likely cause hitting the outer boundary.

Vehicle Trajectory

The scenario outcome can be visualized in Figure 5-12.

The trials that started with high negative lateral deviation ($n_0 \in [-1.8, -0.2]$ meters) were not able to complete the track without hitting the obstacles. This was expected since it is difficult to control drifting maneuvers to accommodate such sharp turns, which are needed to avoid both Obstacle 1 and the outer track limit.
For these failed trials, the control structure tries to lower the total velocity V and to extend the turn radius by further decreasing the side slip angle β. However, the available space is too restrictive to allow for a collision-free trajectory for the starting vehicle states.

The trajectories starting close to central path with $n_0 \in [-0.1, 1.3]$ meters successfully complete the track. This positive bias was anticipated since small positive deviations favor avoiding Obstacle 1 while not extending too close to the track limit. The successful planar trajectories converge before the vehicle completes the full circle, the control structure being able to quickly manage drifting maneuvers and accommodate the difference in initial lateral deviation.

Similar to the inner failed trajectories, high positive initial lateral deviation ($n_0 \in [1.4, 1.8]$) lead to collisions with the outer track limit. The generated state references (β^{ref}, r^{ref}) correspond to a high total velocity V, making it difficult to recover after Obstacle 1 and leading to a collision with the exterior track limit.

There is a significant success bias toward positive initial lateral deviations. This is naturally motivated by the chosen map structure, which favors circular trajectories that start outwards: these trajectories tend to have a positive lateral deviation in the first half (avoiding Obstacle 1), and a negative lateral deviation in the second half (avoiding Obstacle 2).
However, this bias is valid only for the arbitrarily designed obstacle course and does not extend to generic road segments.

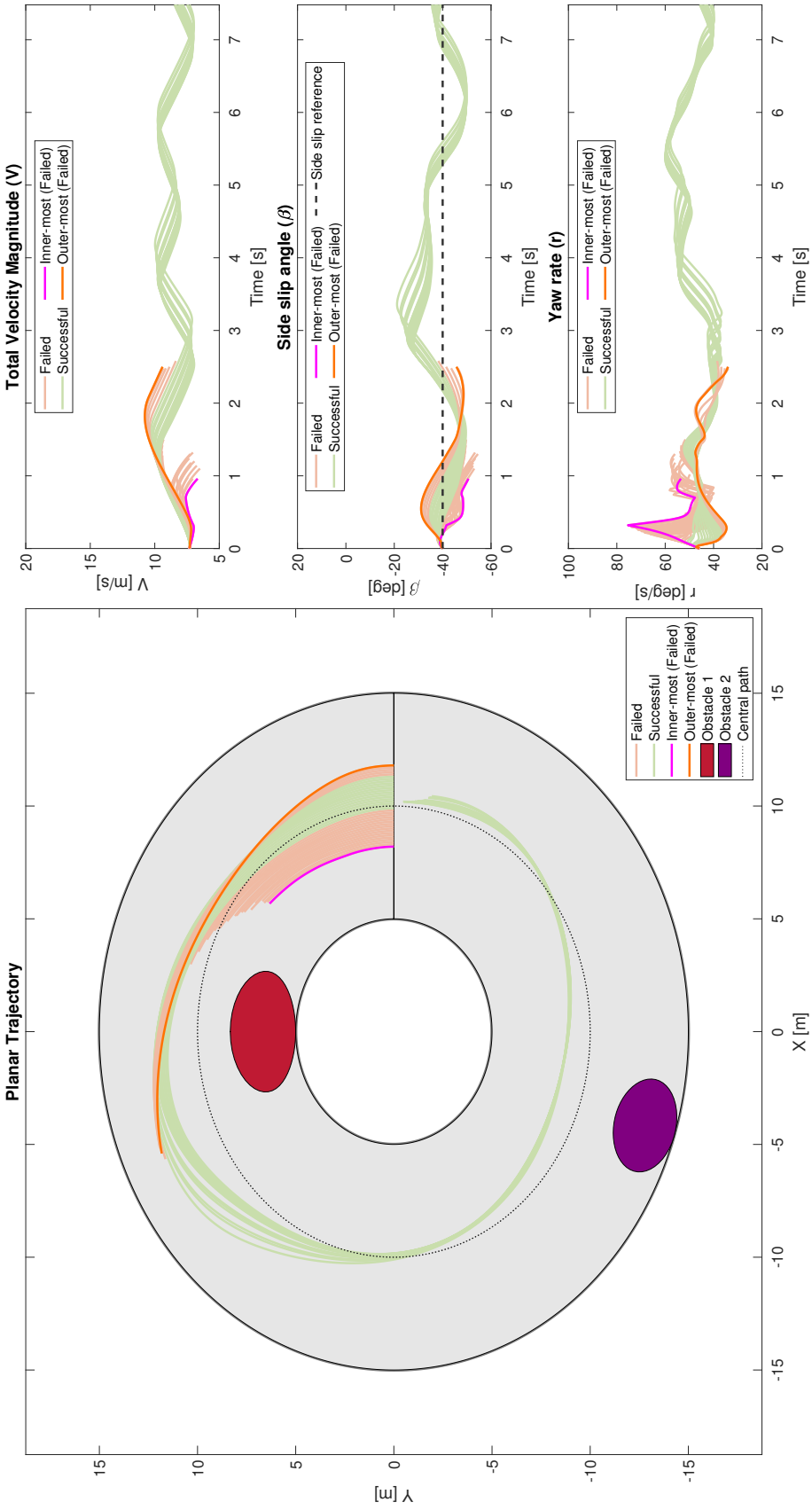


Figure 5-12: Obstacle course trajectories for varying n_0.

Computation time

Figure 5-13 shows the aggregated computation times for each sample, each bar representing the average solving time over all the successful trajectories. The yellow contour represents the maximum total solving time for each sample across all successful trials. The numerical results can be found in Table 5-5.

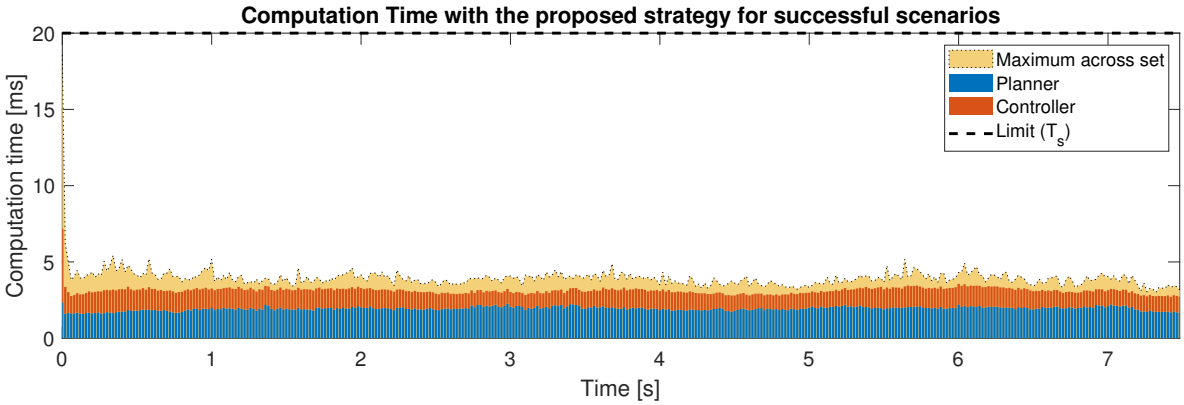


Figure 5-13: Computation time for successful trajectories with varying n_0.

	Planner	Controller	Total Time
Maximum [ms]	4.3507	15.6164	19.9671
90th Percentile [ms]	2.2398	1.4102	3.65
Average [ms]	1.9183	1.1785	3.0968

Table 5-5: Aggregated numerical solving times for successful trajectories with varying n_0.

First, notice that the maximum total solving time almost violates the available computation time. On closer inspection, it is revealed that the Controller solving time reaches more than 15 milliseconds for the outer-most successful trajectory, corresponding to $n_0 = 1.3$. This can be motivated by the provided initial guess for the first sample, in the absence of warm-starting: for the other trajectories, a simple constant initial guess based on the starting vehicle state seems to suffice. However, the outer-most trajectory needs more aggressive maneuvers that depart from the initial guess along the prediction horizon in order to avoid collision with track limits and first obstacle. This in turn leads to a sizable computation time for the first sample, which is then reduced significantly thanks to warm-starting.
A possible solution can be generating an initial guess based on the system dynamics in the absence of any inputs, starting with the initial vehicle state. However, this is worth investigating this in a more realistic scenario.

Interestingly, having excluded the first sample, the solving times hardly vary across successful trajectories. This observation is valid despite that their corresponding state evolution shows noticeable differences, as seen in Figure 5-13. If the initial guess problem is isolated, this demonstrates the computational robustness of the proposed method.

	Planner	Controller	Total Time
Maximum [ms]	6.7504	6.0136	12.764
90th Percentile [ms]	1.2351	1.3668	2.6019
Average [ms]	1.2006	1.105	2.3056

Table 5-6: Aggregated numerical solving times for U-turn trajectories with varying n_0.

5-7 Drift cornering with varying initial lateral deviation

Scenario: The vehicle starting position is varied by changing its initial lateral deviation w.r.t to the central path according to a predefined range. Regardless, the initial vehicle states are kept in the vicinity of a drifting equilibrium. There is a minor model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The goal is to complete the U-turn with minimal lateral deviation from the central path.

In a similar fashion with the scenario outlined in the previous section, the vehicle will start with varying initial lateral deviation n_0. Precisely, n_0 will be varied from -1.8 meters to 1.8 meters, with a step of 0.1 meters. Concisely, the goal in this scenario is to compare the entry and exit spread in lateral deviation as this will reveal the ability to control drift cornering.

Vehicle Trajectory

Figure 5-14 reveals the convergence of such trajectories to the central path at the end of turn.

Notice that the planar trajectories have been trimmed beyond a fixed point at the end of the turn. This explains why trajectories with varying velocities cover the same distance in the same amount of time in the figure. This was done for visualization purposes, to properly align the lateral deviations at turn exit.

The histogram shown in the bottom left corner compares the distribution of the entry and exit lateral deviations. The initial uniformly-distributed samples conveniently converge to small lateral deviations. This further suggests that the control structure is successful in managing drifting maneuvers to steer the vehicle close to the central path.
Moreover, this reduction in lateral deviation is achieved while the side slip angle converges in all the cases to the setpoint, departing by at most 15° in the process.

Computation time

The aggregated solving times are depicted in Figure 5-15 and presented numerically in Table 5-6. Similar to the behavior observed in Figure 5-13, the computation times hardly vary across samples regardless of the difference in state trajectories. However, two differences need to be noted in comparison with the previous simulations.

First, the average solving times tend to be steadier due to the absence of events, as there are no obstacles.

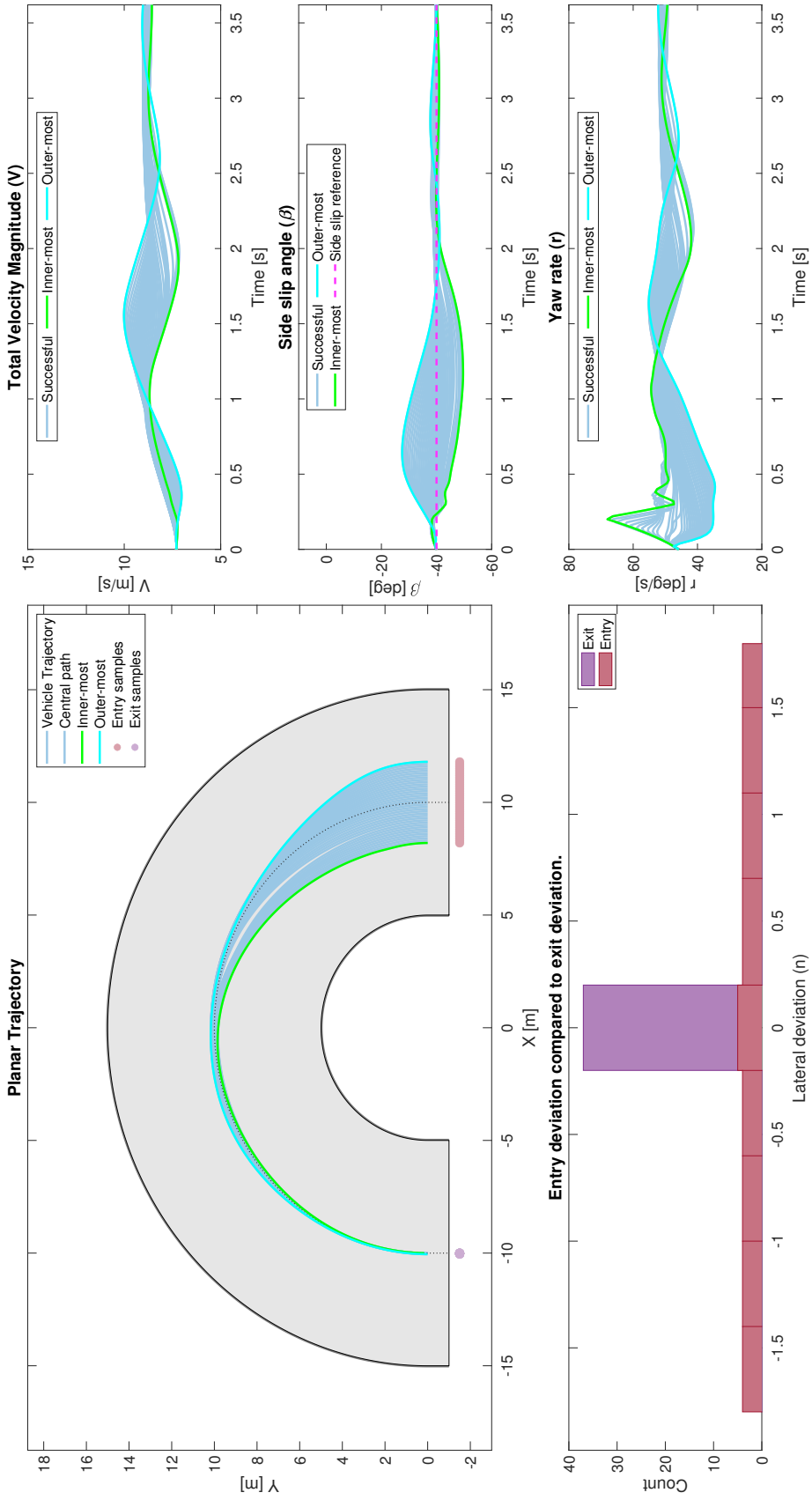


Figure 5-14: U-turn trajectories for varying n_0.

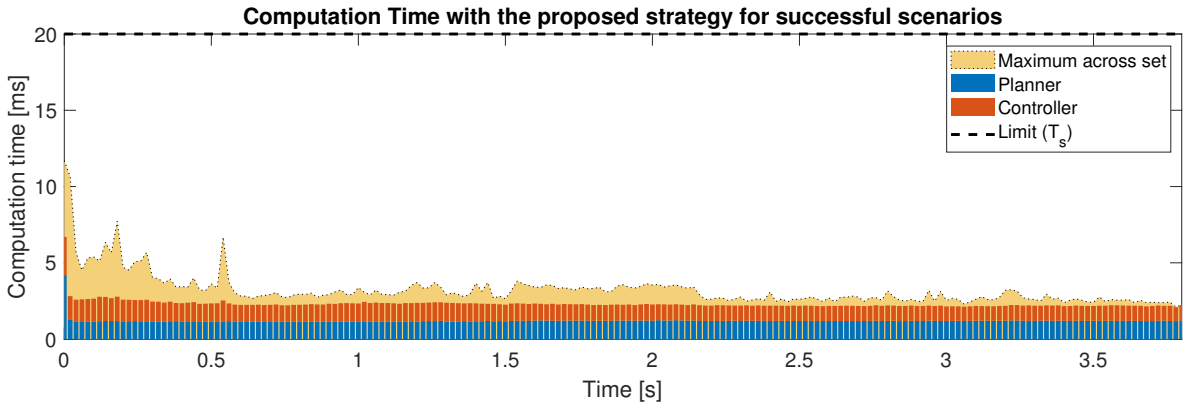


Figure 5-15: Computation time for U-turn trajectories with varying n_0.

Second, the maximum solving time for the first samples is much lower since also the most extreme lateral deviations are not restrictive in the absence of Obstacle 1. Since the initial guess assumes the vehicle is on the center path, for the most extreme cases it generates a longer solving time. However, in this scenario the initial guess is not as unfavorable for the outer trajectories as it is in Section 5-6. This is the central reason behind not experiencing solving times close to the computational availability.

5-8 Sensitivity to friction model

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. On one hand, there is model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. On the other hand and more significant, there is model mismatch caused by the difference in friction model (the actual vehicle has different B, C, D parameters in the Magic Formula). The goal is to measure the effects of this uncertainty on the vehicle behavior and friction estimation accuracy.

This scenario overcomes the simplicity of the testing methodology presented in Section 5-3. There is no longer an abrupt change in friction, and the mismatch no longer consists of only an amplitude difference. Precisely, the mismatch cannot be completely traced back to the D term in the Magic Formula. Therefore, the shapes of the tire force curves are also different than what is assumed in the controller model.

It is important to mention that the Friction Update algorithm is still active. The proposed RLS algorithm will try to converge to a peak value D that matches the observed behavior in the lateral states.

There are six different friction models considered, ranging from a high friction surfaces (such as a well polished track) to more slippery surfaces (similar to wet concrete). The tire force curves as functions of tire slip can be examined in Figure 5-17.

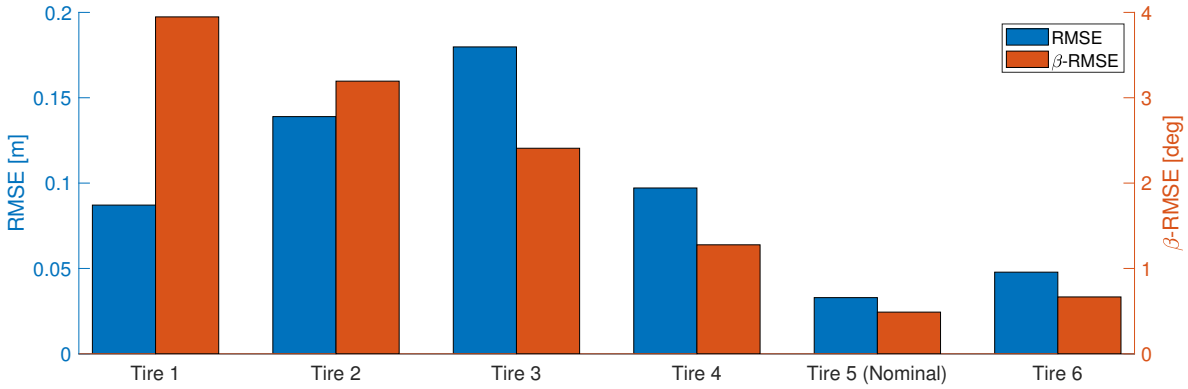


Figure 5-16: RMSE and $\beta-$RMSE for each tire model.

	B	C	D
Tire 1	9.24	0.85	0.7
Tire 2	9.74	1	0.775
Tire 3	10.24	1.15	0.85
Tire 4	10.74	1.3	0.925
Tire 5 **(nominal)**	11.24	1.45	1
Tire 6	11.74	1.55	1.075

Table 5-7: Magic Formula parameters for each tire model.

Vehicle Trajectory

The vehicle trajectories can be observed in Figure 5-18.

Despite that differences in planar trajectories are not sizable, the state trajectories are noticeably different. This can be motivated by the observation that fundamentally the Controller model is affected by this mismatch rather than the Planner model. The interplay between the two layers create trajectories that favor planar tracking instead of side slip tracking. In the absence of a reliable friction estimate, the dynamics are harder to control toward a desired side slip magnitude.

First, in order to assess the impact of this uncertainty on the tracking performance of both central path and side slip setpoint, the Root-Mean-Square Error (RMSE) has been computed for both tasks. Precisely, RMSE denotes the planar tracking error w.r.t to central path, while $\beta-$RMSE represents the side slip tracking error w.r.t to side slips setpoint β^{sp}. These errors are visually represented in Figure 5-16.

The $\beta-$RMSE distribution follows the expected behavior: the furthest away the tire force curve is from the nominal curve (green), the bigger the tracking error is in an almost linear fashion. This is also noticeable in the state trajectories presented in Figure 5-18. Interestingly, the RMSE follows this trend less strictly, achieving better tracking for the most slippery surface despite its poor side slip tracking performance. However, this can be motivated by the duration of the experiment since more passes would allow likely the side slip error to affect the tracking error and make the trend similar to what is observed for β-RMSE.

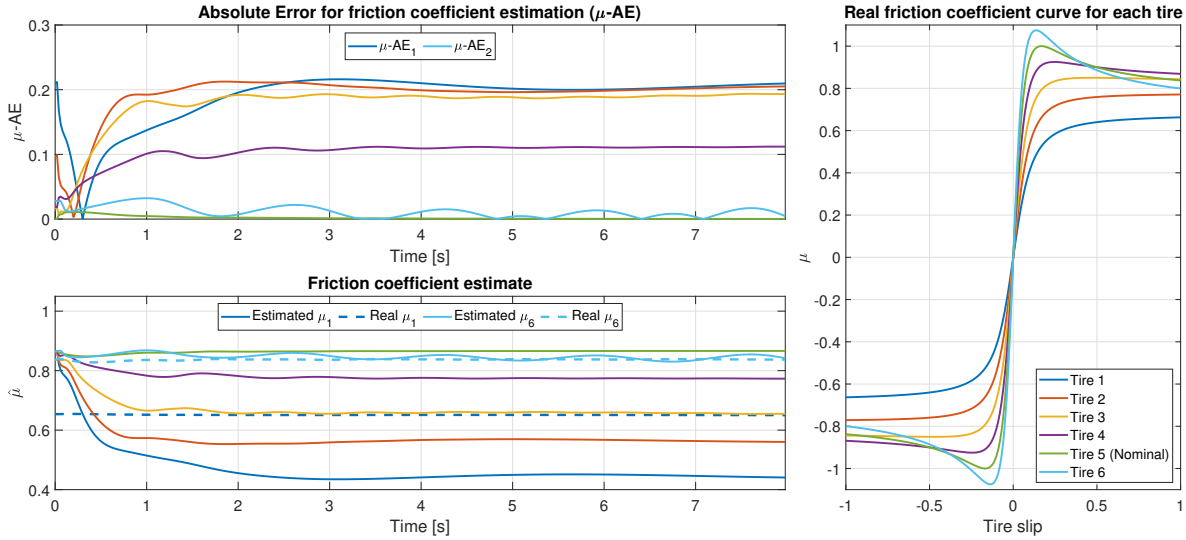


Figure 5-17: Friction coefficient estimate for each tire model.

Second, this experiment reveals the shortcomings of the friction update algorithm. This can be well understood from Figure 5-17.

The top left corner in Figure 5-17 shows the Absolute Error of the friction coefficient estimation. Notice that for the biggest level of model mismatch (in this case, the wet concrete surface depicted in blue), the error becomes as high as $|\hat{\mu} - \mu| \approx 0.2$. This is the reason for the poor side slip tracking that can be observed in Figure 5-16.

The bottom left plot displays the friction coefficient estimates $\hat{\mu}$ for each tire model. For clarity purposes, only the real friction coefficients that correspond to the most extreme tire curves were shown, and their corresponding estimates. Experimentally, the real friction coefficient is computed using the actual tire curves presented in the right half of the figure as functions of the total tire slip for the rear tire s_R.

Summarizing the observed differences, it is clear that the proposed RLS friction update algorithm does not handle well this type of uncertainty. While it still allows the Controller to stabilize drifting maneuvers, the tracking performance is noticeably affected. This is reflected most significantly in the ability to track the desired side slip angle β^{sp}.
For more aggressive maneuvers expected in obstacle-avoidance scenarios, this is a significant limitation. Furthermore, coupled with the sensitivity of the RLS algorithm to abrupt state changes, this can severely limit the ability to drift away from oncoming obstacles. For these reasons, the current work suggests investigating nonlinear friction estimation methods that would directly address both problems. As this study has shown, the remaining computational resources permit the usage of more complex update procedures. Future research avenues will be proposed in Chapter 6.

Lastly, from a **computational perspective** this simulation does not bring a major contribution toward the purpose of this work. The results are similar to what was observed in Figure 5-6. For compactness reasons, neither the numerical results nor the figures are shown in this document.

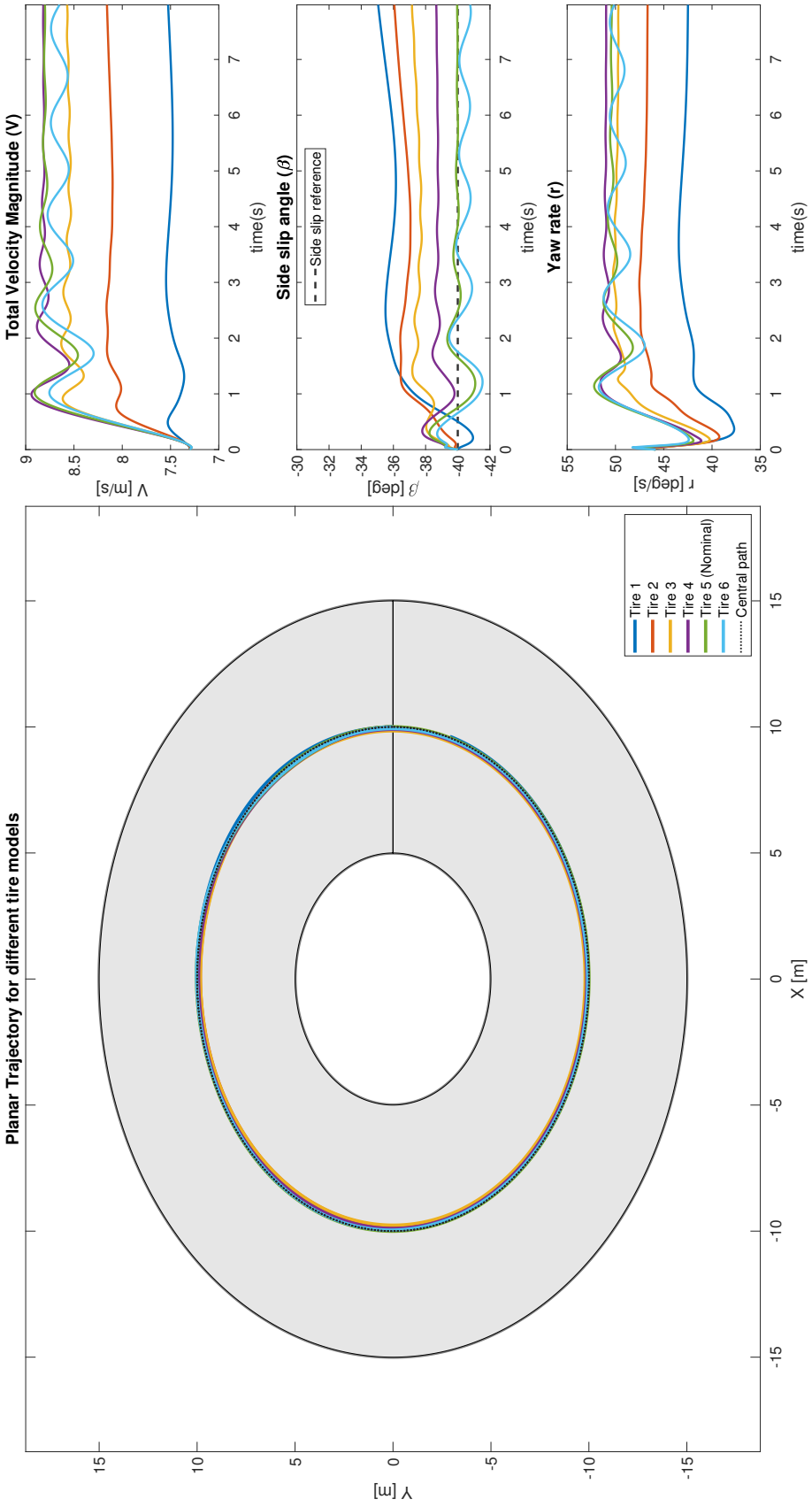


Figure 5-18: Vehicle trajectories for different tire models.

5-9 Sensitivity to total mass and weight distribution

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. First, there is model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. Second, there is additional model mismatch caused by an artificial change in total mass and weight distribution. The goal is to measure the effects of this uncertainty on the vehicle tracking performance and computation times.

This experiment will be targeted at understanding the influence of total mass and weight distribution using the following methodology:

1. A mass difference M is added to the nominal vehicle mass: m_0 ($m = m_0 + M$). M will be varied from -400 kg to 400kg with a step of 50 kg. The vehicle moment of inertia around the vertical axis is also adjusted accordingly.
2. A weight distribution offset is introduced using a fraction γ of the total weight mg. This fraction is varied from -15% to 15% with a step of 1%, where a positive γ corresponds to an increase in front axle load, and negative values to an increase in rear axle load.

Vehicle Trajectory

The successful planar and state trajectories can be seen in Figure 5-19. Expectedly, there is significant variation in vehicle states due to the inaccurate estimation of tire forces caused by mass and weight distribution mismatch. Regardless, the control structure is successful in controlling drifting maneuvers for a sizable level of mismatch and is able to complete the circular track with high side slip angle. The vast majority of successful attempts finish the simulation with β

The vehicle trajectory dependence on M and γ can be better understood from the following two figures: Figure 5-20 depicts the Root-Mean-Square Error (RMSE) of planar trajectory w.r.t central path, and Figure 5-21 shows the RMSE of tracking the side slip setpoint.

In Figure 5-20 Figure 5-21, the plateaus displayed in yellow represent planner or controller infeasibility. This usually represented an attempt in which the vehicle violated the track limits or a numerical problem occurred in the controller optimization problem.
The island of infeasibility is likely the effect of numerical problems as well. From a vehicle dynamics perspective, there is hardly any justification that can motivate its existence.

The RMSE tends to increase in the attempts in which there is lesser vertical load on the front axle. As the maximum lateral force the front tire can exert decreases, the vehicle loses its tracking performance for both objectives (central path and side slip setpoint).
Moreover, this observation is strengthened by the valley that corresponds to increasing γ and decreasing M, being more visible in Figure 5-20. Along this valley, the front axle maintains its load that allows the vehicle to better track the references.

In contrast with this tendency, the direction of increasing γ and increasing M (orthogonal to the previous valley) leads to higher overall cost. This is visibly more pronounced toward negative γ and M, as the cost rapidly grows and leads to infeasibility.

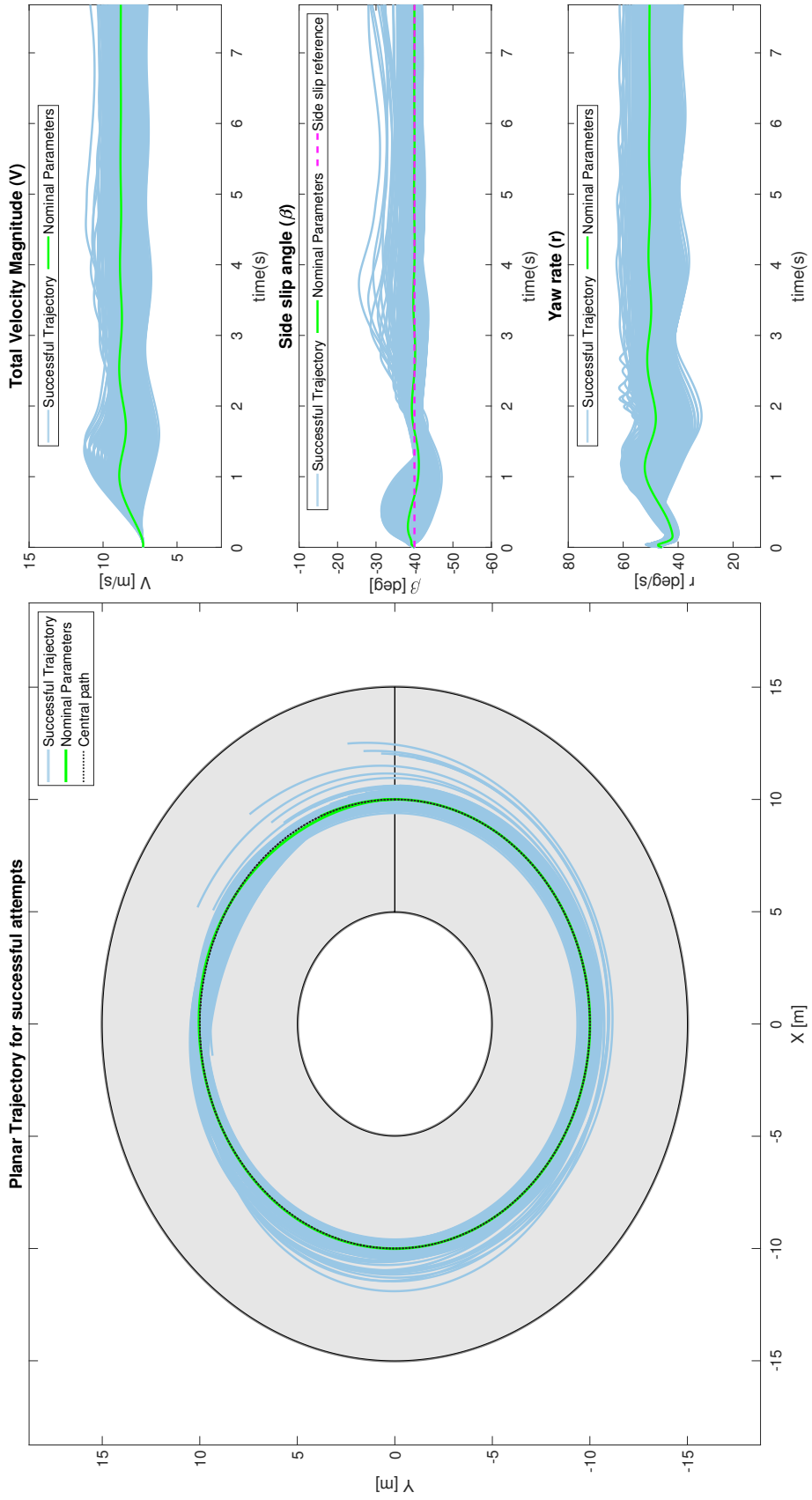


Figure 5-19: Vehicle trajectories with mass and weight distribution mismatch.

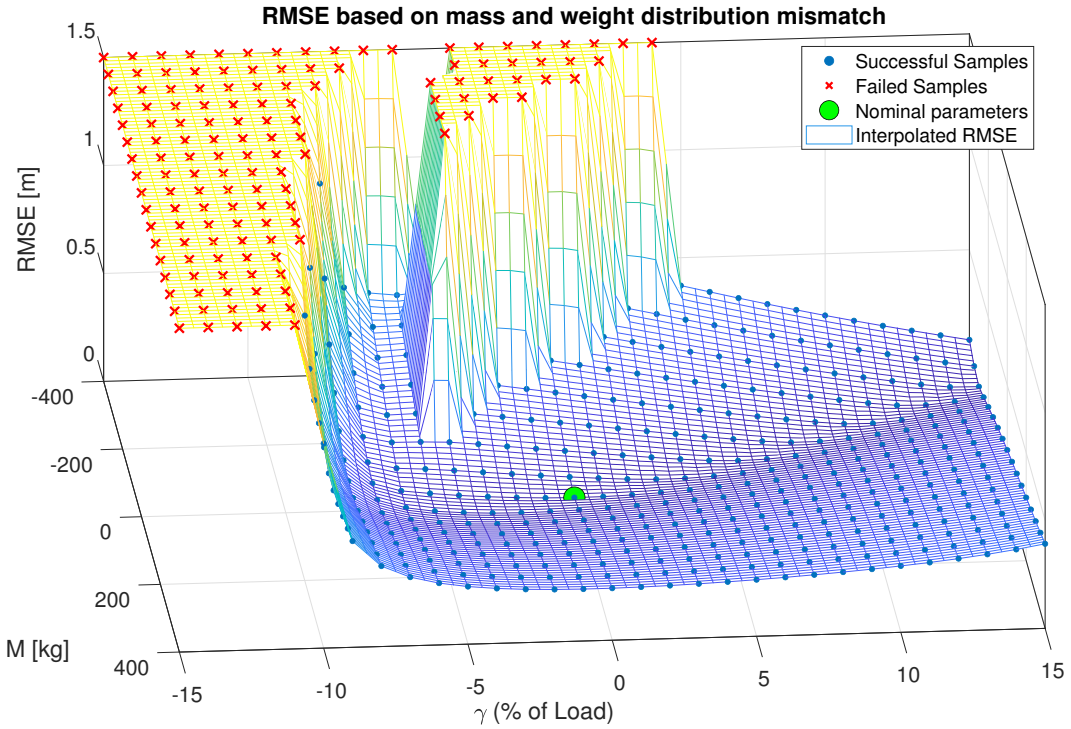


Figure 5-20: RMSE of planar trajectory central path.

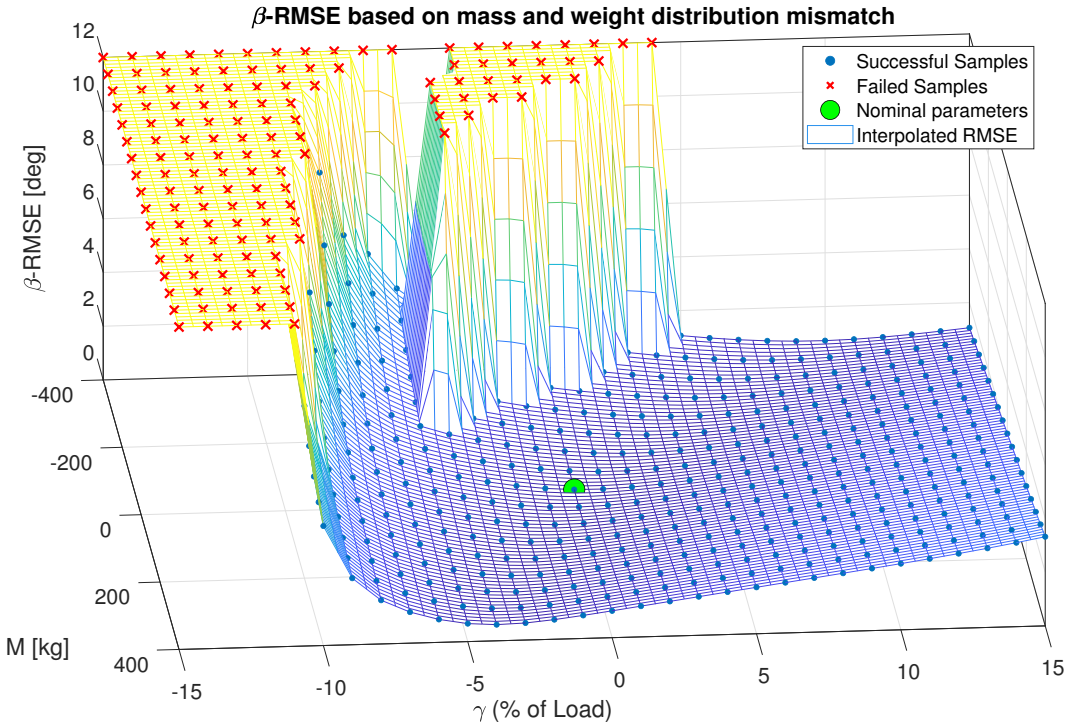


Figure 5-21: RMSE between the actual side slip angle and setpoint (β-RMSE).

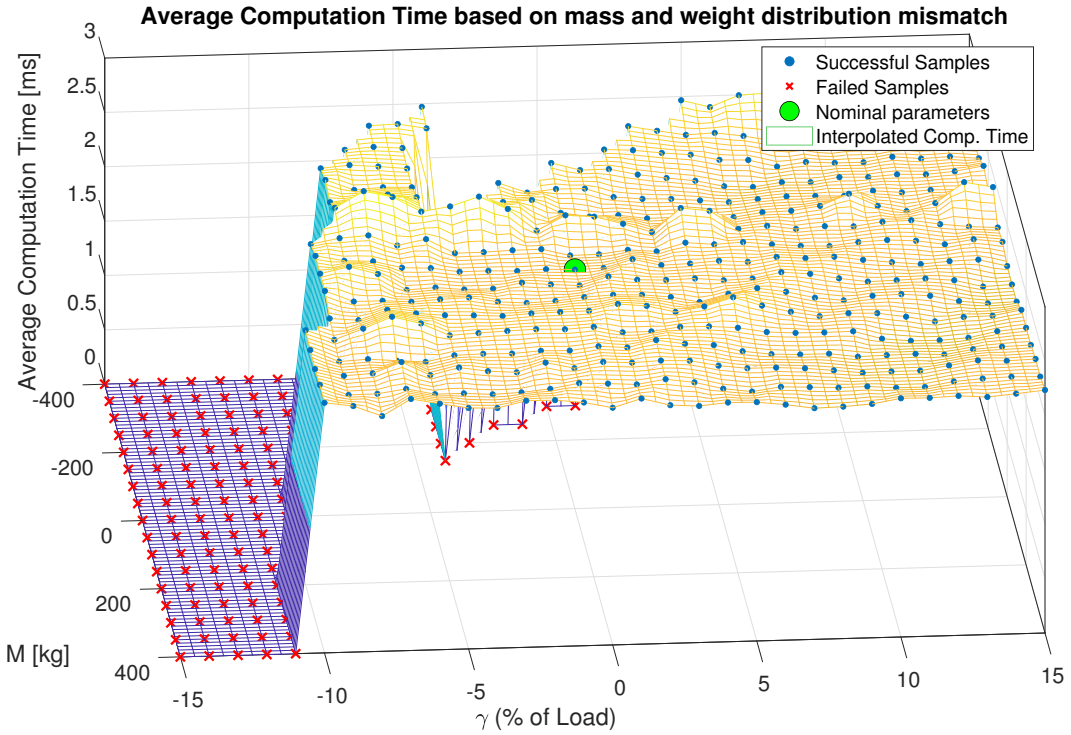


Figure 5-22: Computation time based on mass and weight distribution mismatch

Overall, there is a high level of similarity between the two figures. The reason is the relative high weight the overall control structure has on tracking the side slip angle setpoint. This leads to a structure that prioritizes the stabilization of drifting instead of central path tracking. In certain cases, the setpoint tracking performance is low due to model mismatch, leading to poor planar tracking as a result.

This scenario suggests that the proposed control structure handles well mass and weight distribution mismatch, especially when they favor an increase in vertical load on the front axle. However, the current work proposes validating the introduction of load transfer dynamics in the controller model. As the overall computation opinion stands, there is enough availability to include additional dynamical layers. The advantage will be an extended area of control authority over drifting maneuvers in the presence of this class of uncertainties.

Computation time

The average computation time per experiment for each (M, γ) pair can observed in Figure 5-22. Similar to previous figures, the plateau now set at 0 milliseconds represents planner or controller infeasibility.

The central takeaway here is that the solving time hardly varies for successful attempts. Regarding the higher costs (either RMSE or $\beta-$RMSE), it is observed that the increase in cost is not correlated with a computational drawback (as successive reoptimizations without good initial guess would be visible in the plot). In simple words, the typical scenario shows the Controller rapidly converging to a solution that will satisfy both tracking goals from the Controller perspective. However, due to model mismatch, the generated control action fails to lower the tracking error and the cycle repeats. In the end, this interaction causes an approximately insensitive solving time to the size of the uncertainty, but the tracking cost will be negatively influenced by the increasing level of uncertainty.

5-10 Sensitivity to side slip estimation error

Scenario: The vehicle starts on the center path in the vicinity of a drifting equilibrium. There is model mismatch created by the decoupled wheel dynamics and by the introduction of aerodynamic drag. The side slip angle β is no longer assumed to be accurately measured, rather it is disturbed by Gaussian noise with predefined mean and variance. The goal is to measure the effects of side slip estimation error on the vehicle tracking performance and computation times.

The central motivation behind this simulation is the relative difficulty in estimating the side slip angle β in comparison with other vehicle states such as V. Moreover, as the autonomous drifting framework presented in this document aims to control the side slip angle to a desired value, disturbing one of the weak points of this structure would properly test its resilience.

The experiments will assume the side slip angle measurement is disturbed by a Gaussian noise v of mean b and variance σ^2:

$$\beta = \beta_{real} + v, \quad v \sim \mathcal{N}(\mu, \sigma^2), \tag{5-1}$$

where β_{real} is the perfect reconstruction of the real side slip angle.

The noise parameters will be varied according to the following method:

1. the mean b (or bias in this case) will range from $-0.8°$ to $1.37°$, with a step of $0.11°$.
2. the standard deviation σ is assumed to vary from $0°$ to $1.72°$, with a step of $0.14°$.

For each noise realization (or each pair (b, σ)), five trials were attempted ($Q = 5$). If any of the five runs is unsuccessful, the whole realization is considered to lead to failure.

Vehicle trajectory

The successful vehicle trajectories for each realization and attempt are shown in Figure 5-23. For this experiment, it is insightful to simultaneously examine the state trajectories presented in Figure 5-23, and the tracking error presented in Figure 5-24 or Figure 5-25.

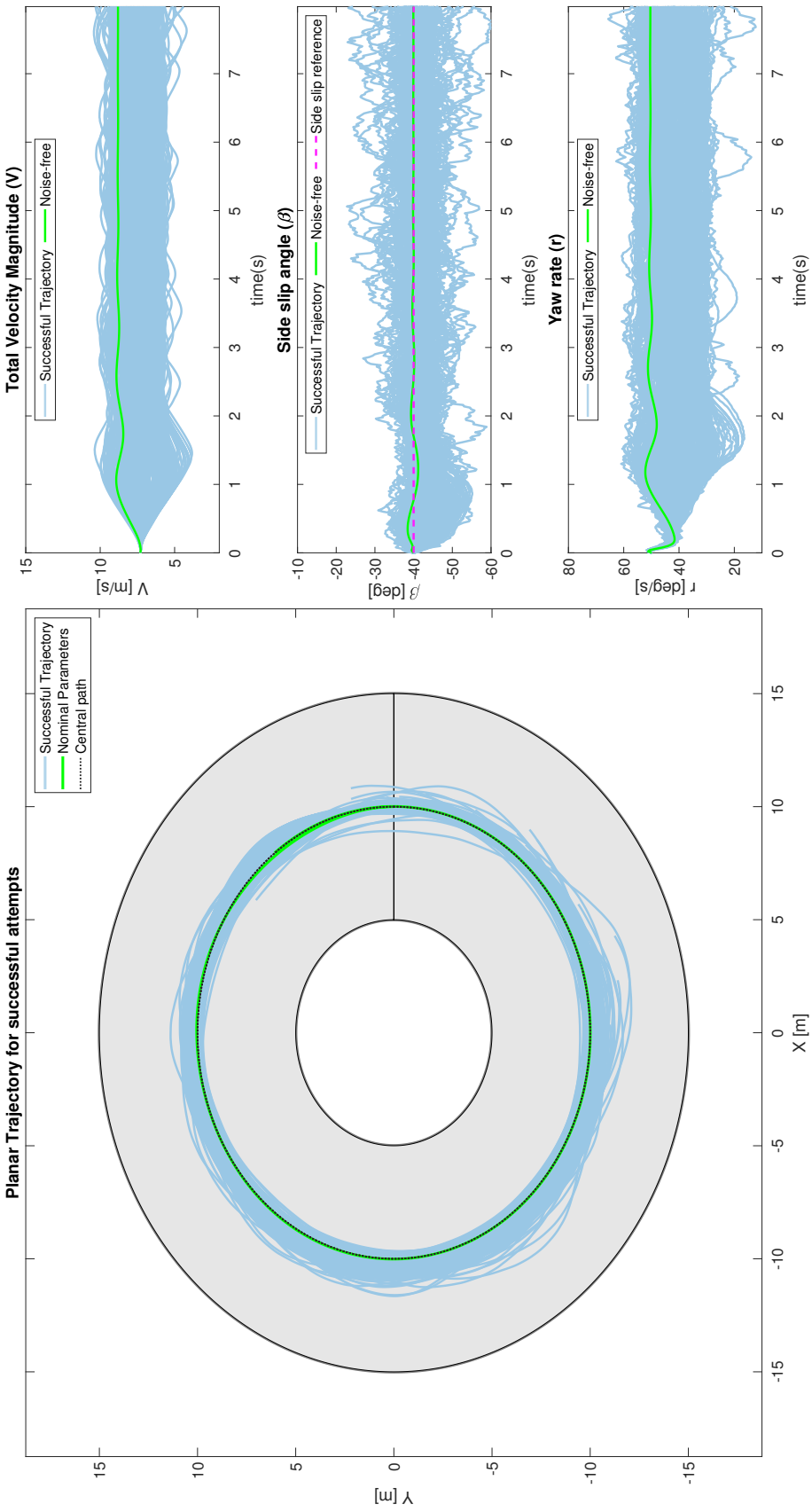


Figure 5-23: Vehicle trajectories with disturbed side slip estimation.

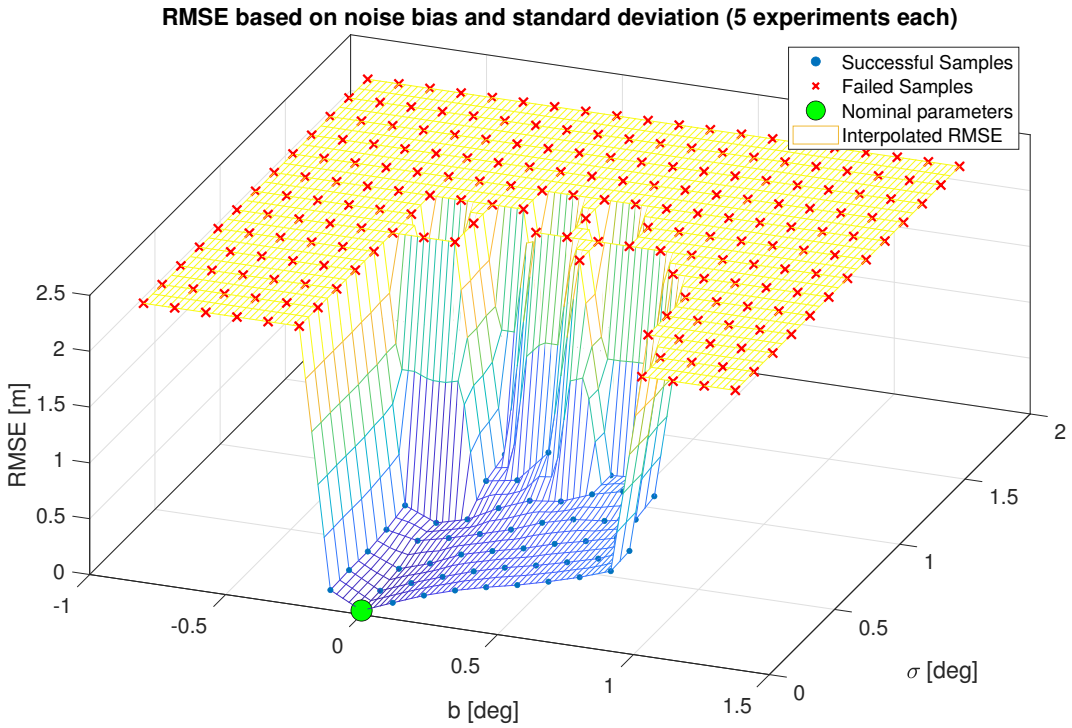


Figure 5-24: RMSE of planar trajectory central path.

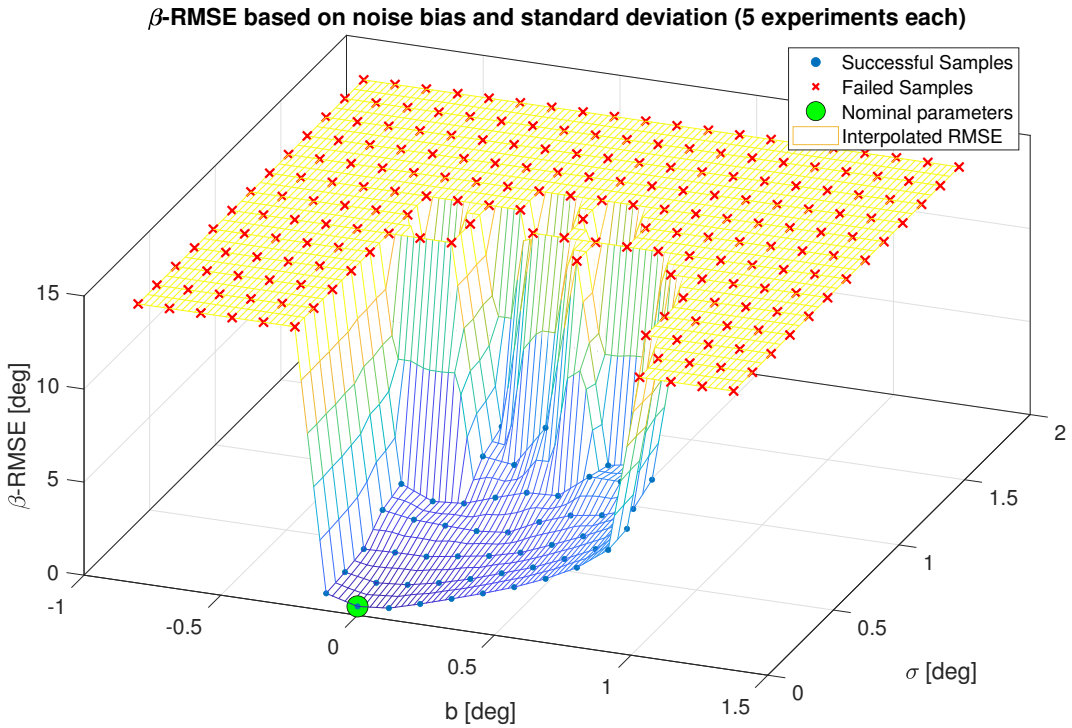


Figure 5-25: RMSE between the actual side slip angle and setpoint (β-RMSE).

Similar to previous scenario, the plateaus displayed in yellow represent planner or controller infeasibility. This usually represented an attempt in which the vehicle violated the track limits or a numerical problem occurred in the controller optimization problem.

In contrast with the investigation on mass and weight distribution mismatch, the reliance on a precise side slip angle estimation is considerably larger. As far as simulations have shown, even minor constant disturbances, e.g., $b = -0.35^\circ$ and $\sigma = 0$, cause problems and lead to collisions with track limits. This strongly suggest investigating robust, accurate methods that can provide high-quality side slip estimation to make this approach feasible in real scenarios.

The previous observation can also be supported by investigating the state trajectories from Figure 5-23. Considering that only successful trajectories are presented, and as this can be traced down to small b and σ, there is a disproportionately large variation in side slip angle over time caused by an arguably lesser noise. This variation in side slip angle further causes sizable tracking errors w.r.t both central path and side slip setpoint β^{sp}.

As noticed in the set of values for b and further observed in either Figure 5-24 or Figure 5-25, there is an asymmetry in the effect of noise mean.
Depending on the context, this can be traced back to either the turn side or the sign of the side slip angle. In this case, a positive bias lowers the actual magnitude of the side slip and puts the solution into a region of the state space where the previous guess is still relevant. A negative bias puts the controller reach to a more nonlinear region and more harshly constrained, leading to numerical problems or to improper solutions that later lead to collisions.

Furthermore, it is valuable to decompose the observed planar tracking error w.r.t central path into its bias and variance components. Formally, this will be done using the following formula:

$$\underbrace{\frac{1}{Q}\sum_{q=1}^{Q}\|E^* - E(q)\|^2}_{\text{Mean Squared Error}} = \underbrace{\|E_{mean} - E^*\|^2}_{\text{Bias}^2} + \underbrace{\frac{1}{Q}\sum_{q=1}^{Q}\|E(q) - E_{mean}\|^2}_{\text{Variance}}, \tag{5-2}$$

where E^* is the nominal error, $E(q)$ is the error for one trial, and E_{mean} is the mean error across the Q trials. This method is similar to how sample mean and variance are computed in scientific studies.

For this investigation, the noise realization that was selected was $(b = 0.01, \sigma = 0.01)$, which is placed well within the feasible region. Numerically, the following results was obtained:

$$\underbrace{0.34097}_{\text{Mean Squared Error}} = \underbrace{0.33276}_{\text{Bias}^2} + \underbrace{0.0082}_{\text{Variance}} \tag{5-3}$$

This suggests that the side slip estimation error largely contributes to an offset in the planar trajectory. Interestingly, this confirms the intuition one develops after observing the state trajectories in Figure 5-23.

Computation time

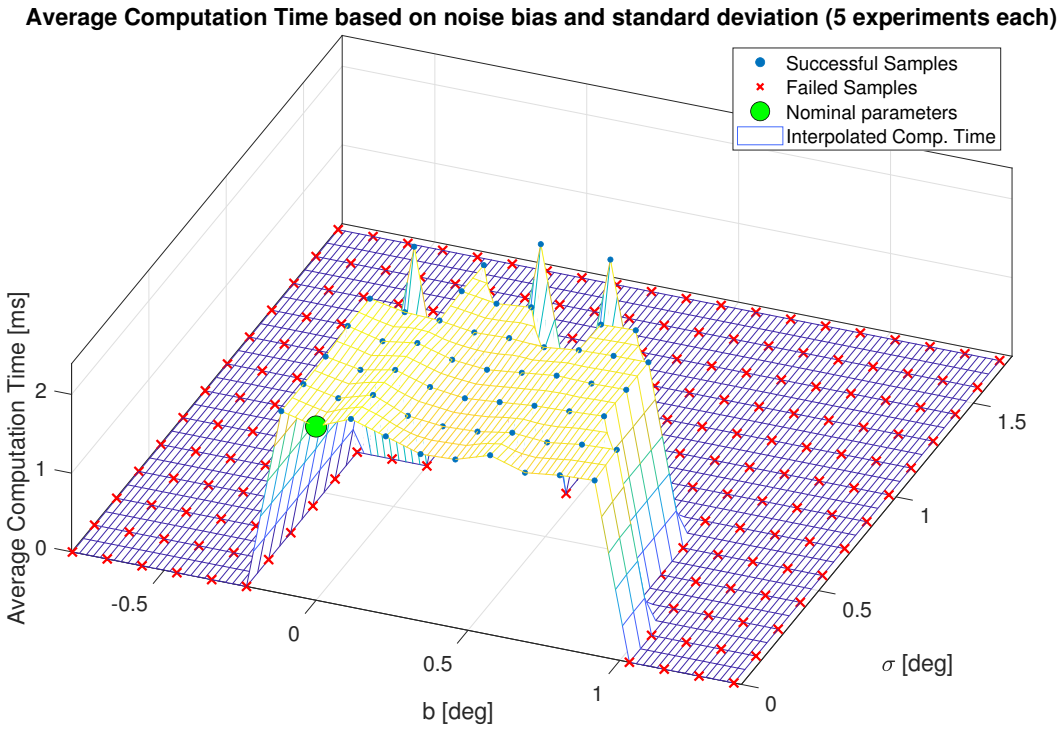


Figure 5-26: Computation time based on noise mean and standard deviation.

From a computational perspective, the conclusion is similar to what was observed in Section 5-9. As Figure 5-26 shows, the average total solving time hardly varies for successful attempts. This suggests the relative steadiness of computation times w.r.t to side slip estimation errors, in contrast with the sensitivity of tracking performance on this class of uncertainty.

However, in the case of side slip estimation errors, there is a profound need to robustify the state estimation methods. For this reason, the current work proposes investigating side slip estimation methods that would greatly extend the applicability of the proposed drift control strategy in the real world. This will be detailed in Chapter 6.

5-11 Summary

This chapter detailed several testing scenarios constructed for evaluating the proposed drift controller. The objectives of these investigations were the ability to converge do drift equilibria, the nominal performance, and the sensitivity to initial conditions, parameter mismatch, and side slip estimation errors. Moreover, these scenarios also validated the computational suitability of the proposed approach for a real-time implementation. In addition to the insights and remarks provided in each section, an aggregated conclusion based on the overall performance and robustness of the proposed control structure will be formulated in the next chapter.

Chapter 6

Conclusion

This chapter is dedicated to closing remarks in the form of a short summary on the contribution of this work and a description of the proposed future work. The information presented in this chapter is primarily intended to correlate the findings from Chapter 5 to the central motivation of this work introduced in Chapter 1. In other words, this chapter should reveal the extent to which the proposed approach satisfies the original problem and to what degree.

6-1 Summary

Full vehicle automation depicts a reality in which the responsibility of handling driving scenarios is passed from human to the artificial drivers. This transition promises increased efficiency, decrease travel costs, and most importantly extended passenger safety. One of the core features of such high levels of automation is the ability to respond well to every encountered scenario, including vehicle control on slippery surfaces that may lead to an increase in body side slip. In essence, these high side slip maneuvers are highly nonlinear, fast dynamical processes, and the available tools to control them are severely limited by structural constraints and the physical environment (road limits, obstacles).

However, it is important to understand the underlying motivation behind the designed test cases. The current work was aimed at developing a control structure that is able to follow a desired side slip angle setpoint, while not compromising its obstacle-avoidance features. Precisely, the other vehicle states are sacrificed to minimize side slip tracking error. This is in contrast with the usual velocity maximizing racing scenarios.

In the context of this work, the fundamental challenge in meeting these criteria is enabling the real-time implementation of the proposed strategy. As drifting is a highly nonlinear, fast dynamical process, the reduced time availability might hinder an unsuitable controller from generating stabilizing commands in due time. This is more pronounced if the objective is beyond drift stabilization, such as path tracking or obstacle-avoidance. For this reason, the current work was focused on reducing the computational burden by incorporating drifting particularities in the solution.

Therefore, this work proposed a hierarchical structure based on Nonlinear Model Predictive Controller (NMPC) to fully control drifting maneuvers as described in Chapter 3. In order to ensure computational feasibility, the original Optimal Control Problem (OCP) was conveniently reformulated, split, and simplified as shown in Chapter 4. Several simulations were constructed to test the limitations of the proposed strategy and to validate its benefits for the initial problem. The test scenarios presented in Chapter 5 are aimed at exciting the weak points of the structure, primarily its sensitivity to parameter uncertainty as the solution is fundamentally a model-based control approach. The outcomes of these scenarios are described in the next section.

6-2 Conclusions

First, the proposed control structure successfully lowers the computational burden of the drifting controller down to levels that make it feasible for real-time implementations. The ability to control drifting maneuvers is not compromised with these numerical improvements. Moreover, it has been shown the proposed split in Navigation and Drift Stabilization objectives does not restrict the vehicle from reaching steady-state drifting, as observed in Section 5-2. The vehicle states asymptotically converge to a Dynamic Bicycle Model (DBM) equilibrium.

Second, the drift controller is able to exploit these maneuvers to avoid static obstacles while still sustaining high side slip angles. These results are strengthened by the relatively reduced available space for maneuvers, as the created maps mimick realistic road dimensions which tend to be rather narrow for movements such as drifting.
However, the arbitrariness of the obstacle-avoidance simulations make it hard to extend the technical findings outside these scenarios. Without comparing to human reactions in similar situations or testing in historical scenarios, it is difficult to formulate a definitive opinion on controller performance. This motivated the creation of more generic, obstacle-free scenarios that were discussed in the second half of Chapter 5.

The resulting problem is less sensitive to the quality of initial guesses. In the original All-in-one formulation, the control structure is heavily reliant on accurate guesses. While this is still valid for the proposed solution as the drifting problem is essentially non-convex, experiments have shown it happens at a much reduced scale with the proposed approach. This is especially important as realistic scenarios require numerical stability in order to provide at least good-enough solution.

Lastly, from a quantitative perspective the reduction in solving times is notable.
The central motivation is to create a structure that is able to provide good-enough control actions in a feasible amount of time. This naturally translates to a feasible worst-case scenario computation time. It has been shown in the Benchmark scenario (Section 5-4) that the proposed solution provides a significant speedup factor of 85. This does not only reduce the computation time below the imposed limit (T_s), but it does so with a considerable margin, occupying approximately 25% of the available time.
Furthermore, the vast majority of cases are computationally feasible with great confidence. Across all simulations, the 90^{th} percentile of solving times did not exceed 4 milliseconds. This holds true even in harsher testing conditions with both sudden friction changes and obstacle-avoidance conditions. This computational robustness is also validated in model mismatch

scenarios.
The only scenario that raises problems from a computational perspective is the outer-most trajectory from Section 5-6. However, empirical findings show that this is a relatively isolated case that can be solved using a method proposed in the next section.

Concisely, the previous observations certify the value of the proposed numerical improvements for autonomous drifting and further suggest the suitability for a real-time implementation.

6-3 Recommendations for future work

Validation of the proposed strategy on a real vehicle

The immediate recommendation is to validate the proposed structure on an experimental test bed. Fundamentally, the literature offers generous example such as [11] that utilize the Dynamic Bicycle Model (DBM) along with the Magic Formula (MF) to control drifting maneuvers. However, the investigation on a real vehicle will show the suitability of the split in dynamics between the Navigation and Drift Stabilization layers when dealing with real car dynamics.

As it was stated in Chapter 1, the current work assumes the vehicle starts in the vicinity of a drift equilibrium for the proposed split in dynamics to work. It is therefore valuable to assess experimentally the sensitivity of the proposed approach to the distance between initial conditions and the drift equilibrium that corresponds to track properties. Supposedly, if this distance is big enough, the structure will fail to converge to a suitable equilibrium that allows to drift along the desired path.

Lastly, it is worth investigating the success of the proposed friction update scheme in the presence of measurement noise and state estimation errors.

Initial guesses based on initial lateral deviation n_0

The effect of the initial lateral deviation n_0 on vehicle state trajectories has also been investigated. The analysis revealed that the control structure is able to reduce the initial spread in lateral deviation and converge to the central path in the case of simple U-turns.

Similarly, the effect of the lateral deviation n_0 was also tested in obstacle-avoidance scenarios to analyze the success rate of safely navigating the track. However, the most extreme initial lateral deviation that still corresponds to a successful trajectory ($n_0 = 1.3$ meters) contributed to a sizable increase in Controller solving time for the first sample (more than 15 milliseconds). The empirical knowledge suggests this increase is largely caused by the different between the initial guess for the first sample (in the absence of warm-starting) and the actual solution. As the starting position is close to the outer track limit, the Controller needs to generate noticeably different maneuvers from the provided guess, increasing the search duration.

A possible solution is the generation of initial guesses by propagating the current vehicle state, position, and track information using either the model dynamics or a predefined open loop maneuver. By arriving at a guess that is closer to the solution of the first sample, the solving time will likely be close to the average. This statement is motivated by the fact that even for

the trajectory in cause, the solving time for the second sample returns to the average solving time thanks to warm-starting.

Nonlinear friction update algorithm

A simplistic friction update rule was proposed that improved the vehicle behavior in the presence of friction uncertainty. For simple scaling mismatch scenarios, it allowed the control structure to fully adapt to the new friction surface and track the desired side slip setpoint. However, a more realistic scenario was constructed in Chapter 5 that proved its limitations when dealing with more complex friction mismatch, as it is expected in realistic scenarios. In this sense, several avenues will be suggested that are more suitable for highly nonlinear processes such as drifting.

The literature offers potential research avenues for this problem:

1. Constructing a lateral force observer and implementing an online gradient descent algorithm for updating the B, C, D parameters, similar to what was achieved in [28]. In the latter, the authors employ a joint estimation technique for computing both side slip angle and the friction model. As it was be observed in Section 5-10, the side slip estimation problem is of high interest for the strategy proposed in this work.
2. A more relevant solution can be found in [29]. The authors propose a Nonlinear Moving Horizon Estimation (NMHE) technique for estimating the parameters of the Pacejka tire model. The results show an estimation rise time of approximately one second. However, the computation time of this algorithm exceeds 20 milliseconds, making it likely unsuitable for the drifting problem, but a more thorough investigation is needed.
3. A different perspective is given in [30], in which the authors construct a neuro-adaptive observer to estimate tire forces as a function of tire slip. This method also allows state estimation, which can be adapted to drifting regions.

A robust side slip estimation method

The current work assumed full state information available. As this is hardly the case for realistic control applications, the literature offers generous examples on the challenges of estimating the side slip angle. As Section 5-10 has shown, the proposed control strategy is highly sensitive to estimation errors of the side slip angle. This holds true even for a Gaussian noise with a mean of less than a degree in magnitude.

Consequently, the current work suggests that future work should be dedicated to improving the robustness of the side slip estimation method. A possible solution is to formulate a joint estimation problem as in [28], tackling the two most promising issues, the friction update and the side slip estimation problems.

Test higher-fidelity models that include load transfer

The mass and weight distribution mismatch scenario showed that the proposed structure is less sensitive to this class of mismatch. However, load transfer dynamics in the pitch axis

can be incorporated into the controller dynamics to further increase its performance outside nominal operating regimes. From a computational perspective, it is yet to be investigated if the additional complexity caused by load transfer mechanics does not violate the real-time constraints. Regardless, empirical observations suggest that this will not be the case.

Moreover, the model fidelity can be increased also by replacing the DBM with a Two-track model, with the possibility to also include roll dynamics. Several works such as [12, 13] have validated the use of the Two-track model for similar tasks, making it a great candidate for the autonomous drifting problem. However, this will further increase the model accuracy and the robustness of the solution at the expanse of increased computational load. As there will be more states to propagate, the Controller problem can become computationally unfeasible in the absence of additional simplifying assumptions.

Appendix A

Numerical values

This chapter will include details about the simulation environment, numerical vehicle parameters, cost function weights, and map dimensions.

A-1 Simulation environment

The simulations were ran on a machine equipped with an Intel i7-11700K processor @3.60GHz, with 32 GB of RAM @3200 MHz, Windows 10, and Matlab R2021a.
The NMPC was implemented using FORCES Pro v6.0.0 and CasADi v3.5.5, using the following *codeoptions*:

1. Maximum number of iterations was set to: $maxit = 2000$
2. Optimization level was set to favor speed: $optlevel = 2$
3. Solve method was set to Primal-Dual Interior Point: *PDIP_NLP*
4. Hessian approximation method was set to Primal-Dual Interior Point: *BFGS*
5. Inequality Residuals Tolerance: $accuracy.ineq = 10^{-5}$
6. Equality Residuals Tolerance: $accuracy.eq = 10^{-5}$
7. Duality Gap Tolerance: $accuracy.mu = 10^{-4}$

A-2 Vehicle parameters

The chassis parameters were taken from [10], and the Magic Formula parameters for the tire model were taken from [13]. The numerical values are summarized in Table A-1.

Symbol	Name	Value	Unit of measure
T_s	Sampling time	0.02	s
L	Length	4.085	m
W	Width	2.4	m
d_{car}	Half-diagonal	2.3689	m
m	Mass	1700	kg
l_F	Distance to front axle	1.392	m
l_F	Distance to rear axle	1.008	m
$F_{z,F}$	Vertical load on front axle	7004	N
$F_{z,R}$	Vertical load on rear axle	9672	N
I_z	Yaw inertia	2385	kg m^2
I_ω	Wheel moment of inertia	3	kg m^2
R_{tire}	Tire radius	0.33	m
g	Gravitational acceleration	9.81	$\frac{m}{s^2}$
$B_F = B_R = B$	Stiffness factor	11.24	-
$C_F = C_R = C$	Shape factor	1.45	-
$D_F = D_R = D$	Peak value	1	-
V^{min}	Minimum total velocity magnitude V	0	$\frac{m}{s}$
V^{max}	Maximum total velocity magnitude V	30	$\frac{m}{s^2}$
dV^{max}	Maximum difference in V^{ref}	2	$\frac{m}{s^2}$
β^{min}	Minimum side slip angle β	-60	deg
β^{max}	Maximum side slip angle β	60	deg
$d\beta^{max}$	Maximum difference in β^{ref}	40	$\frac{deg}{s}$
r^{min}	Minimum yaw rate r	-10	$\frac{rad}{s}$
r^{max}	Maximum yaw rate r	10	$\frac{rad}{s}$
dr^{max}	Maximum difference in r^{ref}	3	$\frac{rad}{s^2}$
ω^{min}	Minimum wheel speed ω	5	$\frac{rad}{s}$
ω^{max}	Maximum wheel speed ω	150	$\frac{rad}{s}$
$d\omega^{max}$	Maximum difference in ω^{ref}	100	$\frac{m}{s^2}$
τ^{max}	Maximum throttle input (torque)	3000	Nm

Table A-1: Numerical vehicle parameters

A-3 Numerical OCP

A-3-1 Planner OCP

The Navigation layer implements a Nonlinear Model Predictive Controller with a prediction horizon $N^{\mathrm{P}} = 40$, and a haste factor $\eta = 4$:

$$Z^P = \begin{bmatrix} \epsilon & dV^{\mathrm{ref}} & d\beta^{\mathrm{ref}} & dr^{\mathrm{ref}} & V^{\mathrm{ref}} & \beta^{\mathrm{ref}} & r^{\mathrm{ref}} & X & Y & \phi \end{bmatrix}^T, \tag{A-1}$$

$$\begin{aligned} \min_{Z^{\mathrm{P}}} \quad & \sum_{k=1}^{N_{\mathrm{P}}} \Big[1000(\theta - \beta^{\mathrm{sp}})^2 + \sin(\theta - \beta^{\mathrm{ref}})^2 + 0.75n^2 \\ & \qquad + 1000(d\beta^{\mathrm{ref}})^2 + 100(dr^{\mathrm{ref}})^2 + 0.4\frac{1}{\epsilon + 10^{-4}} \Big] \end{aligned} \tag{A-2a}$$

$$\text{s.t.} \quad Z^{\mathrm{P}}_{k+1} = f_{\mathrm{planner}}(Z^{\mathrm{P}}_k) \qquad \text{[Planner dynamics]} \tag{A-2b}$$

$$\underline{Z}^{\mathrm{P}} \leq Z^{\mathrm{P}}_k \leq \overline{Z}^{\mathrm{P}} \qquad \text{[Hard constraints (bounds)]} \tag{A-2c}$$

$$\frac{(S - S_{\mathrm{O1}})^2}{(R_{\mathrm{O1}} + \mathrm{c}\, d_{\mathrm{car}})^2} + \frac{(n - n_{\mathrm{O1}})^2}{(R_{\mathrm{O1}} + \mathrm{c}\, d_{\mathrm{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_1\text{]} \tag{A-2d}$$

$$\frac{(S - S_{\mathrm{O2}})^2}{(b_{\mathrm{O2}} + \mathrm{c}\, d_{\mathrm{car}})^2} + \frac{(n - n_{\mathrm{O2}})^2}{(a_{\mathrm{O2}} + \mathrm{c}\, d_{\mathrm{car}})^2} - \epsilon \geq 1 \qquad \text{[Obstacle O}_2\text{]} \tag{A-2e}$$

A-3-2 Controller OCP

The Drift Stabilization layer implements a Nonlinear Model Predictive Controller with a prediction horizon $N^{\mathrm{C}} = 30$:

$$Z^{\mathrm{C}} = \begin{bmatrix} d\omega^{\mathrm{ref}} & d\delta & \omega^{\mathrm{ref}} & \delta & V & \beta & r \end{bmatrix}^T, \tag{A-3}$$

$$\begin{aligned} \min_{Z^{\mathrm{C}}} \quad & \sum_{k=1}^{N_{\mathrm{C}}-1} \Big[120(\beta - \beta^{\mathrm{ref}})^2 + 20(r - r^{\mathrm{ref}})^2 + 0.03(d\omega^{\mathrm{ref}})^2 + 40(d\delta)^2 \Big] \\ & \qquad + 500(\beta - \beta^{\mathrm{ref}})^2 + 80(r - r^{\mathrm{ref}})^2 + 1(d\omega^{\mathrm{ref}})^2 + 400(d\delta)^2 \end{aligned} \tag{A-4a}$$

$$\text{s.t.} \quad Z^{\mathrm{C}}_{k+1} = f_{\mathrm{controller}}(Z^{\mathrm{C}}_k) \qquad \text{[controller dynamics]} \tag{A-4b}$$

$$\underline{Z}^{\mathrm{C}} \leq Z^{\mathrm{C}}_k \leq \overline{Z}^{C} \qquad \text{[hard constraints (bounds)]} \tag{A-4c}$$

A-4 Map properties

This document features a lengthier, circular map, and a shorter U-turn. The latter is simply the circular map truncated after its middle point. The properties that describe both the track and the obstacles can be found in Table A-2.

Symbol	Name	Value	Unit of measure
κ	Curvature of central path	0.1	m^{-1}
hw	Road half-width (lane width)	5	m
L_{path}	Length of central path	62.83	m
S_{O1}	Progress-wise position of O1	15.7	m
n_{O1}	Lateral-deviation of O1	-3.33	m
a_{O1}	Radius on semiminor axis	1.66	m
b_{O1}	Radius on semimajor axis	4	m
S_{O2}	Progress-wise position of O2	43.98	m
n_{O2}	Lateral-deviation of O2	3.45	m
R_{O2}	Radius of O2	1.56	m

Table A-2: Numerical map properties

Appendix B

Simulation panel

B-1 Simulation panel

The simulations presented in Chapter 5 were organized in a Matlab application to provide a better interface with simulation options.

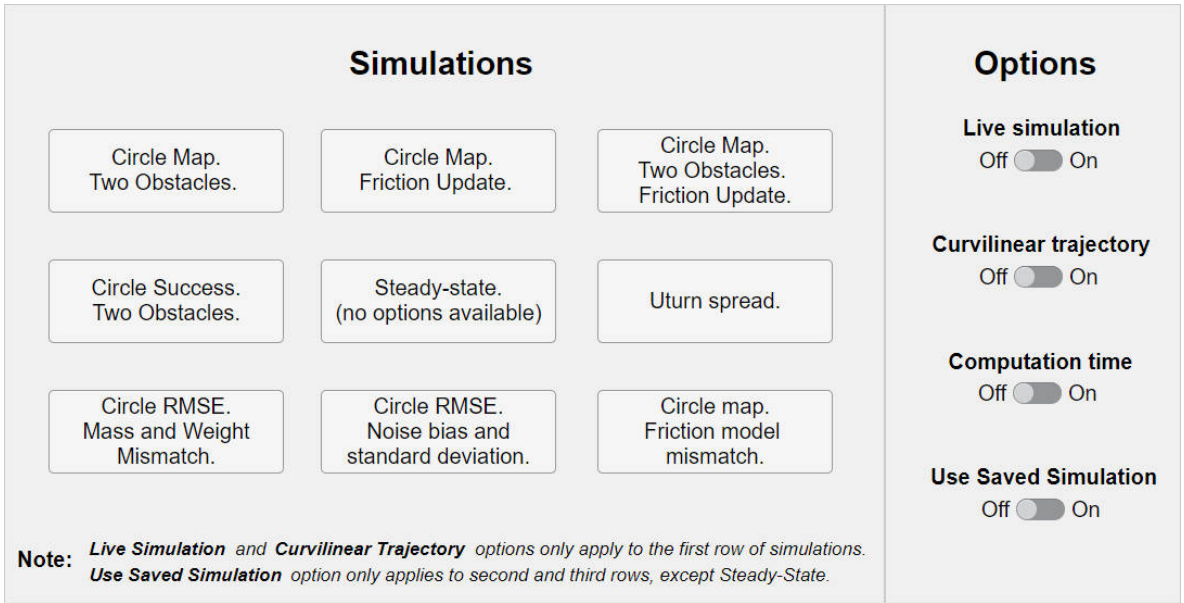


Figure B-1: Simulation panel

The simulation is ran directly after pressing the corresponding button. The sliders on the right pass the selected options to the simulation to minimize user operations. These options include:

1. **Live simulation.** This option is available only for the first three simulations. When activated, a visual dashboard will be created that displays the vehicle position, states

and inputs in real time. State and trajectory predictions are also shown, along with wheel orientation. For example, the obstacle avoidance scenario from Section 5-4 can be visualized in Figure B-2.

2. **Curvilinear trajectory.** Activating this option creates an additional plot that displays vehicle trajectory in curvilinear space, along with side slip and yaw rate trajectories. Similarly, this is available only for the first row of simulations. Using the same example as above, the curvilinear trajectory is presented in Figure B-3.

3. **Computation time.** Checking this option displays the computation time for each sample (or aggregated set of samples) of the experiment. This option is not available for the Steady-State simulations as it conveys little information on the overall performance.

4. **Use Saved Simulation.** This option is especially intended for the lengthy simulations sitting on the bottom row. Checking this option will display the plots based on a previous simulation saved in a *.mat* file. If disabled, the program will generate the solvers and then run the simulation as in the other scenarios.

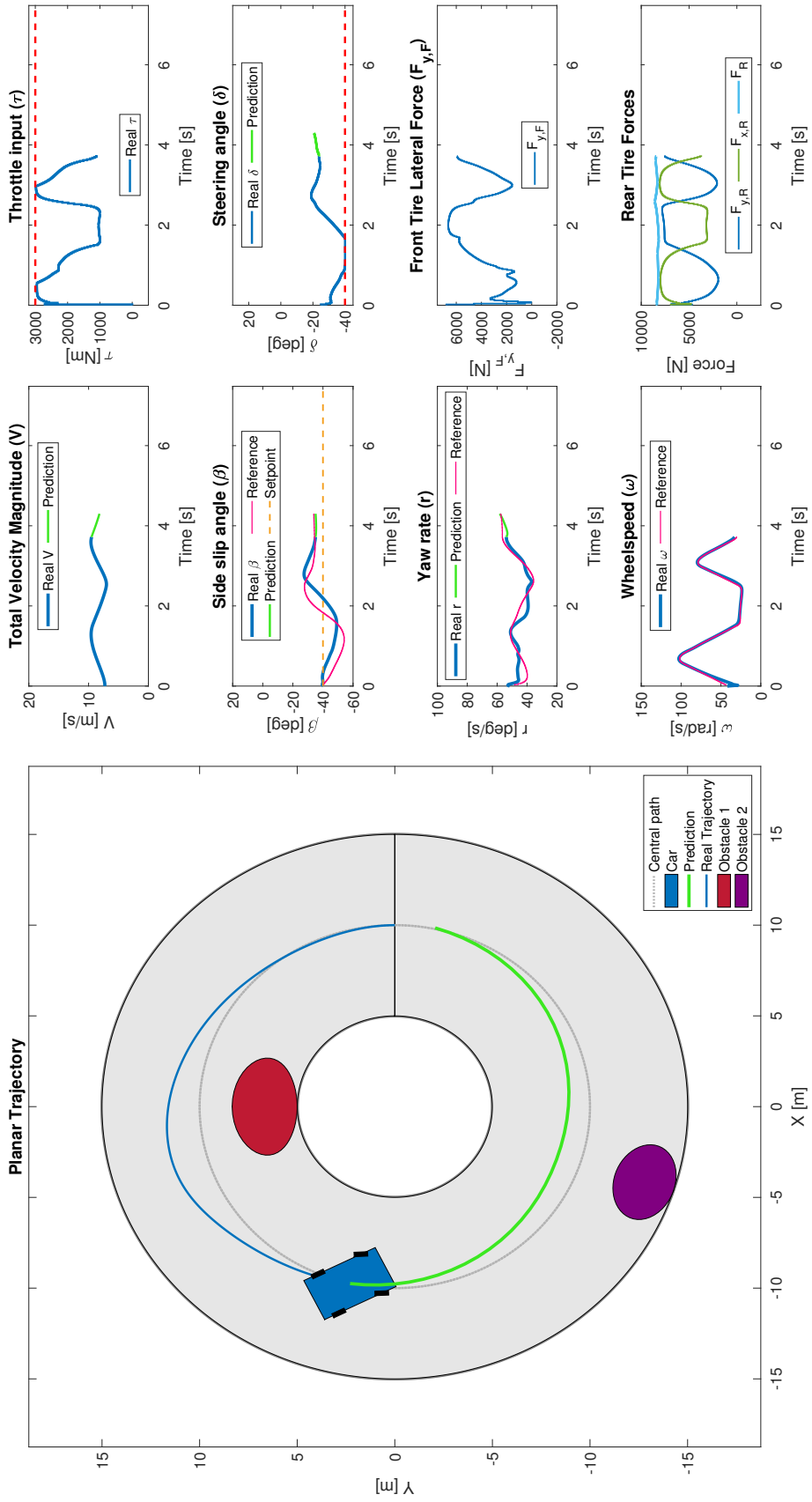


Figure B-2: Live dashboard.

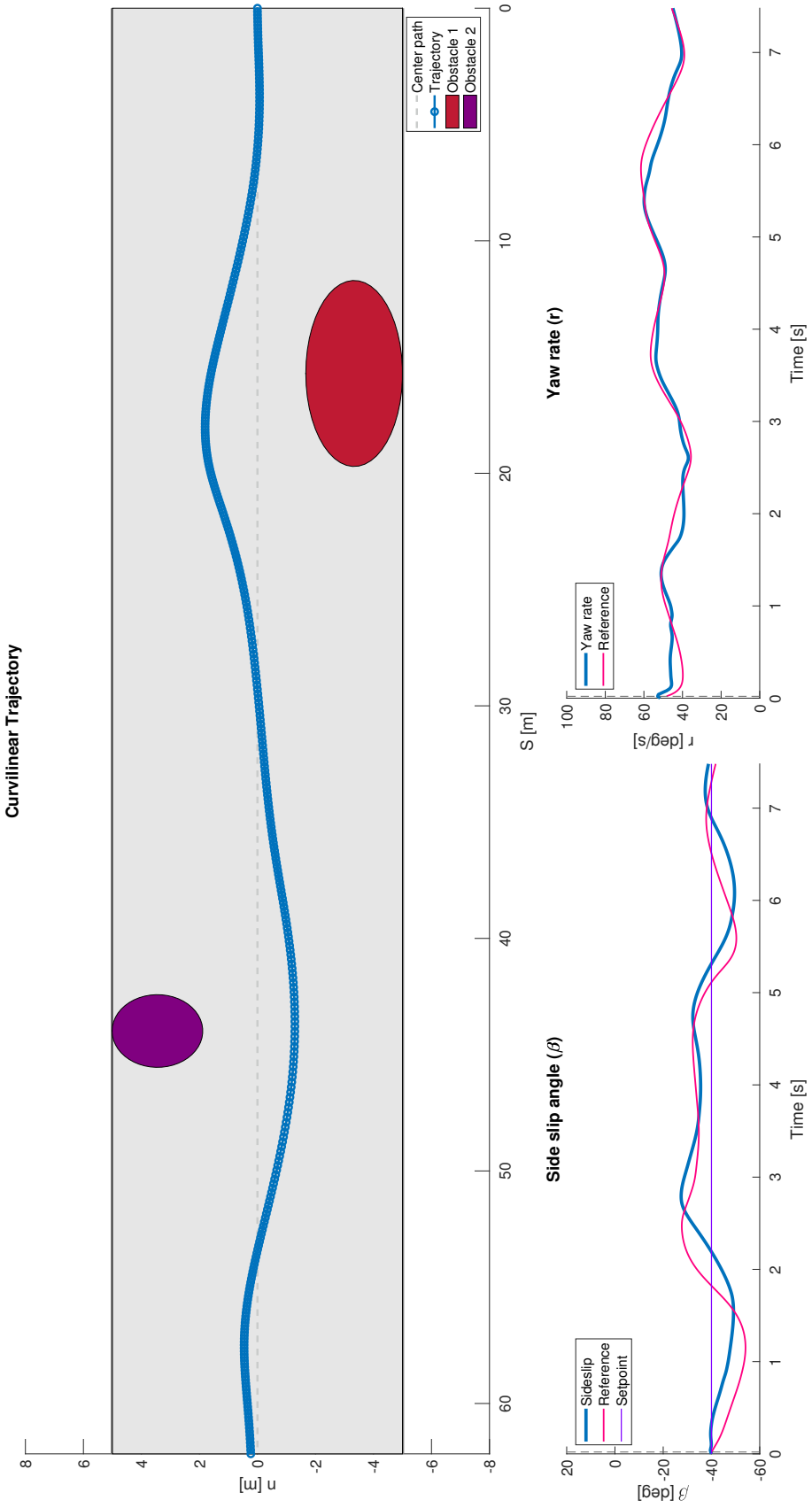


Figure B-3: Obstacle course trajectory in curvilinear space.

Bibliography

[1] AITAD, "What is autonomous driving?." https://aitad.de/en/what-is-autonomous-driving-where-are-we-today-where-will-we-be-tomorrow, 2022.

[2] G. v. L. Campagne, "A Nonlinear Model Predictive Control based Evasive Manoeuvre Assist Function Gijs van Lookeren Campagne," Master's thesis, TU Delft, 2019.

[3] J. Edelmann and M. Plöchl, "Handling characteristics and stability of the steady-state powerslide motion of an automobile," *Regular and Chaotic Dynamics*, vol. 14, no. 6, pp. 682–692, 2009.

[4] R. Y. Hindiyeh, *Dynamics and control of drifting in automobiles*. PhD thesis, Stanford University, 2013.

[5] S. Gros and M. Diehl, "Numerical optimal control (draft)." https://www.syscop.de/files/2020ss/NOC/book-NOCSE.pdf, 2022.

[6] Thales, "7 benefits of autonomous cars." https://www.thalesgroup.com/en/markets/digital-identity-and-security/iot/magazine/7-benefits-autonomous-cars, 2017.

[7] C. Voser, R. Y. Hindiyeh, and J. C. Gerdes, "Analysis and control of high sideslip manoeuvres," *Vehicle System Dynamics*, vol. 48, no. SUPPL. 1, pp. 317–336, 2010.

[8] R. Y. Hindiyeh and J. C. Gerdes, "A controller framework for autonomous drifting: Design, stability, and experimental validation," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME*, vol. 136, no. 5, pp. 1–9, 2014.

[9] J. Y. Goh and J. C. Gerdes, "Simultaneous stabilization and tracking of basic automobile drifting trajectories," *IEEE Intelligent Vehicles Symposium, Proceedings*, vol. 2016-Augus, no. Iv, pp. 597–602, 2016.

[10] J. Y. Goh, T. Goel, and J. Christian Gerdes, "Toward Automated Vehicle Control Beyond the Stability Limits: Drifting Along a General Path," *Journal of Dynamic Systems, Measurement, and Control*, vol. 142, no. 2, pp. 1–10, 2020.

[11] E. Velenis, E. Frazzoli, and P. Tsiotras, "Steady-state cornering equilibria and stabilisation for a vehicle during extreme operating conditions," *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2-4, pp. 217–241, 2010.

[12] E. Velenis, D. Katzourakis, E. Frazzoli, P. Tsiotras, and R. Happee, "Steady-state drifting stabilization of RWD vehicles," *Control Engineering Practice*, vol. 19, no. 11, pp. 1363–1376, 2011.

[13] E. Siampis, E. Velenis, S. Gariuolo, and S. Longo, "A real-time nonlinear model predictive control strategy for stabilization of an electric vehicle at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 6, pp. 1982–1994, 2018.

[14] M. Acosta, *A Drift-based approach to Last-Moment Accident Avoidance Manoeuvres on Loose*. PhD thesis, Coventry University, 2017.

[15] M. Acosta, S. Kanarachos, and M. E. Fitzpatrick, "A hybrid hierarchical rally driver model for autonomous vehicle agile maneuvering on loose surfaces," *ICINCO 2017 - Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics*, vol. 2, pp. 216–225, 2017.

[16] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[17] E. Jelavic, J. Gonzales, and F. Borrelli, "Autonomous Drift Parking using a Switched Control Strategy with Onboard Sensors," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3714–3719, 2017.

[18] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," *SAE Technical Papers*, vol. 96, pp. 190–204, 1987.

[19] J. G. Ziegler, N. B. Nichols, *et al.*, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[20] R. Rajamani, N. Piyabongkarn, J. Lew, K. Yi, and G. Phanomchoeng, "Methods for Active Automotive Safety Applications," *IEEE Control Systems Magazine*, vol. 30, no. 4, pp. 54–69, 2010.

[21] R. Rajamani, G. Phanomchoeng, D. Piyabongkarn, and J. Y. Lew, "Algorithms for real-time estimation of individual wheel tire-road friction coefficients," *IEEE/ASME Transactions on Mechatronics*, vol. 17, no. 6, pp. 1183–1195, 2012.

[22] H. Bock and K. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems*," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984. 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.

[23] A. Domahidi and J. Jerez, "FORCES Professional." Embotech AG, https://embotech.com/FORCES-Pro, 2014–2019.

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

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

[26] M. Vukov, S. Gros, G. Horn, G. Frison, K. Geebelen, J. B. Jørgensen, J. Swevers, and M. Diehl, "Real-time nonlinear MPC and MHE for a large-scale mechatronic application," *Control Engineering Practice*, vol. 45, pp. 64–78, 2015.

[27] T. Herrmann, A. Wischnewski, L. Hermansdorfer, J. Betz, and M. Lienkamp, "Real-Time Adaptive Velocity Optimization for Autonomous Electric Cars at the Limits of Handling," *IEEE Transactions on Intelligent Vehicles*, vol. XX, no. X, pp. 1–13, 2020.

[28] W. Chen, D. Tan, and L. Zhao, "Vehicle Sideslip Angle and Road Friction Estimation Using Online Gradient Descent Algorithm," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 11475–11485, 2018.

[29] M. Zanon, J. V. Frasch, and M. Diehl, "Nonlinear Moving Horizon Estimation for combined state and friction coefficient estimation in autonomous driving," *2013 European Control Conference, ECC 2013*, pp. 4130–4135, 2013.

[30] W. Jeon, A. Chakrabarty, A. Zemouche, and R. Rajamani, "Simultaneous state estimation and tire model learning for autonomous vehicle applications," *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 1941–1950, 2021.

Glossary

List of Acronyms

ADAS	Advanced Driver-Assistance Systems
ESC	Electronic Stability Control
ICR	Instant Center of Rotation
CoG	Center of Gravity
KBM	Kinematic Bicycle Model
DBM	Dynamic Bicycle Model
CoG	Center of Gravity
RWD	Rear-Wheel Drive
MF	Magic Formula
MPC	Model Predictive Controller
NMPC	Nonlinear Model Predictive Controller
NMHE	Nonlinear Moving Horizon Estimation
LQR	Linear Quadratic Regulator
RMSE	Root-Mean-Square Error
RLS	Recurise Least Squares
LQR	Linear Quadratic Regulator
OCP	Optimal Control Problem
SISO	Single-Input Single-Output
MIMO	Multiple-Input Multiple-Output
LQR	Linear Quadratic Regulator
ODE	Ordinary Differential Equation
RK4	4th-order Runge-Kutta
ODE	Ordinary Differential Equation
KKT	Karush-Kuhn-Tucker

NLP	Nonlinear Programming
SQP	Sequential Quadratic Programming
PDIP	Primal-Dual Interior Point
IPM	Interior Point Method
BFGS	Broyden–Fletcher–Goldfarb–Shanno

List of Symbols

α	Tire slip angle
β	Side slip angle
δ	Steering angle
η	Haste factor
Γ	Forgetting factor
$\hat{\mu}$	Friction coefficient estimate
κ	Road Curvature
μ	Friction coefficient
ω	Wheel speed
ϕ	Global orientation (Yaw)
$\phi(k)$	RLS regression vector
ψ	Angular velocity around ICR
τ	Throttle input (torque applied to rear wheel)
θ	Local heading w.r.t to road tangent
$d\beta^{\text{ref}}$	Change in side slip angle reference ($\beta^{\text{ref}}_{k+1} - \beta^{\text{ref}}_{k}$)
$d\delta$	Change in steering angle ($\delta_{k+1} - \delta_k$)
$d\omega^{\text{ref}}$	Change in wheel speed reference ($\omega^{\text{ref}}_{k+1} - \omega^{\text{ref}}_{k}$)
$(a_{\text{O1}}, b_{\text{O1}})$	Radii of Obstacle 1 (ellipse)
$(S_{\text{O1}}, n_{\text{O1}})$	Curvilinear position of Obstacle 1
$(S_{\text{O2}}, n_{\text{O2}})$	Curvilinear position of Obstacle 2
$(X_{\text{O1}}, Y_{\text{O1}})$	Global position of Obstacle 1
$(X_{\text{O2}}, Y_{\text{O2}})$	Global position of Obstacle 2
$\hat{d}$	State reference disturbance estimation
$\overline{Z}$	Upper bounds on decision variables (hard constraints)
O_1	First, elliptical obstacle (depicted in red)
O_2	Second, circular obstacle (depicted in purple)
$\underline{\text{Z}}$	Lower bounds on decision variables (hard constraints)
d_{car}	Vehicle's half-diagonal
dr^{ref}	Change in yaw rate reference ($r^{\text{ref}}_{k+1} - r^{\text{ref}}_{k}$)
dV^{ref}	Change in total velocity magnitude reference ($V^{\text{ref}}_{k+1} - V^{\text{ref}}_{k}$)

e	Wheel speed tracking error at a given instance
e^I	Wheel speed tracking error integral
f	System dynamics
$F_{\mathrm{x,R}}$	Rear tire longitudinal force
$F_{\mathrm{y,F}}$	Front tire lateral force
$F_{\mathrm{y,R}}$	Rear tire lateral force
I_ω	Moment of inertia about wheel's axis of rotation
I_z	Moment of inertia about vehicle's vertical axis (Yaw inertia)
K	RLS Update gain
k	Simulation step
K_I	Integral gain
K_P	Proportional gain
l_{F}	Distance between CoG and front axle
l_{R}	Distance between CoG and rear axle
m	Vehicle mass
n	Lateral deviation from track's central path
n_0	Initial lateral deviation
N_{C}	Controller prediction horizon
N_{P}	Planner prediction horizon
P	RLS Covariance matrix
R	Turn radius
r	Yaw rate
R_{O2}	Radius of Obstacle 1 (circle)
R_{road}	Outer track bound radius
r_{road}	Inner track bound radius
R_{tire}	Tire radius
S	Progress along track
s	Theoretical slip quantities
T_s	Sampling time
V	Total Velocity Magnitude
V^S	Projection of vehicle velocity on path
X	Global longitudinal position
Y	Global lateral position
Z^C	Controller optimization variables
Z^P	Planner optimization variables
B, C, D	Magic Formula parameters
c	Clearance (scalar enlargement of obstacles)
F	Front Tire
R	Rear tire
x	Longitudinal

$_{y}$	Lateral
$_{z}$	Vertical
$_{eq}$	Equilibria
$_{max}$	Maximum
$_{min}$	Minimum
$_{ref}$	Reference
$_{sp}$	Setpoint
$_{wp}$	Waypoint
$^{\circ}$, [deg]	Degrees