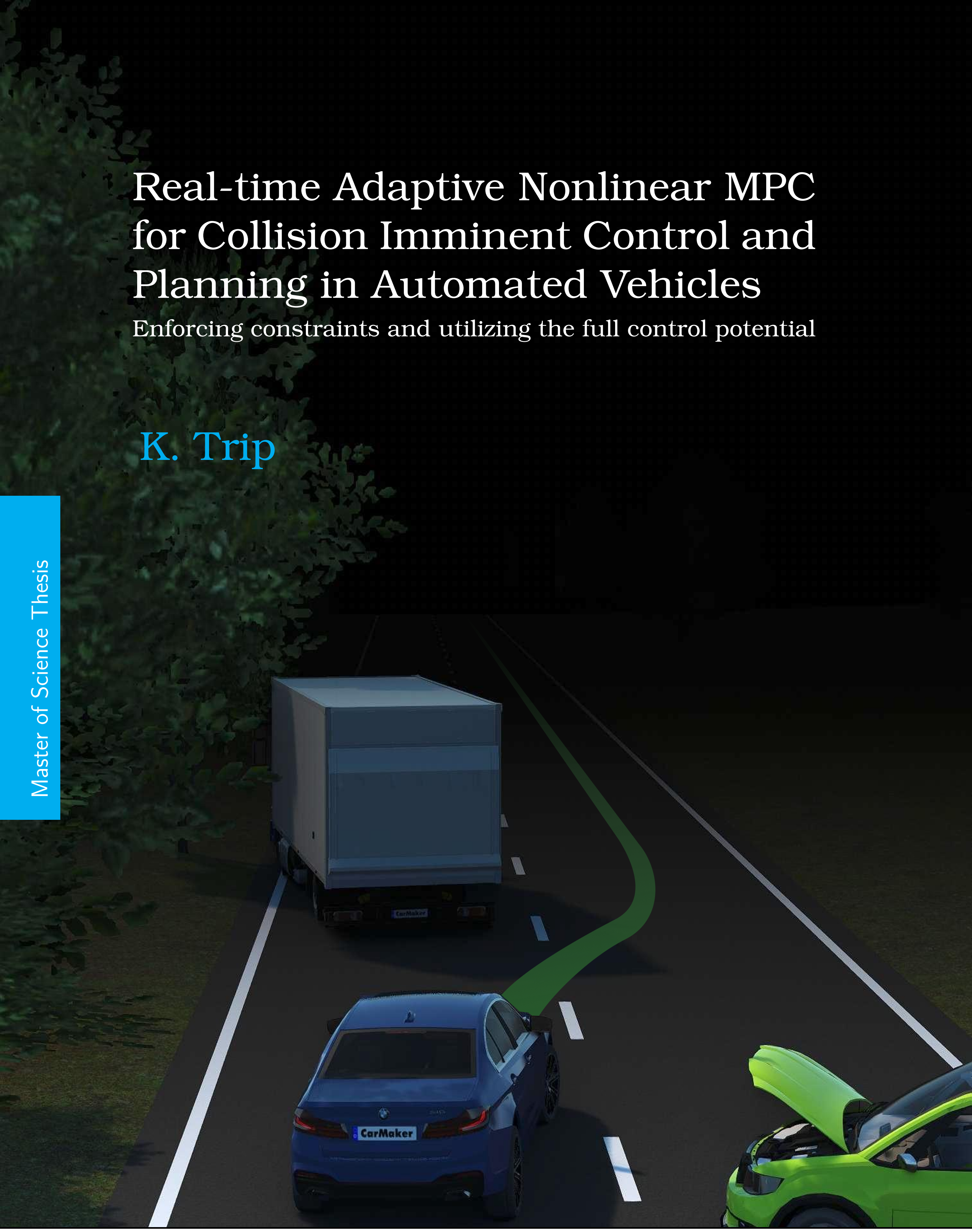
# Real-time Adaptive Nonlinear MPC for Collision Imminent Control and Planning in Automated Vehicles

Enforcing constraints and utilizing the full control potential

K. Trip

Master of Science Thesis



Delft University of Technology

Delft Center for Systems and Control

# Real-time Adaptive Nonlinear MPC for Collision Imminent Control and Planning in Automated Vehicles

**Enforcing constraints and utilizing the full control potential**

Master of Science Thesis

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

Kenrick Trip, 4661826

January 25, 2024

Embedded Systems track: Software and Networking

Faculty of Mechanical Engineering (ME)
Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS)

Delft University of Technology

# Abstract

With the introduction of autonomous vehicles on public roads, their performance in emergency situations has become a strong focus. Collision Imminent Control (CIC) concerns the planning and control of aggressive evasive maneuvers for collision avoidance of automated vehicles. CIC is implemented using adaptive Nonlinear Model Predictive Control (NMPC), which considers obstacles and road barriers for combined trajectory re-planning and control. To achieve real-time performance, the prediction model complexity is often reduced, which can lead to an under-utilization of the control potential. The aim of CIC is to use as much of the control potential of the vehicle as possible while remaining real-time viable.

In this research, CIC is implemented using objective-based collision avoidance based on the distance to obstacles and road boundaries. Different collision avoidance formulations were derived and compared on accuracy and real-time performance. The control potential of the vehicle was further exploited by a computationally efficient vehicle model that employs differential braking. The NMPC problem is solved using Sequential Quadratic Programming (SQP) with Real Time Iterations (RTI). Different techniques that reduce the computation time were compared. Sparse solvers and variable timesteps were found to be most significant.

The robustness of the controller was improved by friction estimation. The controller is furthermore demonstrated to work on highly curved roads and in scenarios with dynamic obstacles. The controller is implemented on the hardware of a real autonomous vehicle and simulated on a closed-loop embedded system. Combining all these elements results in a CIC controller that can apply more control potential and reach control frequencies upwards of $100Hz$, increasing the level of safety in vehicle collision avoidance.

# Table of Contents

# List of Figures

# List of Tables

# Preface

This work is a Master of Science graduation thesis on Collision Imminent Control (CIC) in autonomous vehicles. The aim of CIC is to achieve the highest level of safety in these scenarios. Since vehicle dynamics near the limits of handling are strongly non-linear, CIC controllers commonly apply NMPC. NMPC can exploit more modeling power as it allows for the formulation of nonlinear prediction model dynamics, constraints, and objectives. This allows for a greater utilization of the control potential, which can improve safety. However, NMPC has higher computational complexity, posing a challenge to real-time control.

State-of-the-art works on CIC combine path planning and tracking in a single (one-level) nonlinear optimization problem. This combination can employ more control potential of the vehicle to perform aggressive maneuvers [20]. However, it suffers from increased computational complexity [40]. How to implement such NMPC controllers in real-time using combined steering and differential braking control remains an open question. Therefore, techniques that can reduce the computational complexity of NMPC for CIC have become more relevant.

Different techniques investigated in this work include reductions in prediction model fidelity, formulations of collision avoidance, and solving techniques. Differential braking is applied on a low-fidelity single-track vehicle model using careful friction limit constraint formulations. Collision avoidance is formulated in the objective function of the optimization problem, where violations of a minimum safe distance between the vehicle and the obstacle dominate. Different geometric shapes are used to formulate these distances, aiming to decrease computation time and increase modeling accuracy. The effects of different solvers, variable timesteps, and global reference trajectories on the computational complexity are explored. The influences of changing friction coefficients, road curvatures, and obstacle locations on the CIC controller are evaluated. The best-suited combination of techniques can significantly reduce the computation time to allow for the exploitation of higher modeling power.

The CIC controller is implemented on the embedded controller of a developmental autonomous vehicle of the TUDelft and simulated in real-time using a Hardware in the Loop (HIL) system.

The thesis committee consists of the following members:

- Prof. Dr. M.E. Mazo,
- Prof. Dr. M.T.J. Spaan,
- L. Gharavi.

# Acknowledgements

I would like to thank my supervisor Prof. Dr. M.E. Mazo and my daily supervisor L. Gharavi for their assistance during the writing of this thesis. I want to thank Dr. B. Shyrokau and A. Bertipaglia for their support in implementing the controller on the embedded hardware of the TUDelft autonomous vehicle.

Delft, University of Technology K. Trip
January 25, 2024

“If your car could travel at the speed of light, would your headlights work?”
— *Steven Wright*

Chapter 1

# Introduction

The level of autonomy in modern vehicles has significantly increased, leading to the introduction of autonomous vehicles on public roads. As autonomous vehicles appear on public roads, their performance in emergency situations becomes a strong focus [11]. In these scenarios, autonomous vehicles have the potential to significantly increase the safety of road vehicles [22].

Complex vehicle control techniques can react faster and exploit more control potential of the vehicle in comparison to human drivers [2, 29]. With faster reactions, evasive maneuvers can be executed earlier. By utilizing more control potential, more aggressive maneuvers can be performed.

CIC deals with the planning and control of aggressive evasive maneuvers in collision-imminent scenarios. The aim of CIC is to exploit more control potential of the vehicle with a controller that can be implemented in real-time.

Autonomous vehicles must adhere to traffic laws and aim to achieve the highest level of safety [34]. Safety in vehicle control often focuses on vehicle stability and collision avoidance [29]. Obstacles can be avoided faster if the vehicle can violate stability constraints temporarily [13]. CIC can prioritize different control objectives to find a balance between collision avoidance, stability, and path tracking that provides the highest level of safety [11].

Aggressive maneuvers may lead the vehicle to its handling limits [11], where the dynamics of the vehicle become nonlinear [21]. In CIC, nonlinear models are required for accurate path planning and control [38].

The complexity of CIC is influenced by the dynamic environment in which the vehicle operates [2]. Environmental disturbances, including the tire-road friction coefficient, can pose a robustness challenge to model-based control [12, 39]. Robustness can be improved using constraint tightening or parameter estimation [19, 39].

In vehicle control, Model Predictive Control (MPC) is commonly used as it can incorporate complex control strategies, safety constraints, and control objective prioritization [11, 32]. To account for the nonlinear vehicle dynamics near the handling limits, NMPC has been used to implement CIC [2, 38]. However, the added modeling power in NMPC comes at the expense of increased computational complexity [2].

State-of-the-art works on CIC apply a one-level NMPC controller that combines path re-planning and control in a single Optimal Control Problem (OCP) [2, 38]. This is achieved by incorporating information about the location of obstacles in objectives or constraints. Long prediction horizons ensure that current input signals can regard the future response of the vehicle [11]. Combined braking and steering control can be used to improve collision avoidance performance or stability [2, 15, 35].

The higher computational complexity of a one-level NMPC controller is a challenge to real-time control implementations [40]. In addition, a vehicle controller should use a nonlinear model and optimally distribute braking and steering inputs to improve the handling capabilities in emergency driving [5]. This large input space further increases the computational complexity.

This research aims to utilize techniques that improve the computational performance of NMPC, enabling higher modeling power to be used safely in real-time. Different methods are commonly applied to reduce the computational complexity of CIC. These include using different controller hierarchies [9], problem formulations, integrators, and prediction model fidelities [33].

Several factors can influence the computational complexity of NMPC. Determining which technique outperforms the other can be challenging as various control problem formulations, hardware platforms, and experimental conditions are used in literature. It thus remains an open question as to what combination of techniques is best suited to optimize the time complexity NMPC for CIC. Different techniques that reduce the computational complexity could enable higher modeling power in real-time control.

This research investigates which methods can reduce computational complexity without significantly impacting safety. Therefore, the research question is formalized as:

> "What combination of techniques is best suited to reduce the computational complexity of NMPC for planning and control of aggressive maneuvers for automated vehicles in emergency situations?"

This research builds upon the controller proposed by the work of Brown et al. [2]. This controller uses a one-level NMPC that includes longitudinal brake distribution and steering control. Collision avoidance is formulated as an objective on the distance between the vehicle and obstacles. This controller is implemented as a baseline for the CIC controller developed in this research. Brown et al. [2] executed the controller at a control frequency of 20Hz, requiring a low-level controller to execute the first 5 control inputs open-loop to obtain a real-time implementable controller with a control frequency of 100Hz. This work aims to improve this baseline by investigating various techniques to reduce computational complexity.

This research evaluates the model fidelity, control problem formulation, integration methods, and solving techniques based on their impacts on safety and ability to reduce the computational complexity for a one-level NMPC controller. First, state-of-the-art works on CIC are discussed in chapter 2. In chapter 3, different formulations of collision avoidance objectives are derived. After which, a model that achieves differential braking with a lower model fidelity is developed in chapter 4. The formulation of the optimization problem is discussed in chapter 5. The implementation of the controller and the effects of different solvers and moving block strategies are presented in chapter 6. The impacts of the reference trajectory, vehicle model, and formulation of collision avoidance objectives are evaluated against the baseline implementation [2] in chapter 7. The controller is then pushed to the limits, and its robustness against friction deviations, road curvature variations, and dynamic obstacle behavior is analyzed in chapter 8. Finally, in chapter 9, the controller is implemented on the embedded controller of the TUDelft autonomous vehicle. A full hardware-in-the-loop simulation is performed on embedded hardware, demonstrating the real-life applicability of the proposed controller and computational complexity reduction techniques.

# Chapter 2

# Background

CIC forces the vehicle to its handling limits, where the dynamics are highly nonlinear. Model-based control techniques must be able to capture these dynamics to make accurate predictions. Therefore, NMPC is commonly applied in CIC [2]. However, the computational complexity poses a burden to real-time control. To use CIC in real-time, state-of-the-art works have explored different controller hierarchies, prediction model fidelity, OCP formulation, and NMPC solving techniques to reduce the computational complexity. In addition, methods have been explored to improve the robustness of the CIC in the highly dynamic environment in which autonomous vehicles operate.

## 2-1 Controller hierarchy

Vehicle control is often divided into a modular framework of components, such as presented in Figure 2-1 [18]. This modularity gives rise to different control hierarchies, which can be used to address the problem of computational complexity in NMPC. The elements marked blue in Figure 2-1 form the focus of CIC.

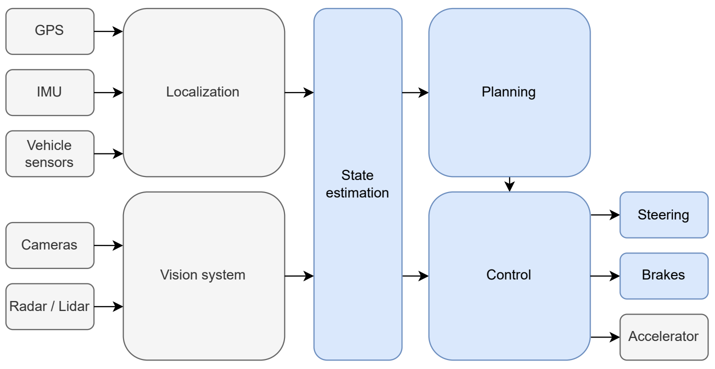


**Figure 2-1:** Vehicle control framework [18]

A common control hierarchy for CIC uses a separate path planner and tracking controller, splitting the problem into two levels [40]. In CIC, this structure is described as a two-level NMPC. This control structure is presented in Figure 2-2.

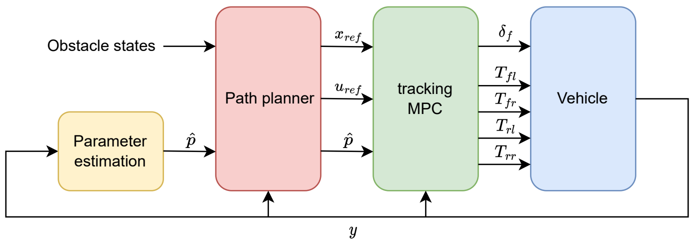


**Figure 2-2:** Two-level NMPC

A two-level hierarchy allows for the use of a simplified vehicle model for path planning [9]. Since the planning and control are divided into different problems, they can be solved at different frequencies [14]. As planning is considered a separate problem, the problem size of the tracking controller is reduced compared to a one-level hierarchy. This allows for higher fidelity tracking controllers [9]. Therefore, a two-level hierarchy can improve tracking performance and reduce computational complexity. However, it can not include guarantees and constraints on collision avoidance as the controller does not consider the location of obstacles. Thereby, collisions may occur due to tracking errors. Furthermore, a trajectory planned with a low-fidelity model can lead to sub-optimal track performance near the handling limits. The trajectory can be infeasible [26] or overly conservative [24] using different models for planning and control [20]. This can lead to frequent recalculation of trajectories [14]. For these reasons, state-of-the-art works on CIC apply a one-level control hierarchy [40, 20, 22].

A one-level hierarchical controller uses a single model that includes the position or dynamics of the object or environment, often expressed in spatial coordinates [26]. This allows for the formulation of collision avoidance as a single optimization problem. Combining planning and tracking removes the need for a separate path planner, allowing the same prediction model to be used for re-planning and control. This hierarchy is presented in Figure 2-3.

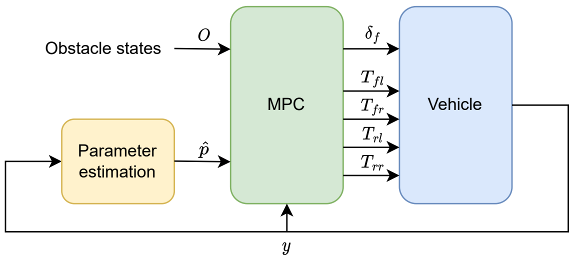


**Figure 2-3:** One-level NMPC

A one-level hierarchy solves a single optimization problem with the same prediction model for planning and control. This can lead to fewer infeasible trajectories and greater exploitation of the control potential [20], thus improving controller fidelity [40]. However, a one-level hierarchy has higher computational costs as integrating the path planning increases the problem size of the OCP. The added computational cost requires careful control problem formulation and the use of methods that can reduce the computation to employ one-level NMPC for CIC in real-time [40, 2].

## 2-2 Model fidelity

State-of-the-art works on CIC achieve real-time control by employing a Single Track (ST) vehicle model [40, 11]. More control potential can be exploited from the vehicle using differential braking [15]. The ST vehicle model generally does not allow for differential braking but has been adapted to include simplified forms of brake distribution [2]. Because of the increased computational complexity, Double Track (DT) models are used less in real-time CIC control [32].

## 2-3 OCP formulation

Stability and collision avoidance objectives are often chosen to dominate the OCP [2, 11]. These objectives can be prioritized by allowing stability constraint violations before collision avoidance violations [11]. Slack variables have been used to implement allowable constraint violations [11] at the expense of increased problem size. For collision avoidance, constraints or objectives can be formulated based on the distance between the vehicle and obstacles [2]. The problem with constraints is that they can reduce the number of feasible solutions and do not always provide a sense of direction for the solver. Therefore, state-of-the-art methods enforce collision avoidance using an objective [2].

## 2-4 NMPC solving techniques

Different methods exist that can improve the computational complexity without significantly impacting performance. Several techniques focus on reducing the prediction horizon [11, 8, 20, 40]. As the CIC controller must plan sufficiently far ahead to achieve successful collision avoidance, integrators are often carefully selected [2, 40]. Time steps can be increased using more accurate integration methods [40]. However, Runga Kutta methods require more evaluations per time instance and are generally more complex than forward Euler methods [40]. A balance must be found between the timestep size and the integrator complexity [2].

Variable timesteps have been applied in Linear Time Varying (LTV) MPC for CIC to reduce the prediction horizon while conserving performance [11]. Cascaded models throughout the prediction horizon [20] likely require more tuning and careful formulation of constraints and objectives. Utilizing different control and replanning frequencies entails that some control actions are derived using open-loop control. As model inaccuracies are inherent in CIC, reliance on open-loop control inputs can pose risks in emergencies.

Other methods employed in the literature on NMPC that reduce the computational complexity are often solver-specific and can be challenging to compare directly. Generally, it can be hard to determine which technique outperforms the other. To a certain degree, each paper solves a different control problem on different hardware platforms with different experimental conditions. This makes it difficult to specify exactly what the solution time benefits of each method are or to compare their performance directly. The time for the nonlinear optimization problem to converge to a solution depends on many factors, which can be difficult to analyze independently. Thus, it remains an open question as to what technique or combination of techniques is best suited to optimize the time complexity NMPC for CIC.

## 2-5 Robustness

To improve robustness, tube-based methods and constraint tightening have been applied to MPC controllers [12, 19]. These methods generally add computational complexity, although more efficient methods have been developed [19]. Robustness can be improved by adding online friction estimation in parallel to solving the OCP [39]. Since friction imposes robustness issues in collision avoidance, it has been added to CIC controllers [39]. The Unscented Kalman Filter (UKF) is well suited for tire-road friction estimation because it is computationally efficient since it avoids the computation of the Jacobian [16]. Furthermore, UKFs can have a lower settling time than Recursive Least Squares (RLS) estimators [39]. The estimation performance of UKF can be compromised by Gaussian noise assumptions [39]. However, low settling times make it well suited to real-time implementations in CIC [39]. Online estimators do not directly affect the computational complexity of the NMPC problem and might aid in convergence by improving estimates.

By combining these techniques, this research aims to improve performance and achieve real-time control for the baseline controller [2]. The baseline controller uses a nonlinear ST vehicle model as a prediction model in a one-level NMPC. This controller configuration can better exploit the vehicle's control potential, increasing safety. However, careful OCP formulation and selection of solving techniques are required to reduce the computational complexity.

# Chapter 3

# Collision avoidance

In this chapter, different formulations of collision avoidance are derived. Collision avoidance in CIC can be divided into road boundary and obstacle collision avoidance. Collision avoidance can be achieved by enforcing constraints or penalizing safe-distance violations in the objective function of the OCP. As collision avoidance has the highest priority in CIC, the formulation of these constraints or objectives can significantly impact the controller's performance. Therefore, these formulations must be carefully considered to reduce computational complexity and excessive conservatism while ensuring accuracy and safety.

Collision avoidance is assumed to be a problem in which obstacles and road boundaries must be avoided. Obstacle collision avoidance is complex due to the wide range of obstacle geometries and dynamic or stochastic behavior. Road boundaries can be difficult to formulate due to changing widths and curvatures of the road. The vehicle's dynamics can be projected onto a trajectory that follows the road curvature to simplify the modeling of road boundaries. This leads to the formulation of the spatial dynamics commonly used in vehicle control [26, 10].

## 3-1 Spatial dynamics

The spatial dynamics are formed by projecting the dynamics of the vehicle $(\dot{x}, \dot{y}, \dot{\psi})$ onto a road-fixed trajectory $\sigma$. Here, $\dot{x}$ and $\dot{y}$ are the longitudinal and lateral velocities of the vehicle in the body fixed frame $\mathcal{F}_b$ and $\dot{\psi}$ is the angular velocity around the yaw axis.

The trajectory $\sigma$ can be represented by a lane boundary or center of the road [10]. This projection creates three states $(e_\psi, s, e_y)$. The state $e_\psi$ is the angle between the vehicle velocity and the road direction, $s$ is the distance along the trajectory $\sigma$, and $e_y$ is the perpendicular distance from the trajectory $\sigma$. Graphically, this is represented in Figure 3-1.

The relations in Equation 3-1 hold for the projection in Figure 3-1 [10].

$$e_y = \left\| [X_v, Y_v]^T - [X_\sigma, Y_\sigma]^T \right\|_2, \quad e_\psi = \psi - \psi_\sigma \tag{3-1}$$

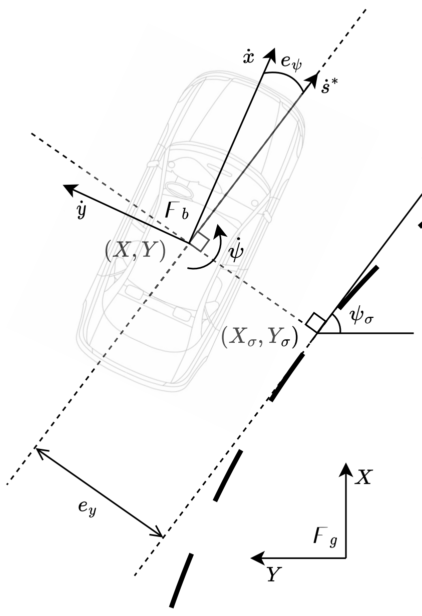


**Figure 3-1:** Spatial coordinate transformation [10]

Here, $(X_v, Y_v)$ and $(X_\sigma, Y_\sigma)$ are the coordinates in a global frame $\mathcal{F}_g$. $\psi$ and $\psi_\sigma$ are the yaw angles with respect to the global frame $\mathcal{F}_g$.

The equations of motion of the spatial coordinates $(e_\psi, s, e_y)$ can be expressed using the vehicle states $(\dot{x}, \dot{y}, \dot{\psi})$ and the road curvature $\kappa(s)$. The dynamics of $e_\psi$, $s$, and $e_y$ can be expressed as shown in Equation 3-2 [2].

$$\begin{aligned}\dot{e}_\psi &= \dot{\psi} - \kappa(s)\dot{s} \\ \dot{s} &= \frac{1}{1-\kappa(s)e_y}\left(\dot{x}\cos\left(e_\psi\right) - \dot{y}\sin\left(e_\psi\right)\right) \\ \dot{e}_y &= \dot{x}\sin(e_\psi) + \dot{y}\cos(e_\psi)\end{aligned} \tag{3-2}$$

As the curvature can change along the road, it can be expressed as a function of the distance along the trajectory $s$. If the vehicle's location is known, $\kappa(s)$ could be assumed available from maps, estimated from visual systems on the vehicle, or derived from cartesian road coordinates as will be discussed in chapter 8.

The lateral distance $e_y$ and rotation $e_\psi$ with respect to the reference trajectory $\sigma$ can be included in the OCP. Road boundary collision avoidance can be formulated as objectives or constraints on $e_y$.

As an extension, the vehicle dynamics $(\dot{x}, \dot{y}, \dot{\psi})$ can be divided by $\dot{s}$ to create a spatial-based model that is distance variant instead of time-variant [10]. However, in real-time systems, it can be impractical to have a model that is not time-dependent. For example, the prediction horizon of a spatial-based model is some distance that varies with the vehicle velocity. Therefore, a time-dependent model is developed, presented in chapter 4.

## 3-2 Objective and constraint-based collision avoidance

The spatial dynamics can be used to formulate objectives or constraints, allowing the controller to avoid collision with obstacles and road barriers.

Constraint-based collision avoidance limits the state-space of the vehicle to the drivable space confined between the road boundaries and obstacles. Generally, these constraints can be formulated as distance-varying lateral bounds on the projected state $e_y$. Since the road geometry can change, these bounds on $e_y$ can be defined as a function of the distance $s$ [26]. This forms the limits $e_{yl}(s)$ and $e_{yr}(s)$. The limits $e_{yl}(s)$ and $e_{yr}(s)$ specify the perpendicular distance from the left and right road boundary to the trajectory, respectively. These constraints can be extended to include obstacles. The limits $e_{yl}(s)$ and $e_{yr}(s)$ can be selected based on the minimum value of the obstacle or round boundary, defining the drivable space using a form of corridor planning [26]. This is exemplified in Figure 3-2.

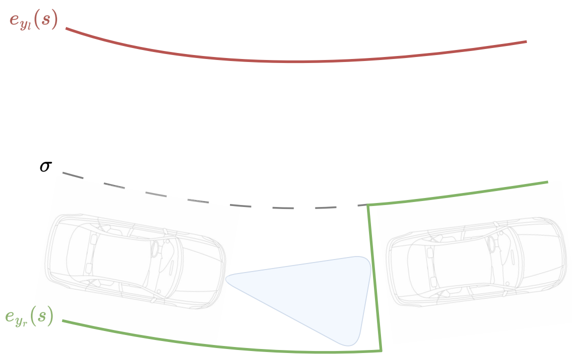


**Figure 3-2:** Spatial varying collision avoidance bounds

### 3-2-1 Limitations of constraint-based collision avoidance

Constraint-based collision avoidance has several limitations. Constraints of the obstacle boundaries can only be specified when a decision has been made on which side the obstacle will be evaded. This can be trivial in cases such as depicted in Figure 3-2, where the left side of the obstacle is included in the right bound $e_{yr}(s)$ to initiate an evasive maneuver to the left. However, in some cases, choosing the direction of the evasive maneuver is less trivial.

Furthermore, constraint-based collision avoidance can have undesirable effects in emergency collision avoidance, as constraints limit the feasible space of the OCP. In CIC, a large set of trajectories can be infeasible depending on the difficulty of the collision avoidance problem. When a current solution is infeasible, it does not always give the solver a sense of direction. Furthermore, it remains a question of what should be done when no feasible solution is found within the periodic execution of the controller.

Objective-based collision avoidance alleviates these issues to some extent. Including an objective that maximizes the distance to an obstacle or road boundary until a safe distance is achieved can supply crucial information in the form of a Jacobian and Hessian to the solver. This can be beneficial in real-time CIC to create faster sub-optimal trajectories. Therefore, collision avoidance is formulated as an objective in the CIC controller, similar to the baseline implementation [2].

## 3-3 Collision avoidance formulation

Objective-based collision avoidance aims to maximize the distance $d$ between the vehicle and obstacle or road boundaries. This can be achieved by defining some safe distance boundary $d_{\min}$ and formulating the distance cost in the objective function as presented in Equation 3-3 [2].

$$c_{dist} = \begin{cases} Q_{dist}(d - d_{\min})^2 & \text{if } d < d_{\min} \\ 0 & \text{otherwise} \end{cases} \tag{3-3}$$

By including multiple distances, an Artificial Potential Field (APF) is created. An APF is a superposition of potential functions that can be minimized to find an optimal trajectory [37]. APFs have commonly been used in path planning in highly dynamic environments [23], such as vehicle control. A superposition of multiple distances costs $c_{dist}$ for multiple obstacles and road boundaries can be seen as a APFs that can directly be minimized in an OCP. This achieves simultaneous planning and control, common for one-level hierarchical control. The distance function is scaled by a parameter $Q_{dist}$.

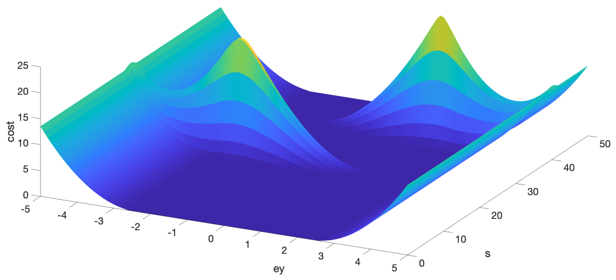


**Figure 3-3:** APF collision avoidance

A superposition of four distance cost functions for two obstacles and two road boundaries leads to the APF presented in Figure 3-3. For this APF, the driveable space of the vehicle is presented by regions with low-cost values, and the regions that should be avoided include a higher cost. A benefit of this method is that it defines a quadratic cost that provides a sense of direction to a solver of the OCP in the form of a Jacobian and a Hessian. When these costs dominate the cost function of the OCP, fast, locally optimal solutions can be generated without extensively limiting feasibility compared to constraint-based collision avoidance.

The computational complexity of objective-based collision avoidance correlates to the formulation of the distance $d$. For this reason, different methods of formulating this instance are explored and compared on accuracy, conservatives, and computational complexity.

## 3-4 Distance formulations

In objective-based collision avoidance, different distance formulations can describe the distance between the vehicle, obstacles, and road boundaries. These distance formulations can be created using different vehicle geometry formulations, as the vehicle's geometry does not change.

The minimum distance between the vehicle and the left and right road boundaries $d_l$ and $d_r$ can be defined as presented in Equation 3-4 [2]. Each vehicle geometry formulation has a different expression of the maximum lateral coordinate $e_{y_{\max}}$. The location of the left and right road boundary is expressed as $rb_l$ and $rb_r$ respectively.

$$d_l = rb_l - e_y - e_{y_{\max}}, \quad d_r = e_y - e_{y_{\max}} - rb_r \tag{3-4}$$

To formulate the distance between the vehicle and an obstacle $d_o$, it is considered that the potential obstacles can have a varied range of different geometries. Therefore, describing an obstacle by a set of circles of different radii is opted for as it can describe any potential geometry [2]. For each circle fitted to an obstacle $r_o$, the distance $d_o$ is derived as presented in Equation 3-5 [2]. Here, $L$ is the center distance between the obstacle and the geometric shape fitted to the vehicle. The distance from the center to the edge of the geometric shape fitted to the vehicle in the direction of the obstacle is described by $d_v$, as illustrated in Figure 3-4.

$$d_o = L - d_v - r_o \tag{3-5}$$

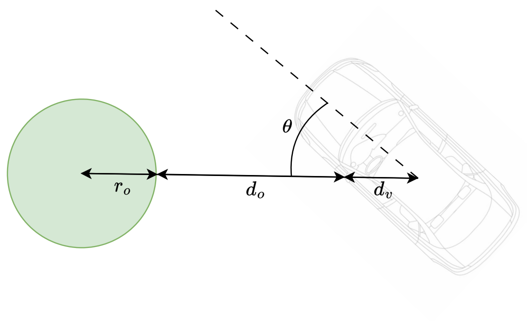


**Figure 3-4:** Derivation of $d_o$

The distances can be expressed in different coordinate frames, including the spatial coordinates of the road frame $(s, e_y)$ and the Cartesian vehicle body fixed frame $(x, y)$ [28]. Expressing convex geometric shapes in the spatial coordinate frame leads to a non-convex transformation [28]. The downside of using cartesian coordinates is that the states $(x, y, \psi)$ from the vehicle model must be included in the control problem, which can increase the problem size as the states states $(e_\psi, s, e_y)$ are required to describe road boundaries.

To avoid expressing non-convex geometric shapes using spatial coordinates, the distances $L$, $d_v$, and $r_o$ are defined in the Cartesian frame but are expressed as functions of $(e_\psi, s, e_y)$ and the road curvature $\kappa(s)$. As is done in the baseline [2].

Using the linearized signed distance $L$ derived in the baseline [2], $\theta$ is derived so that different geometric shapes can be fitted to the vehicle. The baseline method fits multiple circles to the vehicle [2]. As an extension, an ellipsoid and a rectangle are used to represent the geometry of the vehicle. The angle $\theta$ and the distance $L$ to the obstacle can be derived by assuming that the vehicle is at location $(s_1, e_1)$ and that the obstacle is at location $(s_2, e_2)$. This is depicted in Figure 3-5.

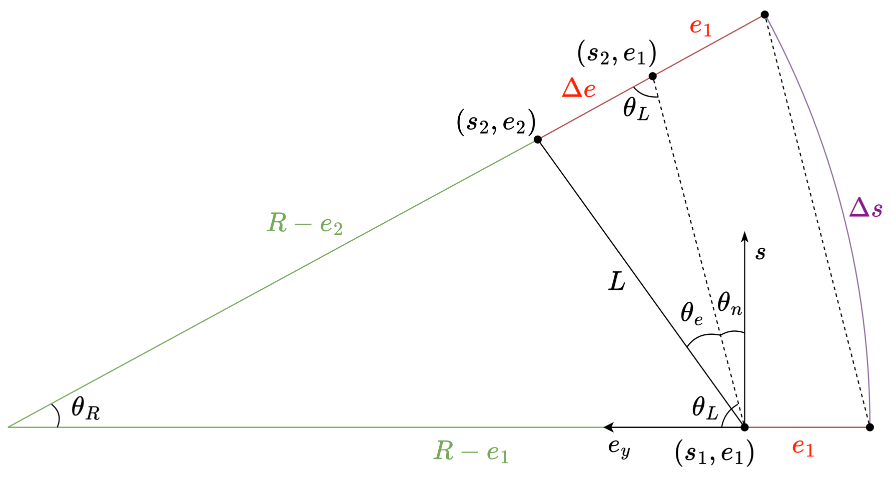


**Figure 3-5:** Derivation of $L$ and $\theta$ [2]

Here $\Delta e = e_2 - e_1$ and $\Delta s = s_2 - s_1$. Since the distance is only added to the cost function when the vehicle is in relative proximity to the obstacle, $\Delta s$ is relatively small, and $\kappa(s) = \kappa$ is assumed constant over $\Delta s$. This leads to the approximations of $\theta_R$ and $L$ presented in Equation 3-6 [2].

$$\theta_R \approx \kappa \Delta s, \quad L \approx \sqrt{(1 - e_1\kappa)(1 - e_2\kappa)(s_2 - s_1)^2 + (e_2 - e_1)^2} \tag{3-6}$$

To find an expression for the angle between the vehicle and the obstacle $\theta$, an isosceles triangle can be created to form the expression of $\theta_L$ and $\theta_n$ in Equation 3-7.

$$\theta_L = \frac{\pi - \kappa\Delta s}{2}, \quad \theta_n = \frac{\kappa \Delta s}{2} \tag{3-7}$$

The Sine rule can form an expression of $\theta_e$, presented in Equation 3-8. To describe the angle correctly after the obstacle is passed, a sign function for $\Delta s$ is added. To reduce the complexity of this expression, an approximation can be used by assuming that the road curvature is relatively small and the vehicle is close to the obstacle. The accuracy of this approximation is very high near the obstacle, and the error is generally below 5 degrees, as presented in Appendix B.

$$\theta_e = \text{sign}(\Delta s) \arcsin\left(\frac{\Delta e}{L} \sin(\theta_L)\right) \approx \arctan\left(\frac{\Delta e}{\Delta s}\right) \tag{3-8}$$

As the vehicles rotated with respect to the road frame $(s, e_y)$ with an angle $e_\psi$, the total angle between the vehicle and the obstacle $\theta$ can be expressed as presented in Equation 3-9.

$$\theta = \theta_e + \theta_n - e_\psi \approx \arctan\left(\frac{\Delta e}{\Delta s}\right) + \frac{\kappa \Delta s}{2} - e_\psi \tag{3-9}$$

### 3-4-1 Multiple circle distance

In the baseline, multiple circles can be projected on the agent vehicle and obstacles to describe the distance accurately and reduce conservativeness [2]. Graphically, this is presented in Figure 3-6.

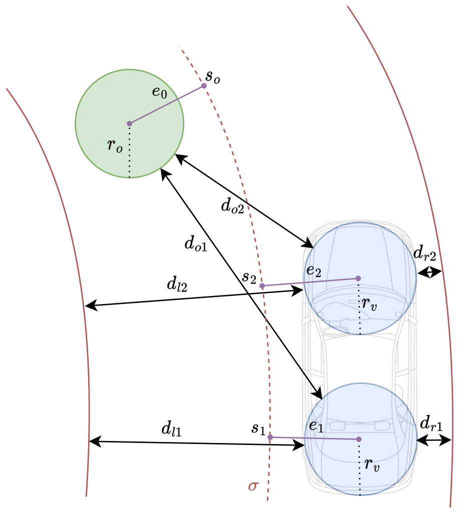


**Figure 3-6:** Multiple circle collision model

### Obstacle distance

In each obstacle distance $d_o$, $d_v$ equals the radius $r_v$ of one of the circles fitted to the vehicle. For the scenario depicted in Figure 3-6, the obstacle distances are expressed in Equation 3-10 [2]. Here, $L_1$ and $L_2$ are the center distances between the obstacle and both circles fitted to the vehicle.

$$d_{o1} = L_1 - r_v - r_o, \quad d_{o2} = L_2 - r_v - r_o \tag{3-10}$$

### Road boundary distance

For each circle fitted to the vehicle, $e_{y_{\max}} = r_v$ [2]. For the situation depicted in Figure 3-6, the road boundary distances are expressed in Equation 3-11 [2].

$$\begin{aligned} d_{l1} &= rb_l - e_1 - r_v, \quad d_{r1} = e_1 - r_v - rb_r \\ d_{l2} &= rb_l - e_2 - r_v, \quad d_{r2} = e_2 - r_v - rb_r \end{aligned} \tag{3-11}$$

This method is more accurate and less conservative than using a single circle around the vehicle and each obstacle [2], as it explicitly captures the vehicle's rotation. However, the multiple circle method has a complexity of $O(2 \cdot n_d \cdot n_o)$ distances, where $n_d$ are the number of circles fitted to the vehicle and $n_o$ are the number of obstacles in the collision avoidance scenario. Using two circles leaves a significant region of the vehicle out of the collision avoidance expression, which can pose safety risks.

## 3-4-2 Elliptical distance

An ellipse can be used to fit a single geometry to the vehicle. Due to a vehicle's shape, an ellipse can describe the geometry of a vehicle relatively well.

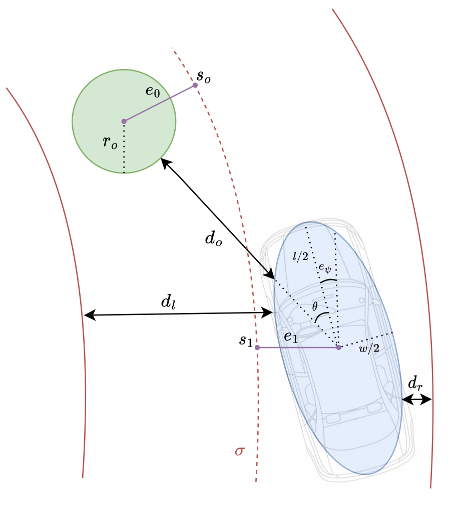


**Figure 3-7:** Elliptical distance approximation

## Obstacle distance

The distance between the vehicle and the obstacle can be expressed using the polar expression of Cartesian coordinates of an ellipse. These expressions can be used to derive $d_v$ as presented in Equation 3-12.

$$\begin{aligned} d_e &= \sqrt{x_e^2 + y_e^2}, \quad x_e = \frac{l}{2}\sin\theta, \quad y_e = \frac{w}{2}\cos\theta \\ &= \frac{1}{2}\sqrt{l^2\sin^2\theta + w^2\cos^2\theta^2} \\ &= \frac{1}{2}\sqrt{(l^2 - w^2)\sin^2\theta + w^2} \end{aligned} \tag{3-12}$$

Using the expression of $d_v$, distance $d_o$ to each obstacle is presented in Equation 3-13.

$$d_o = L - \frac{1}{2}\sqrt{(l^2 - w^2)\sin^2\theta + w^2} - r_o \tag{3-13}$$

## Road boundary distance

The minimum distance between the ellipse and the road boundaries can be found by finding the extrema points of the ellipse along the $e_y$ axis of the road frame. The ellipse is defined in the body-fixed frame of the vehicle in $(x, y)$. In the road frame $(s, e_y)$ the ellipse is centered at $(s_1, e_1)$ and is rotated by $e_\psi$. The extrema lateral points of a rotated ellipse centered at the origin are derived in Appendix B and expressed in Equation 3-14.

$$\begin{aligned} e_{y_{\max}} &= \frac{1}{2}\sqrt{(w\sin e_\psi)^2 + (l\cos e_\psi)^2} \\ &= \frac{1}{2}\sqrt{(w^2 - l^2)\sin^2 e_\psi + l^2} \end{aligned} \tag{3-14}$$

Using this expression of $e_{y_{\max}}$, the distance to the left and right road boundaries $d_l$ and $d_r$ can be defined as described by Equation 3-15.

$$
\begin{aligned}
d_l &= rb_l - e_y - \frac{1}{2}\sqrt{(w^2 - l^2)\sin^2 e_\psi + l^2} \\
d_r &= e_y - \frac{1}{2}\sqrt{(w^2 - l^2)\sin^2 e_\psi + l^2} - rb_r
\end{aligned}
\tag{3-15}
$$

This method is also more accurate and less conservative than using a single circle around the vehicle and each obstacle, as it also explicitly captures the vehicle's rotation. Compared to the multiple circle method, the time complexity of the distance calculations is reduced to $O(2 \cdot n_o)$, where $n_o$ are the number of obstacles in the collision avoidance scenario. This is because a single geometric shape is fitted to the vehicle instead of a set of circles. A limitation of the ellipse is that it does not accurately capture the corners of the vehicle, which might be critical in collision avoidance.

### 3-4-3 Rectangular distance

Using the approximation of $\theta$, a rectangle can be used to describe the vehicle's geometry. A rectangular or polytopic expression accurately represents the vehicle's geometry [2], as vehicles tend to be more angular than ellipsoidal or circular.

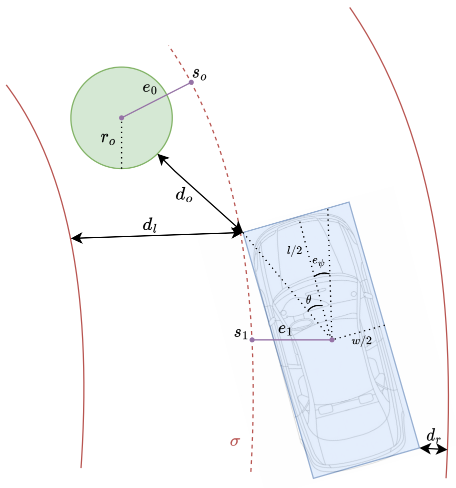


**Figure 3-8:** Rectangular distance approximation

#### Obstacle distance

The distance to an obstacle $d_v$ can be expressed in coordinates $(s, e_y)$ using the previously derived approximation for $\theta$, as expressed in Equation 3-16.

$$
\begin{aligned}
\theta_C &= \arctan\left(\frac{l}{w}\right) \\
\theta_R &= |\mathrm{mod}(\theta, \pi) - \frac{\pi}{2}| \approx \frac{\pi}{4}(\cos\theta + 1) \\
d_v &= \min\left(\frac{w}{2\cos\theta_R}, \frac{l}{2\sin\theta_R}\right)
\end{aligned}
\tag{3-16}
$$

Here, $d_v$ is a piece-wise defined function that describes the distance from the center to the edge of the rectangle fitted to the vehicle for the angle $\theta$ between the vehicle and obstacle.

#### Road boundary distance

Another benefit of using a rectangular geometry is that the lateral extrema point is always situated at a corner of the rectangle.

The rectangle is defined in the body-fixed frame of the vehicle in $(x, y)$. In the road frame $(s, e_y)$ the rectangle is centred at $(s_1, e_1)$ and is rotated by $e_\psi$. Therefore, the lateral extrema occurs at the rectangle's corners, rotated by $e_\psi$. This leads to the expression of $e_{y_{\max}}$ in Equation 3-17.

$$\begin{aligned} \theta_E &= |\text{mod}(\theta, e_\psi) - \frac{\pi}{2}| \approx \frac{\pi}{4}\left(\cos e_\psi + 1\right) \\ e_{y_{\max}} &= \frac{\sqrt{l^2 + w^2}}{2} \cdot \sin\left(\theta_C + \theta_E\right) \end{aligned} \tag{3-17}$$

For the rectangle, the distance to the left and right road boundaries $d_l$ and $d_r$ can be defined as:

$$d_l = rb_l - e_y - \frac{\sqrt{l^2 + w^2}}{2} \cdot \sin\left(\theta_C + \theta_E\right) \tag{3-18}$$

$$d_r = e_y - \frac{\sqrt{l^2 + w^2}}{2} \cdot \sin\left(\theta_C + \theta_E\right) - rb_r \tag{3-19}$$

### 3-4-4 Dynamic obstacle behavior

If it is possible to estimate the velocities of an obstacle, they can be assumed constant throughout the prediction horizon and updated based on the controller's frequency or the estimation algorithm. A set of obstacle locations $L_o$ with size $[2 \times N]$ can be provided to the controller for each obstacle $o$.

Using the initial estimates: $\hat{s}_o(0), \hat{e}_o(0)$ for the location and $\dot{\hat{s}}_o, \dot{\hat{e}}_o$ for the velocity of the obstacle in the road frame $(s, e_y)$, the set $L_o$ can be explicitly defined:

$$L_o(i) = [\hat{s}_o(i-1) + \dot{\hat{s}}_o \cdot dt, \hat{e}_o(i-1) + \dot{\hat{e}}_o \cdot dt]^T, \quad \forall i = \{1, \cdots, N\} \tag{3-20}$$

## 3-5 Distance formulation comparison

The distance formulations can be compared based on the computation times and the distance cost function values $c_{dist}$. These can be found by evaluating the different distance formulations on an uncontrolled exemplary trajectory. This trajectory ensures the vehicle gets close to two obstacles and both road boundaries, as presented in Figure 3-9.

The distance formulations result in the distance cost functions $c_{dist}$ presented in Figure 3-10, where $Q_{dist} = 1$. The rectangular representation is the most accurate as it is the closest to the vehicle geometry [2]. However, the rectangular distance function leads to a non-smooth

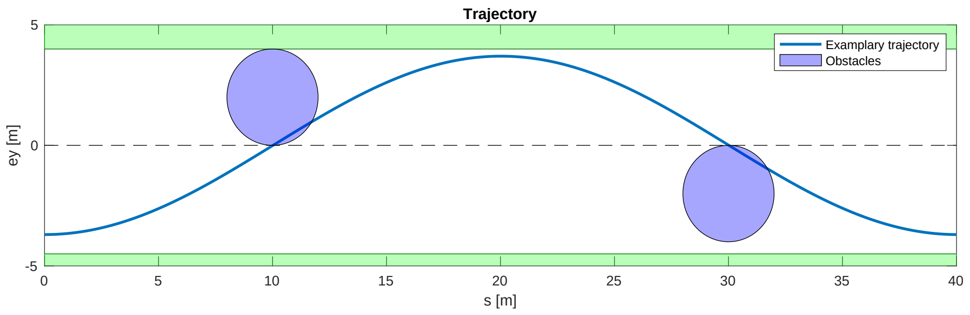


**Figure 3-9:** Uncontrolled trajectory simulation

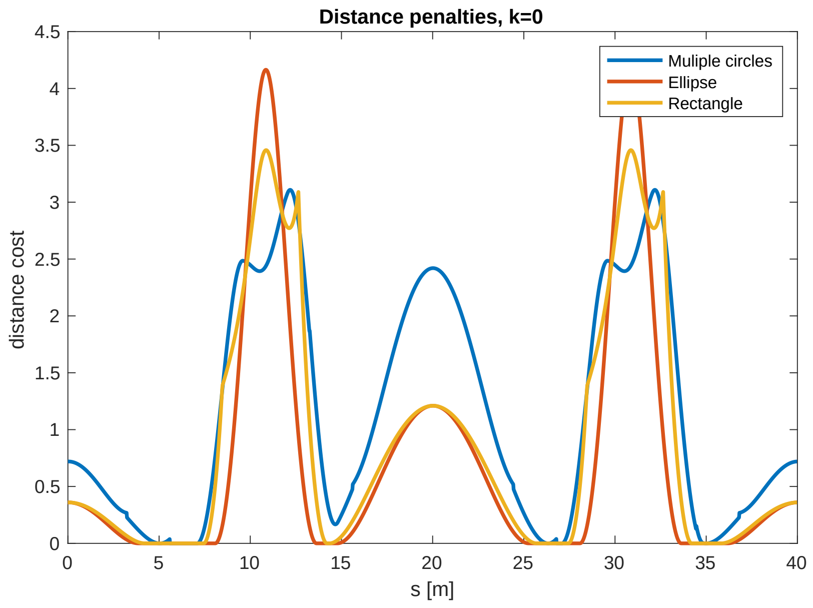


**Figure 3-10:** Cost function values for a specified trajectory

cost function, which could be a challenge for solvers as the Jacobian and Hessian can be hard to define near these points.

From Figure 3-10, it can be observed that the multiple-circle method produces wider cost function peaks with dips near the middle of the vehicle near the obstacles. This can be explained by the location of the two circles fitted to the vehicle, as fitting a circle on the front and rear axles is limited in representing the geometry near the vehicle's center. Furthermore, the costs near the road boundaries are greater because both circles are near the edge. The multiple-circle formulation can create some discrepancies compared to the rectangular distance formulation.

The ellipsoidal formulation has narrower peaks near the obstacles and a lower cost function near the road boundaries. This can be explained by the modeling errors created by fitting an ellipse to the vehicle geometry. An ellipse does not capture the corners of the vehicle accurately if the longitudinal and lateral axis dimensions are defined as the length and width of the vehicle.

The computation times are presented in Table 3-1. The ellipse is the fastest, followed by the rectangle and the multiple-circle method. The rectangular formulation can be slower due to conditional formulation and additional angular expressions. The multiple-circle method has to evaluate more distances, proportional to the number of circles fitted to the vehicle, creating higher computational complexity.

| | |
|---|---|
| Multiple circles [2] | 2.31 $\mu$s |
| Ellipsoid | 1.61 $\mu$s |
| Rectangle | 2.01 $\mu$s |

**Table 3-1:** Distance formulation computation times

The effects of these distance models are on the computation time of the closed-loop CIC controller, which is evaluated in chapter 7.

# Chapter 4

# Vehicle prediction model

This chapter derives a low-fidelity vehicle model that can apply differential braking. Different vehicle models can be used to model the dynamics of a vehicle. The choice of a model can depend on many factors, including the control objective, required prediction accuracy, and computational complexity. Prediction model fidelity can significantly impact the computational performance of model-based control techniques such as NMPC. An ideal prediction model is computationally efficient and captures the dynamics of the plant well enough to achieve the perceived control objective. In CIC, the objective is to apply the full control potential of the vehicle in emergency collision avoidance scenarios. More control potential can be applied by utilizing differential braking [15].

Torque vectoring control can also be applied to improve vehicle stability [31]. However, it has complex powertrain requirements as it needs the ability to control the torque output at each wheel. This has been possible in some modern electric vehicles but is rare in production vehicles. However, since each wheel has a separate brake actuator and electronic braking systems have become very common, differential braking can be possible in CIC.

In the domain of emergency collision avoidance in road vehicles, ST and DT models with three Degrees of Freedom (DoF) are most common [32]. Differential braking can not be applied to a conventional ST model as each wheel is not modeled separately. Although it can be applied to a Double Track (DT) model, more tire function evaluations increase computational complexity. By combining the dynamical expressions of a ST and DT model, a low-fidelity vehicle model is derived that allows for differential braking. This model aims to apply differential braking with fewer tire function evaluations than the DT model.

The vehicle model derived in this chapter combines the computational efficiency of the ST model with the higher modeling power of the DT model to achieve efficient differential braking. This model is referred to as the CIC vehicle model and can be seen as an evolution of the prediction model derived in the baseline [2].

The dynamics of the ST and DT models are first presented to derive this model. These models commonly have three degrees of freedom around the vehicle's body-fixed lateral, longitudinal, and yaw axes [32]. The ST and DT models have different expressions for the lateral acceleration $\ddot{y}$, longitudinal acceleration $\ddot{x}$, and the angular yaw acceleration $\ddot{\psi}$.

The ST model is presented in Figure 4-1. This model resembles a bicycle with separate front and rear tire dynamics. This is done to simplify the model, reducing the computational complexity while still describing the most significant effects of the steering angle on the vehicle dynamics. The front axle of the ST model has tire forces $F_{x_f}, F_{y_f}$ and the rear axle has tire

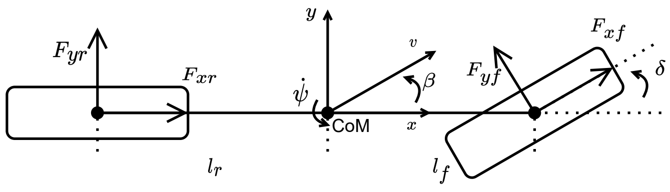


**Figure 4-1:** ST vehicle model

forces $F_{x_r}, F_{y_r}$. The equations of motion for the ST model are described in Equation 4-1 [2].

$$\begin{aligned} m\ddot{x} &= m\dot{y}\dot{\psi} + F_{x_f}\cos\delta - F_{y_f}\sin\delta + F_{x_r} - F_{drag} \\ m\ddot{y} &= -m\dot{x}\dot{\psi} + F_{x_f}\sin\delta + F_{y_f}\cos\delta + F_{y_r} \\ I_z\ddot{\psi} &= l_f(F_{x_f}\sin\delta + F_{y_f}\cos\delta) - l_r F_{y_r} \end{aligned} \tag{4-1}$$

Here, $F_{i,j}$ denote the tire forces, $v$ is the velocity of the vehicle with angle $\beta$ to the $x$-axis, $\delta$ is the front steering angle, $\psi$ is the global yaw angle, and $\dot{\psi}$ is the angular velocity of the vehicle. This model includes the following parameters: the distance between the Center of Mass (CoM) and the front axle $l_f$ and rear axle $l_r$, the inertia around the $z$-axis $I_z$, and the mass of the vehicle $m$.

The steady-state normal load at each axle can then be expressed using the sum of moments around the $y$-axis, gravitational constant $g$, and the height of the CoM of the vehicle $h$ [2].

$$F_{zf} = \frac{l_r mg - hm\ddot{x}}{l_f + l_r}, \quad F_{zr} = \frac{l_r mg + hm\ddot{x}}{l_f + l_r} \tag{4-2}$$

The DT model is presented in Figure 4-2. In a DT model, the tire forces of each tire are modeled separately. This improves the accuracy of the model but also increases the computational complexity. This is due to the increased complexity of the expressions and the increase in the number of evaluations of the tire model. The DT model further enables differential braking between the left and right sides of the vehicle as the effect of each tire force is modeled.

In the double-track model, the forces at each wheel are considered separately. Therefore, the tire forces are denoted $F_{i,j,k}$, with $i = (x, y)$ for the direction, $j = (f, r)$ for the axle, and $k = (l, r)$ for the side of the vehicle that the tire is on. This leads to the expression for the vehicle dynamics presented in Equation 4-3 [33].

$$\begin{aligned} m\ddot{x} &= m\dot{y}\dot{\psi} + (F_{xfl} + F_{xfr})\cos\delta - (F_{yfl} + F_{yfr})\sin\delta + F_{xrl} + F_{xrr} - F_{drag} \\ m\ddot{y} &= -m\dot{x}\dot{\psi} + (F_{yfl} + F_{yfr})\cos\delta + (F_{xfl} + F_{xrl})\sin\delta + F_{yrl} + F_{yrr} \\ I_z\ddot{\psi} &= (F_{yfl}\sin\delta - F_{xfl}\cos\delta + F_{xfr}\cos\delta - F_{yfr}\sin\delta + F_{xrr} - F_{xrl})\frac{w}{2} - \\ &\quad (F_{xfl}\sin\delta + F_{yfl}\cos\delta + F_{xfr}\sin\delta + F_{yfr}\cos\delta)\, l_f + (F_{yrr} + F_{yrl})\, l_r \end{aligned} \tag{4-3}$$

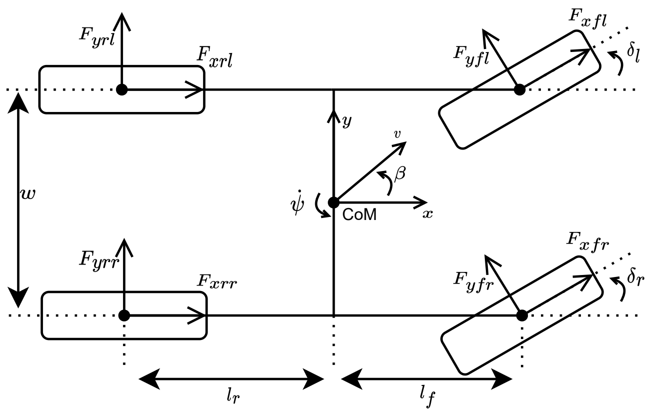


**Figure 4-2:** Double track vehicle model

Similar to the ST model, the steady state normal load at each tire can be expressed using the sum of moments around the $y$-axis and the width of the vehicle $w$ [6].

$$\begin{aligned} F_{zfl} &= \frac{l_r mg - hm\ddot{x}}{2\,(l_f + l_r)} - \frac{l_r hm\ddot{y}}{(l_f + l_r)\,w}, \quad F_{zfr} = \frac{l_r mg - hm\ddot{x}}{2\,(l_f + l_r)} + \frac{l_r hm\ddot{y}}{(l_f + l_r)\,w} \\ F_{zrl} &= \frac{l_f mg + hm\ddot{x}}{2\,(l_f + l_r)} - \frac{l_f hm\ddot{y}}{(l_f + l_r)\,w}, \quad F_{zrr} = \frac{l_f mg + hm\ddot{x}}{2\,(l_f + l_r)} + \frac{l_f hm\ddot{y}}{(l_f + l_r)\,w} \end{aligned} \tag{4-4}$$

In both models, the drag force $F_{drag}$ can be approximated by $F_{drag} = Cd_0 + Cd_1\dot{x}$, where $Cd_0$ and $Cd_0$ scale the linear drag force [2].

Using assumptions on the operating conditions, steering system, and tire model, it can be shown that the yaw dynamics is the main difference between the ST and DT dynamics. These assumptions include:

**Assumption 1.** *The road surface is flat.*

**Assumption 2.** *All wheels have contact with the road surface at all times.*

**Assumption 3.** *The steering angle of both front wheels is equal.*

**Assumption 4.** *The lateral tire force can be expressed as:* $F_y = \mu F_z \cdot g(\dot{x}, \dot{y}, \dot{\psi}, \delta)$.

**Assumption 5.** *The longitudinal force* $F_x$ *on each axle* $(f, r)$ *is equal to the sum of each wheel's longitudinal force on that axle.*

Here, $g(\dot{x}, \dot{y}, \dot{\psi}, \delta)$ is some nonlinear function. Assumptions 1 and 2 allow for the use of the normal load expressions of the ST and DT models in Equation 4-2 and Equation 4-4. With these assumptions, it can be derived that the normal forces of each tire can be combined as presented in Equation 4-5.

$$F_{zfl} + F_{zfr} = \frac{l_r mg - hm\ddot{x}}{2\,(l_f + l_r)} = F_{z_f}, \quad F_{zrl} + F_{zrr} = \frac{l_r mg + hm\ddot{x}}{2\,(l_f + l_r)} = F_{z_r} \tag{4-5}$$

Assumptions 3, 4, and 5 simplify longitudinal and lateral tire force expressions so that they can be combined as described by Equation 4-6.

$$\begin{aligned}
&F_{xfr} + F_{xfl} = F_{x_f}, \quad F_{xrr} + F_{xrl} = F_{x_r} \\
&F_{yfl} + F_{yfr} = \mu(F_{zfl} + F_{zfr}) \cdot g(\dot{x}, \dot{y}, \dot{\psi}, \delta) = \mu F_{z_f} \cdot g(\dot{x}, \dot{y}, \dot{\psi}, \delta) = F_{yf} \\
&F_{yrl} + F_{yrr} = \mu(F_{zrl} + F_{zrr}) \cdot g(\dot{x}, \dot{y}, \dot{\psi}, \delta) = \mu F_{z_r} \cdot g(\dot{x}, \dot{y}, \dot{\psi}, \delta) = F_{yr}
\end{aligned} \tag{4-6}$$

Substituting these tire force expressions into the equations of motion of the DT results in the updated dynamics presented in Equation 4-7. Under the assumptions above, the DT model can be written as a ST model with an additional yaw moment $M_{DT}$.

$$\begin{aligned}
m\ddot{x} &= m\dot{y}\dot{\psi} + F_{x_f}\cos\delta - F_{y_f}\sin\delta + F_{x_r} - F_{drag} \\
m\ddot{y} &= -m\dot{x}\dot{\psi} + F_{x_f}\sin\delta + F_{y_f}\cos\delta + F_{y_r} \\
I_z\ddot{\psi} &= l_f(F_{x_f}\sin\delta + F_{y_f}\cos\delta) - l_r F_{y_r} + \\
&\quad (F_{yfl}\sin\delta - F_{xfl}\cos\delta + F_{xfr}\cos\delta - F_{yfr}\sin\delta + F_{xrr} - F_{xrl})\,\frac{w}{2} \\
&= l_f(F_{x_f}\sin\delta + F_{y_f}\cos\delta) - l_r F_{y_r} + M_{DT}
\end{aligned} \tag{4-7}$$

For relatively small steering angles, the main contributions to $M_{DT}$ stem from differences between the longitudinal forces on either side of the vehicle. This phenomenon is exploited in torque vectoring and differential braking control. In Equation 4-8, this moment can be divided into parts impacted by either longitudinal or lateral forces, $M_x$ and $M_y$ respectively.

$$M_{DT} = \underbrace{((F_{xfr} - F_{xfl})\cos\delta + F_{xrr} - F_{xrl})\,\frac{w}{2}}_{M_x} + \underbrace{((F_{yfl} - F_{yfr})\sin\delta)\,\frac{w}{2}}_{M_y} \tag{4-8}$$

The moment $M_y$ requires the tire model evaluations of both front tires. If this term is neglected, such as in the ST model, only two tire model evaluations are needed for $F_{y_f}$ and $F_{y_r}$. To reduce model fidelity and promote computational complexity, it is assumed that $M_y = 0$, so fewer tire model evaluations can be used.

**Assumption 6.** *The yaw moment* $M_y = 0$.

The moment $M_x$ is used to implement differential braking. With assumption 6, the dynamics of the reformulated DT model are equal to the dynamics of the ST model with an additional yaw moment $M_x$. This model is referred to as the CIC vehicle model and is presented in Figure 4-3.

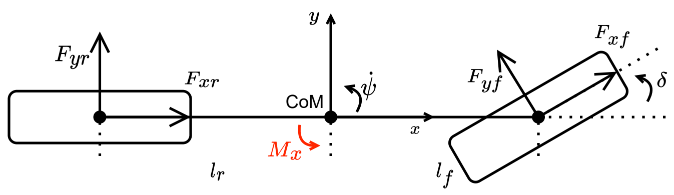


**Figure 4-3:** CIC vehicle model

The expression of $M_x$ can be further simplified by employing braking logic to derive the individual longitudinal tire forces $(F_{xfl}, F_{xfr}, F_{xrl}, F_{xrr})$.

## 4-1 Braking logic

The braking force at each wheel can be controlled using three inputs, namely $\dot{F}_x$, $\lambda_x$, and $\lambda_y$. This can be seen as an expansion on previous works that only apply a longitudinal brake bias $\lambda_x$ [2]. Here, $\dot{F}_x$ controls the rate of change in longitudinal force. Having $\dot{F}_x$ as a control input can minimize braking jerk by adding a penalty on $\dot{F}_x$ in the objective [2]. $\lambda_x$ and $\lambda_y$ specify the braking bias in longitudinal and lateral directions. Fewer inputs are required compared to introducing a longitudinal force input for each wheel [2]. Using these three inputs, the longitudinal tire forces at each wheel can be derived as presented in Equation 4-9.

$$
\begin{aligned}
F_{xfl} &= \begin{cases} \lambda_x \lambda_y F_x & F_x \leq 0 \\ \lambda_{\text{drive}} \frac{F_x}{2} & \text{otherwise} \end{cases} \\
F_{xfr} &= \begin{cases} \lambda_x (1-\lambda_y) F_x & F_x \leq 0 \\ \lambda_{\text{drive}} \frac{F_x}{2} & \text{otherwise} \end{cases} \\
F_{xrl} &= \begin{cases} (1-\lambda_x) \lambda_y F_x & F_x \leq 0 \\ (1-\lambda_{\text{drive}}) \frac{F_x}{2} & \text{otherwise} \end{cases} \\
F_{xrr} &= \begin{cases} (1-\lambda_x)(1-\lambda_y) F_x & F_x \leq 0 \\ (1-\lambda_{\text{drive}}) \frac{F_x}{2} & \text{otherwise} \end{cases}
\end{aligned}
\tag{4-9}
$$

The small angle approximation in Equation 4-10 can simplify the expression of $M_x$. This approximation sinmplifies the process of rewriting $M_x$ as a function of $(\delta, \dot{F}_x, \lambda_x, \lambda_y)$.

$$
M_x = \left( (F_{xfr} - F_{xfl}) \left( 1 - \frac{\delta^2}{2} \right) + F_{xrr} - F_{xrl} \right) \frac{w}{2} \tag{4-10}
$$

Using the braking logic defined in Equation 4-9, it is clear that $M_x = 0$ whenever the vehicle is not braking ($F_x > 0$). This aligns with the desire to apply only differential braking, not torque vectoring. In the case that the vehicle is braking ($F_x \leq 0$), $M_x$ can be expressed as a function of $(\delta, \dot{F}_x, \lambda_x, \lambda_y)$, as shown in Equation 4-11.

$$
\begin{aligned}
M_x \Big|_{F_x \leq 0} &= \left( (\lambda_x - 2\lambda_x\lambda_y) \left( 1 - \frac{\delta^2}{2} \right) + 1 - \lambda_x - 2\lambda_y + 2\lambda_x\lambda_y \right) \frac{wF_x}{2} \\
&= \left( 2 - 4\lambda_y - \lambda_x \left( 1 - 2\lambda_y \right) \delta^2 \right) \frac{wF_x}{4}
\end{aligned}
\tag{4-11}
$$

This expression can be further simplified if the $\delta^2$ factor is neglected. The maximum error introduced by this approximation can be derived when it is assumed that the steering angle is bounded.

**Assumption 7.** *The steering angle at the wheels $\delta$ is bounded by approximately* $\pm 20\,\text{deg}$.

Since the steering angle and the distribution values $\lambda$ are bounded by $[0, 1]$, the worst error of the additional moment $M_x$ due to these simplifications is around 6% and is independent of $\lambda_x$, as derived in Appendix A. This maximum error occurs when the vehicle brakes on a single wheel at the maximum steering angle. As this error is acceptable, the simplification in Equation 4-12 is applied.

$$M_x \approx \begin{cases} \frac{wF_x}{2}\left(1-2\lambda_y\right) & F_x < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4-12}$$

### Load dynamics

The longitudinal tire forces are controlled using $(\delta, \dot{F}_x, \lambda_x, \lambda_y)$. However, these forces must adhere to the friction limits. Since the longitudinal force of each tire is controlled individually, four constraints must be added to ensure that each tire remains unsaturated. The friction limits depend on the friction coefficient $\mu$ and the normal load $F_z$. The lateral and longitudinal tire forces influence the tire load at each wheel $F_{z_{i,j}}$.

To simplify the constraint formulation, the load dynamics can be added to the state space of the vehicle model [33]. This is realized using longitudinal and lateral tire force deviation parameters $\Delta F_{zx}$ and $\Delta F_{zy}$ respectively [33]. These parameters describe the longitudinal and lateral load distribution deviation from the front and rear static axle loads $F_{sf}$ and $F_{sr}$ respectively. The load forces on each tire are expressed in Equation 4-13 [33].

$$\begin{aligned} F_{zfl} &= \frac{1}{2}\left(F_{sf} - \Delta F_{zx}\right) - \gamma \Delta F_{zy}, \qquad & F_{zfr} &= \frac{1}{2}\left(F_{sf} - \Delta F_{zx}\right) + \gamma \Delta F_{zy} \\ F_{zrl} &= \frac{1}{2}\left(F_{sr} + \Delta F_{zx}\right) - \left(1-\gamma\right) \Delta F_{zy}, \qquad & F_{zrr} &= \frac{1}{2}\left(F_{sr} + \Delta F_{zx}\right) + \left(1-\gamma\right) \Delta F_{zy} \end{aligned} \tag{4-13}$$

The distribution of lateral load deviation $\Delta F_{zy}$ on the front axle is described using the vehicle handling parameter $\gamma$. The value of $\gamma$ can account for oversteer or understeer behavior of the vehicle [33].

The dynamics of $\Delta F_{zx}$ and $\Delta F_{zy}$ can be expressed as a linear combination of the total longitudinal and lateral tire forces, $F_{x_{\text{tot}}}$ and $F_{y_{\text{tot}}}$ respectively. The load dynamics are presented in Equation 4-14 [33]. Here, $k_x$ and $k_y$ are the longitudinal and lateral load derivative scaling parameters.

$$\begin{aligned} \Delta \dot{F}_{zx} &= k_x h \frac{F_{x_f}\cos\delta - F_{y_f}\sin\delta + F_{x_r}}{l_f + l_r} = \frac{k_x h F_{x_{\text{tot}}}}{l_f + l_r} \\ \Delta \dot{F}_{zy} &= k_y h \frac{F_{x_f}\sin\delta + F_{y_f}\cos\delta + F_{y_r}}{w} = \frac{k_y h F_{y_{\text{tot}}}}{w} \end{aligned} \tag{4-14}$$

## 4-2 Tire model

Different tire models can be considered, but the scope is limited to nonlinear models as CIC aims to perform maneuvers near the handling limits. Linear tire models fail to capture the dynamics under these conditions [17]. The Fiala tire model is considered in this work as it has a relatively small number of parameters, and it can capture the nonlinear relation between $F_y$ and $F_x$, leading to simplified friction limit constraints on the control problem formulation discussed in chapter 5.

The brush Fiala tire model is commonly used in control for aggressive maneuvers at the limits of handling [1, 11]. It is a nonlinear model that is piecewise-defined based on the current tire slip angle $\alpha$, tire saturation slip angle $\alpha_s$, tire stiffness $C_\alpha$, and the maximum lateral tire force $F_{y\,\mathrm{max}}$. The model can be formalized as presented in Equation 4-15 [2].

$$F_y = \begin{cases} -C_\alpha \tan\alpha + \frac{C_\alpha^2}{3F_{y\,\mathrm{max}}}|\tan\alpha|\tan\alpha - \frac{C_\alpha^3}{27(F_{y\,\mathrm{max}})^2}\tan^3\alpha, & |\alpha| < \alpha_s \\ -F_{y\,\mathrm{max}}\,\mathrm{sign}(\alpha), & \text{otherwise} \end{cases}$$

$$\alpha_s = \tan^{-1}\left(\frac{3F_{y\,\mathrm{max}}}{C_\alpha}\right), \quad F_{y\,\mathrm{max}} = \sqrt{(\mu F_z)^2 - F_x^2} \tag{4-15}$$

## 4-3 Combined state space model

The CIC vehicle model can be formulated into a continuous nonlinear state space model. The vehicle dynamics, load dynamics, and collision avoidance model equations can be combined to form a single prediction model. The dynamics of the CIC prediction model $f(\xi, u)$ in Equation 4-16 combines all these expressions for state space $\xi$ and input space $u$.

$$\dot{\xi} = f(\xi, u), \quad \xi = (\delta, F_x, \dot{\psi}, \dot{x}, \dot{y}, e_\psi, s, e_y, \Delta F_{zx}, \Delta F_{zy}), \quad u = (\dot{\delta}, \dot{F}_x, \lambda_x, \lambda_y) \tag{4-16}$$

This vehicle model is used as a prediction model in the planning and control of evasive maneuvers for CIC.

### Baseline model

To evaluate the performance of this model, a state-of-the-art baseline prediction model developed by Brown et al. [2] is implemented to compare the performance of the CIC model in chapter 7. The CIC model can be seen as an evolution of this model. The baseline model only includes longitudinal brake distribution and does not provide differential braking. Therefore, the baseline model is equal to the CIC model for $\lambda_y = 0.5$ for which $M_x = 0$. Since the brake distribution is greatly simplified, only two constraints of friction limit are needed, which do not require load dynamics. Therefore, this model has a more compact state space. Furthermore, the model has one less input as $\lambda_y$ is not a control variable. The state and input space of the baseline prediction model are presented in Equation 4-17.

$$\dot{\xi} = f(\xi, u), \quad (\delta, F_x, \dot{\psi}, \dot{x}, \dot{y}, e_\psi, s, e_y), \quad u = (\dot{\delta}, \dot{F}_x, \lambda_x) \tag{4-17}$$

## 4-4 Model parameter identification

To ensure the accuracy of the prediction model, the model parameters can be identified to ensure that the response of the prediction model response best matches the controlled plant. This is referred to as gray-box system identification. The BMW 5-series vehicle model from IPG CarMaker [1] with an RT_255_55R17_p2.50 tire model is used as the plant of the

[1]IPG CarMaker: https://ipg-automotive.com/en/products-solutions/software/carmaker/

controller. This is a high-fidelity model with complex dynamics that closely approximates the behavior of a real car.

To control this model, the control input $u = (\dot{\delta}, \dot{F}_x, \lambda_x, \lambda_y)$ have to be converted to inputs that relate to the plant. In this case, the vehicle can be controlled using the steering wheel angle $\delta_{sw}$, the accelerator $a_{cc}$, and the braking torques $T_{b,i,j}$ at axle $i = (f, r)$ and side $j = (l, r)$. The power train is included using the accelerator instead of a driving torque. Individual braking torques are used to employ differential braking.

The control inputs $u$ can be converted to plant inputs $u_p$ using the conversion presented in Equation 4-18. Here, $\Delta t$ is the timestep of the controller.

$$
\begin{aligned}
\delta_{sw} &= n_{sw} \cdot \left(\delta + \dot{\delta}\Delta t\right) \\
T_{bfl} &= \begin{cases} -\lambda_x \lambda_y \left(F_x + \dot{F}_x \Delta t\right) & \text{if } F_x + \dot{F}_x \Delta t < 0 \\ 0 & \text{otherwise} \end{cases} \\
T_{bfr} &= \begin{cases} -\lambda_y \left(1 - \lambda_x\right) \left(F_x + \dot{F}_x \Delta t\right) & \text{if } F_x + \dot{F}_x \Delta t < 0 \\ 0 & \text{otherwise} \end{cases} \\
T_{brl} &= \begin{cases} -\lambda_x \left(1 - \lambda_y\right) \left(F_x + \dot{F}_x \Delta t\right) & \text{if } F_x + \dot{F}_x \Delta t < 0 \\ 0 & \text{otherwise} \end{cases} \\
T_{brr} &= \begin{cases} -\left(1 - \lambda_x\right)\left(1 - \lambda_y\right) \left(F_x + \dot{F}_x \Delta t\right) & \text{if } F_x + \dot{F}_x \Delta t < 0 \\ 0 & \text{otherwise} \end{cases} \\
a_{cc} &= n_{a_{cc}} \left(F_x + \dot{F}_x \Delta t\right), \quad 0 \le a_{cc} \le 1
\end{aligned}
\tag{4-18}
$$

The braking torques $T_{b,i,j}$ are denoted as positive. Furthermore, the steering rate $n_{sw}$ and accelerator scaling parameter $n_{a_{cc}}$ can be added to the set of prediction model parameters to be identified. With this input conversion, the CIC and baseline prediction models can be used to control the system plant.

Since the CIC and baseline prediction model have different dynamics, constraints, and input and state spaces, the parameters of each model are identified separately to offer a fair comparison. As these models have different input spaces, a different input sequence is used to identify the parameters of each model. Both input sequences force the vehicle to perform an aggressive double-lane change maneuver to capture the vehicle's dynamics near the handling limits. For each input sequence, the model parameters of the corresponding prediction model can be fitted to the state response of the plant model.

### 4-4-1 Baseline prediction model identification

The baseline and CIC prediction models are identified using the control input sequence presented in Figure 4-4. The simulated state space of the IPG vehicle model and the baseline model for these control inputs are presented in Figure 4-5 and Figure 4-7. Here, the spatial dynamics $(e_\psi, s, e_y)$ are excluded as these can be directly derived using the cartesian dynamics and the curvature profile.

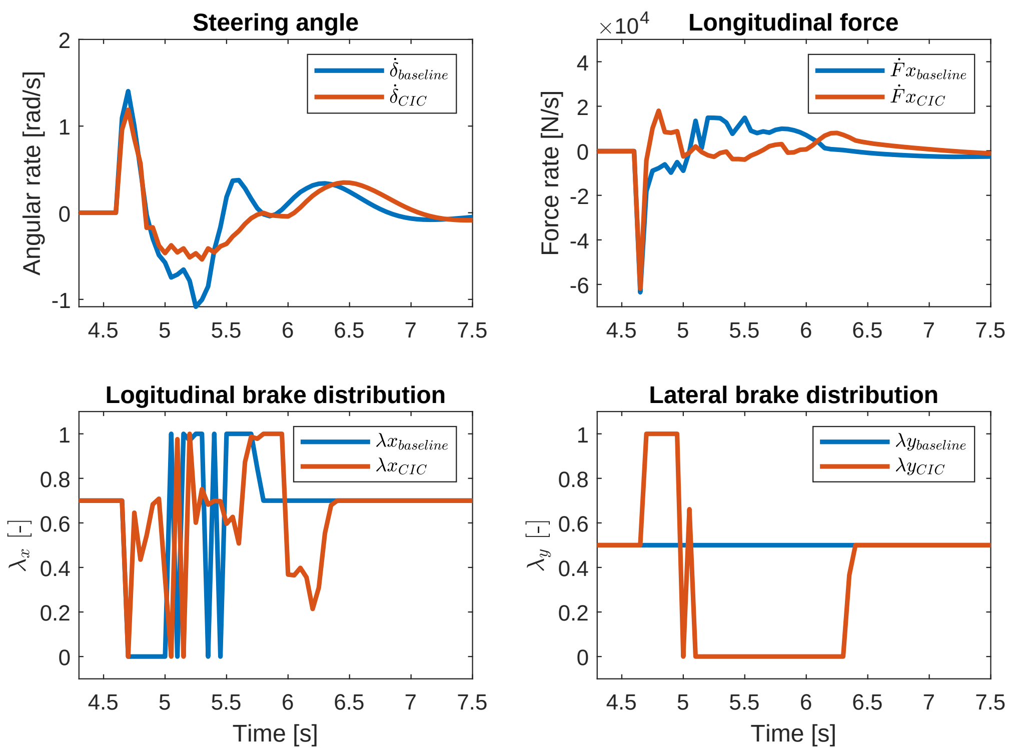


**Figure 4-4:** Control inputs for model indentification

From Figure 4-5, it is clear that the identified model closely represents the IPG vehicle model. The most significant deviations between the models exist in the longitudinal dynamics due to the more complex drivetrain and tire model of the plant model. The gray box identification leads to the set of identified model parameters in Table 4-1. Since there is a deviation between the modeled tire forces and the tire force of the plant, the drag dynamics are challenging to estimate in this scenario.

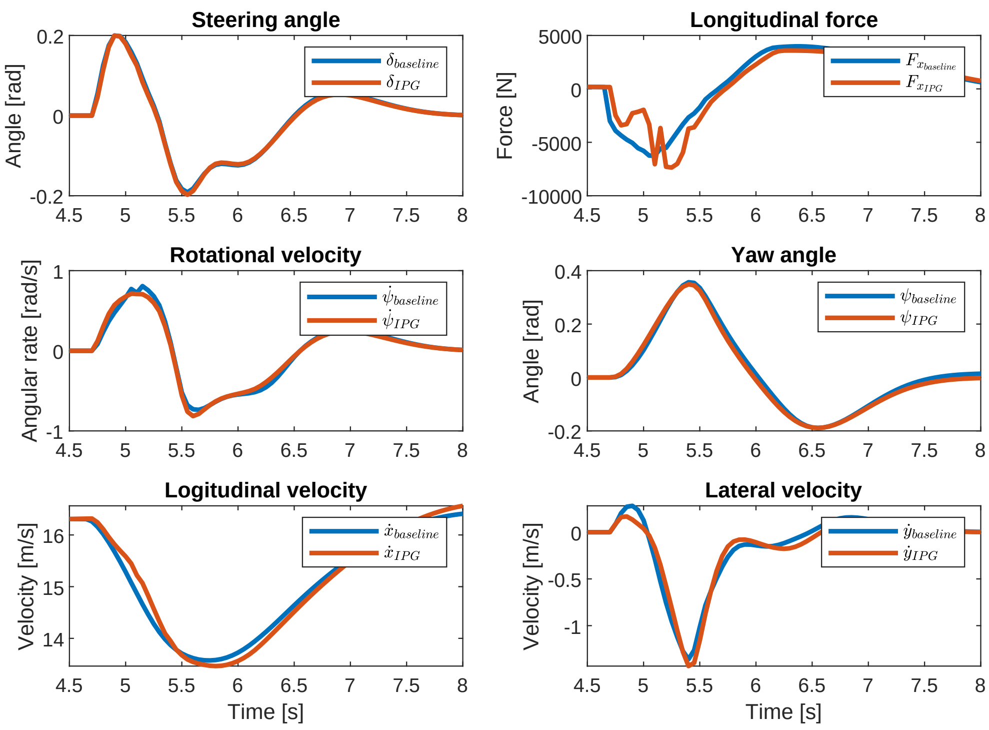


**Figure 4-5:** States for baseline model indentification

## 4-4-2 CIC prediction model identification

The identification process for the baseline model can be repeated for the CIC prediction model. However, since the CIC prediction model has more parameters, the load dynamics are identified separately to simplify the identification process. The input sequence in Figure 4-4 is used to identify the parameters of the load dynamics.

### Identification of load dynamics

The load forces on each tire can be expressed using the CIC model states $\Delta F_{zx}$ and $\Delta F_{zx}$, presented in chapter 4. These states depend on the longitudinal and lateral tire forces, which makes it impossible to identify the load dynamics separately from the rest of the CIC model. To enable separate identification of the load dynamics, these states can be approximated from the accelerations using scaling parameters $k_{ax}$ and $k_{ay}$ as presented in Equation 4-19.

$$\Delta F_{zx} = k_{ax}\ddot{x}, \quad \Delta F_{zx} = k_{ay}\ddot{y} \tag{4-19}$$

Fitting the load dynamics to the tire loads of the IPG model for the control inputs in Figure 4-4 leads to the response presented in Figure 4-6. The identified model for the load dynamics captures the tire loads well, with most of the error present in the peak load forces. The remainder of the parameters for the CIC prediction model can be identified using the previously identified load dynamics. The simulated state space of the IPG vehicle model and the CIC prediction model for the control inputs in Figure 4-4 is presented in Figure 4-7.

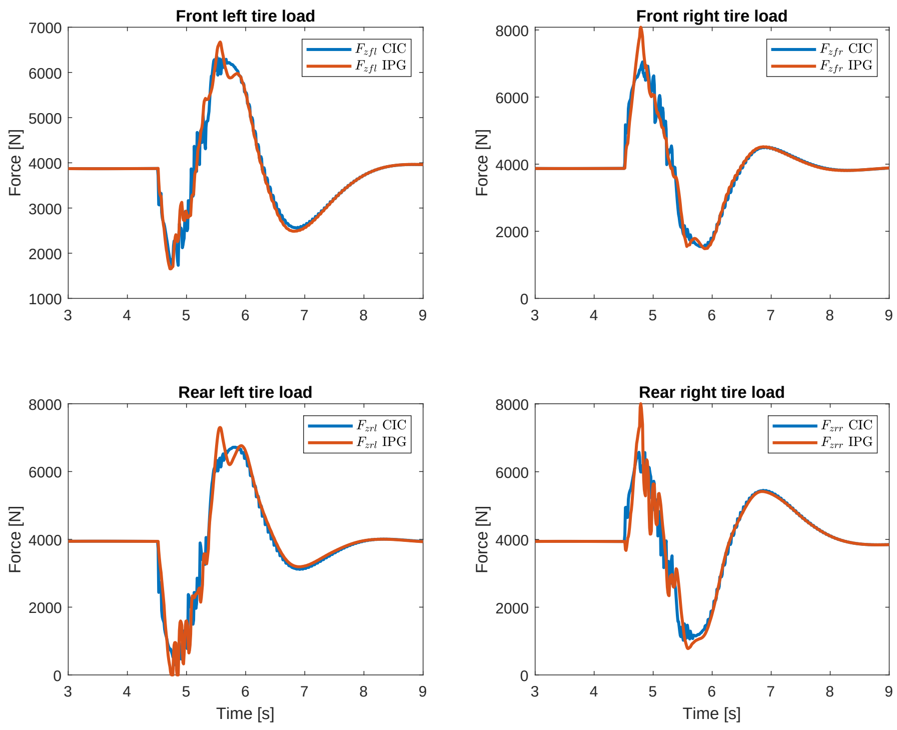


**Figure 4-6:** Load dynamics identification

More significant discrepancies can be observed in the lateral dynamics, likely caused by modeling errors in applying differential braking. However, the magnitude of the lateral velocity is also significantly lower than in the baseline model. All other states are described equally well.

## 4-4-3 Identified parameters

For the baseline model, nine parameters are identified. The CIC model has an additional eight parameters for the load dynamics. Since these models are nonlinear, constrained SQP gray-box optimization is used to identify the parameters. The identified model parameters are presented in Table 4-1, Table 4-2, and Table 4-3. The parameters of the IPG input conversion were identified as $n_{acc} = 1.4e^{-4}$ and $n_{sw} = 14$.

Generally, both prediction models can describe the dynamics of the IPG CarMaker vehicle validation model well, with most uncertainty caused by prediction model inaccuracies of the powertrain and tire model.

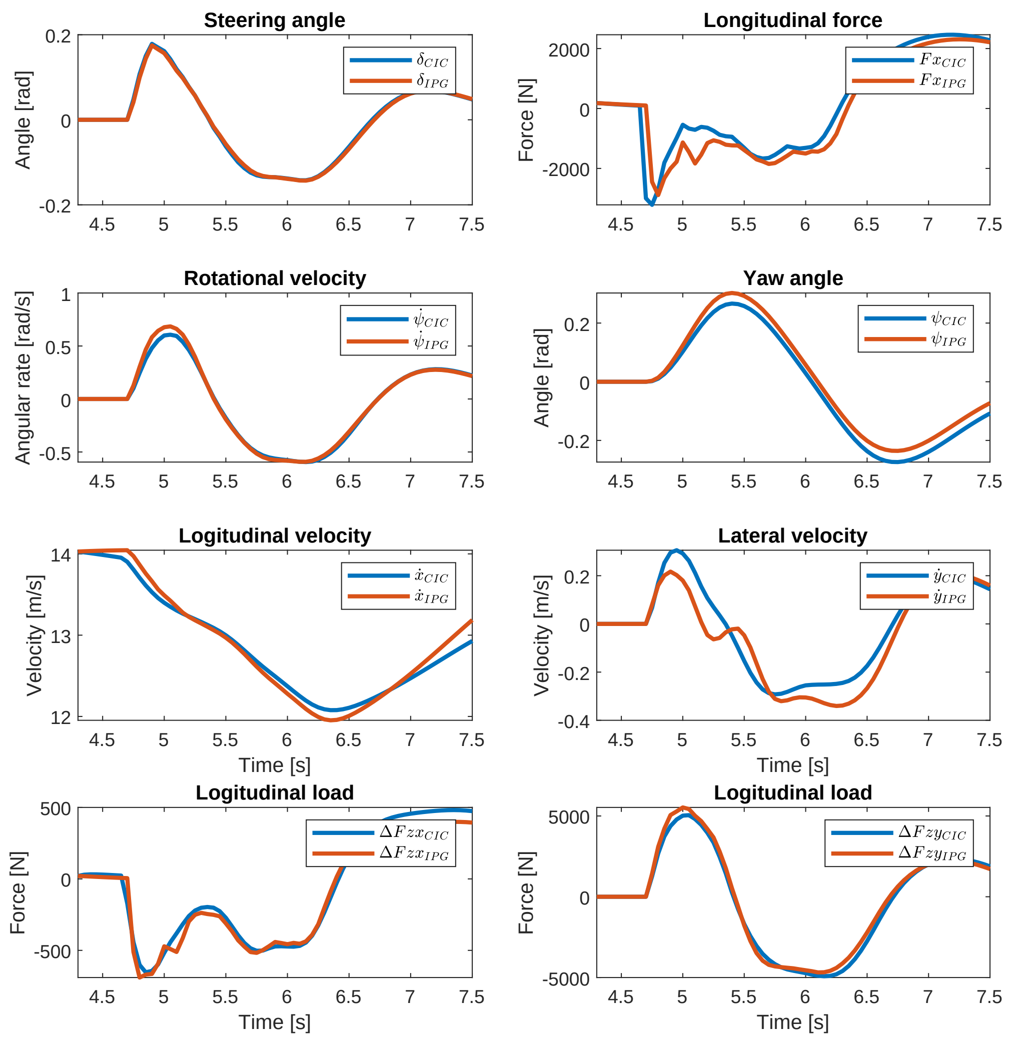


**Figure 4-7:** States for CIC model identification

| symbol | value | unit |
|---|---|---|
| $m$ | 2011.2 | $kg$ |
| $I_z$ | 3763.2 | $m^4$ |
| $l_r$ | 1.377 | $m$ |
| $l_f$ | 1.511 | $m$ |
| $h$ | 0.380 | $m$ |
| $C\alpha_f$ | $252 \cdot 10^3$ | - |
| $C\alpha_r$ | $178 \cdot 10^3$ | - |
| $Cd_0$ | 43.83 | - |
| $Cd_1$ | 1.42 | - |

**Table 4-1:** Baseline model parameters

| symbol | value | unit |
|---|---|---|
| $m$ | 1972.2 | $kg$ |
| $I_z$ | 3294.2 | $m^4$ |
| $l_r$ | 1.373 | $m$ |
| $l_f$ | 1.514 | $m$ |
| $h$ | 0.400 | $m$ |
| $C\alpha_f$ | $215 \cdot 10^3$ | - |
| $C\alpha_r$ | $280 \cdot 10^3$ | - |
| $Cd_0$ | 423 | - |
| $Cd_1$ | 1.42 | - |

**Table 4-2:** CIC model parameters

| symbol | value | unit |
|---|---|---|
| $k_x$ | 6.406 | - |
| $k_y$ | 21.05 | - |
| $nF_x$ | 0.625 | - |
| $F_{s1}$ | 7757.1 | $N$ |
| $F_{s2}$ | 7871.1 | $N$ |
| $k_{ax}$ | 309.0 | — |
| $k_{ay}$ | 634.5 | — |
| $\gamma$ | 0.456 | — |

**Table 4-3:** Load dynamics parameters

Chapter 5

# Optimization problem formulation

With the definition of the dynamics, constraints, and objectives for the prediction model in chapter 3 and chapter 4, the NMPC problem can be formulated. The OCP formulation is derived from the baseline [2]. This OCP can be defined as an Nonlinear Program (NLP). This is achieved by formulating a constrained minimization problem of the objective function $J(\xi, u)$ for the system states $\xi$ and inputs $u$ over a finite horizon prediction $N$. The NMPC problem can be formulated into a NLP using multiple shooting over $N$ intervals between $t_0$ and $t_N$ [3]. This is demonstrated in Equation 5-1 [3].

$$
\begin{aligned}
\min_{\xi_k, u_k} \quad & \sum_{k=0}^{N-1} J(\xi_k, u_k) + J_N(\xi_N) \\
\text{s.t.} \quad & \xi_0 = \hat{\xi}_0 \\
& \xi_{k+1} = \Phi(\xi_k, u_k) \\
& c_L \leq c(\xi_k, u_k) \leq c_U \\
& c_{NL} \leq c(\xi_N) \leq c_{NU}
\end{aligned}
\tag{5-1}
$$

In Equation 5-1, $J_N(\xi, u)$ is the terminal cost, $\xi_N$ the terminal state, $c_L$ and $c_U$ the lower and upper bounds of the nonlinear stage constraints, and $c_{NL}$ and $c_{NU}$ the lower and upper bounds of the nonlinear terminal constraints. The functions $c(\xi_k, u_k)$ and $c(\xi_N)$ describe the nonlinear stage and terminal constraint functions. The discrete state update is described by $\Phi(\xi_k, u_k)$.

Since the dynamics are nonlinear, most constraints of the NMPC problem have to be formulated as nonlinear constraints in the NLP problem. Input constraints are the only linear constraints. These nonlinear constraints can be divided into constraints on states, system dynamics, and general constraints.

## 5-1 State constraints

The CIC prediction model defined in chapter 4 has ten states, these include:

$$\xi = [\delta, F_x, \dot{\psi}, \dot{x}, \dot{y}, e_\psi, s, e_y, \Delta F_{zx}, \Delta F_{zy}] \tag{5-2}$$

A constraint is introduced on $\delta$ to limit the steering angle. Furthermore, The total longitudinal force $F_x$ is limited by the powertrain. This results in the following two state constraints [2]:

$$\delta_{\min} \leq \delta \leq \delta_{\max}, \quad F_{x_{\min}} \leq F_x \leq F_{x_{\max}} \tag{5-3}$$

## 5-2 Input constraints

The CIC prediction model defined in chapter 4 has four inputs, $u = [\dot{\delta}, \dot{F}_x, \lambda_x, \lambda_y]$. Input constraints can incorporate actuator limits and ensure system feasibility. The steering rate $\dot{\delta}$ is confined to the specifications of the steering system. No bounds are defined on the rate of change of the longitudinal force $\dot{F}_x$. However, a cost will be placed on this input to prevent jitter, similar to the baseline [2]. To distribute the braking force over the four wheels, $\lambda_x$ and $\lambda_y$ are bounded by $[0, 1]$. This results in the following three input constraints presented in Equation 5-4.

$$\dot{\delta}_{\min} \leq \dot{\delta} \leq \dot{\delta}_{\max}, \quad 0 \leq \lambda_x \leq 1, \quad 0 \leq \lambda_y \leq 1 \tag{5-4}$$

## 5-3 System dynamics constraints

The prediction model dynamics can be encoded in the NLP by enforcing the constraint $\xi_{k+1} = \Phi(\xi_k, u_k)$. Here $\Phi(\xi_k, u_k)$ is a function that calculates and integrates the state derivative $\dot{\xi}_k$ using the current state $\xi_k$ and input $u_k$. This derivative is then integrated over some time $dt = t_{k+1} - t_k$ to produce the next state $\xi_{k+1}$. The state derivate is calculated using the state space dynamics of the CIC model defined in chapter 4.

Different integration methods can be exploited to balance accuracy and computational complexity for real-time performance [40]. This is further discussed in chapter 6.

## 5-4 Friction limit constraints

When differential braking is applied, the braking force on each tire can be controlled individually. However, the friction at each wheel limits the available braking force. The tire saturation constraints are defined in Equation 5-5 [2].

$$F_{x_{i,j}}^2 + F_{y_{i,j}}^2 \leq \mu F_{z_{i,j}} \tag{5-5}$$

The lateral tire force constraints are encoded in the Fiala tire model. However, the longitudinal tire forces must be constrained according to the friction limits at each wheel. These constraints can be formulated as presented in Equation 5-6.

$$|F_{x_{i,j}}| \leq \mu F_{z_{i,j}}, \quad i = (f, r), \, j = (l, r) \tag{5-6}$$

These constraints ensure that lateral tire forces are feasible and the friction limit bounds the longitudinal force. This constraint can be expressed in terms of $F_x$, $\lambda_x$, and $\lambda_y$ using the load dynamics at each wheel.

Using these load dynamics and the previously defined braking forces on each tire, four friction limit constraints are derived in Appendix A. These constraints are expressed in Equation 5-7. Here, $\mu$ can be substituted by $\eta\mu$, where $\eta$ is a scaling factor smaller than 1 to account for uncertainty in $\mu$ [2].

$$\begin{aligned}
&\operatorname{sign}(F_x)\cdot\begin{bmatrix}\frac{\lambda_1}{\mu} & \frac{1}{2} & \gamma\end{bmatrix}\begin{bmatrix}F_x\\ \Delta F_{zx}\\ \Delta F_{zy}\end{bmatrix}\le F_{gf}, \qquad \operatorname{sign}(F_x)\cdot\begin{bmatrix}\frac{\lambda_2}{\mu} & \frac{1}{2} & -\gamma\end{bmatrix}\begin{bmatrix}F_x\\ \Delta F_{zx}\\ \Delta F_{zy}\end{bmatrix}\le F_{gf}\\
&\operatorname{sign}(F_x)\cdot\begin{bmatrix}\frac{\lambda_3}{\mu} & -\frac{1}{2} & 1-\gamma\end{bmatrix}\begin{bmatrix}F_x\\ \Delta F_{zx}\\ \Delta F_{zy}\end{bmatrix}\le F_{gr}, \quad \operatorname{sign}(F_x)\cdot\begin{bmatrix}\frac{\lambda_4}{\mu} & -\frac{1}{2} & \gamma-1\end{bmatrix}\begin{bmatrix}F_x\\ \Delta F_{zx}\\ \Delta F_{zy}\end{bmatrix}\le F_{gr}
\end{aligned} \tag{5-7}$$

With:

$$\begin{aligned}
&\lambda_1=\begin{cases}-\lambda_x\lambda_y & F_x\le 0\\ \frac{\lambda_{\text{drive}}}{2} & \text{otherwise}\end{cases}, \quad \lambda_2=\begin{cases}-\lambda_x(1-\lambda_y) & F_x\le 0\\ \frac{\lambda_{\text{drive}}}{2} & \text{otherwise}\end{cases}\\
&\lambda_3=\begin{cases}-(1-\lambda_x)\lambda_y & F_x\le 0\\ \frac{(1-\lambda_{\text{drive}})}{2} & \text{otherwise}\end{cases}, \quad \lambda_4=\begin{cases}-(1-\lambda_x)(1-\lambda_y) & F_x\le 0\\ \frac{(1-\lambda_{\text{drive}})}{2} & \text{otherwise}\end{cases}\\
&F_{gf}=\frac{l_r mg}{2\left(l_f+l_r\right)}, \qquad F_{gr}=\frac{l_f mg}{2\left(l_f+l_r\right)}
\end{aligned} \tag{5-8}$$

## 5-5 Stability

Stability can be enforced using constraints on tire slip [31], yaw rate, or lateral velocity [11]. However, collision avoidance should be prioritized over stability in CIC [11]. As collision avoidance performance can be improved when stability constraints are temporarily violated, these constraints should not always be enforced. Allowing constraint violations requires slack variables, which increase the problem size.

In some cases, stability constraints can be left out of the OCP entirely in CIC [2]. Stability is then indirectly enforced as unrecoverable unstable conditions lead to collisions with road boundaries or obstacles later on in the prediction horizon. Therefore, unrecoverable vehicle instability is penalized by penalizing collision avoidance with road boundaries and obstacles, combined with long prediction horizons already required by collision avoidance control. Temporary and marginal vehicle stability violations are not penalized or constrained in this way.

## 5-6 Control hierarchy

Different control hierarchies can be used to implement CIC. Combining the collision avoidance objective into the OCP leads to a controller that can achieve collision avoidance without needing a reference trajectory. However, it might be desirable to incorporate a global planner so that the controller can be used as a tracking controller outside of collision imminent scenarios. Furthermore, having a reference trajectory available during collision avoidance can incorporate other objectives, such as adherence to traffic laws by specifying a target velocity. This reference trajectory should always be tracked as well as possible unless following it leads to a collision. In chapter 6, it is explored whether this reference trajectory can improve the collision avoidance performance of CIC.

Similar to the baseline [2], a global planner that specifies a trajectory $ey_{ref}(s)$ as a function of the distance $s$ and a reference velocity $\dot{x}_{ref}$ is implemented. This leads to the controller hierarchy presented in Figure 5-1.

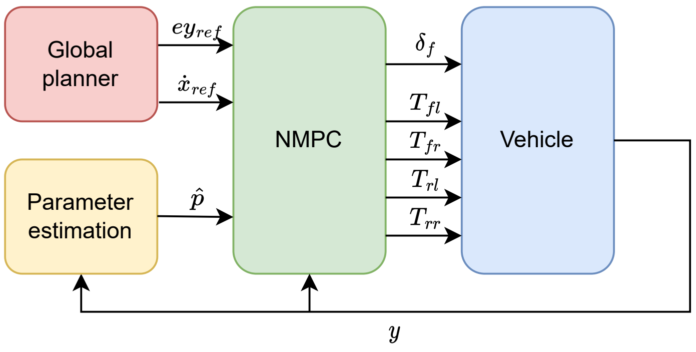


**Figure 5-1:** CIC controller hierachy

This hierarchy is somewhere between a one-level and two-level controller. The location of obstacles is directly considered in the optimization problem using the collision avoidance objective presented in chapter 3. The location of the obstacles is included in the estimated parameter vector $\hat{p}$.

The CIC controller has ten parameters in $\hat{p}$ that describe elements of the dynamic environment in which the vehicle operates. These are presented in Equation 5-9. Here, it is assumed that there are two obstacles described by the location $(s_o, e_o)$ and radius $r_o$. Other time-varying parameters, such as the road boundaries, friction coefficient, and road curvature, are also included.

$$\hat{p}(s) = [s_{o1}, e_{o1}, r_{o1}, s_{o2}, e_{o2}, r_{o2}, rb_r, rb_l, \mu, \kappa] \tag{5-9}$$

Some parameters are a function of $s$ as they can vary over distance. As the parameters are not assumed constant, each step in the prediction horizon has a separate set of parameters. This leads to $\hat{p}(s)$ having a size of a $[N \times 10]$ when there are two obstacles. The distance $s$ can be extrapolated throughout the prediction horizon based on the initial location $s_0$ and the velocity along the road $\dot{s}$.

The estimation of these parameters is further addressed in chapter 8. Some parameters can be manually selected without estimation to specify different experimental setups discussed in chapter 6.

## 5-7 Objective

CIC has multiple control objectives that must be combined and weighted to achieve the desired behavior. The objectives are formulated similarly to the baseline [2]. The objectives include tracking the global planner's reference trajectory and velocity, avoiding obstacles and road boundaries, limiting jitter in the steering and longitudinal acceleration, and limiting vehicle instability. Energy-based costs can furthermore promote convergence.

These objectives are formalized as a combination of stage costs $J(\xi_k, u_k)$ and terminal costs $J_N(\xi_N)$ in Equation 5-10.

$$\begin{aligned} J(\xi_k, u_k) &= J(\xi_k, u_k) + J_N(\xi_N) \\ &= (h(\xi_k, u_k) - h_{\text{ref}})\, Q\, (h(\xi_k, u_k) - h_{\text{ref}})^T + (h_N(\xi_N) - h_{N_{\text{ref}}})\, Q_N\, (h_N(\xi_N) - h_{N_{\text{ref}}})^T \end{aligned} \tag{5-10}$$

Here, $Q$ and $Q_N$ are weighted diagonal matrices, and $h$ and $h_N$ are nonlinear cost functions. The cost functions are defined in Equation 5-11.

$$\begin{aligned} h(\xi_k, u_k) &= [\dot{x}, e_y, \dot{\delta}, \dot{F}_x, \lambda_x, \lambda_y, d_o^-, d_r^-] - h_{\text{ref}} \\ h_{\text{ref}} &= [\dot{x}_{\text{ref}}, e_{y_{\text{ref}}}, 0, 0, \lambda_{x_{\text{nat}}}, \lambda_{y_{\text{nat}}}, d_{o_{\min}}, d_{r_{\min}}] \\ h_N(\xi_N) &= [\dot{x}, e_y] - h_{N_{\text{ref}}} \\ h_{N_{\text{ref}}} &= [\dot{x}_{\text{ref}}, e_{y_{\text{ref}}}] \end{aligned} \tag{5-11}$$

The weighted diagonal matrices $Q$ and $Q_N$ are presented in Equation 5-12. The values were tuned so that the controller achieved the desired performance, starting from the initial values from the baseline [2]. As collision avoidance has the highest priority in CIC, the weight of collision avoidance was set to dominate the cost function.

$$Q = \text{diag}([0.1, 4, 100, 10^{-6}, 0.1, 0.1, 100, 100]), \quad Q_N = \text{diag}([0.1, 4]) \tag{5-12}$$

Here, $d_o^-$ and $d_r^-$ describe the distance to all obstacle and road boundaries distances in the OCP when the minimum distances $d_{o_{\min}}$ and $d_{r_{\min}}$ are violated. These are presented in Equation 5-13 to implement the function $c_{\text{dist}}$ defined in chapter 3. The formulations for $d_o$ and $d_r$ derived in that chapter can be used to create different OCP formulations. In vector form, $d_o$ and $d_r$ can include all distances to the obstacles and road boundaries according to the distance models and scenario.

$$d_o^- = \begin{cases} d_o & \text{if } d_o < d_{o_{\min}} \\ d_{o_{\min}} & \text{otherwise} \end{cases}, \quad d_r^- = \begin{cases} d_r & \text{if } d_r < d_{r_{\min}} \\ d_{r_{\min}} & \text{otherwise} \end{cases} \tag{5-13}$$

The parameters $\lambda_{x_{\text{nat}}}$ and $\lambda_{y_{\text{nat}}}$ describe the natural longitudinal and lateral brake distribution respectively. These costs ensure the brake distribution returns to the natural configuration after an aggressive maneuver [2].

The values of the parameters used in $h_{\text{ref}}$ are presented in Table 5-1. These values aim to capture that vehicles naturally distribute more braking force to the front tires. Furthermore, more distance margin is desired around obstacles than road boundaries.

| $\lambda_{x_{\mathbf{nat}}}$ [-] | $\lambda_{y_{\mathbf{nat}}}$ [-] | $d_{o_{\min}}$ [m] | $d_{r_{\min}}$ [m] |
|---|---|---|---|
| 0.7 | 0.5 | 0.7 | 0.5 |

**Table 5-1:** Objective reference parameters

Chapter 6

# Solving techniques

The OCP problem can be implemented in various ways, allowing for different solving techniques. These techniques can be exploited by using different solvers or problem formulations. This chapter evaluates the performance of CIC for different condensed or sparse solvers, integration grids, and toolboxes. For this evaluation, an experimental setup is created in which a collision is imminent. The resulting maneuver should push the vehicle and the controller to the limits.

## 6-1 Experimental setup

The performance of CIC is analyzed when the vehicle must perform a double-lane change on a dual two-lane road to avoid two obstacles in either lane, similar to the baseline [2]. This scenario is depicted in Figure 6-1. Both obstacles spontaneously appear in front of the vehicle

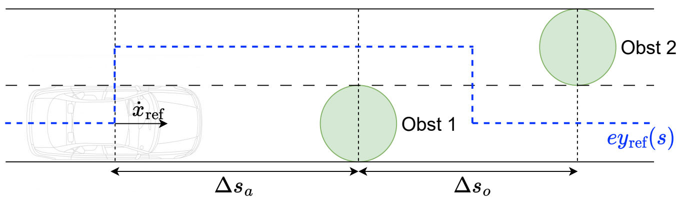


**Figure 6-1:** Experimental setup for CIC

when the vehicle is some distance $\Delta s_a$ from the first obstacle. The vehicle must stay within the road boundaries during the maneuver. This experimental setup can be parameterized by specifying the reference velocity $\dot{x}_{\text{ref}}$, the distance between obstacles $\Delta s_o$, the distance $\Delta s_a$, and the friction coefficient $\mu$. Generally, smaller values for $\Delta s_a$, $\Delta s_o$, and $\mu$ and larger values of $\dot{x}_{\text{ref}}$ make the problem more difficult to solve. These parameters can be specified using the parameter vector $\hat{p}$, defined in chapter 5.

A default obstacle radius $r_o$ of 2 meters is selected. Furthermore, a reference trajectory $ey_{\text{ref}}$ is assumed available. The reference tracks the left lane at a distance $\Delta s_a$ from the obstacle and returns to the right lane between the obstacles.

All PC-based simulations are performed on a Windows 10 PC with an i7-7700 CPU running at 2.80GHz.

## 6-2 Controller implementation

The implementation of the controller can impact the real-time performance of the controller. To solve the nonlinear OCP in real-time, the problem can be rewritten to a Quadratic Programming (QP) problem using Newton-based methods such as SQP [7]. SQP is commonly used in NMPC, as it can solve NMPC problems in real-time [27]. Therefore, SQP forms the focus of this research.

### 6-2-1 Sequential quadratic programming

SQP exploits the Karush-Kuhn-Tucker (KKT) conditions of a local optimal solution of the non-linear OCP to re-write the problem as a QP problem.

The KKT conditions for a general NLP presented in chapter 5 can be expressed as presented in Equation 6-1 [7]. Here, $z$ is the state and input space of the OCP, $f_{ec}(z)$ and $f_{ic}(z)$ are the equality and inequality constraints for the NLP.

$$z = [\xi, u]^T, \quad f_{ec}(z) = 0, \quad f_{ic}(z) \leq 0 \tag{6-1}$$

For some local optimum $z^*$, the KKT conditions are expressed in Equation 6-2 [7].

$$\begin{aligned} \nabla_z \mathcal{L}\left(z^*, \lambda_z^*, \mu_z^*\right) = \nabla_z(f(z^*) + f_{ec}(z^*)^T \lambda_z^* + f_{ic}(z^*)^T \mu_z^*) = 0 \\ f_{ec}\left(z^*\right) = 0, \quad f_{ic}\left(z^*\right) \leq 0, \quad \mu_z^* \geq 0 \\ f_{ic}\left(z^*\right)_i \mu_{z_i}^* = 0, \quad \forall i = 1 \cdots n_h \end{aligned} \tag{6-2}$$

It is assumed that no exact Hessian is available and that the generalized Gauss-Newton method is applied to approximate the Hessian in real-time. This is applicable as the objective function of the OCP $J(\xi, u)$ in chapter 5 is defined as a sum of squares [7]. The objective is rewritten in Equation 6-3.

$$J(\xi, u) = \frac{1}{2}||J_1(z)||_2^2 \tag{6-3}$$

The QP problem can be formulated by iteratively finding a solution that satisfies the KKT conditions [7]. In this way, a local optimum is iteratively found. The Hessian $H$, QP objective $f_{QP}(z)$, and the constraints can be expressed at each iteration $k$ as presented in Equation 6-4 [7].

$$\begin{aligned} H_k &= \nabla J_1(z_k) \nabla J_1(z_k)^T \\ f\mathrm{QP}_k(z) &= \nabla f\left(z_k\right)^T z + \frac{1}{2}\left(z - z_k\right)^T H_k \left(z - z_k\right) \\ &= \frac{1}{2}\left\| J_1\left(z_k\right) + \nabla J_1\left(z_k\right)^T\left(z - z_k\right) \right\|_2^2 \end{aligned} \tag{6-4}$$

In this research, the toolboxes MATMPC [3], ACADO, and Acados are leveraged to perform these operations in real time. Real Time Iterations (RTI) are used, in which only a single SQP iteration is performed for $k = 0$. In RTI, the performance of SQP is further exploited using direct multiple-shooting discretization [27].

### 6-2-2 SQP toolboxes

For real-time control, embedded software tools that leverage $C++$ and $C$ languages are explored. Three toolboxes are used to implement the CIC controller. These toolboxes use a definition of the OCP as presented in chapter 5 and apply SQP to generate a Quadratic Programming (QP) problem implemented in $C$-code.

The generated controllers can be compiled for Matlab execution using the MEX compiler. This outputs an S-function that can be directly used in Simulink. The S-function of the controllers can then be inserted into the Simulink configuration of IPG CarMaker, which simulates the plant's response. The integrated closed loop Simulink model is presented in Appendix D.

#### MATMPC

MATMPC is a recent toolbox that utilizes Casadi to generate a QP problem in the $C$ coding language using symbolic variables in Matlab. The interface of MATMPC is in Matlab, where executable Mex functions are generated upon building the $C$-code. No direct embedded implementation is possible by default, as the generated Mex functions are executed from Matlab.

Casadi offers flexibility by allowing simple and efficient integration with other programming languages like Python and Matlab. However, it can result in poorly readable $C$ code. This makes it more complex to add external features, limiting available options to those implemented by default.

#### ACADO

ACADO is the most common toolbox used in vehicle control [32]. ACADO can be interfaced with through Matlab, or the model can be specified directly in the $C^{++}$ coding language. The benefit is that this results in more readable code that allows custom features to be added, such as variable timesteps.

A downside of using ACADO is that it uses custom datatypes for the variables used in the OCP. This means that standard operators do not work. ACADO provides basic operators for its datatypes but does not include extrema or conditional operators. A workaround must be found since the Fiala tire model and the collision avoidance formulation use such operators.

To approximate conditional operators, a tangent hyperbolic function can be used. This can be done for the conditional operator in the collision avoidance model as presented in Equation 6-

5.

$$\begin{aligned} c_{dist} &= \begin{cases} Q_{dist}(d - d_{\min})^2 & \text{if } d < d_{\min} \\ 0 & \text{otherwise} \end{cases} \\ &\approx Q_{dist}\left(d - d_{\min}\right)^2 \cdot \left(0.5 - \frac{\arctan\left(n_{\arctan} \cdot (d - d_{\min})\right)}{\pi}\right) \end{aligned} \tag{6-5}$$

$n_{\arctan}$ is a scaling parameter. Larger values result in a better approximation of the conditional operator. Applying the same method to the Fiala tire model results in a highly complex expression. Instead, the Fiala tire model presented in chapter 4 is approximated by a simplified Pacejka tire model presented in Equation 6-6 [36, 25]. This tire model does not require any conditional operators.

$$F_y = -\mu F_z \sin\left(B \arctan\left(C\alpha - E\left(C\alpha - \arctan\alpha\right)\right)\right) \cdot \cos\left(\arctan\left(B_x F_x\right)\right) \tag{6-6}$$

The Pacejka tire model parameters $B, C, E, B_x$ in Table 6-1 are identified for the front and rear axle of the vehicle.

| **Parameter** | **Front axle** | **Rear axle** |
|---|---|---|
| $B$ | 1.62 | 1.43 |
| $C$ | 17.03 | 17.78 |
| $E$ | 1.00 | 1.00 |
| $B_x$ | $2.5e^{-4}$ | $2.5e^{-4}$ |

**Table 6-1:** Pajecka tire model parameters

ACADO primarily uses the QPoases solver. However, others can be added [27]. ACADO is most commonly used in vehicle control [32], but its developers have started developing the newer Acados toolbox.

### Acados

Acados is an extension of ACADO. In Acados, the codebase written in $C^{++}$ has been replaced by an implementation in $C$-code that can be interfaced with in Matlab of Python using Casadi. Like MATMPC, the optimization problem and settings can be defined in Matlab, and $C$-code can be generated. Acados also provides options for embedded implementations on modern dSpace hardware. However, no standard method exists to implement variable timesteps.

These toolboxes can apply different solvers and solving techniques to improve the computation time. The effects of different solvers and variable timesteps are first evaluated to compare these toolboxes in their optimal configurations.

## 6-3 QP solvers

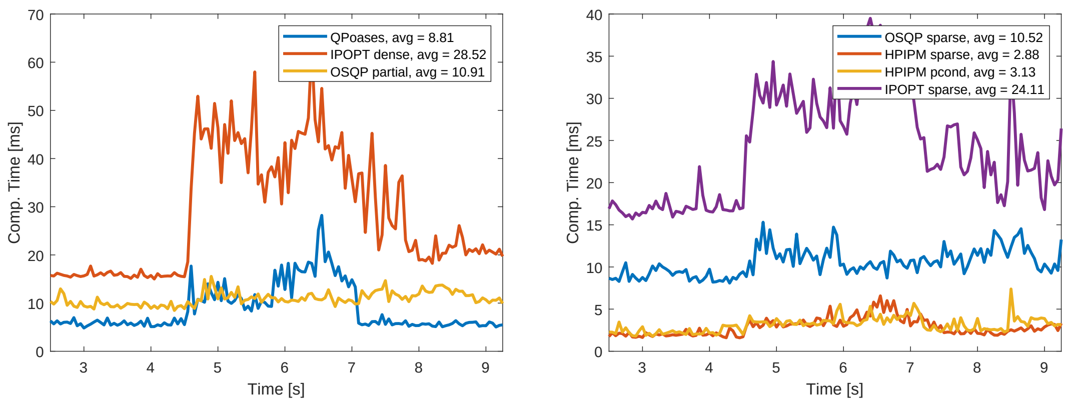
Different solvers can be used to solve the QP problem generated by the different toolboxes. There are important distinctions between these solvers.

Solvers can be divided into condensed and sparse solvers. Different condensing methods exist for condensed solvers, including partial and full condensing. Condensing exploits the principle that a new state can be expressed as a function of the current state and control inputs and eliminates it from the OCP [7]. Sparse solvers exploit sparse matrices, where the dimensionality can be reduced by eliminating zero entries. This eliminates algebraic variables and reduces the problem size [7]. Generally, reduced-sized problems can be solved faster.

Using the MATMPC toolbox, the computation time of different solvers is analyzed. The experimental setup in section 6-1 is used with the parameters specified in Table 6-2. The computation time for each solver class is presented in Figure 6-2. The sparse HPIMP solver has the lowest computation time of the solvers considered. The condensed solver QPoases has the second-lowest average computation time. It is found that condensed solvers have a greater maximum computation time compared to sparse solvers.

| $\dot{x}_{\mathbf{ref}}$ [m/s] | $\Delta s_a$ [m] | $\Delta s_o$ [m] | $\mu$ | $N$ |
|---|---|---|---|---|
| 17 | 20 | 25 | 0.9 | 50 |

**Table 6-2:** Scenario parameters QP solver comparison



**(a)** Condensed QP solvers **(b)** Sparse QP solvers

**Figure 6-2:** Compuatation time of QP solvers

## 6-4 Integration

In NMPC, candidate control trajectories are mapped to state trajectories [40]. State trajectories are used to compute the value of the objective function and check for feasibility [40]. This mapping is achieved through numerical integration of the nonlinear dynamics [40].

In CIC, there is a trade-off between the accuracy and computational complexity of the integrator. Higher integrator accuracy allows for larger timesteps that reduce the number of steps in the prediction horizon $N$ for the same time window. Reducing the number of steps in the prediction horizon decreases computational complexity. However, more complex integrators require more function evaluations of the system dynamics. Runge Kutta methods can be applied to increase the integrator's accuracy [40]. Runge-Kutta 2th order (RK2) has been found to offer a good balance for CIC [2].

An increased integrator timestep can reduce the number of steps $N$ in a time horizon $P_T$, which reduces computational complexity. However, the integrator must continue to capture the system dynamics with acceptable accuracy. In one-level CIC, there is another limitation. Larger distances between evaluation points can lead to problems in avoiding smaller obstacles, as exemplified in Figure 6-3. The controller encounters no costs for collision avoidance as all integration points lay outside the minimum obstacle distance.

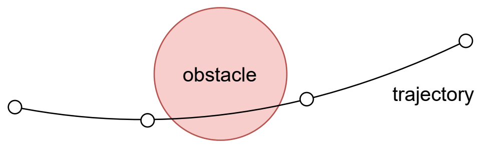


**Figure 6-3:** Integrating points in collision avoidance

### 6-4-1 Variable timesteps

Variable timesteps can be applied to decrease the number of steps in the prediction horizon [8]. Furthermore, variable timesteps allow for larger timesteps without significantly impacting collision avoidance performance.

The timestep between integrations can be changed from short to long over the prediction horizon. Reducing the prediction horizon can reduce the computation time, reducing the input, search, and solution space of the OCP for the same prediction time window.

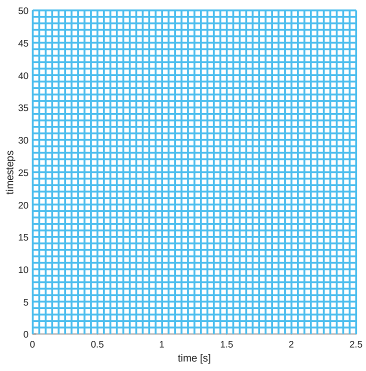
Increasing the time step along the prediction horizon ensures accuracy at the start and provides predictions increasingly further ahead [15]. Variable timesteps can achieve equal prediction time windows with fewer steps than smaller timesteps throughout the horizon.

In CIC, initial small time steps are primarily used for stability control and tracking decisions, and larger time steps capture the effects on collision avoidance [11]. Variable timesteps can reduce the computational complexity of NMPC by reducing the number of steps in the prediction horizon while ensuring accuracy and predictions sufficiently far ahead.

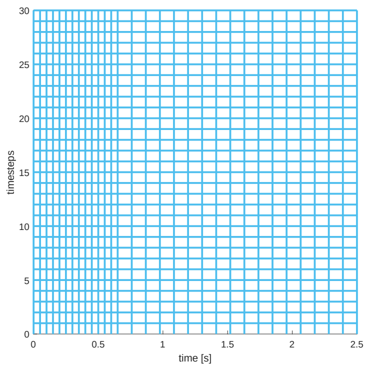
Variable timesteps can reduce the computation time. However, careful implementation is required to prevent undesirable behavior. Due to timestep differences, this behavior can be introduced through artificially scaled cost functions and constraint equations. In Linear Time Invariant (LTI) MPC, the costs and constraints must be weighted to account for the different timesteps [11].

Moving block strategies address this problem by evaluating the dynamics at a fixed lower timestep and keeping control actions constant for some integration points [30]. This allows variable timesteps to be implemented in SQP. Moving block strategies can be implemented by specifying a non-uniform integration grid for the input space $u$ of the OCP problem. A conventional uniform and non-uniform integration grid are presented in Figure 6-4. From Figure 6-4, it can be observed that the number of points on the integration grid is greatly reduced. The same prediction time window is obtained using small timesteps in the beginning and larger timesteps near the end of the prediction horizon.

Variable timesteps are implemented by assuming a constant prediction time window $P_T$, initial timestep $dt_1$, initial timestep horizon $N_1$, and total prediction horizon $N$. A moving block



**(a)** Uniform grid ($N = 50$)



**(b)** Non-uniform grid ($N = 30$)

**Figure 6-4:** Integration grids

strategy is implemented using a non-uniform integration grid created with an integration point at each variable timestep $dt(n)$. This timestep $dt(n)$ is specified as:

$$dt(n) = \begin{cases} dt_1 & \text{if } n < N_1 \\ dt_2 & \text{otherwise} \end{cases} \tag{6-7}$$

$$dt_2 = \frac{P_T - dt_1 N_1}{N - N_1} \approx n_{dt} dt_1, \quad n_{dt} \in \mathbb{Z} \| n_{dt} \geq 1 \tag{6-8}$$

Variable timesteps are applied to the best condensed and sparse solvers, namely QPoases and HPIPM. The effects of variable timesteps are explored for an initial timestep $dt_1$ of 0.05. The first entry uses no variable timesteps $N_1 = N = 50$, which is applied to the baseline [2]. Different total timesteps $N$ and initial timesteps $N_1$ are applied, and the average and maximum computation times are found for the following scenarios:

| $\dot{x}_{\mathbf{ref}}$ [m/s] | $\Delta s_a$ [m] | $\Delta s_o$ [m] | $\mu$ |
|---|---|---|---|
| 20 | 22 | 15 | 0.9 |

**Table 6-3:** Scenario HPIPM

| $\dot{x}_0$ [m/s] | $\Delta s$ [m] | $\Delta s_o$ [m] | $\mu$ |
|---|---|---|---|
| 17 | 20 | 15 | 0.9 |

**Table 6-4:** Scenario QPoases

These scenarios are different as the performance of the solvers near the limits is explored. As HPIPM is faster, it generally handles more complex scenarios as the faster computations allow for more exploration. The computation times are averaged over three measurements. The computation times and the standard deviation error are presented in Tables Table 6-5 and Table 6-6.

These results show that the computation time can be greatly reduced by applying variable timesteps. Generally, smaller prediction horizons $N$ yield faster computation times as $N$ is proportional to the problem size. The average computation time was reduced by 40% and 75% and the maximum by 50% and 88% for HPIPM and QPoases, respectively. The computation time reduction is greater in non-sparse solvers as these benefit most from the reduced problem size. For HPIPM, the average and maximum computation was almost reduced to half the original value. For HPIPM, a higher value of $N_1$ generally yields better results. This trend does not appear for QPoases. The reductions in computation time indicate that variable timesteps can play an important role in improving the real-time performance of NMPC.

| $N$ | $N_1$ | **CPT avg** | **CPT max** |
|---|---|---|---|
| 50 | 50 | 3.42±0.04 | 14.42±3.20 |
| 40 | 10 | 2.77±0.03 | 13.88±4.71 |
| 40 | 20 | 2.83±0.10 | 13.72±1.78 |
| 40 | 30 | 2.86±0.03 | 10.55±3.51 |
| 35 | 10 | 2.63±0.03 | 9.40±2.20 |
| 35 | 20 | 2.45±0.04 | 8.70±2.02 |
| 35 | 25 | 2.65±0.12 | 8.33±0.85 |
| 30 | 5 | 2.31±0.09 | 7.71±0.83 |
| 30 | 10 | 2.51±0.13 | 14.32±3.34 |
| 30 | 20 | 2.20±0.02 | 7.26±0.95 |
| 25 | 5 | 1.93±0.05 | 7.02±1.34 |
| 25 | 10 | 1.94±0.07 | 7.83±1.30 |

**Table 6-5:** CPT HPIPM sparse

| $N$ | $N_1$ | **CPT avg** | **CPT max** |
|---|---|---|---|
| 50 | 50 | 15.13±0.29 | 121.17±4.53 |
| 40 | 10 | 9.85±0.12 | 86.62±12.47 |
| 40 | 20 | 8.67±0.26 | 54.93±5.25 |
| 40 | 30 | 8.88±0.15 | 67.3±3.80 |
| 35 | 10 | 7.33±0.26 | 39.98±5.46 |
| 35 | 20 | 6.65±0.21 | 35.36±2.87 |
| 35 | 25 | 6.98±0.20 | 47.94±4.36 |
| 30 | 5 | 5.07±0.08 | 22.66±1.12 |
| 30 | 10 | 5.2±0.11 | 25.35±3.33 |
| 30 | 20 | 5.04±0.19 | 27.49±2.16 |
| 25 | 5 | 4.08±0.40 | 15.11±1.47 |
| 25 | 10 | 3.87±0.10 | 20.34±1.06 |

**Table 6-6:** CPT QPoases condensed

## 6-5 Toolbox comparison

With the best solver and variable timesteps settings, the toolboxes used to perform SQP can be compared using PC-based simulations in IPG CarMaker. The toolboxes are compared in their best configuration to make a fair comparison. MATMPC can use variable timesteps as standard, and a custom non-uniform integration grid is implemented in the $C^{++}$ codebase of ACADO. Acados has no standard implementation for variable timesteps. MATMPC and ACADO can exploit the best variable timestep configuration of $N = 25$ and $N_1 = 5$. MATMPC and Acados can use the HPIPM solver, whereas ACADO is equipped with the QPoases solver. This forms the set of parameters in Table 6-7 for the scenario on which these toolboxes can be compared.

| Toolbox | $\dot{x}_{ref}$ **[m/s]** | $\Delta s_a$ **[m]** | $\Delta s_o$ **[m]** | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|---|---|
| ACADO | 15 | 20 | 20 | 0.9 | 25 | QPoases |
| Acados | 15 | 20 | 20 | 0.9 | 50 | HPIPM |
| MATMPC | 15 | 20 | 20 | 0.9 | 25 | HPIPM |

**Table 6-7:** Scenario parameters toolbox comparison

The computation times of CIC using these toolboxes for the scenario described by Table 6-7 are presented in Figure 6-5. It can be observed that ACADO has the lowest average computation time, and Acados has the highest average computation time. The longer prediction horizon in Acados can explain this. ACADO has relatively high maximum computation times, likely due to the QPoases solver. HPIPM has a relatively constant computation time.

The trajectories presented in Figure 6-6 show that MATMPC and ACADO have relatively similar trajectories. The longer prediction horizon used in Acados likely increases the costs of tracking the reference $ey_{ref}$, leading to a tighter trajectory around the first obstacle. MATMPC is used for PC-based simulations as it comes prepared with a large set of options and offers customizability. The downside of MATMPC is that it is not suited for embedded implementations by default. The controller is also validated on embedded hardware using ACADO. Information about the embedded implementation is presented in chapter 9.

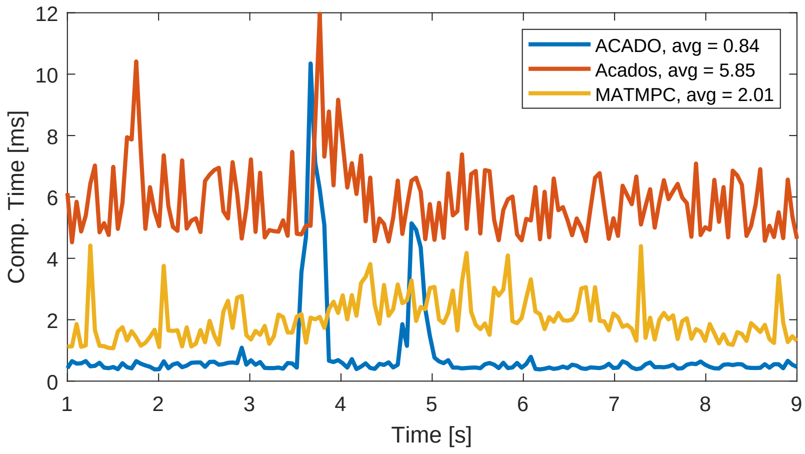


**Figure 6-5:** Computation times SQP toolboxes

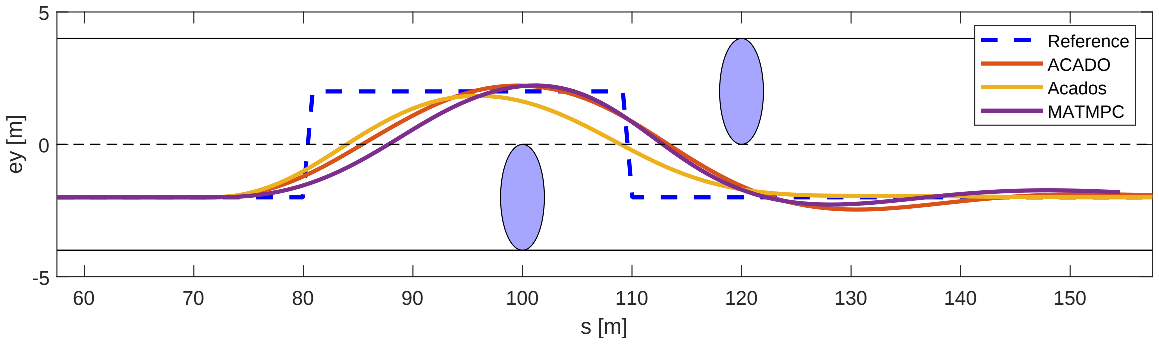


**Figure 6-6:** Trajectories SQP toolboxes

Chapter 7

# Closed-loop performance assessment

This chapter evaluates the closed-loop performance of the CIC controller. The effects of the reference trajectory, vehicle model, and distance formulation on the performance and computation time of the CIC controller are evaluated.

## 7-1 Influence of reference trajectory

The reference trajectory $ey_{\text{ref}(s)}$, is assumed to be provided by a global planner, as introduced in chapter 5. A simple trajectory can be used since collision avoidance is included in the OCP. The controller's performance is evaluated using the experimental setup presented in section 6-1 using the parameters in Table 7-1. The obstacles initially have a radius of 2 meters.

| $\dot{x}_{\textbf{ref}}$ **[m/s]** | $\Delta s_a$ **[m]** | $\Delta s_o$ **[m]** | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|---|
| 18 | 20 | 15 | 0.9 | 50 | HPIPM |

**Table 7-1:** Scenario parameters for trajectory evaluation

Two different trajectories are considered. The first is a straight trajectory that follows the initial lane. The second is a trajectory from a lane selector that selects the left lane at a distance $\Delta s_a$ from the obstacle and returns to the right lane between the obstacles. The vehicle's reference trajectories and trajectories are presented in Figure 7-1. The distance to the obstacles for both scenarios is presented in Figure 7-2.

The vehicle takes a much wider path around the first obstacle when the lane selector provides a reference. The computation times for both scenarios are stated in Table 7-2. Next to increasing the distance to the obstacles, the reference trajectory of the lane selector also leads to a reduced computation time. However, the increased computation time for the first trajectory is mainly caused by the high maximum time.

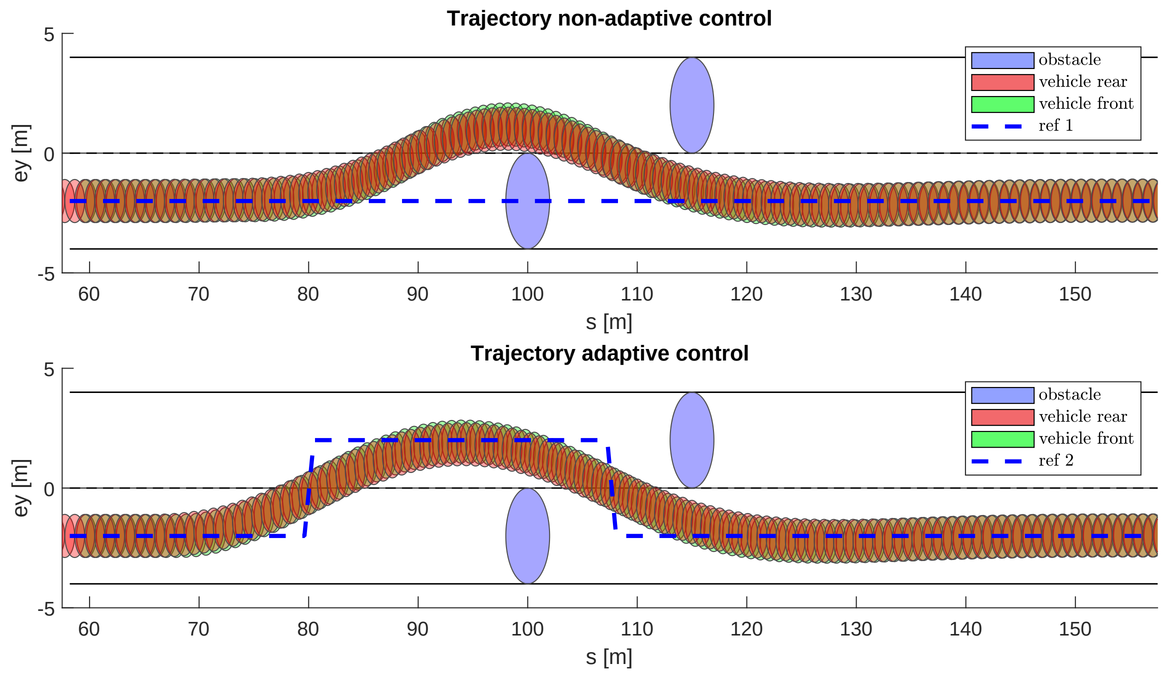


**Figure 7-1:** Reference trajectory tracking and collision avoidance

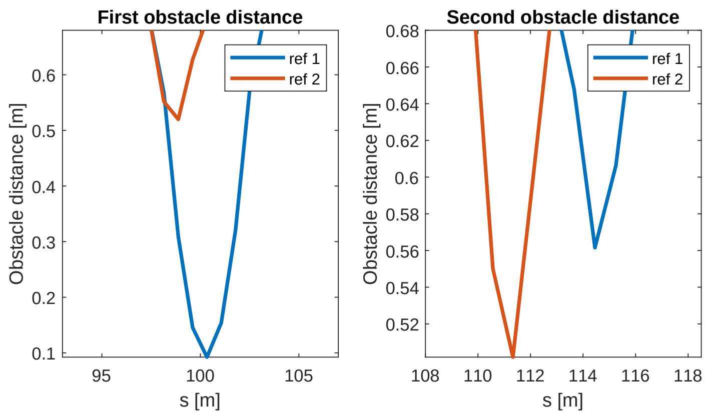


**Figure 7-2:** Obstacle distance for different reference trajectories

The same experiment is repeated to evaluate the impact of the obstacle's size. Figure 7-3 presents the trajectories where the size of the obstacles is increased to 3 meters. The computation times of this scenario are stated in Table 7-3.

| Distance model | CPT avg [ms] | CPT max [ms] |
|---|---|---|
| Constant reference | 6.71 | 239.01 |
| Lane selecting | 3.27 | 12.17 |

**Table 7-2:** Computation time reference types

These results show that providing a simple reference trajectory can improve the safety and computation time of the CIC controller in scenarios with small obstacles. For larger obstacles,

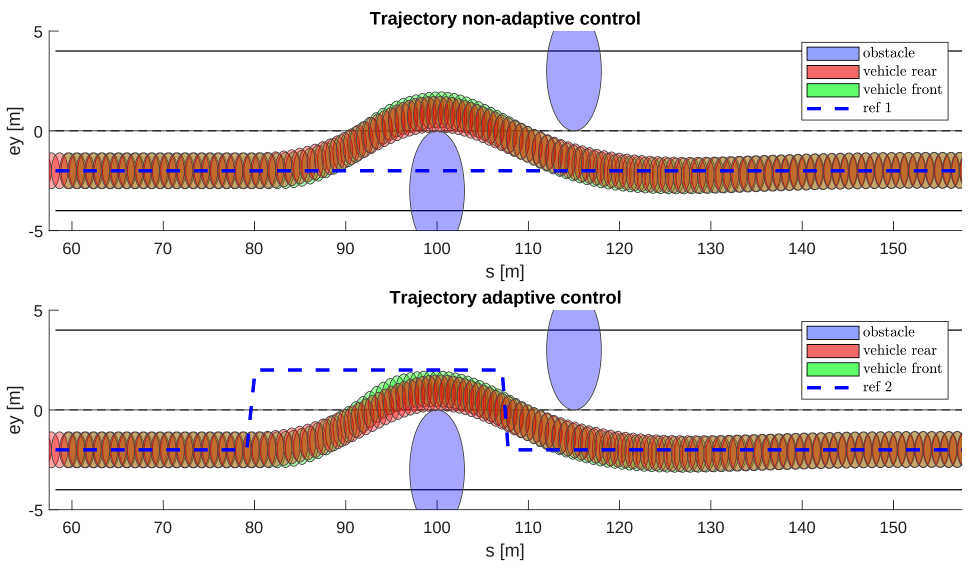


**Figure 7-3:** Reference trajectory tracking and collision avoidance

| **Distance model** | **CPT avg [ms]** | **CPT max [ms]** |
|---|---|---|
| Constant reference | 6.71 | 10.34 |
| Lane selecting | 4.05 | 10.12 |

**Table 7-3:** Computation time reference types for large obstacles

the reference trajectory from the lane selector offers fewer improvements. As the lane selector trajectory increases the performance of CIC, it is assumed to be available for the remaining experiments in this research.

## 7-2 Vehicle models

The CIC model presented in chapter 4 can be compared directly to the baseline model [2]. The CIC model includes an additional yaw moment resulting from differential braking. Compared to the baseline model, the CIC model has two more states, one extra input, and two additional constraints.

Both models are simulated using the experimental setup presented in section 6-1. The parameters of this maneuver are presented in Table 7-4. Executing this scenario results in the

| $\dot{x}_{\mathbf{ref}}$ **[m/s]** | $\Delta s_a$ **[m]** | $\Delta s_o$ **[m]** | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|---|
| 20.5 | 25 | 15 | 0.9 | 50 | HPIPM |

**Table 7-4:** Scenario parameters for vehicle model evaluation

trajectories presented in Figure 7-4. From these trajectories, a clear control strategy can be observed. The vehicle tends to get close to the first obstacle, making the second obstacle and road boundaries easier to avoid. While both trajectories are similar, the CIC vehicle model results in a marginally tighter trajectory, allowing it to recover from the maneuver faster.

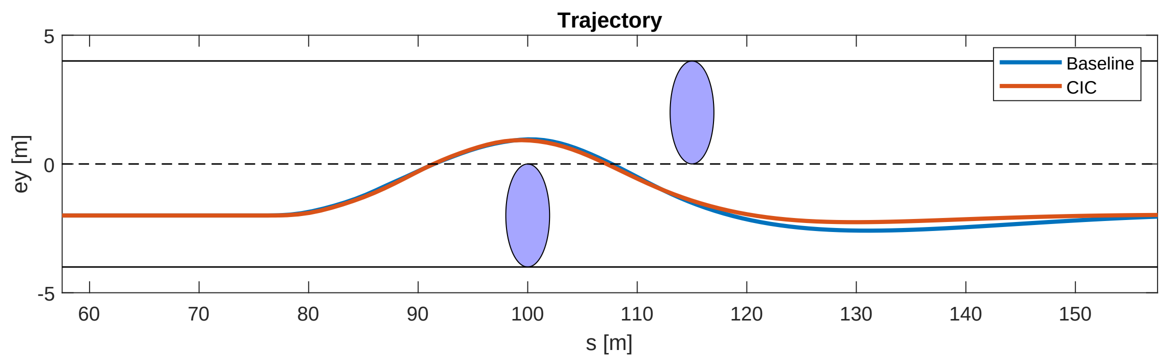


**Figure 7-4:** Trajectory of vehicle models

The input trajectory of both models is presented in Figure 7-5. The CIC model has a fourth control input $\lambda_y$ that allocates the brake distribution from left to right. The direction of the lateral brake allocation coincides with the direction of the turn. Generally, the control inputs of the CIC model are smoother, indicating that the baseline is pushed further to its control limits.

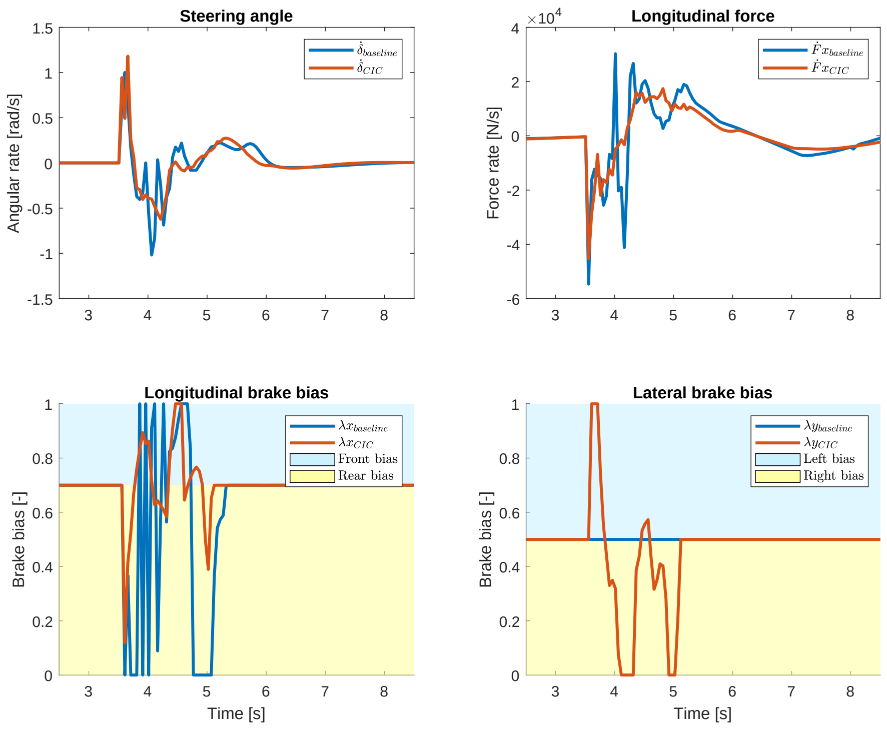


**Figure 7-5:** Vehicle model control inputs

The state trajectory of both models is presented in Figure 7-6. Because of differential braking, more control potential can be applied. This results in the vehicle driving a tighter trajectory around the first obstacle, yielding a maneuver with a lower lateral velocity. This means the car can brake less and keep a higher longitudinal velocity.

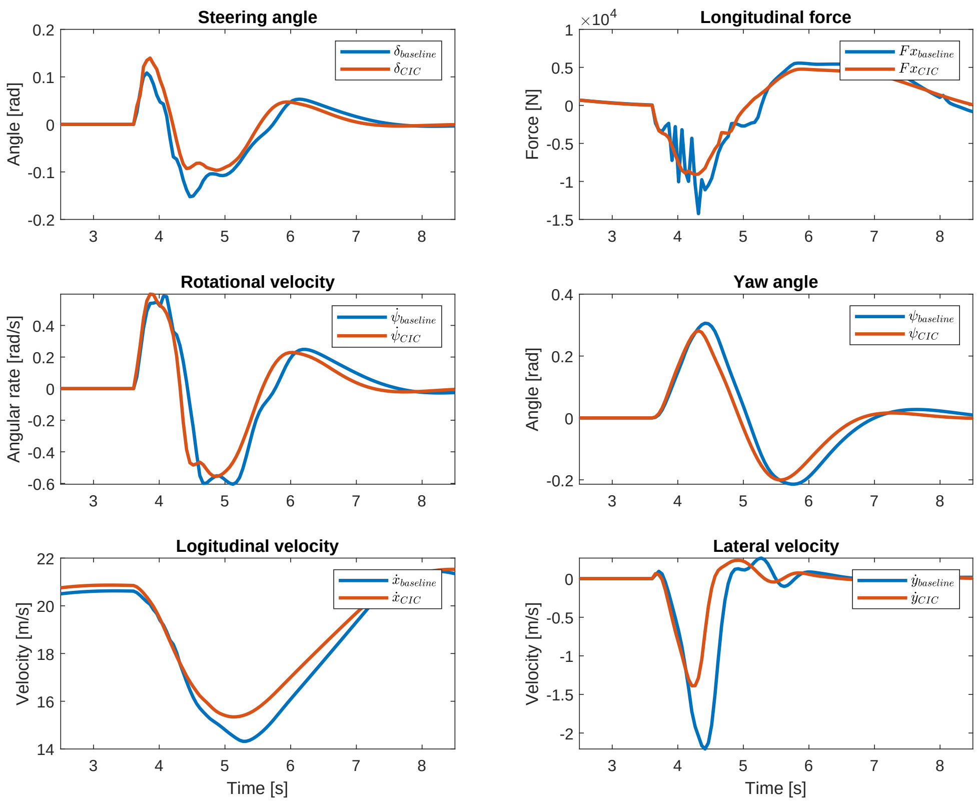


**Figure 7-6:** Inputs of vehicle models

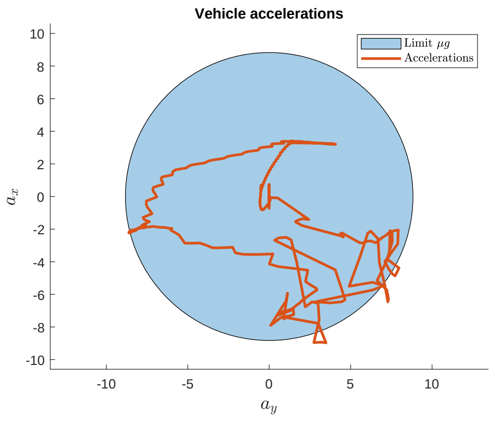
Employing more control potential by applying differential braking can make the CIC problem easier to solve. This claim is supported by the computation times of both models, presented in Table 7-5.

| **Vehicle model** | **CPT avg [ms]** | **CPT max [ms]** |
|---|---|---|
| Baseline [2] | 6.71 | 13.70 |
| CIC | 4.05 | 11.26 |

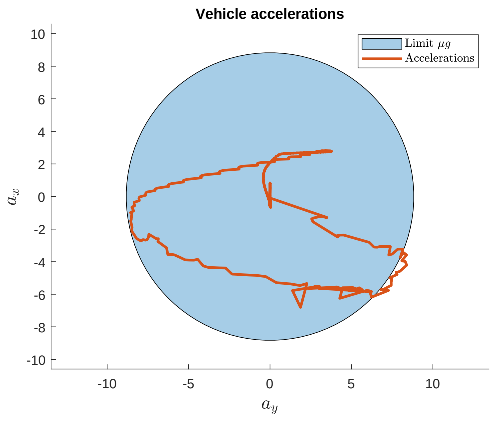
**Table 7-5:** Computation time with different vehicle models

The average computation time of the CIC model is lower despite having a greater input and state space and more nonlinear constraints. Therefore, adding differential braking can allow for a greater exploitation of the control potential. By exploiting more control potential, the OCP becomes easier to solver, reducing the average computation time.

The aggressiveness of the maneuver is demonstrated in the *GG* curve, which presents the accelerations relative to the acceleration $\mu g$. The *GG* curve is presented in Figure 7-7. The vehicle's accelerations reach the edges of the circle defined by $\mu g$, indicating that the maneuver is aggressive with an acceleration between $0.8g$ and $1.1g$ for $\mu = 0.9$. To verify that the CIC model adheres to the friction limits, the baseline and CIC friction limits are presented in Figure 7-8 and Figure 7-9 respectively.



**(a)** Accelerations baseline



**(b)** Accelerations CIC

**Figure 7-7:** Vehicle model accelerations

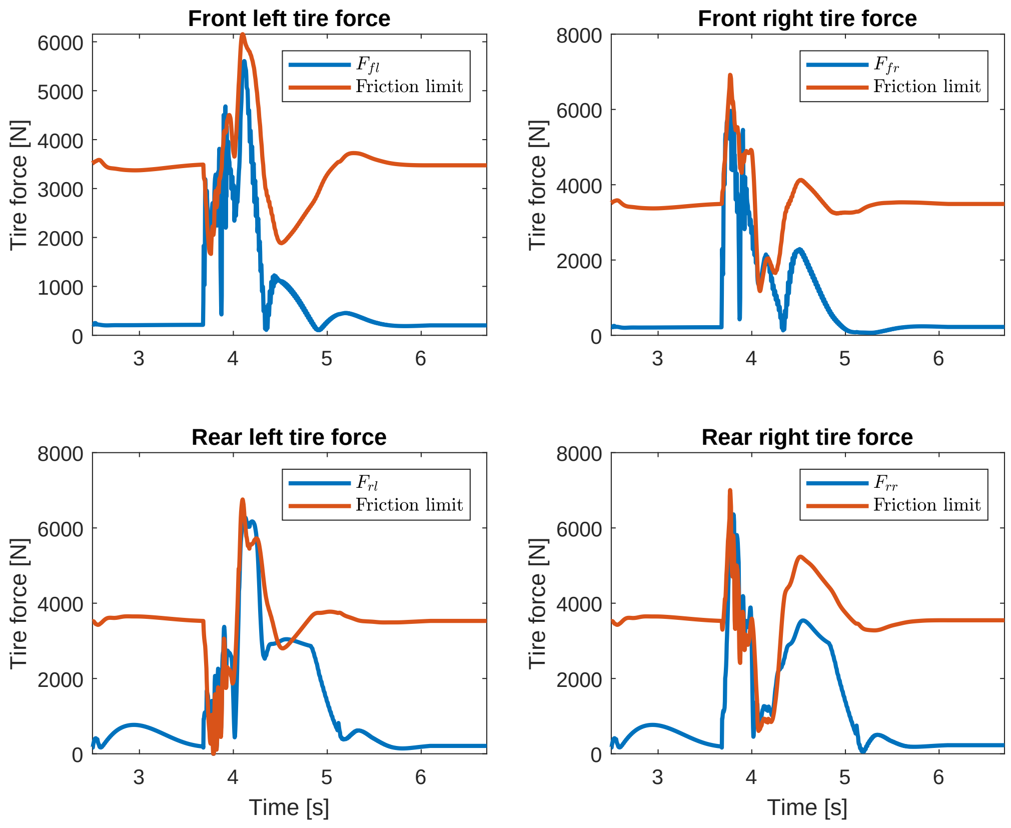


**Figure 7-8:** Friction limit constraints baseline model

There are minor temporary violations of the friction limits in both the baseline model and the CIC model, likely due to a difference in control frequency and the frequency of the plant

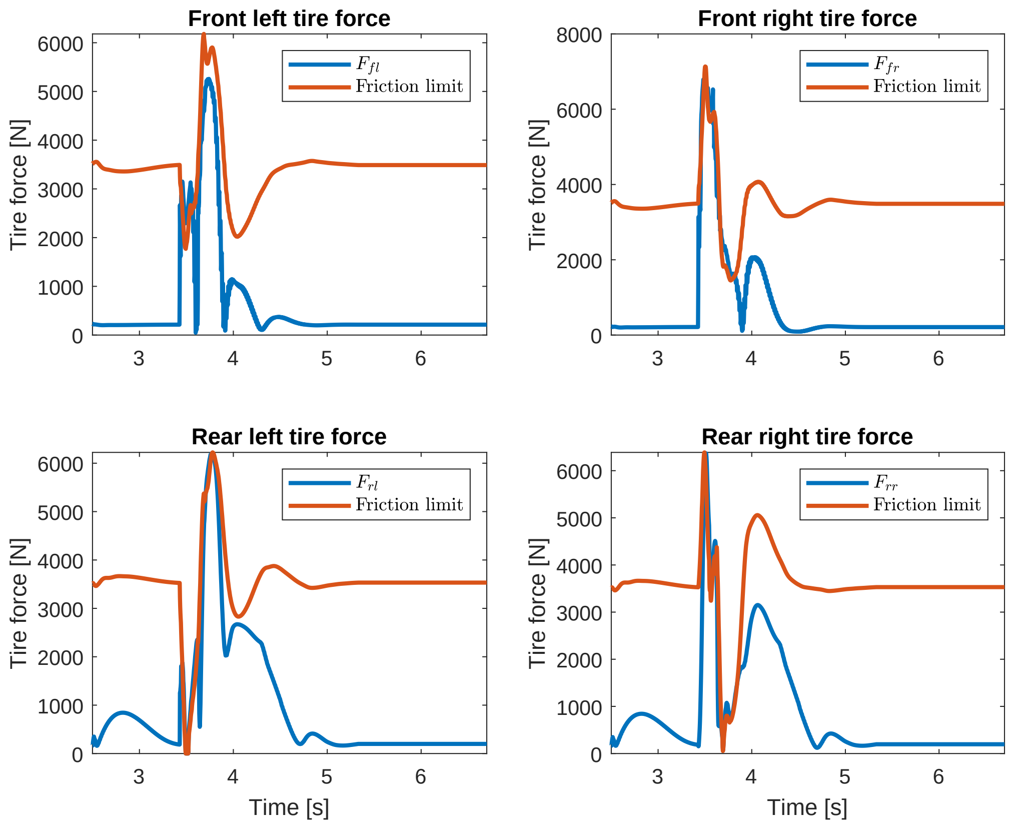


**Figure 7-9:** Friction limit constraints CIC model

simulations. Generally, both models allow for high exploitation of the available tire forces. However, the CIC model has smoother tire forces that almost perfectly adhere to the limits, allowing for greater exploitation of the control potential.

In this scenario, the CIC model employed more control potential through differential braking. More control potential utilization entails that the OCP becomes easier to solve, resulting in lower computation times despite the larger problem size. Furthermore, all tire friction constraints remain satisfied despite using an adjusted single-track model.

### 7-2-1 Vehicle model stability

The vehicle model used in the prediction model of the OCP can impact the controller's stabilizing performance. The OCP formulation derived in chapter 5 does not have explicit stability constraints. Instead, the controller indirectly penalizes instability if it leads to a collision during its long prediction horizon.

## 7-2-2 Stability limits

Stability limits can be formulated as steady-state limits on the yaw rate and lateral velocity, as shown in Equation 7-1 [11]. The limits form the stability envelope in Figure 7-10 [1].

$$\dot{\psi} \leq \min\left(\frac{F_{yf,\max}(1+l_f/l_r)}{m\dot{x}}, \frac{F_{yr,\max}(1+l_r/l_f)}{m\dot{x}}\right), \quad \dot{y} \leq \dot{x}\alpha_{r_{sat}} + l_r\dot{\psi} \tag{7-1}$$

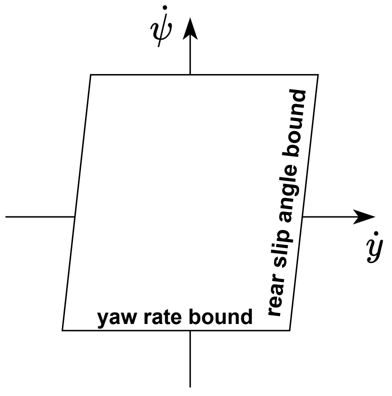


**Figure 7-10:** Stability envelope [1]

Alternatively, stability limits can be enforced on the sideslip angle $\beta$ [8]. $\beta$ is often limited in driver-focused vehicle stability control techniques such as Electronic Stability Control (ESC) [4]. In ESC, $|\beta|$ is often limited to around 4 or 5 degrees [4].

The stability constraints in Equation 7-1 are not enforced in the OCP definition described in chapter 5. However, these limits can provide insight into the vehicle stability conditions for both vehicle models. The stability conditions of the baseline and CIC vehicle models are presented in Figure 7-11 and Figure 7-12, respectively. The sideslip angles of both models are compared in Figure 7-13. These figures show that the CIC vehicle model can execute a similar collision avoidance maneuver with lower lateral velocity and sideslip angles than the baseline model. This demonstrates that differential braking could improve the stabilizing performance of the vehicle.

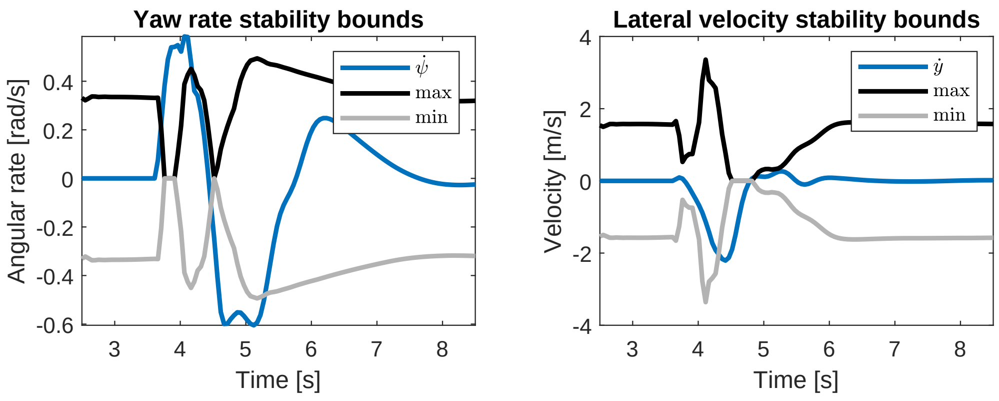


**Figure 7-11:** Stability baseline model

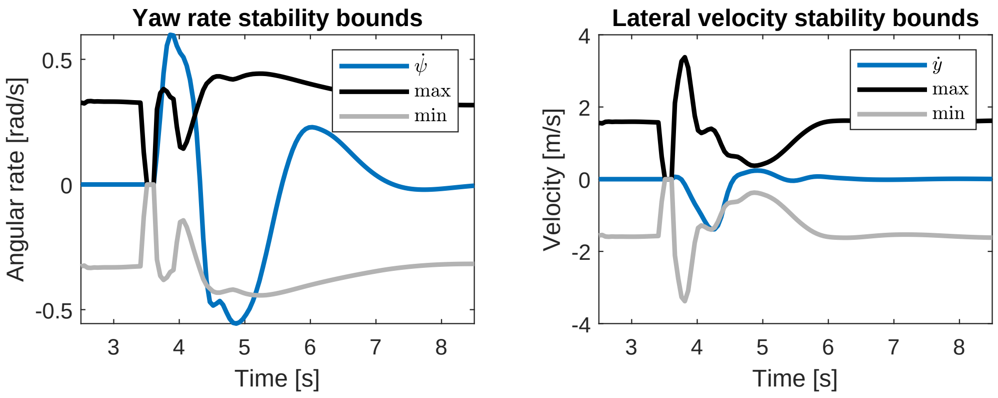


**Figure 7-12:** Stability CIC model

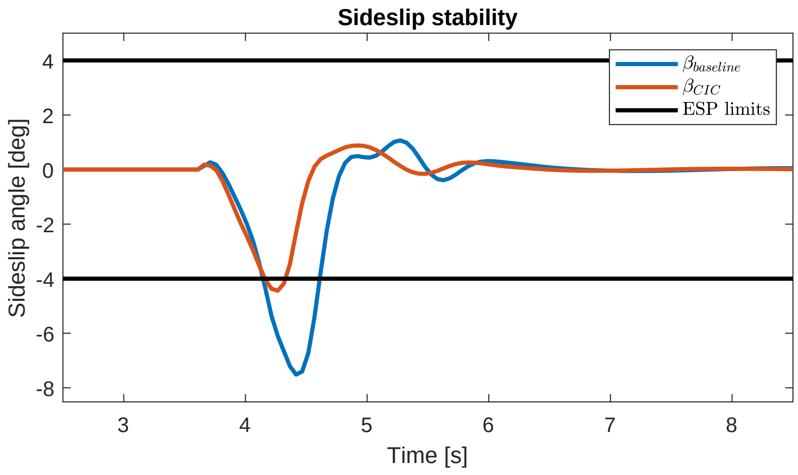


**Figure 7-13:** Sideslip angles of vehicle models

## 7-3 Collision avoidance formulations

The impact of model fidelity on the computation time is further investigated using different collision avoidance models presented in chapter 3. Here, the baseline distance model using multiple circles [2] is compared against the derived ellipsoidal and rectangular distance models.

The collision avoidance models are simulated using the experimental setup presented in section 6-1 with the parameters stated in Table 7-6. For this scenario, the performance of these models is analyzed based on time and accuracy.

| $\dot{x}_{\textbf{ref}}$ **[m/s]** | $\Delta s_a$ **[m]** | $\Delta s_o$ **[m]** | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|---|
| 18 | 25 | 15 | 0.9 | 50 | HPIPM |

**Table 7-6:** Scenario parameters collision avoidance formulation evaluation

For this scenario, the trajectories of each collision model are presented in Figure 7-14. Figure 7-14 shows relatively similar trajectories for the distance models. After avoiding the first obstacle, the trajectories differ more.

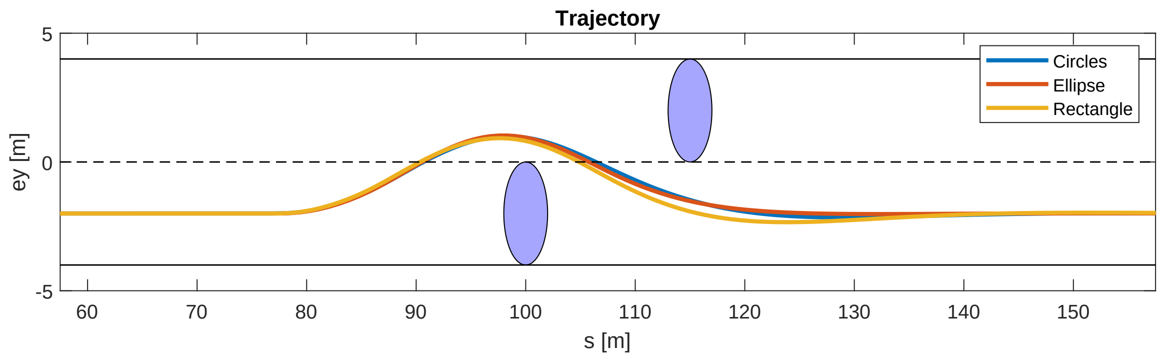


**Figure 7-14:** Collision model trajectories

The rectangular method is the most conservative of the distance models, capturing the full vehicle geometry by fitting right-angled corners to the vehicle. It is expected that the rectangular model would result in the largest distance to the obstacles as it would encounter greater costs for an equal trajectory than the other methods due to the more conservative nature of the formulation. This seems not to be the case in this scenario.

The execution time of distance model function model evaluations in chapter 3 demonstrates that the ellipsoidal and rectangular models have a reduced computation time compared to the multiple circle method. This does not directly entail that the computation time of the entire OCP will also be reduced. The computation times of the OCP for the distance models are presented in Table 7-7.

| **Distance model** | **CPT avg [ms]** | **CPT max [ms]** |
|---|---|---|
| Multiple circles | 3.09 | 8.38 |
| Ellipse | 3.11 | 8.48 |
| Rectangle | 3.15 | 7.45 |

**Table 7-7:** Computation time distance models

For this scenario, the average computation times are relatively equal, with the multiple circles having the minimal average and the rectangle having the lowest maximum time. Despite faster model evaluations, the computation time is not improved.

Since each distance model leads to a somewhat different optimization problem, their behavior can be difficult to analyze. To limit the effects of the reference trajectory, the obstacle radius $r_o$ is increased to 3 meters. Furthermore, the vehicle's reference velocity was increased to evaluate the performance at the controller's limits. Leading to the scenario in Table 7-8.

| $\dot{x}_{\textbf{ref}}$ **[m/s]** | $\Delta s_a$ **[m]** | $\Delta s_o$ **[m]** | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|---|
| 25 | 35 | 16 | 0.9 | 50 | HPIPM |

**Table 7-8:** Scenario parameters collision avoidance formulation evaluation

The trajectories for this scenario are presented in Figure 7-15. In this case, the multiple-circles method fails to find a collision-free trajectory, leading to constraint violations that saturate the tires. The ellipsoidal and rectangular methods are similar to the first scenario, where the rectangular model applies marginally more aggressive steering.

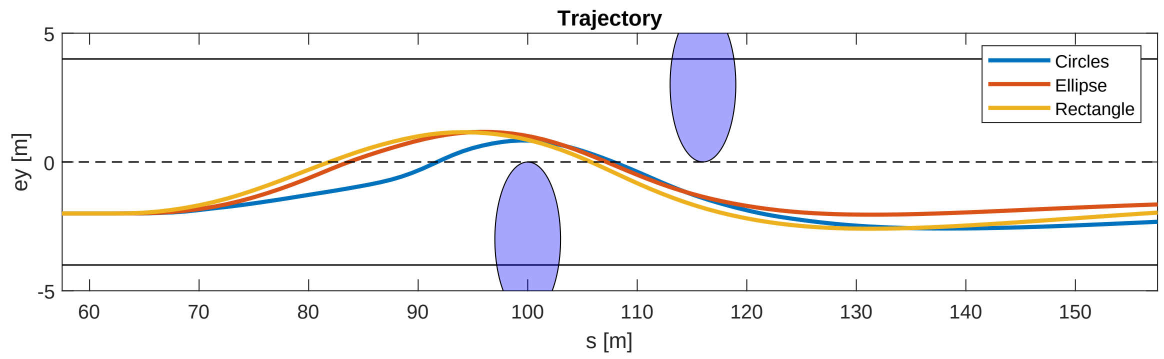


**Figure 7-15:** Collision model trajectories with large obstacles

The computation times at the limits of the OCP are shown in Table 7-9. In this scenario, the rectangular method strongly outperforms both other models, achieving the lowest average and maximum computation time.

It remains the case that comparing the distance models based on the closed-loop computation time remains complex, as it can depend on several factors. Results might differ significantly between scenarios. However, the rectangular and multiple circles methods consistently outperformed the ellipsoidal method.

| Distance model | CPT avg [ms] | CPT max [ms] |
|---|---|---|
| Multiple circles | 4.90 | 48.98 |
| Ellipse | 5.02 | 215.10 |
| Rectangle | 3.91 | 20.23 |

**Table 7-9:** Computation time distance models with large obstacles

A potential contributing factor to the lower computation times for the multiple circles method might be that the Jacobian and Hessian could be easier to approximate as they have a more simplified mathematical expression. The state space of the OCP is presented in Figure 7-16.

The state space illustrates that the rectangular method is the first to apply a large steering angle, indicating that it converges faster to a locally optimal trajectory. The ellipsoidal and multiple circles methods initially apply more braking and less steering.

The multiple circles method requires more time to converge to a trajectory that avoids both obstacles. This results in excessive braking, which could have temporarily violated saturation constraints. From the rotational velocity and steering angle of the multiple circles model, it can be derived that the controller is likely correcting a drift.

It is demonstrated that all distance models could be used in a real-time implementable CIC controller. The multiple-circle and rectangular methods generally achieved the best real-time performance.

The rectangular model can be considered the best distance model for CIC. It captures the vehicle's geometry most accurately, leading to a more conservative distance formulation that considers all parts of the vehicle in the expression. Despite the increased conservatives, the closed-loop computation time is lower or equal to the baseline method, which captures significantly less of the vehicle geometry.

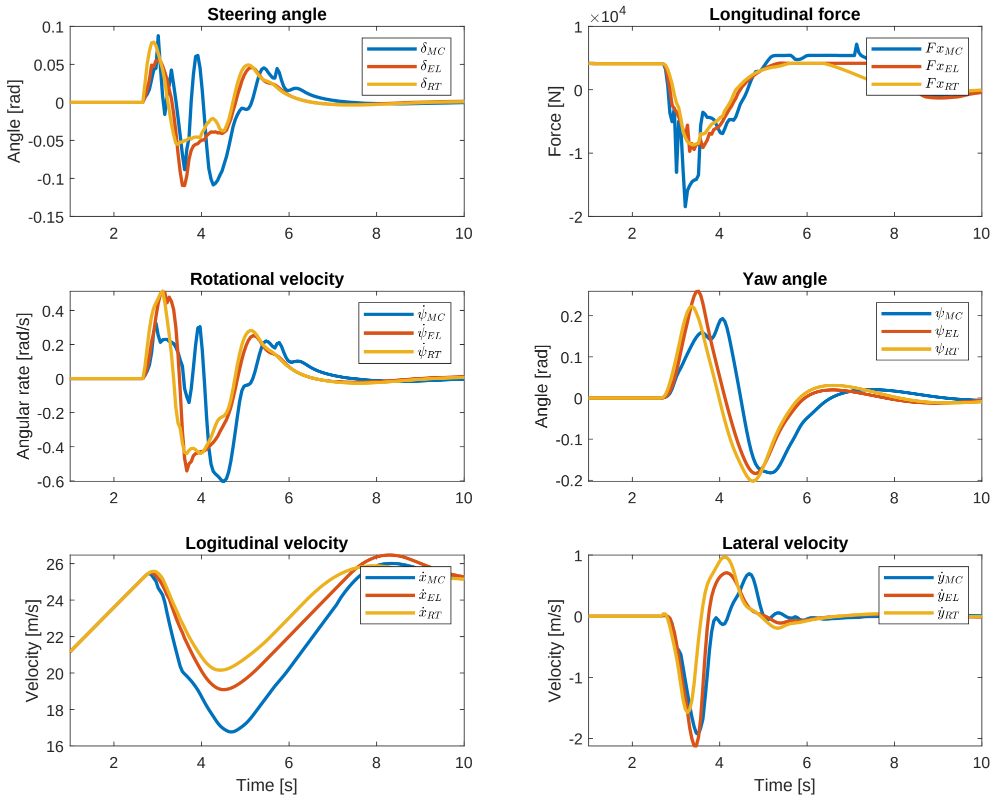


**Figure 7-16:** Vehicle state space for collision models with large obstacles

Chapter 8

# Robustness evaluation

The CIC controller must cope with the complex dynamic environment in which autonomous vehicles operate. Robustness challenges can be posed by changing road-tire friction coefficients and road curvatures or by dynamic obstacle behavior. This chapter evaluates the performance of the best controller from chapter 7 in scenarios where these robustness challenges are posed.

As described in chapter 5, the CIC controller with two obstacles has ten parameters describing the dynamic environment. As shown in Equation 8-1.

$$\hat{p} = [s_{o1}, e_{oy1}, r_{o1}, s_{o2}, e_{oy2}, r_{o2}, rb_r, rb_l, \mu, \kappa] \tag{8-1}$$

This chapter evaluates robustness for changing values of $\mu$, $\kappa$, and the location of obstacles.

## 8-1 State and friction estimator

The road-tire friction coefficient can be estimated to improve robustness and model accuracy [39]. Estimating the road-tire friction coefficient is important in differential braking control for the accuracy of tire saturation constraints and prediction model dynamics. Furthermore, prediction model states might not be available as some states can not be measured or measuring is economically unviable.

A UKF is constructed to estimate the states of the CIC prediction model and the tire-road friction coefficient. The CIC vehicle model is used to generate a model-based prediction, and a Monte Carlo simulation is used to describe the dynamics of the tire-road friction coefficient [16].

To obtain an accurate value of $\mu$ using the UKF, it is chosen to use the same prediction model as used by the controller. This ensures that the estimated parameter of $\mu$ reduces the error between the CIC prediction model and the plant.

Two state and friction estimators were developed in Appendix C. A UKF that uses the rotational wheel dynamics to determine the longitudinal tire forces and a UKF that assumes ideal tire force actuation. The longitudinal tire forces can be estimated using the rotational dynamics of a wheel presented in Equation 8-2 [16].

$$I_w \dot{\omega}_f = T_w - M_r - F_x r_w \tag{8-2}$$

Here, $\omega_w$, $I_{w_i}$, and $r_{w_i}$ are the rotational velocity, inertia and radius of the wheel respectively. $T_w$ is the total applied torque at the wheel and $M_r$ is the torque of wheel resistance. To describe the rotational dynamics accurately, each wheel should be considered separately. No set of parameters could be identified for which the wheel dynamics closely approximated the artificial measurements from the IPG vehicle model. However, Since the control inputs from the CIC controller respect the friction limits, the approximation in Equation 8-3 is applied.

$$F_{x_f} \approx \frac{T_f}{r_w}, \quad F_{x_r} \approx \frac{T_r}{r_w} \tag{8-3}$$

Therefore, the second UKF developed in Appendix C is implemented in closed-loop simulations. The parameter space $\hat{\xi}$ of this UKF is created using the cartesian dynamics of the CIC vehicle model $(\dot{x}, \dot{y}, \dot{\psi})$ and friction coefficient $\mu$. For the control inputs of the filter $u_{UKF}$, the control inputs to the IPG model are considered. It is assumed that only standard vehicle sensors can be used. These include accelerometers, gyroscopes, and wheel speed sensors. Artificial measurements can be taken from the IPG model that forms the measurement vector $y_{UKF}$. The parameter, input, and measurement vectors are presented in Equation 8-4.

$$\begin{aligned} \hat{\xi} &= [\dot{x}, \dot{y}, \dot{\psi}, \mu]^T \\ u_{UKF} &= [\delta_{sw}, T_f, T_r]^T \\ y &= [\dot{x}, \ddot{x}, \ddot{y}, \dot{\psi}]^T \end{aligned} \tag{8-4}$$

The design and the tools used to implement the UKF are derived from the work of Hamann et al. [16]. The details of the UKF algorithm used are presented in Appendix E.

## 8-1-1 Friction estimation

The friction estimator is tuned on a trajectory for which the friction coefficient changes during an aggressive double-lane change maneuver. This is achieved in IPG CarMaker by specifying a road section with a lower friction coefficient just before the vehicle encounters two obstacles.

The effects of online fiction estimation on the success rate of aggressive maneuvers for collision avoidance have been evaluated in previous works [39]. Online friction estimation can increase the success rate significantly as the robustness issues posted by the changing friction coefficient can be reduced [39]. A scenario is constructed to demonstrate that this extends to CIC and to study the effect on the computation time. This scenario is created using the experimental setup in section 6-1, and the environment parameters are specified in Table 8-1. In the experiment, the friction coefficient is suddenly reduced from 0.9 to 0.4 when the vehicle is at a distance $\Delta s_a$ to the first obstacle. The friction-adaptive closed-loop controlled is evaluated against a CIC controller that assumes a constant friction coefficient of $\mu = 0.9$. The UKF is first tested offline in this scenario, resulting in the estimated states and friction coefficient presented in Figure 8-1.

| $\dot{x}_0$ [m/s] | $\Delta s_a$ [m] | $\Delta s_o$ [m] | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|---|
| 15 | 30 | 15 | 0.9-0.4 | 50 | HPIPM |

**Table 8-1:** Scenario parameters uncertain friction

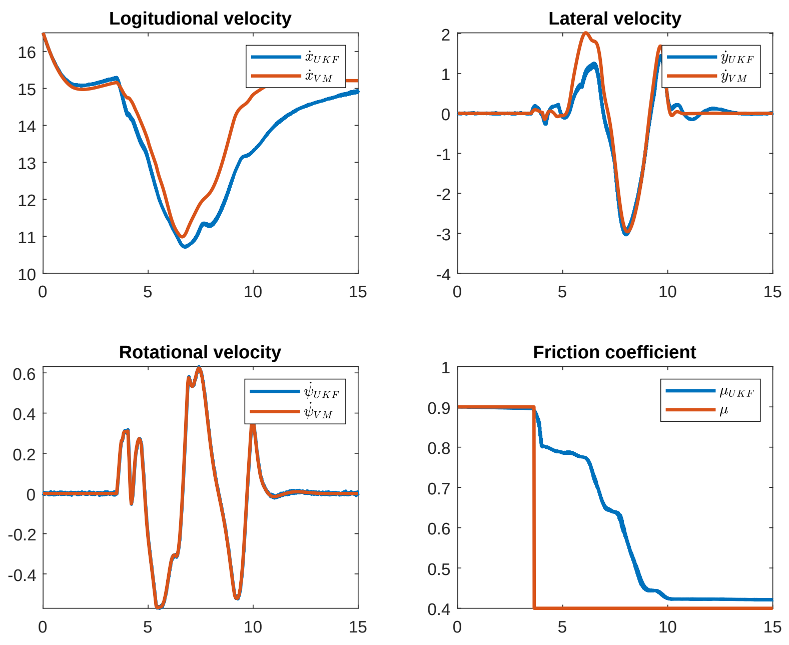


**Figure 8-1:** Offline UKF state and friction estimates

The UKF can estimate the state space and friction coefficient. The real-time collision avoidance performance of the CIC with friction estimation can be compared to the case where a fixed friction coefficient $\mu = 0.9$ is assumed to demonstrate the impact of online friction estimation.

The online estimated friction coefficient during the executed maneuver is presented Figure 8-2. When the friction coefficient changes at $s = 70$, the friction estimate decreases to 0.49 when $s = 97$ around 1.3 seconds later. Here, the prediction is the most accurate, likely since this is near the limit of handling of the vehicle. Near the limits of handling the friction coefficient estimates are generally more accurate [17].

The non-adaptive and adaptive controller trajectories are presented in Figure 8-3. The non-adaptive controller collides with the second obstacle and causes violent oscillations when correcting a drift.

The non-adaptive controller results in a collision, and the adaptive controller performs a successful evasive maneuver. The difference between the two controllers can be more clearly evaluated based on their state space, presented in Figure 8-4.

For the values of the system states, it can be observed that the non-adaptive controller performs greater steering actions and fewer braking actions. The planned control actions saturate the tires as the prediction model assumes a friction coefficient of ($\mu = 0.9$). This discrepancy between the prediction model and the plant thus leads to a colliding trajectory.

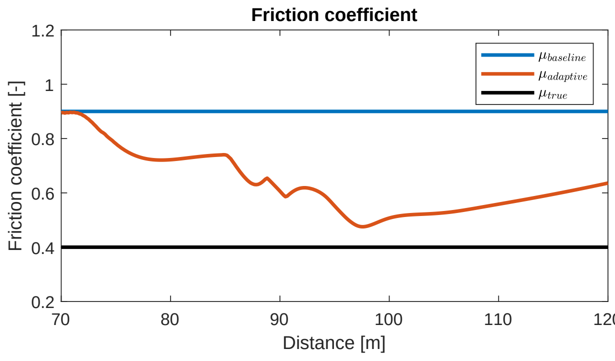


**Figure 8-2:** Online estimated friction coefficient

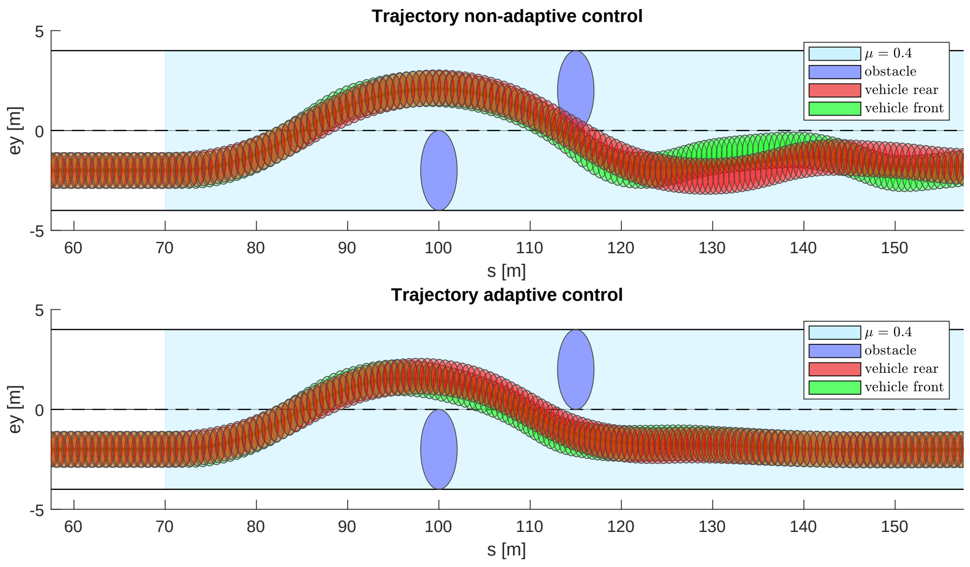


**Figure 8-3:** IPG scenario with uncertain friction

The adaptive controller applies significantly more braking as it derives that with the initial velocity and the predicted friction coefficient, the trajectory is not collision-free.

The effect of online friction estimation on the computation time is shown in Table 8-2. The adaptive controller did not have a significantly lower average computation time for this scenario. However, the adaptive controller did perform a collision-free evasive maneuver.

| Distance model | CPT avg [ms] | CPT max [ms] |
|---|---|---|
| non-adaptive | 3.96 | 8.01 |
| adaptive | 3.59 | 10.47 |

**Table 8-2:** Computation time for adaptive CIC

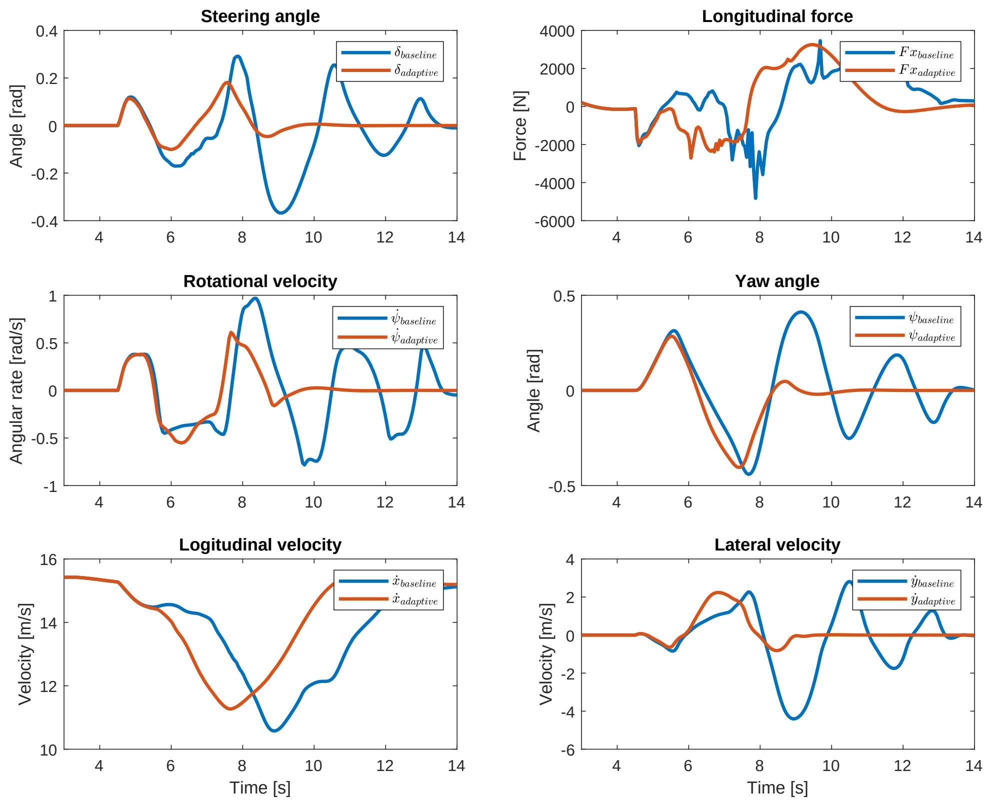


**Figure 8-4:** State space adaptive CIC

## 8-2 Road curvature estimator

Thus far, CIC has been mainly tested on straight roads. In the real world, the CIC controller must cope with changing road curvatures. This can be a source of disturbance to the controller, posing robustness issues.

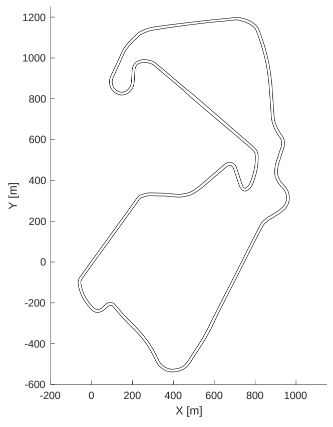
For the CIC controller to work on highly curved roads with no road curvature profile, the road curvature can be estimated based on a cartesian-based definition of the road, such as available on maps. The road curvature can be estimated using three points in the cartesian frame. However, the curvature approximation can be greatly simplified if these points are assumed to be an equal distance $\Delta s$ apart. Both methods are as presented in Appendix C.

These curvature estimator were used to derive a curvature profile for the Silverstone racetrack, presented in Figures 8-5a and 8-5b. This curvature profile simulates the behavior of the CIC controller on strongly curved roads. The track data was retrieved from the TUM Institute of Automotive Technology [1]

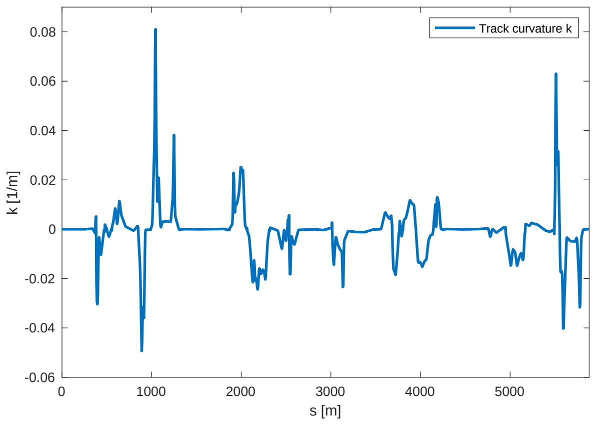
This curvature profile simulates the CIC controller on the racetrack. The parameters in Table 8-3 are used to describe the scenario.

With the estimated curvature profile $\kappa(s)$, the CIC controller was simulated on the Silverstone

[1]Racetrack database: https://github.com/TUMFTM/racetrack-database



(a) Silverstone layout



(b) Silverstone curvature profile

**Figure 8-5:** Silverstone curvature estimation

| $\dot{x}_{\mathbf{ref}}$ **[m/s]** | $ey_{\mathbf{ref}}$ **[m]** | $\mu$ | $N$ | **Solver** |
|---|---|---|---|---|
| 25 | 0 | 0.9 | 50 | HPIPM |

**Table 8-3:** Scenario parameters racetrack

racetrack. The full trajectory is presented in Appendix C. A particular section of the track is highlighted in Figure 8-6. In this track segment, it can be observed that the controller follows a racing line as it aims to minimize the error to its target velocity $\dot{x}_{\text{ref}}$. Since $ey_{\text{ref}} = 0$, the vehicle drives in the middle of the track on the straights.

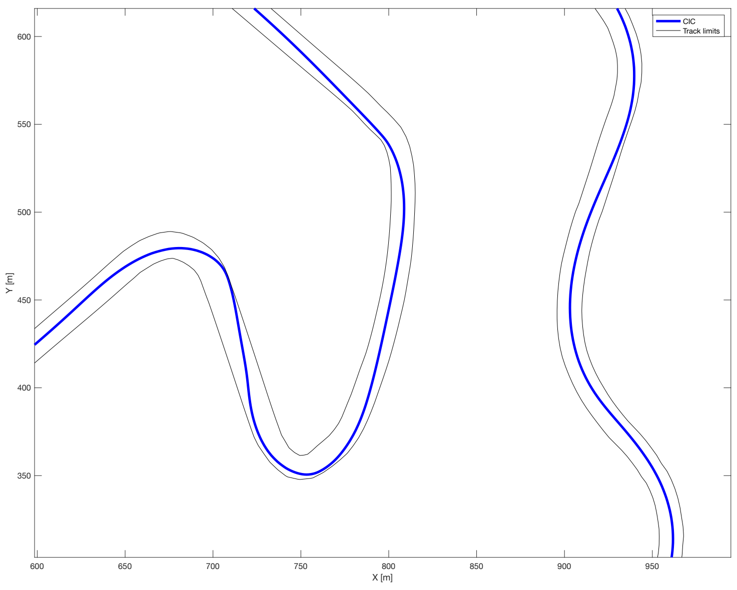


**Figure 8-6:** CIC on Silverstone

This scenario demonstrates that CIC can avoid the road boundaries on highly curved road profiles. The smallest momentary turning radius of the track is around 12 meters. This is close to the minimum turning radius of the vehicle. This demonstrates that the controller can effectively apply braking, acceleration, and steering control inputs to negotiate these tight turns at high velocity.

## 8-3 Dynamic obstacles

Another source of disturbance and uncertainty stems from obstacles' dynamic and uncertain behavior. To model the behavior of obstacles, it is assumed that the position and velocity of each obstacle are available at the beginning of the prediction horizon.

The position $p_o$ and the velocity $v_o$ of an obstacle are described in spatial coordinates $(s, e_y)$. In the controller, the velocity of the obstacles is assumed to be constant throughout the prediction horizon. When a new estimate of the obstacle location and position is available, the parameters can be updated in a future execution of the controller. The location of the obstacles is updated as described in chapter 3.

Two scenarios are investigated to evaluate the effects of dynamic obstacle behavior on the CIC controller. In the first scenario, two obstacles are considered traveling in opposite directions on a two-lane road. The obstacles travel much slower than the agent vehicle and are observed around 1 second before a collision would occur. Since an obstacle is approaching in the other lane, the slower obstacle in the same lane as the vehicle can not be overtaken at first. The vehicle must brake hard to prevent an accident with the significantly slower obstacle and wait to overtake until the approaching obstacle in the other lane has passed.

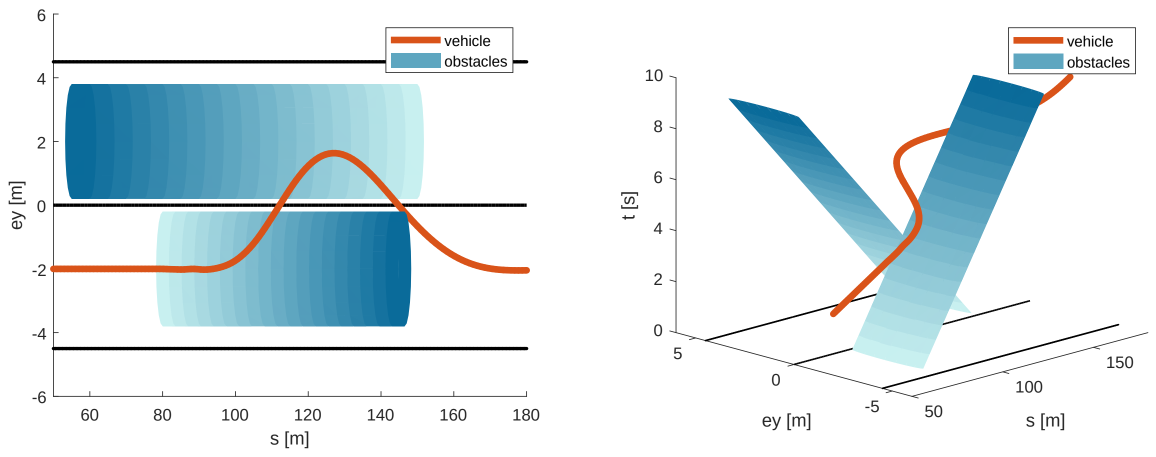


**Figure 8-7:** Dynamic obstacles blocked overtaking

The trajectories of the vehicle and obstacles in this scenario are shown in Figure 8-7. On the left, a top-down view is presented, where the positions of the obstacles are shaded from light to dark as the obstacle moves along their paths. On the right, these trajectories are shown in three dimensions with respect to $(s, e_y)$ and the time. In the right plot, the trajectory's slope indicates the velocity. The vehicle stays in its lane despite quickly approaching the much slower obstacle. Instead of changing lanes, the vehicle brakes and waits with the overtaking maneuver until the obstacle in the opposite lane has passed. Figure 8-7 precisely describes the controller's expected behavior.

Another possible scenario is that the obstacle changes lanes while the agent vehicle attempts an overtake. In this scenario, the vehicle must abandon its overtake maneuver, brake, and return to the original lane. This maneuver is presented in Figure 8-8, where the trajectories are presented similarly to Figure 8-7. The slope of both trajectories with respect to $s$ is relatively similar, indicating that the vehicle and obstacle are traveling at similar velocities. When the vehicle initiates an overtake of the obstacle, the obstacle moves over to the overtaking lane. In this scenario, the drivable space of the controller rapidly shrinks as the obstacle moves in front of it. The vehicle correctly extrapolates the velocity of the obstacle and decides to change back to the original lane to overtake from the left.

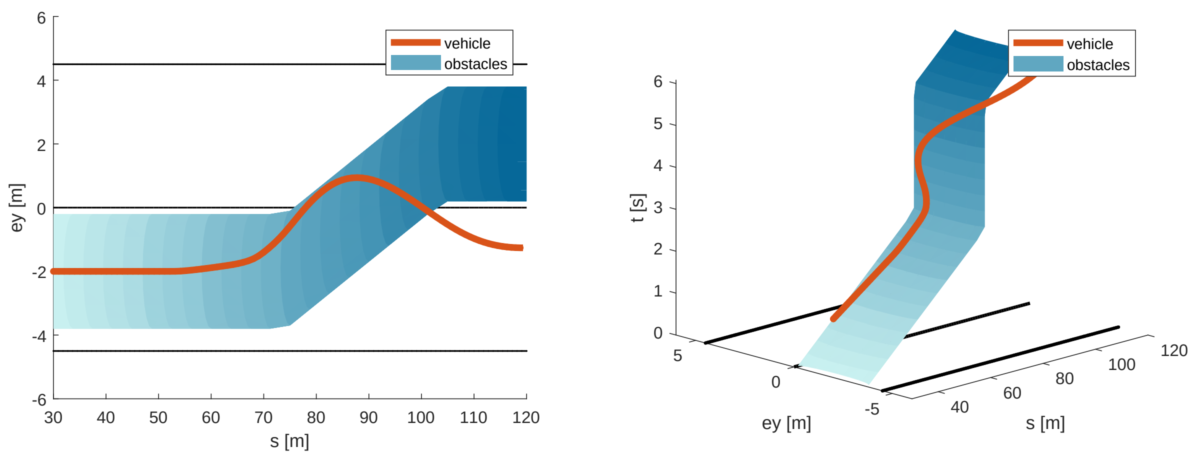


**Figure 8-8:** Dynamic obstacles changing lanes

These two scenarios demonstrate that the CIC controller can avoid dynamic obstacles in complex collision avoidance scenarios.

Chapter 9

# Hardware in the loop

The CIC controller is implemented on several dSpace embedded control platforms to evaluate the real-time performance. These platforms are purpose-built for the automotive sector to allow for rapid implementation and testing of vehicle control systems. Controllers can be implemented using $C$-code or, more commonly, by building a Simulink model and compiling it to $C$-code.

## 9-1 Platforms

Out of the toolboxes evaluated in chapter 6, ACADO and Acados can be applied on dSpace embedded platforms. Three of such platforms were employed to perform HIL simulations of the CIC controller, these include:

- AutoBox (ABX) DS1007,
- MicroAutoBox (MABX) DS1404,
- SCALEXIO (SCX).

The MABX controller presented in Figure 9-1 was first used to implement the CIC controller. It was found that with a long prediction horizon ($N = 50$), the CIC problem developed in ACADO can exceed its memory of 16MB. Next to reducing the computation time, variable timesteps also reduce the storage size of the OCP.

The aim is to exploit the full performance of the embedded controller. Therefore, it is desired to solve the OCP and simulate the measurements of the validation model on separate systems. To achieve this, the ABX controller from the developmental TUDelft autonomous vehicle can be used. This vehicle is presented in Figure 9-2.

Using the vehicle CAN network, the ABX controller can send control signals to the vehicle actuators. Setting up a CAN network between the controller and a validation model simulator can allow for closed-loop simulations. Since variable timesteps can not be used when implementing the CIC controller in Acados, it might also lead to memory issues. Furthermore,

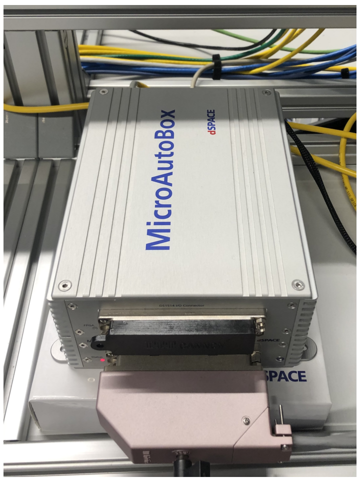


**Figure 9-1:** MicroAutoBox (MABX)

Acados does not provide a default method to deploy it on older dSpace controllers. Because of these considerations and the lower computation time of ACADO in PC-based simulations, the HIL simulations are performed using the ACADO toolbox.

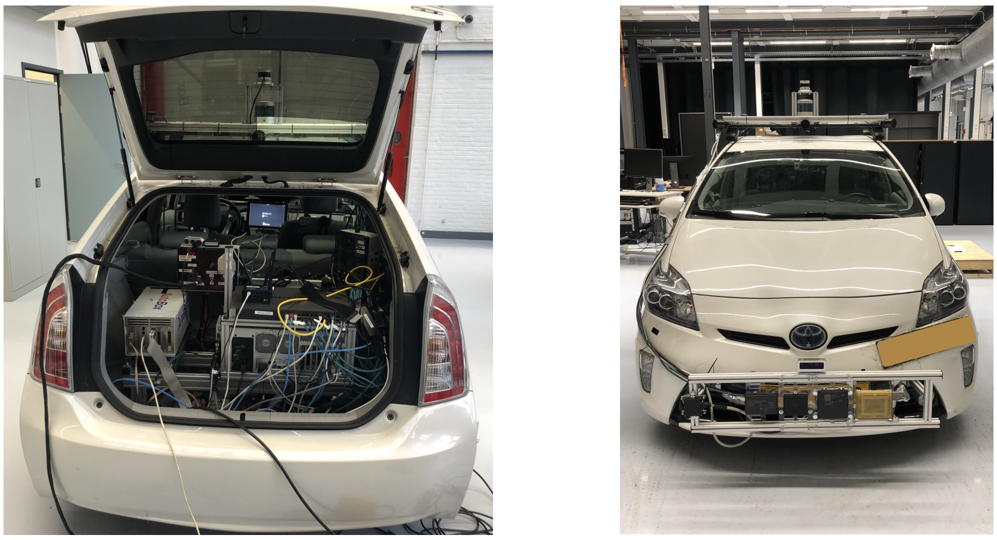


**Figure 9-2:** Autonomous vehicle test platform

## 9-2 Closed loop embedded simulation

Closed loop embedded HIL simulations are performed by sending control signals over a CAN network. These control signals form the inputs of a HIL IPG CarMaker validation model executed on a separate simulator. Artificial measurements are then returned to the controller, from which a new initial state is derived. The ABX runs the CIC controller using ACADO, and a SCX simulator is used to execute the HIL IPG CarMaker validation model. Both controllers send and receive messages over a custom-implemented CAN network, as depicted in Figure 9-3.

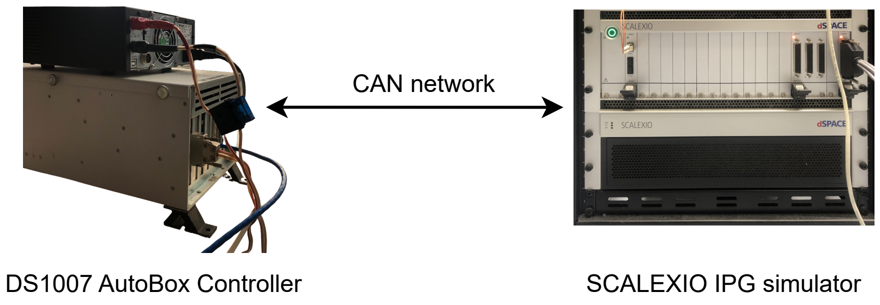


**Figure 9-3:** Closed loop embedded CAN network

### 9-2-1 CAN network

The CAN network was set up using the dSpace RTI CAN Simulink blocks on the controller side and dSpace ConfigurationDesk on the simulator side.

The CAN network structure was configured using a DBC file created using CANdb++. This DBC file specifies the signal names and the data types. The DBC file is used in the RTI Simulink blocks and ConfigurationDesk to specify the RX and TX message structures.

Using data type conversions from the Simulink environment to the CAN network is essential. Signals with low values were multiplied by a certain factor according to the data type. For example, the control signal for the accelerator $\in (0, 1)$ can be multiplied on the TX side and divided on the RX side by a large factor so that it is not converted to a binary signal. More details on the implementation of the CAN network are available in Appendix E.

The closed loop embedded HIL simulation is executed using the scenario presented in section 6-1 with the parameters stated in Table 6-7. The computation times of the controller are shown in Figure 9-4.

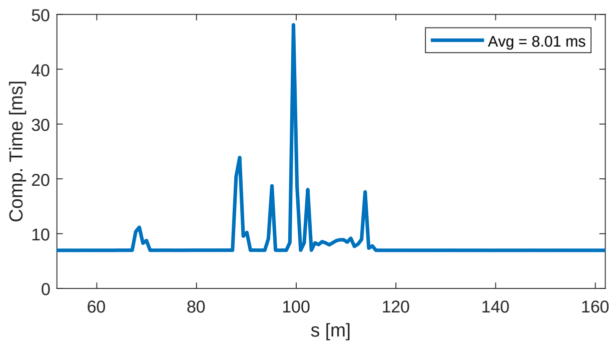


**Figure 9-4:** Computation times SQP toolboxes

The computation time on the embedded system is significantly higher than in the PC-based simulations. This can have multiple causes, such as delays in the CAN communication, reduced signal accuracy due to data type conversions in the CAN network, and reduced computation power of the hardware. The average computation time of 8ms makes it suitable for high control frequencies. The maximum computation time could be further reduced by using a sparse solver such as HPIPM, employing different stopping conditions of the OCP, or implementing advanced real-time task scheduling.

The trajectory of the HIL simulation is presented in Figure 9-5. Even though the computation time is higher, the vehicle's trajectory in this scenario is similar to the PC-based simulation.

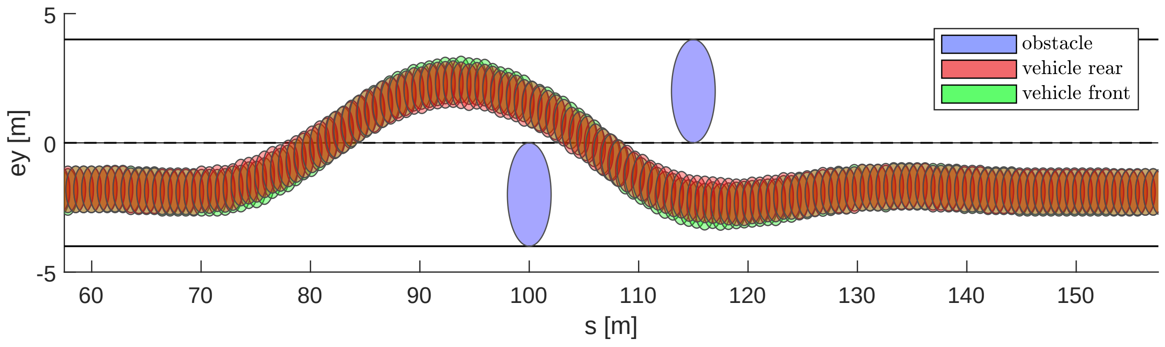


**Figure 9-5:** Trajectory HIL simulation

# Chapter 10

# Discussion

This research investigated several methods to improve the safety of CIC. Safety can be increased by employing more control potential of the vehicle. Typically, this can be achieved by using higher modeling power at the expense of computational complexity. More control potential can be applied using a one-level NMPC controller with a single nonlinear prediction model for re-planning and control.

Different ways of decreasing the computational complexity were explored, including solving techniques, model fidelity, and control problem formulation.

## 10-1 Solving techniques

The nonlinear optimization problem was solved using SQP with RTI. Different solvers can be used to solve the resulting QP problem. Solvers can significantly impact the computation time of the NMPC CIC problem. Sparse solvers outperformed condensed solvers for CIC with long prediction horizons.

To improve convergence, a global reference trajectory provides a sense of direction. This trajectory is not strictly necessary for collision avoidance, as demonstrated in chapter 6. However, it can improve the computation time and the distance to the obstacles, specifically for smaller obstacles.

## 10-2 Model fidelity

The model fidelity of the prediction model can significantly impact safety and computation time. Safety can be improved by exploiting more control potential. Differential braking was found to employ more control potential, which can reduce computation time, promote stability, and achieve collision avoidance in more complex scenarios, compared to the baseline vehicle model [2].

Collision avoidance is integrated into the NMPC problem as an objective. Multiple distance formulations were compared based on their modeling accuracy and computation time. These formulations result from describing the vehicle with different geometric shapes, including multiple circles [2], an ellipse, and a rectangle. Although the ellipse had the lowest computation time for model evaluations, integrating this collision model in the OCP increased the computation time of the controller. Due to the intricate nature of nonlinear optimization, comparing the effects of the distance models directly is complex. A possible explanation could be that the higher complexity of the distance expression leads to less sparsity and more complex Jacobian and Hessian expressions, even though fewer distances must be computed.

It was found that the rectangular model best describes the vehicle geometry. This leads to greater conservativeness, as all areas of the vehicle are included in the expression. The computation time of the rectangular distance model is lower or equal to the baseline method [2] despite the increased conservativeness. Therefore, the rectangular model can improve safety in CIC.

## 10-3 OCP formulation

The control problem can be formulated in various ways, depending on the objectives, constraints, and requirements of the controller. Reducing the problem size was found to reduce computational complexity. Several methods exist that can reduce the problem size. Firstly, stability can be enforced indirectly by penalizing eventual collisions with road boundaries and obstacles that may result from instability [2]. This reduces the problem size compared to using stability constraints with slack variables that allow temporary constraint violations. Secondly, using braking distribution parameters instead of individual tire longitudinal tire forces can reduce the input space [2]. Furthermore, moving block strategies can be applied to reduce the amount of steps in the prediction horizon. Reducing the prediction horizon decreases the problem size, affecting the input and state space, together with the number of constraints. Compared to applying bigger fixed timesteps, variable timesteps can capture the dynamics accurately in the initial phase of the prediction horizon.

A lower bound on the number of timesteps in the prediction horizon was identified. Below this bound, the integration accuracy of the vehicle dynamics and collision avoidance formulation is potentially reduced past a limit where collision avoidance performance suffers significantly.

## 10-4 Real-time performance

The best real-time performance was achieved on PC-based simulations using the MATMPC framework, the HPIPM solver, and variable timesteps. The computation time of the HIL controller is significantly higher than in the PC-based simulations. This can have multiple causes, such as delays in the CAN communication, reduced signal accuracy due to data type conversions in the CAN network, and reduced computation power of the hardware. Even the HIL could be executed at real-time control frequencies due to the low average computation time of 8 ms. The effects of high maximum computation times could be further reduced by using the sparse HPIPM solver in ACADO, employing earlier stopping conditions of the nonlinear optimization problem, or using advanced real-time task scheduling.

# Chapter 11

# Conclusion

An adaptive real-time implementable one-level NMPC controller is proposed for CIC that can achieve higher levels of safety by exploiting greater modeling power. To employ higher modeling power in real-time, SQP with RTI was used together with sparse QP solvers. Starting from a baseline controller [2], various methods that decrease the computational complexity were explored, including solving techniques, model fidelity, and control problem formulation.

It was found that the solver can significantly impact computational complexity, with sparse solvers generally outperforming condensed solvers. Furthermore, moving block strategies employing variable timesteps can considerably reduce the problem size. This greatly reduces computational complexity, with the most significant impact on condensed solvers.

The model fidelity of a vehicle model that can employ differential braking was minimized by employing differential braking on an adjusted single-track vehicle model. This vehicle prediction model can increase the control potential that can be applied, resulting in higher levels of safety demonstrated by improved collision avoidance and stabilizing performance. It was found that the increase in control potential utilization can decrease the computation time despite increasing the problem size.

Safety was increased by employing more accurate collision avoidance formulations that accurately capture the vehicle geometry. A rectangular shape in the collision avoidance formulation provided the best modeling accuracy and computational complexity.

The most significant factors influencing the computation time are the solver, problem size, model complexity, and the exploitation of control potential. With these factors carefully selected or exploited, the CIC controller can achieve control frequencies exceeding 100Hz in PC-based simulations. Compared to the PC-based implementation used by Brown et al. [2], the control frequency was increased by a factor of 5. This removes the need for open-loop control, as the lower computation time of the CIC controller allows it to be closed-loop implemented in real-time. Furthermore, the CIC controller can be considered safer as it can apply more control potential through differential braking, and the (rectangular) collision avoidance formulation is more conservative.

The controller was implemented on the embedded controller of a real autonomous vehicle. HIL simulations were performed using a separate controller and simulator that interface over a CAN network. A control frequency exceeding 50Hz could be feasible with more advanced scheduling. The main problem facing the embedded implementation is the high maximum computation time.

The controller was forced to the limits, achieving collision avoidance in scenarios with uncertain friction and dynamic obstacles. The controller also successfully avoided road boundaries at high speed on highly curved tracks. This demonstrates that the proposed controller can cope with some aspects of the dynamic environment in which autonomous vehicles operate.

With careful formulation and applying different techniques, the safety of CIC can be improved by exploiting more control potential. This can be achieved since the lower computational complexity allows for higher modeling power. However, several challenges to autonomous driving remain, such as perception and global decision-making. Implementing the controller on a real vehicle might bring other unforeseen problems, such as coping with other vehicle safety systems, delays, and other actuator-specific limitations.

In some experiments, it was observed that the CIC controller had a high maximum computation time. Future research could investigate the effects of different solver settings on the performance and maximum computation time of CIC. Furthermore, for UKF friction estimation in vehicle control with differential braking, future works could use a high-fidelity prediction model that includes the rotational dynamics of individual wheels.

To further improve the computational complexity, future works could investigate the effects of different non-uniform integration grids with non-binary variable timestep step increases, implement promising novel solvers such as PRESAS [27], or explore other methods that might offer further reductions in computation time. In addition, the performance of the collision avoidance formulations could be investigated for scenarios with more obstacles.

# Appendix A

# Vehicle model

This chapter in the appendix presents the analysis of approximations and derivation of constraints for the CIC vehicle model. The error of the approximation of $M_x$ is analyzed, followed by derivations of the friction limit constraints.

## A-1 Simplifying the differential braking moment

In chapter 4, an added yaw moment $M_x$ is included in the CIC model to enable differential braking. The approximation of $M_x$ is based on a small angle approximation. Equation A-1 shows the full expression and approximation of the added moment when $M_x < 0$.

$$\begin{aligned} M_x\bigg|_{F_x \leq 0} &= \frac{wF_x}{2}\left(\lambda_x\left(1-2\lambda_y\right)\cos\delta + \left(1-\lambda_x\right)\left(1-2\lambda_y\right)\right) \\ &\approx \frac{wF_x}{2}\left(1-2\lambda_y\right) \end{aligned} \tag{A-1}$$

Hence, the error is expressed as:

$$\text{Error}_{M_x} = |\frac{M_{x_{\max}} - M_{x_{\text{approx}}}}{M_{x_{\max}}}| \cdot 100 = |\lambda_x\left(\cos\delta - 1\right)| \cdot 100 \tag{A-2}$$

This maximum error occurs if $\lambda_x = 1$ and $\delta = 20\,\text{deg}$ and amounts to 6.03%.

## A-2 Friction limit constraints

Generally, the friction limit can be expressed in Equation A-3 [2].

$$|F_x| \leq \mu F_z \tag{A-3}$$

When using differential braking, this constraint must be applied to each individual tire. The normal loads of each tire can be expressed as in Equation 4-13, and the longitudinal force can be derived from Equation A-10. This can be used to derive four friction limit constraints.

**First, the constraints can be formalized in the case that:** $F_x < 0$. In this case, the following constraint can be formulated for each tire $(i, j)$:

$$F_{x_{i,j}} \geq -\mu F_{z_{i,j}} \tag{A-4}$$

Filling in the equations for the front left tire results in:

$$\begin{aligned} F_{xfl} &= \lambda_x \lambda_y F_x \geq -\mu F_{zfl} \\ -F_{xfl} &= -\lambda_x \lambda_y F_x \leq \mu F_{zfl} \\ &= -\lambda_x \lambda_y F_x \leq \frac{\mu}{2} \left( \frac{l_r}{l_f + l_r} F_z - \Delta F_{zx} \right) - \gamma \Delta F_{zy} \end{aligned} \tag{A-5}$$

This process can be repeated for the other tires to derive constraints for $F_{xfr}$, $F_{xfr}$, and $F_{xfr}$. Moving the vehicle state-dependent terms to the left results in the following linear constraints with respect to the states:

$$\begin{aligned} \begin{bmatrix} -\lambda_x \lambda_y & \frac{\mu}{2} & \mu\gamma \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq \frac{l_r \mu m g}{2\left(l_f + l_r\right)} \\ \begin{bmatrix} -\lambda_x \left(1 - \lambda_y\right) & \frac{\mu}{2} & -\mu\gamma \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq \frac{l_r \mu m g}{2\left(l_f + l_r\right)} \\ \begin{bmatrix} -\left(1 - \lambda_x\right) \lambda_y & -\frac{\mu}{2} & \mu\left(1 - \gamma\right) \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq \frac{l_f \mu m g}{2\left(l_f + l_r\right)} \\ \begin{bmatrix} -\left(1 - \lambda_x\right)\left(1 - \lambda_y\right) & -\frac{\mu}{2} & -\mu\left(1 - \gamma\right) \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq \frac{l_f \mu m g}{2\left(l_f + l_r\right)} \end{aligned} \tag{A-6}$$

**Second, the constraints can be formalized in the case that:** $F_x \geq 0$. In this case, the following constraint can be formulated for each tire $(i, j)$:

$$F_{x_{i,j}} \leq \mu F_{z_{i,j}} \tag{A-7}$$

If $F_x$ is positive, there is no differential braking or torque vectoring as there exists a fixed $F_x$ distribution $\lambda_{\text{drive}}$, resulting in the following constraints:

$$\begin{aligned} \begin{bmatrix} \lambda_{\text{drive}} & \mu \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \end{bmatrix} &\leq \frac{l_r \mu m g}{\left(l_f + l_r\right)} \\ \begin{bmatrix} 1 - \lambda_{\text{drive}} & -\mu \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \end{bmatrix} &\leq \frac{l_f \mu m g}{\left(l_f + l_r\right)} \end{aligned} \tag{A-8}$$

The full constraint can be expressed as:

$$\begin{aligned}
\begin{bmatrix} \frac{\lambda_1}{\mu} & \frac{1}{2} & \gamma \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq F_{gf} \\
\begin{bmatrix} \frac{\lambda_2}{\mu} & \frac{1}{2} & -\gamma \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq F_{gf} \\
\begin{bmatrix} \frac{\lambda_3}{\mu} & -\frac{1}{2} & 1-\gamma \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq F_{gr} \\
\begin{bmatrix} \frac{\lambda_4}{\mu} & -\frac{1}{2} & \gamma-1 \end{bmatrix} \begin{bmatrix} F_x \\ \Delta F_{zx} \\ \Delta F_{zy} \end{bmatrix} &\leq F_{gr}
\end{aligned} \tag{A-9}$$

With:

$$\begin{aligned}
\lambda_1 &= \begin{cases} -\lambda_x\lambda_y & F_x \leq 0 \\ \frac{\lambda_{\text{drive}}}{2} & \text{otherwise} \end{cases}, \quad & \lambda_2 &= \begin{cases} -\lambda_x(1-\lambda_y) & F_x \leq 0 \\ \frac{\lambda_{\text{drive}}}{2} & \text{otherwise} \end{cases} \\
\lambda_3 &= \begin{cases} -(1-\lambda_x)\lambda_y & F_x \leq 0 \\ \frac{(1-\lambda_{\text{drive}})}{2} & \text{otherwise} \end{cases}, \quad & \lambda_4 &= \begin{cases} -(1-\lambda_x)(1-\lambda_y) & F_x \leq 0 \\ \frac{(1-\lambda_{\text{drive}})}{2} & \text{otherwise} \end{cases} \\
F_{gf} &= \frac{l_r mg}{2\left(l_f + l_r\right)}, & F_{gr} &= \frac{l_f mg}{2\left(l_f + l_r\right)}
\end{aligned} \tag{A-10}$$

# Appendix B

# Distance formulation derivation

Certain approximations and derivations are relied upon in the formulation of the distance models presented in chapter 3. These are presented and analyzed in this chapter.

## B-1 Error in angle estimation

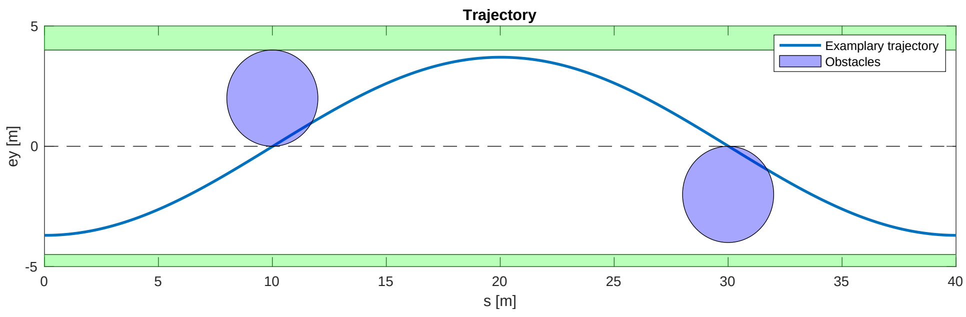


**Figure B-1:** Exemplary trajectory simulation

The expression of $\theta$ is significantly simplified but influenced by $\kappa$ in the $\theta_L$ term. To evaluate the accuracy of this approximation, it can be evaluated against the original expression on an uncontrolled exemplary trajectory, such as presented in Figure B-1. The error in the approximation increases for larger road curvatures. In this trajectory, the road is assumed to have an instantaneous turning radius of 10 meters.

The approximation error of the angle to each obstacle during the trajectory is shown in Figure B-2. Generally, the error in the angle is less than 5 degrees. Near the obstacles at $s = 10$ and $s = 30$, the error in the angle to the obstacle is minimal.

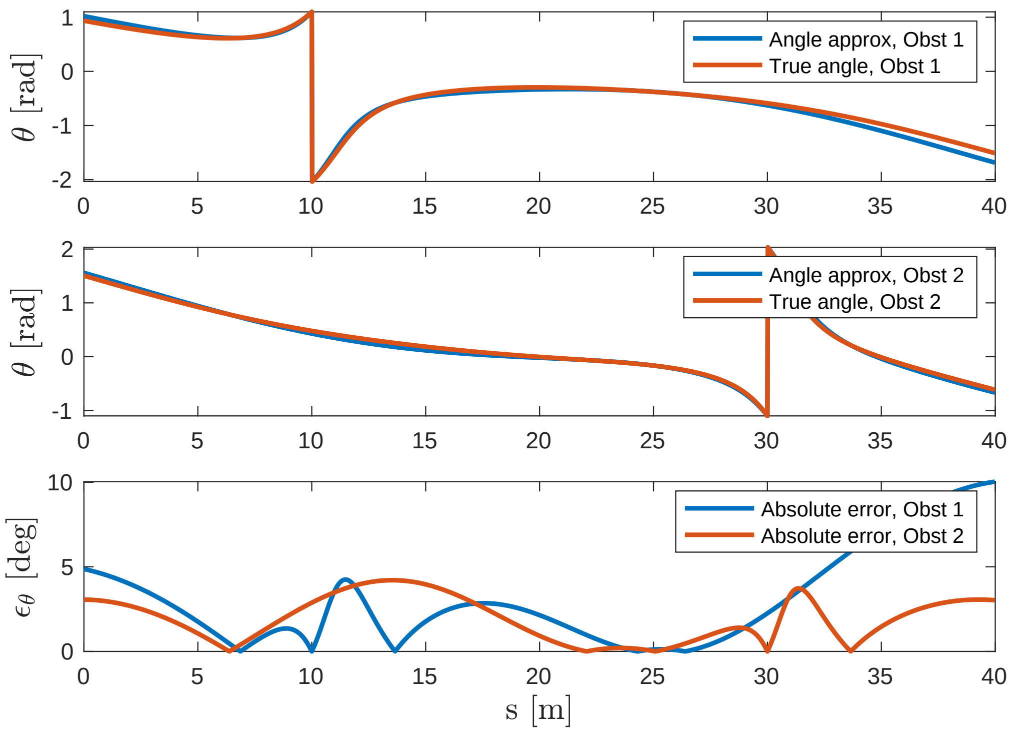


**Figure B-2:** Approximation errors in $\theta$

## B-2 Derivation of extrema coordinates of a rotated ellipse

The extrema coordinates of a rotated ellipse with respect to the y-axis are derived in this chapter. The equations of a rotated ellipse with angle $\theta$ in the cartesian space can be expressed as:

$$\frac{(x\cos\theta - y\sin\theta)^2}{a^2} + \frac{(x\sin\theta + y\cos\theta)^2}{b^2} = 1 \tag{B-1}$$
$$(x\cos\theta - y\sin\theta)^2 b^2 + (x\sin\theta + y\cos\theta)^2 a^2 = a^2b^2$$

This equation expands to:

$$(x^2\cos^2\theta - 2xy\cos\theta\sin\theta + y^2\sin^2\theta)b^2 + (x^2\sin^2\theta + 2xy\cos\theta\sin\theta + y^2\cos^2\theta)a^2 = a^2b^2$$

The maximum point can be found by finding the derivative with respect to $x$ and $y$ of the LHS:

$$f(x,y) = (x^2\cos^2\theta - 2xy\cos\theta\sin\theta + y^2\sin^2\theta)b^2 + (x^2\sin^2\theta + 2xy\cos\theta\sin\theta + y^2\cos^2\theta)a^2$$

Differentiating $f(x,y)$, with respect to $x$ and $y$ results in the following partial derivative functions:

$$\begin{aligned}\frac{\partial f(x,y)}{\partial x} &= (2x\cos^2\theta - 2y\cos\theta\sin\theta)b^2 + (2x\sin^2\theta + 2y\cos\theta\sin\theta)a^2 \\ &= b^2x\cos^2\theta + a^2x\sin^2\theta + (a^2-b^2)y\cos\theta\sin\theta \qquad \text{(B-2)}\\ \frac{\partial f(x,y)}{\partial y} &= (-2x\cos\theta\sin\theta + 2y\sin^2\theta)b^2 + (2x\cos\theta\sin\theta + 2y\cos^2\theta)a^2 \\ &= b^2y\sin^2\theta + a^2y\cos^2\theta + (a^2-b^2)x\cos\theta\sin\theta \qquad \text{(B-3)}\end{aligned}$$

$\frac{\partial y}{\partial x}$ can be found by dividing the partial derivative equations.

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial f(x,y)} \cdot \frac{\partial f(x,y)}{\partial x} = \frac{b^2x\cos^2\theta + a^2x\sin^2\theta + (a^2-b^2)y\cos\theta\sin\theta}{b^2y\sin^2\theta + a^2y\cos^2\theta + (a^2-b^2)x\cos\theta\sin\theta} \qquad \text{(B-4)}$$

$\frac{\partial y}{\partial x}$ equals zero if:

$$b^2x\cos^2\theta + a^2x\sin^2\theta + (a^2-b^2)y\cos\theta\sin\theta = 0 \qquad \text{(B-5)}$$

Hence:

$$\begin{aligned} y_{\max} &= -\frac{b^2x\cos^2\theta + a^2x\sin^2\theta}{(a^2-b^2)\cos\theta\sin\theta} \\ &= -x\frac{a^2\sin^2\theta + b^2\cos^2\theta}{(a^2-b^2)\cos\theta\sin\theta} \qquad \text{(B-6)}\end{aligned}$$

$\frac{\partial x}{\partial y}$ equals zero if:

$$b^2y\sin^2\theta + a^2y\cos^2\theta + (a^2-b^2)x\cos\theta\sin\theta = 0 \qquad \text{(B-7)}$$

Hence:

$$\begin{aligned} x_{\max} &= -\frac{b^2y\sin^2\theta + a^2y\cos^2\theta}{(a^2-b^2)\cos\theta\sin\theta} \\ &= -y\frac{a^2\cos^2\theta + b^2\sin^2\theta}{(a^2-b^2)\cos\theta\sin\theta} \qquad \text{(B-8)}\end{aligned}$$

The maximum $y$-coordinate is then found by filling in the equation for $x_{\max}$ in the function of the rotated ellipse:

$$b^2\left(x\cos\theta + x\frac{a^2\sin^2\theta + b^2\cos^2\theta}{(a^2-b^2)\cos\theta}\right)^2 + a^2\left(x\sin\theta - x\frac{a^2\sin^2\theta + b^2\cos^2\theta}{(a^2-b^2)\sin\theta}\right)^2 = a^2b^2 \qquad \text{(B-9)}$$

The following simplification can be applied:

$$f(a,b,\theta) = a^2\sin^2\theta + b^2\cos^2\theta \qquad \text{(B-10)}$$

The equation of the ellipse expands to:

$$\begin{aligned}
&a^2x^2\left(\frac{f^2}{(a^2-b^2)\sin\theta}\right)^2+b^2x^2\left(\frac{f^2}{(a^2-b^2)\cos\theta}\right)^2+2x^2\frac{f\left(b^2-a^2\right)}{(a^2-b^2)}+b^2x^2\cos^2\theta+a^2x^2\sin^2\theta=a^2b^2\\
&a^2x^2\left(\frac{f^2}{(a^2-b^2)\sin\theta}\right)^2+b^2x^2\left(\frac{f^2}{(a^2-b^2)\cos\theta}\right)^2-y^2f=a^2b^2\\
&x^2\left(\frac{a^2f^2}{(a^2-b^2)^2\sin^2\theta}+\frac{b^2f^2}{(a^2-b^2)^2\cos^2\theta}-f\right)=a^2b^2\\
&x^2\left(\frac{f^2}{(a^2-b^2)^2}\cdot\frac{a^2\cos^2\theta+b^2\sin^2\theta}{\sin^2\theta\cos^2\theta}-f\right)=a^2b^2\\
&x^2\left(\frac{f}{(a^2-b^2)^2}\cdot\frac{\left(a^2\cos^2\theta+b^2\sin^2\theta\right)\cdot\left(a^2\sin^2\theta+b^2\cos^2\theta\right)}{\sin^2\theta\cos^2\theta}-f\right)=a^2b^2\\
&x^2\left(\frac{f}{(a^2-b^2)^2}\cdot\frac{\left(a^4\sin^2\theta\cos^2\theta+a^2b^2\cos^4\theta+a^2b^2\sin^4\theta+b^4\sin^2\theta\cos^2\theta\right)}{\sin^2\theta\cos^2\theta}-f\right)=a^2b^2\\
&x^2\left(\frac{f}{(a^2-b^2)^2}\cdot\frac{\left(a^4\sin^2\theta\cos^2\theta+a^2b^2\left(1-2\sin^2\theta\cos^2\theta\right)+b^4\sin^2\theta\cos^2\theta\right)}{\sin^2\theta\cos^2\theta}-f\right)=a^2b^2\\
&x^2\left(\frac{f}{(a^2-b^2)^2}\cdot\frac{a^2b^2+(a^2-b^2)^2\sin^2\theta\cos^2\theta}{\sin^2\theta\cos^2\theta}-f\right)=a^2b^2\\
&x^2\left(f\cdot\frac{a^2b^2}{(a^2-b^2)^2\sin^2\theta\cos^2\theta}\right)=a^2b^2\\
&x^2=\frac{(a^2-b^2)^2\sin^2\theta\cos^2\theta}{a^2\sin^2\theta+b^2\cos^2\theta} \qquad \text{(B-11)}
\end{aligned}$$

Using the previously found

$$y_{\max}=-x\frac{a^2\sin^2\theta+b^2\cos^2\theta}{(a^2-b^2)\cos\theta\sin\theta} \qquad \text{(B-12)}$$

$$\begin{aligned}
\therefore y_{\max}^2&=x^2\frac{\left(a^2\sin^2\theta+b^2\cos^2\theta\right)^2}{(a^2-b^2)^2\cos^2\theta\sin^2\theta}\\
&=\frac{(a^2-b^2)^2\sin^2\theta\cos^2\theta}{a^2\sin^2\theta+b^2\cos^2\theta}\cdot\frac{\left(a^2\sin^2\theta+b^2\cos^2\theta\right)^2}{(a^2-b^2)^2\cos^2\theta\sin^2\theta}=a^2\sin^2\theta+b^2\cos^2\theta \qquad \text{(B-13)}
\end{aligned}$$

This leads to the value of the maximum $y$-coordinate of a rotated ellipse:

$$y_{\max}=\pm\sqrt{a^2\sin^2\theta+b^2\cos^2\theta} \qquad \text{(B-14)}$$

# Appendix C

# Environmental estimators

This section provides the derivations of the estimators used in this work.

## C-1 State and friction estimator

Two different Unscented Kalman Filters (UKFs) are developed in this research. The first filter includes the rotational dynamics of the wheels, and the second filter simplifies the dynamics by assuming that the longitudinal tire forces can be derived from the control inputs $u$.

### C-1-1 UKF with rotational dynamics

To account for discrepancies between the commanded longitudinal tire forces by the controller and the actual longitudinal tire forces, the tire rotational dynamics described in Equation C-1 can be used [16].

$$I_{w_f}\dot{\omega}_f = T_f - M_{r_f} - F_{x_f} r_{w_f}, \quad I_{w_r}\dot{\omega}_r = T_r - M_{r_r} - F_{x_r} r_{w_r} \tag{C-1}$$

In Equation C-1, $\omega_{w_i}$ is the rotational velocity, $I_{w_i}$ is the inertia, and $r_{w_i}$ is the radius of the wheel at $i = (f, r)$. $T_f$ and $T_r$ represent the torques at the front and rear axle.

The rotational dynamics of the wheel and the dynamics of the ST model lead to the following estimation vector $\hat{\xi}$ for the UKF shown in Equation C-2. These estimates can be integrated and scaled to represent the full state space of the CIC prediction model.

$$\hat{\xi} = [\dot{x}, \dot{y}, \dot{\psi}, \mu, F_{xf}, F_{xr}, \omega_f, \omega_r]^T \tag{C-2}$$

Artificial measurements are taken from the plant in IPG CarMaker. These form the measurement vector $y_{UKF}$, presented in Equation C-3.

$$y = [\dot{x}, \ddot{x}, \ddot{y}, \dot{\psi}, \omega_f, \omega_r]^T \tag{C-3}$$

The input space of the UKF $u_{UKF}$ is chosen to be equal to the inputs of the IPG plant model. Including the torques directly as an input simplifies the wheel rotational dynamics in the prediction model.

$$u_{UKF} = [\delta_{sw}, T_f, T_r]^T \tag{C-4}$$

The UKF prediction model is expressed in Equation C-5.

$$\begin{aligned}
m\ddot{x} &= m\dot{y}\dot{\psi} + F_{x_f}\cos\delta - F_{y_f}\sin\delta + F_{x_r} \\
m\ddot{y} &= -m\dot{x}\dot{\psi} + F_{x_f}\sin\delta + F_{y_f}\cos\delta + F_{y_r} \\
I_z\ddot{\psi} &= l_f(F_{x_f}\sin\delta + F_{y_f}\cos\delta) - l_r F_{y_r} \\
\dot{\mu} &= 0 \\
\dot{F}_{x_f} &= (F_{x_f} - T_f r_{w_r})\Delta t \\
\dot{F}_{x_r} &= (F_{x_r} - T_f r_{w_r})\Delta t \\
I_w\dot{\omega}_f &= T_f - M_{r_f} - F_{x_f} r_{w_r} \\
I_w\dot{\omega}_r &= T_r - M_{r_r} - F_{x_r} r_{w_r}
\end{aligned} \tag{C-5}$$

Similar to the CIC prediction model, a Fiala tire model is used for the lateral dynamics. Gray-box parameter identification was performed to find the model parameters. However, it proved challenging to fit the rotational dynamics to the response of the IPG model. This is likely due to differential braking and a ST model in the prediction model of the UKF.

## Measurement model

The measurement model $H(\xi_{k+1})$ finds the prediction of the measurements $\hat{y}_{k+1}$ according the predicted model states $\xi_{k+1}$ [16]. For the UKF with rotational dynamics, the precision model can be defined as:

$$\begin{aligned}
\hat{y}1_{k+1} &= \xi1_{k+1} \\
\hat{y}2_{k+1} &= \frac{\xi1_{k+1} - \xi1_k}{\Delta t} - \xi3_{k+1}\xi2_{k+1} \\
\hat{y}3_{k+1} &= \frac{\xi2_{k+1} - \xi2_k}{\Delta t} + \xi3_{k+1}\xi1_{k+1} \\
\hat{y}4_{k+1} &= \xi3_{k+1} \\
\hat{y}5_{k+1} &= \xi7_{k+1} \\
\hat{y}6_{k+1} &= \xi8_{k+1}
\end{aligned} \tag{C-6}$$

## UKF parameters

The UKF was tuned to minimize the error in the lateral velocity. This resulted in the variables of the covariance matrices $P_0, R_v, R_n$ and the UKF tuning parameters $\alpha_{UKF}$, $\beta_{UKF}$, and $\kappa_{UKF}$.

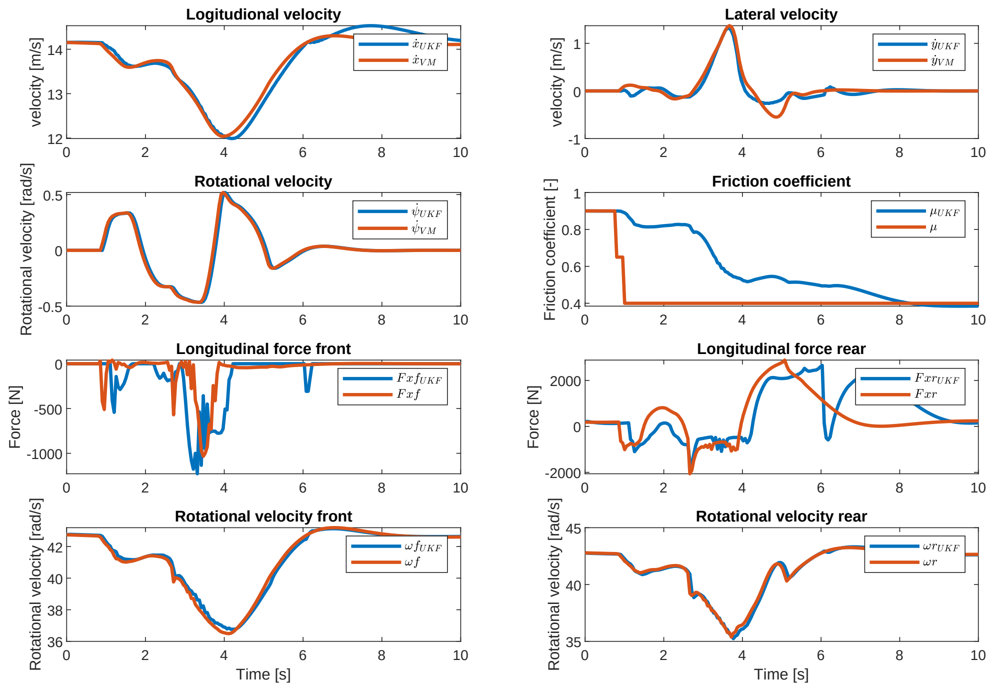


**Figure C-1:** UKF with wheel rotational dynamics

$$
\begin{aligned}
P_0 &= \text{diag}([1.2e^{-6}, 1.7e^{-5}, 1.0e^{-5}, 1.1e^{-4}, 2.0e^{-5}, 1.8e^{-5}, 1.2e^{-6}, 1.0e^{-6}]) \\
R_v &= \text{diag}([1.0e^{-5}, 1.2e^{-5}, 1.0e^{-5}, 9.0e^{-4}, 2.2e^{-4}, 2.0e^{-4}, 1.0e^{-5}, , 1.1e^{-5}]) \\
R_n &= \text{diag}([1.0e^{-5}, 5.0e^{-4}, 5.8e^{-5}, 1.1e^{-8}, 1.1e^{-5}, 1.0e^{-5}]) \\
\alpha_{UKF} &= 0.1 \\
\beta_{UKF} &= 2.2 \\
\kappa_{UKF} &= 96000
\end{aligned}
$$

## Estimation results

The estimator was tuned on a double lane change maneuver in IPG. The estimated states and friction coefficient are presented in Figure C-1. Although the UKF can estimate the states and friction coefficient relatively well on the double lane change maneuver dataset, the friction coefficient would blow up in closed-loop simulations. Since the wheel rotational dynamics seemed to significantly impact $\hat{\mu}$, not being able to fit the rotational velocities to the IPG model response was likely the reason for this behavior.

## C-1-2 Simplified UKF

Since the rotational dynamics proved difficult to fit the response of the IPG model, a simplified UKF was implemented. Here, the longitudinal tire forces are approximated as:

$$F_{x_f} \approx \frac{T_f}{r_w}, \quad F_{x_r} \approx \frac{T_r}{r_w} \tag{C-7}$$

Compared to the UKF with rotational wheel dynamics, this approximation eliminates multiple variables. Again, the measurements are taken from a BMW 5-series vehicle model from IPG CarMaker and form measurement vector $y$:

$$y = [\dot{x}, \ddot{x}, \ddot{y}, \dot{\psi}]^T \tag{C-8}$$

The input space is also chosen to be equal to the inputs of the IPG plant.

$$u = [\delta_{sw}, T_f, T_r]^T \tag{C-9}$$

The prediction model of the simplified UKF can be expressed as:

$$\begin{aligned} m\ddot{x} &= m\dot{y}\dot{\psi} + F_{x_f}\cos\delta - F_{y_f}\sin\delta + F_{x_r} \\ m\ddot{y} &= -m\dot{x}\dot{\psi} + F_{x_f}\sin\delta + F_{y_f}\cos\delta + F_{y_r} \\ I_z\ddot{\psi} &= l_f(F_{x_f}\sin\delta + F_{y_f}\cos\delta) - l_r F_{y_r} \\ \dot{\mu} &= 0 \end{aligned} \tag{C-10}$$

A Fiala tire model is used for the lateral dynamics. Gray-box parameter identification was performed to find the model parameters. Without the rotational dynamics of the wheels, this prediction model could closely approximate the vehicle dynamics.

### Measurement model

The measurement model $H(\xi_{k+1})$ finds the prediction of the measurements $\hat{y}_{k+1}$ according the predicted model states $\xi_{k+1}$ [16]:

$$\begin{aligned} \hat{y}1_{k+1} &= \xi 1_{k+1} \\ \hat{y}2_{k+1} &= \frac{\xi 1_{k+1} - \xi 1_k}{\Delta t} - \xi 3_{k+1}\xi 2_{k+1} \\ \hat{y}3_{k+1} &= \frac{\xi 2_{k+1} - \xi 2_k}{\Delta t} + \xi 3_{k+1}\xi 1_{k+1} \\ \hat{y}4_{k+1} &= \xi 3_{k+1} \end{aligned}$$

### UKF parameters

The UKF was tuned to minimize the error of the estimated parameters. This resulted in the following set of variables:

$$
\begin{aligned}
P_0 &= \text{diag}([1.2e^{-6}, 2.1e^{-5}, 1.1e^{-5}, 1.0e^{-4}]) \\
R_n &= \text{diag}([1e^{-5}, 5e^{-3}, 5e^{-5}, 1e^{-8}]) \\
R_v &= \text{diag}([1e^{-5}, 1e^{-5}, 1e^{-5}, 1e^{-5}]) \\
\alpha_{UKF} &= 0.5 \\
\beta_{UKF} &= 1.9 \\
\kappa_{UKF} &= 88000
\end{aligned}
$$

## Estimation results

The estimator was tuned on a double lane change maneuver in IPG. The estimated states and friction coefficient are presented in Figure C-2. Eliminating the rotational dynamics of the wheels has somewhat decreased the accuracy of the UKF. However, as the prediction model could fit the response of the plant better, this filter was implementable in closed-loop control.

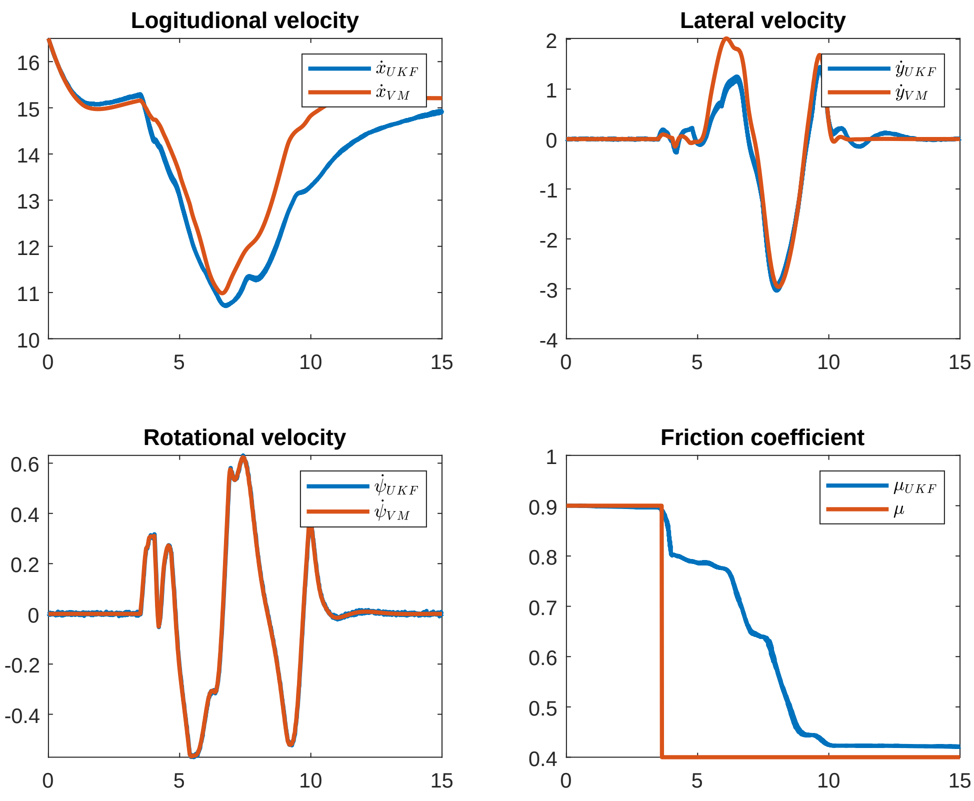


**Figure C-2:** UKF without wheel rotational dynamics

**implementation**

This UKF filter was integrated inside the IPG Simulink model, as presented in Figure C-3.

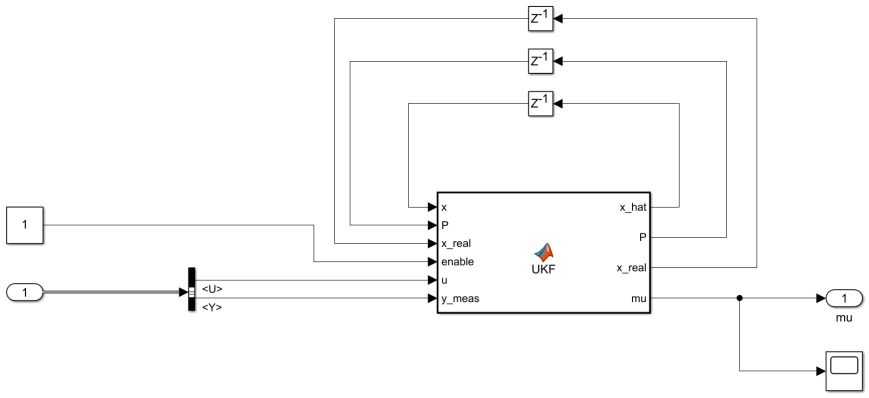


**Figure C-3:** UKF Simulink implmentation

## C-2 Road curvature estimator

The road curvature at some point can be found using a previous location and a future location in cartesian coordinates. This can be taken from maps and integrated with GPS data. Two different methods are presented to estimate this curvature. The first method fits a circle to three points. The second method uses the dynamics of the curvature, assuming that the points along the trajectory are sampled at equal distances apart.

### C-2-1 Circle fitting method

To find the curvature, a circle can be fitted to three points. A system of equations can be solved to find the center $(x_c, y_c)$ and radius of the circle for which Equation C-11 holds for each point $p = (x, y)$.

$$\begin{aligned}(x - x_c)^2 + (y - y_c)^2 &= r^2 \\ \therefore x^2 + y^2 - 2x_c x + x_c^2 - 2y_c y + y_c^2 - r^2 &= 0 \\ \therefore 2x_c x + 2y_c y + r^2 - x_c^2 - y_c^2 &= x^2 + y^2\end{aligned} \tag{C-11}$$

Assuming we have 3 points: $p_1 = (x_1, y_1)$, $p_2 = (x_2, y_2)$, and $p_3 = (x_3, y_3)$. We can find $r$ by solving a system of equations which can be expressed in matrix form:

In matrix from:

$$\begin{bmatrix} 2x_1 & 2y_1 & 1 \\ 2x_2 & 2y_2 & 1 \\ 2x_3 & 2y_3 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x_1^2 + y_1^2 \\ x_2^2 + y_2^2 \\ x_3^2 + y_3^2 \end{bmatrix} \tag{C-12}$$

where, $a = x_c$, $b = y_c$, and $c = r^2 - x_c^2 - y_c^2$. Then we can find curvature $k$ over these points as:

$$k = \frac{1}{r} = \frac{1}{\sqrt{a^2 + b^2 + c}} \tag{C-13}$$

However, the sign of the curvature is required to indicate if the road is curving to the left or right.

$$\theta_1 = \arctan\left(\frac{y_2 - y_1}{x_2 - x_1}\right), \quad \theta_2 = \arctan\left(\frac{y_3 - y_2}{x_3 - x_2}\right), \quad \Delta\theta = \theta_2 - \theta_1 \tag{C-14}$$

Using the change of angle of the track, the road curvature with the corresponding sign can be derived:

$$k = \frac{\text{sign}(\Delta\theta)}{\sqrt{a^2 + b^2 + c}} \tag{C-15}$$

### C-2-2 Simplified curvature estimator

In some cases, it can be assumed that the distance between points on a trajectory is constant. This means that the distance between the points $p_i$ and $p_{i+1}$ is some distance $\Delta s$ for all $i \in T$ where $T$ is some trajectory. In this case, the curvature can be simplified to:

$$k_i = \frac{\Delta\theta}{\Delta s} = \frac{\Delta\theta}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}} \tag{C-16}$$

### C-2-3 Race track simulation

The CIC controller is implemented on the Silverstone race circuit. Here, the curvature is estimated using the curvature estimators above. The full trajectory of the CIC controller on the Silverstone track is shown in Figure C-4 on the next page.

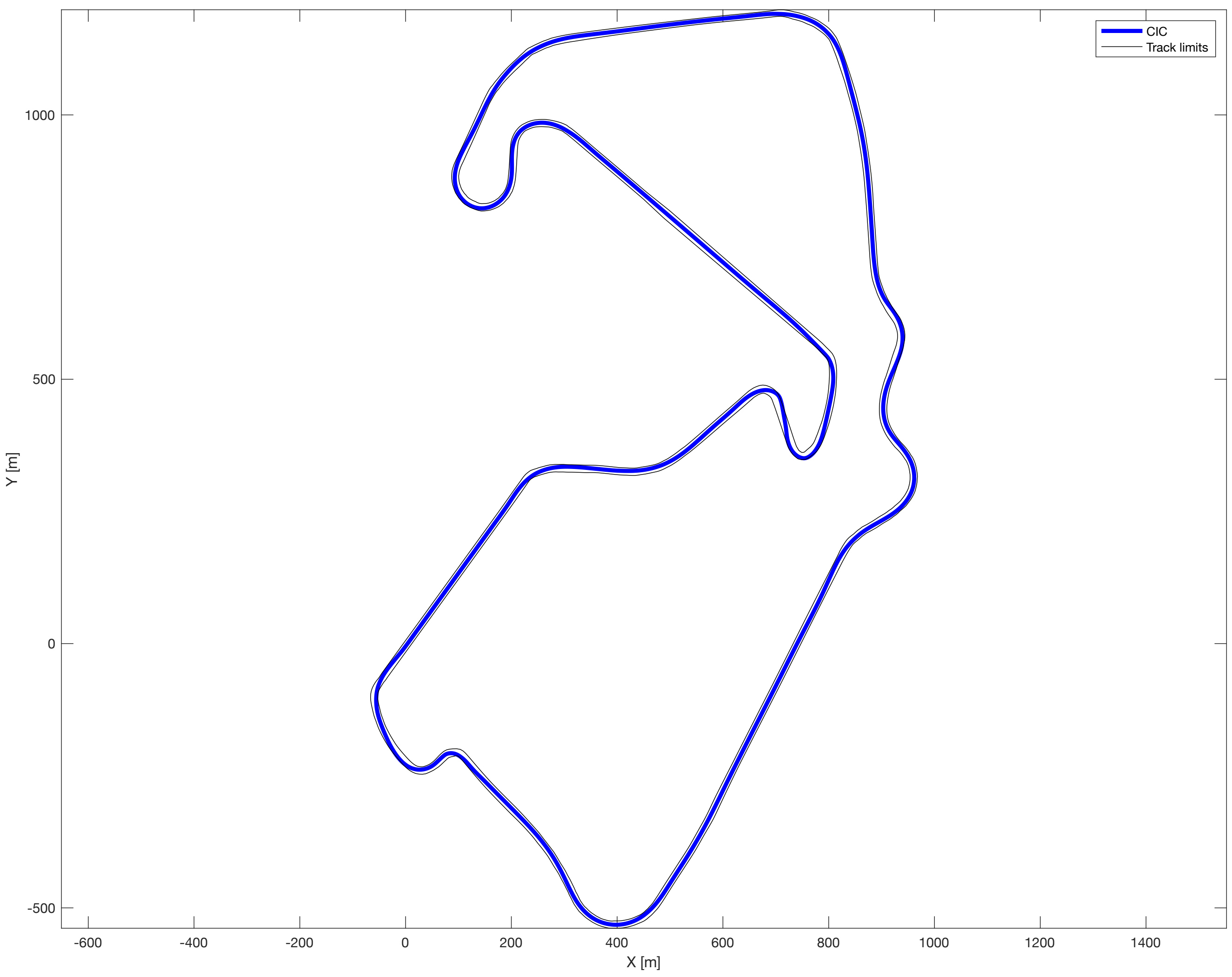


**Figure C-4:** CIC on the Silverstone race circuit

# Appendix D

# IPG vehicle model

This chapter presents the integration of the CIC controller in IPG CarMaker for closed-loop control. IPG CarMaker is configured to use the BMW 5-series model. A 3D scenario is created in IPG, where two square obstacles are avoided, this is presented in Figure D-1.

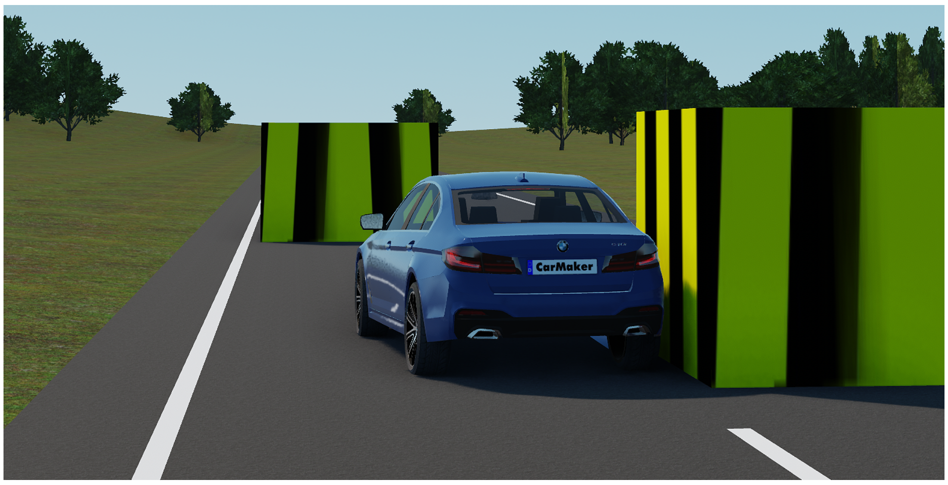


**Figure D-1:** IPG render

## D-1 IPG Simulink implementation

Measurements from IPG can be transformed to the states of the prediction model using identified parameters from chapter 4. This transformation is presented in Figure D-2. The standard IPG Simulink model includes different subsystems, presented in Figure D-3. CIC is implemented in the VehicleControl and IPG-Vehicle subsystems. All custom blocks are marked in blue. The VehicleControl subsystem is shown in Figure D-4. The VehicleControl subsystem includes the controller shown in Figure D-5, state imports from Figure D-6, and the VhclCtrl bus creator presented in Figure D-7. The IPG Vehicle subsystem that implements differential braking is shown in Figure D-8.

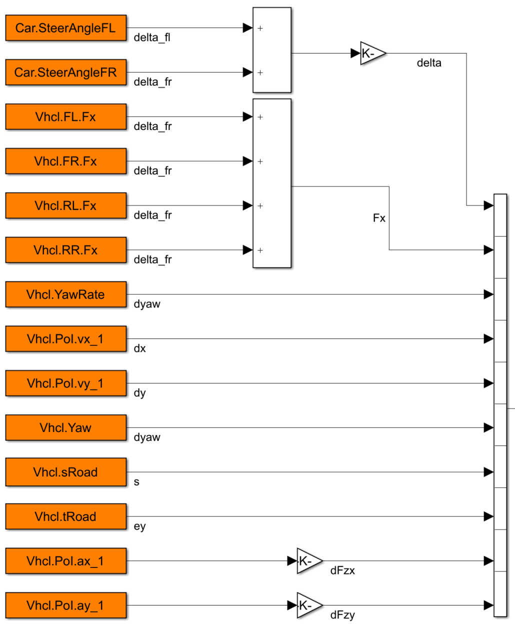


**Figure D-2:** IPG state measurements

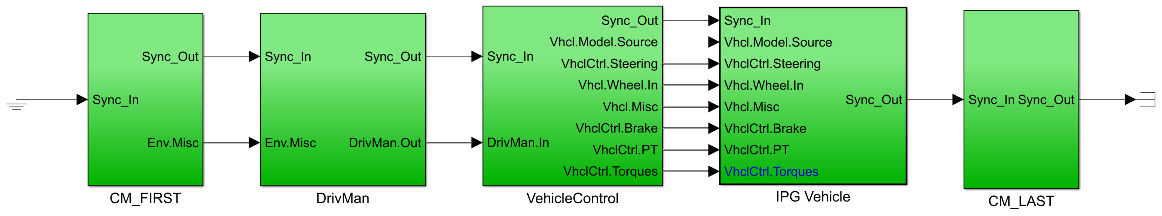


**Figure D-3:** Full IPG Simulink model

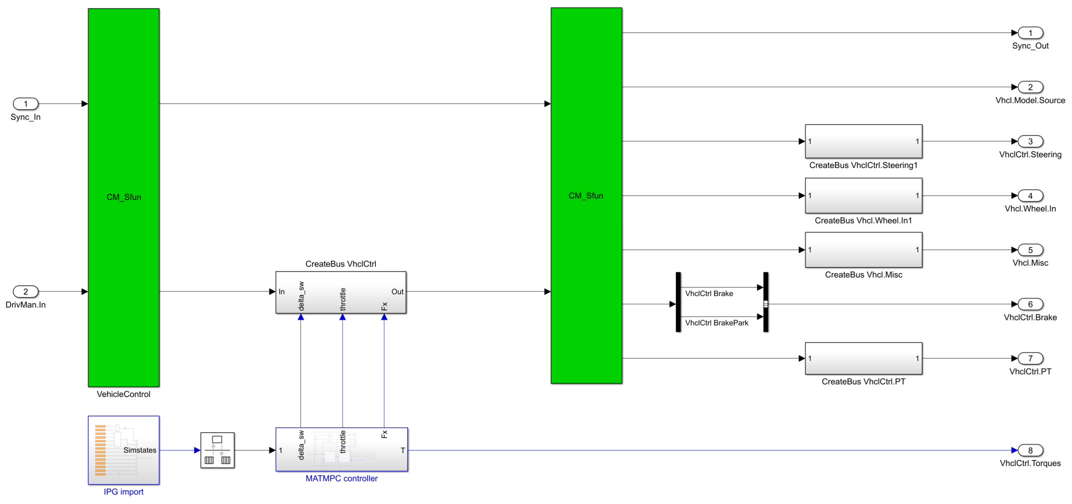


**Figure D-4:** IPG VehicleControl subsystem

The vehicle control subsystem is shown in Figure D-5. The VehicleControl subsystem includes the controller, state imports, and VhclCtrl bus creator.

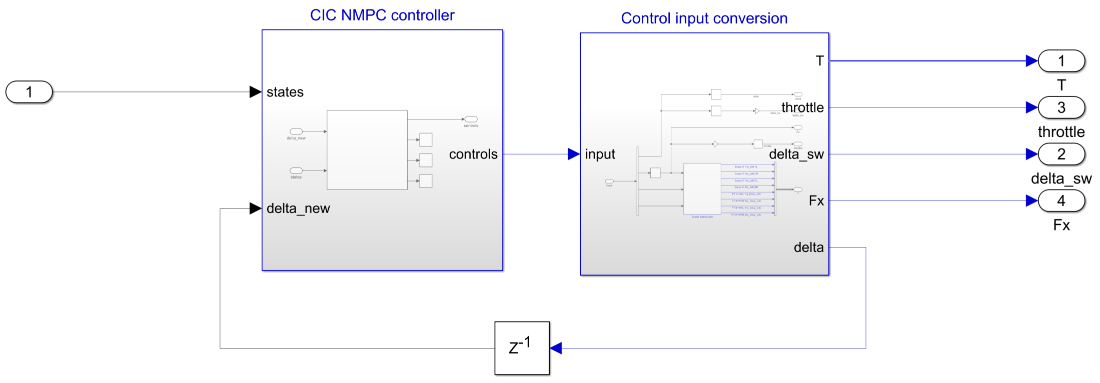


**Figure D-5:** IPG Controller subsystem

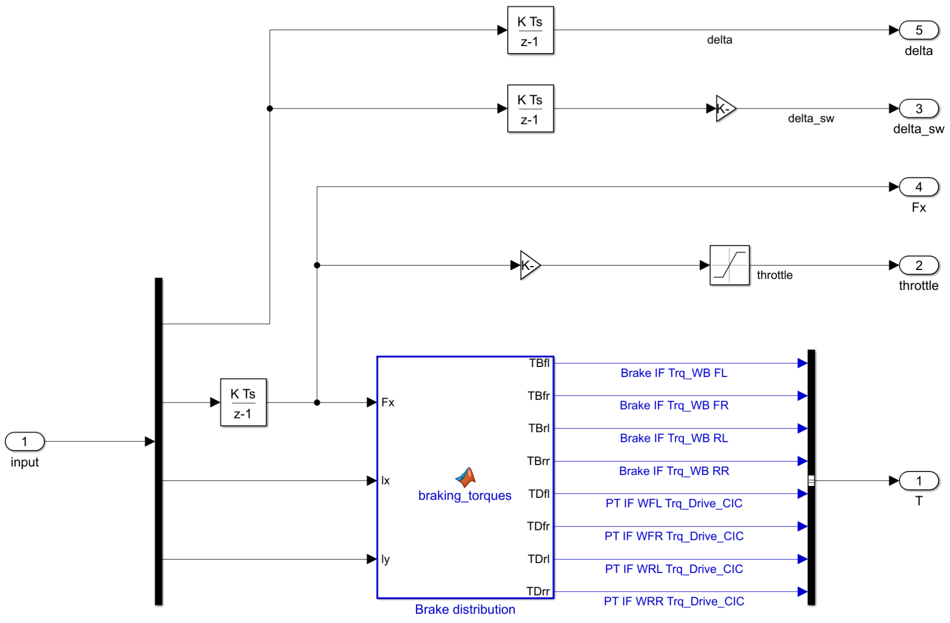


**Figure D-6:** IPG control input conversion

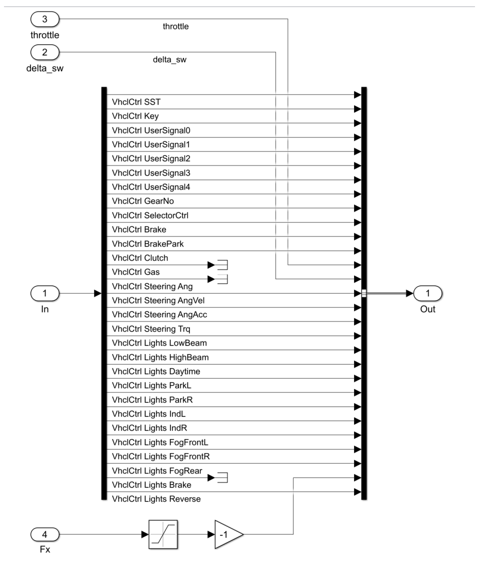


**Figure D-7:** IPG VhlCtrl bus creator

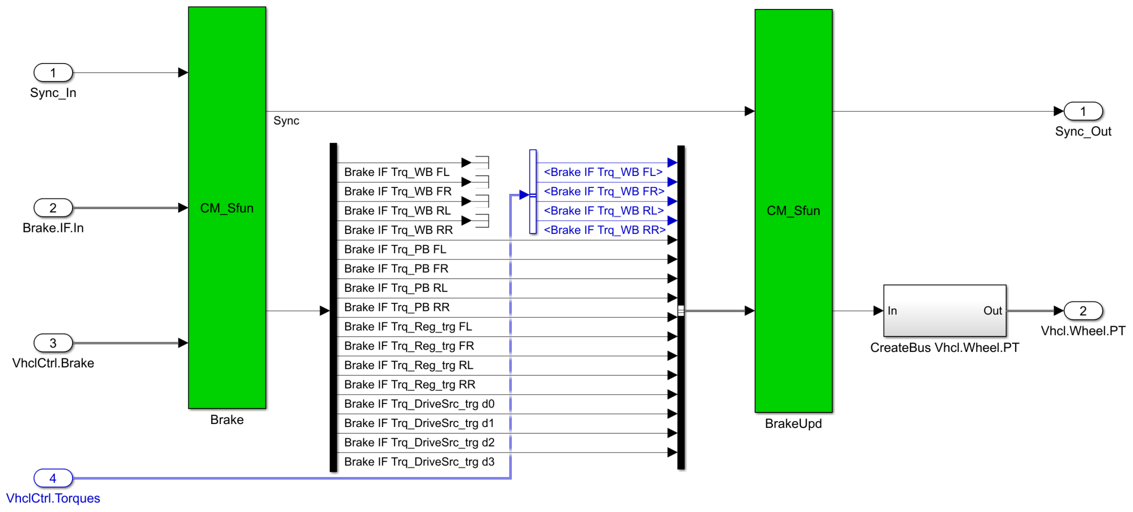


**Figure D-8:** IPG Vehicle subsystem

## D-2 NMPC toolbox integration

In Figure D-9, Figure D-10, and Figure D-11, the Simulink implementation of the NMPC toolboxes MATMPC, ACADO and Acados are presented.

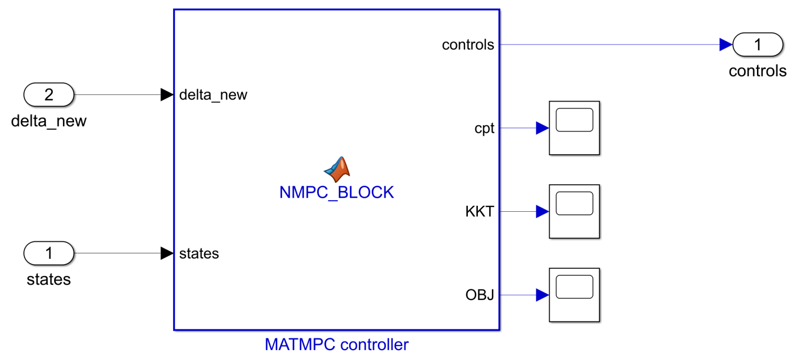


**Figure D-9:** IPG MATMPC closed loop controller

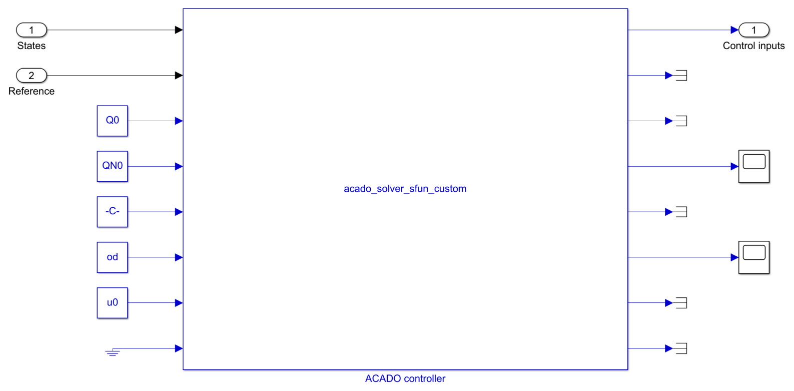


**Figure D-10:** IPG ACADO closed loop controller

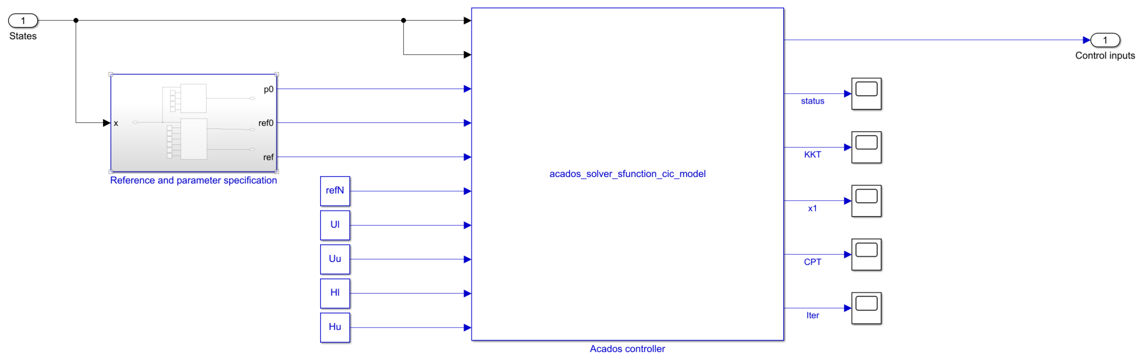


**Figure D-11:** IPG Acados closed loop controller

# Appendix E

# Hardware in the loop

As presented in chapter 9, HIL simulations are performed using a CAN network. This CAN network was set up using the dSpace RTI CAN Simulink blocks on the controller side and dSpace ConfigurationDesk on the simulator side. Here, dSpace AutoBox is used as the controller and dSpace SCALEXIO is used for embedded HIL simulations of the IPG model.

The CAN network structure was configured using a DBC file created using CANdb++. This DBC file specifies the signal names and the data types. On the SCALEXIO simulator, the CAN network is implemented using HIL IPG. This is shown in Figure E-3 and Figure E-2. On the AutoBox side, the controller is implemented on the CAN network using RTI Simulink blocks from the dSpace RTI CAN library, as depicted in Figure E-3. These figures show the data type conversion on the RX and TX side between the CAN network and Simulink. Signals on the TX side are multiplied by an array of values that exploit the maximum accuracy of the specified datatype of the CAN network. On the RX side, these values are divided using the same array of values. In future works, more advanced conversion techniques could be used.

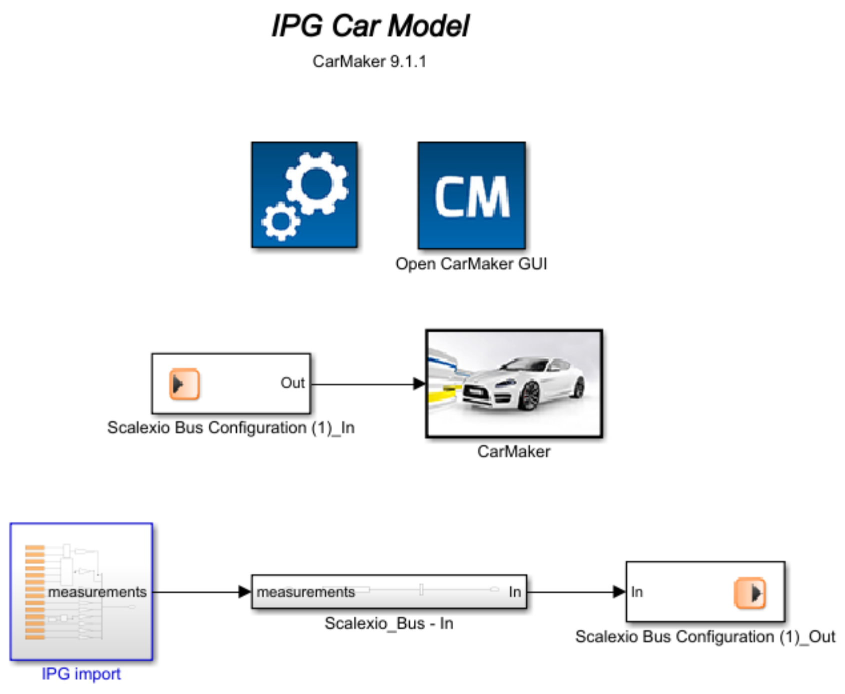


**Figure E-1:** CAN setup simulator

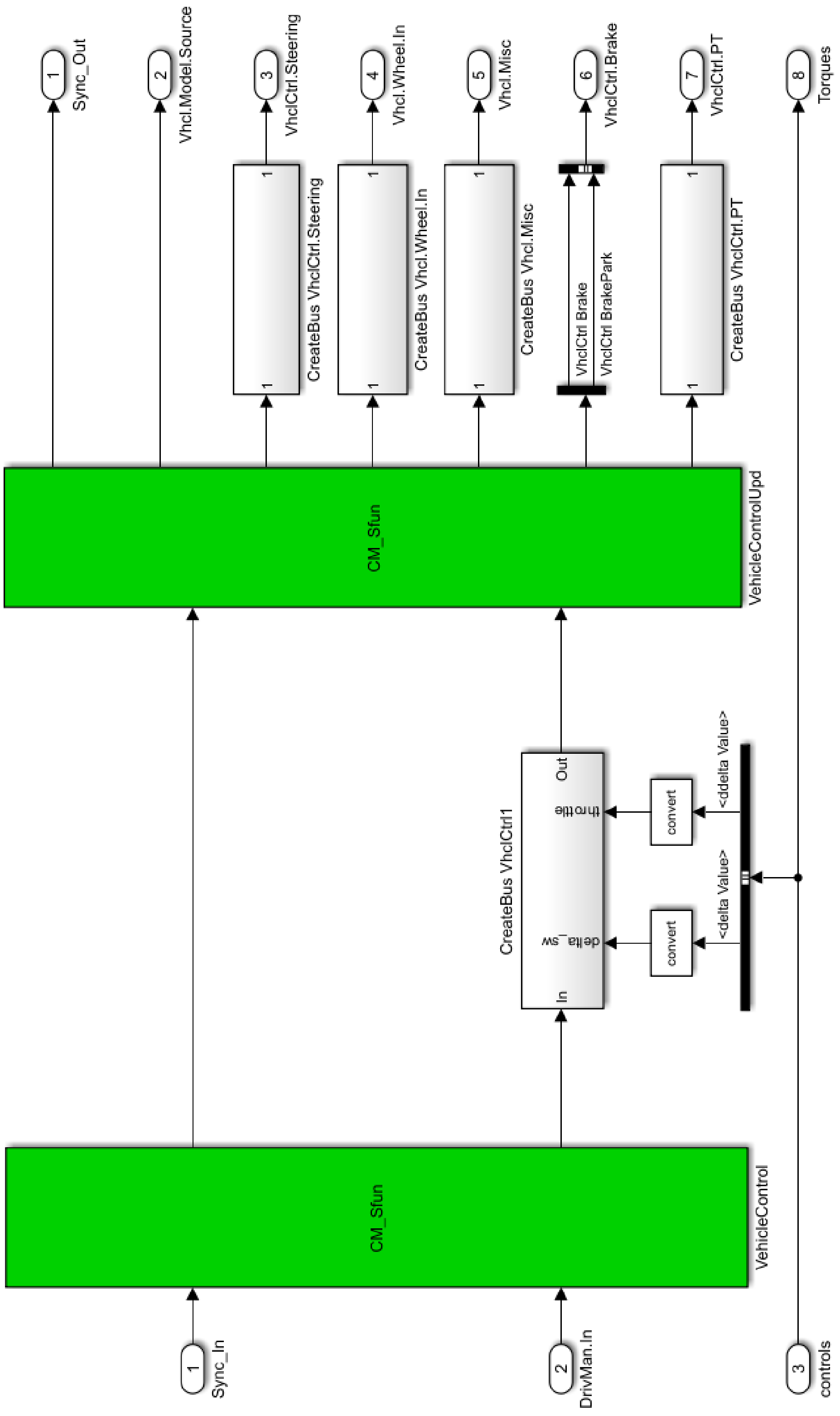


**Figure E-2:** HIL IPG setup

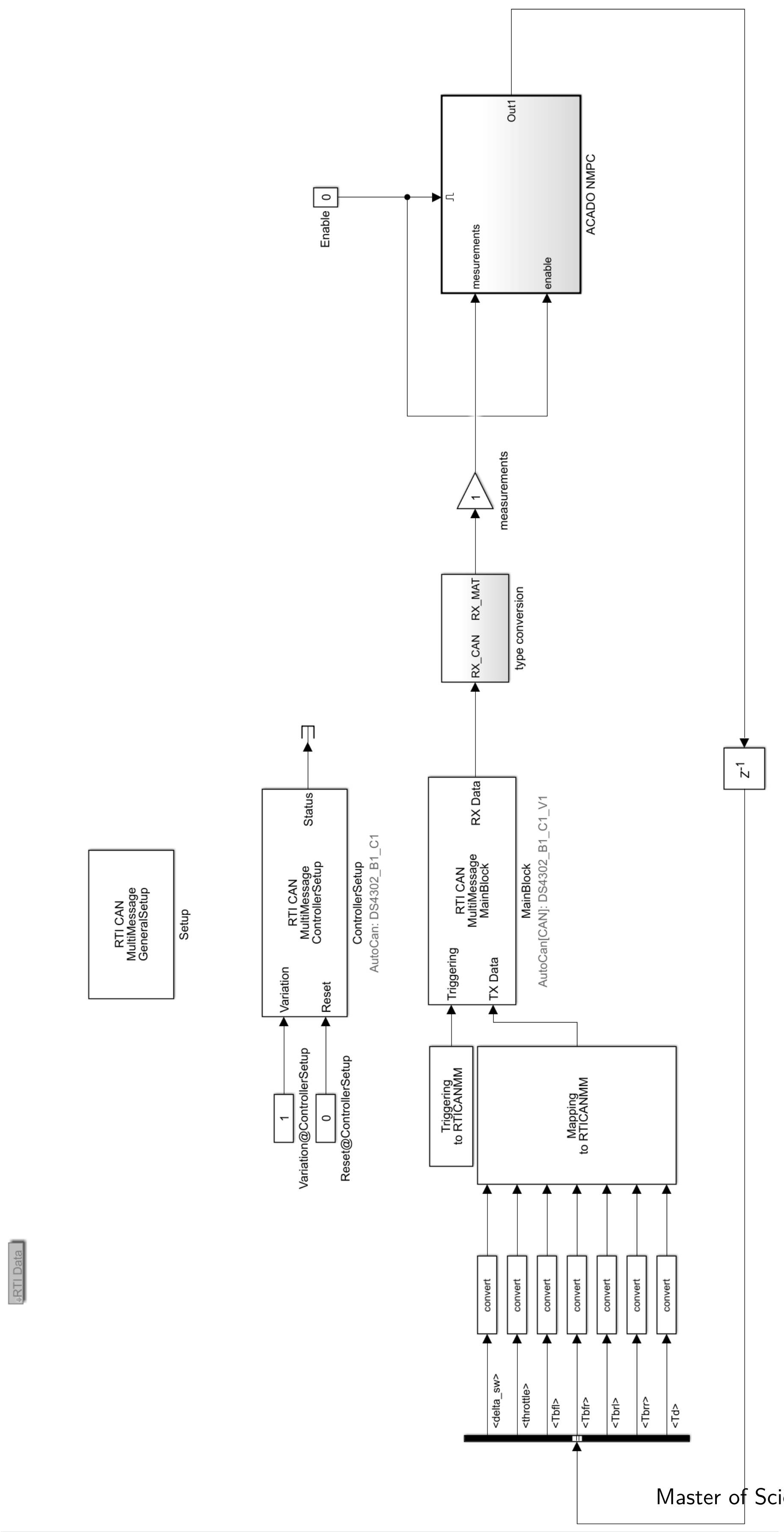


**Figure E-3:** CAN setup controller

# Bibliography

[1] Beal, C. E. and Gerdes, J. C. Model predictive control for vehicle stabilization at the limits of handling. *IEEE Transactions on Control Systems Technology*, 21:1258–1269, 2013.

[2] Brown, M. and Gerdes, J. C. Coordinating tire forces to avoid obstacles using nonlinear model predictive control. *IEEE Transactions on Intelligent Vehicles*, 5:21–31, 2020.

[3] Chen, Y., Bruschetta, M., Picotti, E., and Beghi, A. Matmpc, a matlab based toolbox for real-time nonlinear model predictive control. *European Control Conference (ECC)*, pages 3365–3370, 2019.

[4] Choi, M. and Choi, S. B. Model predictive control for vehicle yaw stability with practical concerns. *IEEE Transactions on Vehicular Technology*, 63(8):3539–3548, 2014.

[5] Chowdhri, N., Ferranti, L., Iribarren, F. S., and Shyrokau, B. Integrated nonlinear model predictive control for automated driving. *Control Engineering Practice*, 106:104654, 2021.

[6] Dakhlallah, J., Glaser, S., Mammar, S., and Sebsadji, Y. Tire-road forces estimation using extended kalman filter and sideslip angle evaluation. *the American Control Conference*, pages 4597–4602, 2008.

[7] Diehl, M., Ferreau, H. J., and Haverbeke, N. *Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[8] Erlien, S. M., Fujita, S., and Gerdes, J. C. Shared steering control using safe envelopes for obstacle avoidance and vehicle stability. *IEEE Transactions on Intelligent Transportation Systems*, 17:441–451, 2016.

[9] Falcone, P., Borrelli, F., Tseng, H. E., Asgari, J., and Hrovat, D. A hierarchical model predictive control framework for autonomous ground vehicles. *American Control Conference*, pages 3719–3724, 2008.

[10] Frasch, J. V., Gray, A., Zanon, M., Ferreau, H. J., Sager, S., Borrelli, F., and Diehl, M. An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles. *European Control Conference (ECC)*, pages 4136–4141, 2013.

[11] Funke, J., Brown, M., Erlien, S. M., and Gerdes, J. C. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios. *IEEE Transactions on Control Systems Technology*, 25:1204–1216, 2017.

[12] Gao, Y., Gray, A., Tseng, H. E., and Borrelli, F. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52:802–823, 2014.

[13] Gray, A., Gao, Y., Lin, T., Hedrick, J. K., Tseng, H. E., and Borrelli, F. Predictive control for agile semi-autonomous ground vehicles using motion primitives. *2012 American Control Conference (ACC)*, pages 4239–4244, 2012.

[14] Gros, S., Buccieri, D., Mullhaupt, P., and Bonvin, D. *A Two-Time-Scale Control Scheme for Fast Unconstrained Systems*, pages 551–563. Springer Berlin Heidelberg, 2014.

[15] Hajiloo, R., Abroshan, M., Khajepour, A., Kasaiezadeh, A., and Chen, S. K. Integrated steering and differential braking for emergency collision avoidance in autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22:3167–3178, 2021.

[16] Hamann, H., Hedrick, J. K., Rhode, S., and Gauterin, F. Tire force estimation for a passenger vehicle with the unscented kalman filter. *IEEE Intelligent Vehicles Symposium*, page 814 – 819, 2014.

[17] Hsu, Y. H. J., Laws, S. M., and Gerdes, J. C. Estimation of tire slip angle and friction limits using steering torque. *IEEE Transactions on Control Systems Technology*, 18:896–907, 2010.

[18] Jo, K., Kim, J., Kim, D., Jang, C., and Sunwoo, M. Development of autonomous car - part ii: A case study on the implementation of an autonomous driving system based on distributed architecture. *IEEE Transactions on Industrial Electronics*, 62:5119–5132, 2015.

[19] Kohler, J., Soloperto, R., Muller, M. A., and Allgower, F. A computationally efficient robust model predictive control framework for uncertain nonlinear systems. *IEEE Transactions on Automatic Control*, 66:794–801, 2021.

[20] Laurense, V. A. and Gerdes, J. C. Long-horizon vehicle motion planning and control through serially cascaded model complexity. *IEEE Transactions on Control Systems Technology*, 30:166–179, 2022.

[21] Laurense, V. A., Goh, J. Y., and Gerdes, J. C. Path-tracking for autonomous vehicles at the limit of friction. *the American Control Conference*, pages 5586–5591, 2017.

[22] Liang, Y., Li, Y., Khajepour, A., Huang, Y., Qin, Y., and Zheng, L. A novel combined decision and control scheme for autonomous vehicle in structured road based on adaptive model predictive control. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[23] Malone, N., Chiang, H. T., Lesser, K., Oishi, M., and Tapia, L. Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field. *IEEE Transactions on Robotics*, 33:1124–1138, 2017.

[24] Metzler, M., Tavernini, D., Gruber, P., and Sorniotti, A. On prediction model fidelity in explicit nonlinear model predictive vehicle stability control. *IEEE Transactions on Control Systems Technology*, 29:1964–1980, 2021.

[25] Pacejka, H. B. *Tyre and vehicle dynamics.* Elsevier Butterworth-Heinemann, 2009.

[26] Plessen, M. G., Bernardini, D., Esen, H., and Bemporad, A. Spatial-based predictive control and geometric corridor planning for adaptive cruise control coupled with obstacle avoidance. *IEEE Transactions on Control Systems Technology*, 26:38–50, 2018.

[27] Quirynen, R., Berntorp, K., and Cairano, S. D. Embedded optimization algorithms for steering in autonomous vehicles based on nonlinear model predictive control. *2018 Annual American Control Conference (ACC)*, pages 3251–3256, 2018.

[28] Reiter, R., Nurkanović, A., Frey, J., and Diehl, M. Frenet-cartesian model representations for automotive obstacle avoidance within nonlinear mpc. *European Journal of Control*, page 100847, 2023.

[29] Schwarting, W., Alonso-Mora, J., Paull, L., Karaman, S., and Rus, D. Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model. *IEEE Transactions on Intelligent Transportation Systems*, 19:2994–3008, 2018.

[30] Schwickart, T., Voos, H., Darouach, M., and Bezzaoucha, S. A flexible move blocking strategy to speed up model-predictive control while retaining a high tracking performance. *European Control Conference (ECC)*, pages 764–769, 2016.

[31] Siampis, E., Velenis, E., Gariuolo, S., and Longo, S. 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*, 26:1982–1994, 2018.

[32] Stano, P., Montanaro, U., Tavernini, D., Tufo, M., Fiengo, G., Novella, L., and Sorniotti, A. Model predictive path tracking control for automated road vehicles: A review. *Annual Reviews in Control*, 55:194–236, 2023.

[33] Subosits, J. and Gerdes, J. C. Impacts of model fidelity on trajectory optimization for autonomous vehicles in extreme maneuvers. *IEEE Transactions on Intelligent Vehicles*, 2021.

[34] Thornton, S. M., Pan, S., Erlien, S. M., and Gerdes, J. C. Incorporating ethical considerations into automated vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, 18:1429–1439, 2017.

[35] Tjonnas, J. and Johansen, T. A. Stabilization of automotive vehicles using active steering and adaptive brake control allocation. *IEEE Transactions on Control Systems Technology*, 18:545–558, 2010.

[36] Wielitzka, M., Dagen, M., and Ortmaier, T. State and maximum friction coefficient estimation in vehicle dynamics using ukf. *the American Control Conference*, 0:4322–4327, 2017.

[37] Wolf, M. T. and Burdick, J. W. Artificial potential functions for highway driving with collision avoidance. *IEEE International Conference onRobotics and Automation*, 2008.

[38] Wurts, J., Stein, J. L., and Ersal, T. Collision imminent steering at high speeds on curved roads using one-level nonlinear model predictive control. *IEEE Access*, 9:39292–39302, 2021.

[39] Wurts, J., Dallas, J., Stein, J. L., and Ersal, T. Adaptive nonlinear model predictive control for collision imminent steering with uncertain coefficient of friction. *2020 American Control Conference (ACC)*, pages 4856–4861, 2020.

[40] Wurts, J., Stein, J. L., and Ersal, T. Design for real-time nonlinear model predictive control with application to collision imminent steering. *IEEE Transactions on Control Systems Technology*, 30:2450–2465, 2022.

# Glossary

## List of Acronyms

| | |
|---|---|
| **MPC** | Model Predictive Control |
| **APF** | Artificial Potential Field |
| **DoF** | Degrees of Freedom |
| **UKF** | Unscented Kalman Filter |
| **ESC** | Electronic Stability Control |
| **NMPC** | Nonlinear Model Predictive Control |
| **RK2** | Runge-Kutta 2th order |
| **UKF** | Unscented Kalman Filter |
| **RLS** | Recursive Least Squares |
| **CIC** | Collision Imminent Control |
| **CoM** | Center of Mass |
| **OCP** | Optimal Control Problem |
| **SQP** | Sequential Quadratic Programming |
| **RTI** | Real Time Iterations |
| **KKT** | Karush-Kuhn-Tucker |
| **LTI** | Linear Time Invariant |
| **LTV** | Linear Time Varying |
| **HIL** | Hardware in the Loop |
| **NLP** | Nonlinear Program |
| **OCP** | Optimal Control Problem |
| **QP** | Quadratic Programming |
| **ST** | Single Track |
| **DT** | Double Track |
| **ABX** | AutoBox |

| | |
|---|---|
| **MABX** | MicroAutoBox |
| **SCX** | SCALEXIO |

# Nomenclature

## List of Symbols

| | |
|---|---|
| $\dot{\psi}$ | Yaw axis angular velocity [rad/s] |
| $\alpha$ | Tire slip angle [rad] |
| $\alpha_s$ | Tire saturation slip angle [rad] |
| $\alpha_{UKF}$ | UKF sigma point spread [-] |
| $\beta$ | Vehicle sideslip angle [rad] |
| $\beta_{UKF}$ | UKF prior [-] |
| $\delta$ | Vehicle front steering angle [rad] |
| $\delta_{sw}$ | Steering wheel angle [rad] |
| $\Delta F_{zx}$ | Longitudinal load deviation [N] |
| $\Delta F_{zy}$ | Lateral load deviation [N] |
| $\Delta t$ | Controller time step [-] |
| $\mathcal{F}_b$ | Vehicle body fixed frame [-] |
| $\mathcal{F}_g$ | Global frame [-] |
| $\dot{\psi}$ | Vehicle yaw rate [rad/s] |
| $\eta$ | Friction limit conservativeness factor [-] |
| $\gamma$ | Vehicle front axle lateral load distribution [-] |
| $\hat{\xi}$ | Estimation vector of the UKF [-] |
| $\kappa_{UKF}$ | UKF tuning parameter [-] |
| $\mu$ | Friction coefficient [-] |
| $\Phi(\xi_k, u_k)$ | Discrete state update function [-] |
| $\psi$ | Vehicle yaw angle in the global frame [rad] |
| $\psi_\sigma$ | Yaw angle of the trajectory [rad] |
| $\sigma$ | Road fixed trajectory [-] |
| $\theta$ | Angle between the vehicle and an obstacle [rad] |
| $\xi$ | Prediction model state space [-] |
| $\xi_N$ | NMPC terminal state [-] |

| | |
|---|---|
| $(X_v, Y_v)$ | Global vehicle coordinates [m] |
| $(s_o, e_o)$ | Obstacle location [m] |
| $(X_\sigma, Y_\sigma)$ | Global trajectory coordinates [m] |
| $\ddot{\psi}$ | Angular yaw acceleration of the vehicle [rad/s$^2$] |
| $\ddot{x}$ | Longitudional acceleration of the vehicle [m/s$^2$] |
| $\ddot{y}$ | Lateral acceleration of the vehicle [m/s$^2$] |
| $\Delta s_a$ | Distance from the first obstacle where obstacles appear in experiments [m] |
| $\Delta s_o$ | Distance between obstacles in experiments [m] |
| $\dot{\hat{s}}_o, \dot{\hat{e}}_o$ | Dynamic obstacle velocity [m/s,m/s] |
| $\dot{F}_x$ | Longitudinal tire force rate [N/s] |
| $\dot{x}$ | Longitudinal vehicle velocity [m/s] |
| $\dot{x}_{\text{ref}}$ | Reference loditudinal velocity [m/s] |
| $\dot{x}_{ref}$ | Reference velocity [m/s] |
| $\dot{y}$ | Lateral vehicle velocity [m/s] |
| $\hat{p}$ | Estimated parameters |
| $\hat{p}$ | Parameter vector [-] |
| $\hat{s}_o(0), \hat{e}_o(0)$ | Initial dynamic obstacle location [m,m] |
| $\kappa(s)$ | Road curvature [1/m] |
| $\lambda_x$ | Longitudinal brake distribution parameter [-] |
| $\lambda_y$ | Lateral brake distribution parameter [-] |
| $\lambda_{x_{\text{nat}}}$ | Natural logitudional brake distribution [-] |
| $\lambda_{y_{\text{nat}}}$ | Natural lateral brake distribution [-] |
| $\omega_{w_i}$ | Wheel speed of the wheel at $i = (f, r)$ [rad/s] |
| $\omega_w$ | Wheel rotational velocity [rad/s] |
| $a_{cc}$ | Accelerator control signal [-] |
| $B, C, E, B_x$ | Pacejka tire model parameters [-] |
| $c(\xi_k, u_k)$ | NMPC nonlinear stage constraint function [-] |
| $c(\xi_N)$ | NMPC nonlinear terminal constraint function [-] |
| $C_\alpha$ | Tire stiffness factor [-] |
| $c_L$ | NMPC lower bound of nonlinear stage constraints [-] |
| $c_U$ | NMPC upper bound of nonlinear stage constraints [-] |
| $c_{dist}$ | Obstacle distance cost [-] |
| $c_{NL}$ | NMPC lower bound of nonlinear terminal constraints [-] |
| $c_{NU}$ | NMPC upper bound of nonlinear terminal constraints [-] |
| $Cd_0$ | Linear drag offset parameter [N] |
| $Cd_0$ | Linear drag scaling parameters [Ns/m] |
| $d$ | Distance between the vehicle and an obstacle or road boundary [-] |
| $d_l$ | Minimum distance between the vehicle and the left road boundary [m] |
| $d_o$ | Minimum distance between the vehicle and obstacle [m] |
| $d_o^-$ | Obstacle distances lower than the minimum [m] |

| | |
|---|---|
| $d_r$ | Minimum distance between the vehicle and the right road boundary [m] |
| $d_r^-$ | Road boundary distances lower than the minimum [m] |
| $d_v$ | Distance from the center to the edge of geometric shape fitted to the vehicle [m] |
| $d_{\min}$ | Safe distance boundary [-] |
| $d_{o_{\min}}$ | Minimum safe obstacle distance [m] |
| $d_{r_{\min}}$ | Minimum safe road boundary distance [m] |
| $dt(n)$ | Variable timestep [s] |
| $dt_1$ | Initial timestep [s] |
| $e_\psi$ | Angle between the vehicle velocity and the road direction [rad] |
| $e_y$ | Vehicle distance perpendicular to the trajectory [m] |
| $e_{y_{\max}}$ | Maximum lateral coordinate of the vehicle geometry formulation [m] |
| $e_{yl}(s)$ | Left distance varying lateral bound [m] |
| $e_{yr}(s)$ | Right distance varying lateral bound [m] |
| $ey_{ref}(s)$ | Reference trajectory [m] |
| $f(\xi, u)$ | Prediction model dynamics [-] |
| $F_{drag}$ | Drag force on the vehicle [N] |
| $f_{ec}(z)$ | NLP equality constraints |
| $F_{i,j,k}$ | Tire forces at axis $i = (x, y, z)$, axle $j = (f, r)$, and side $k = (l, r)$ [N] |
| $F_{i,j}$ | Single track tire forces along axis $i = (x, y, z)$, at axle $j = (f, r)$ [N] |
| $f_{ic}(z)$ | NLP inequality constraints [-] |
| $f_{QP}(z)$ | QP objective function [-] |
| $F_{sf}$ | Static load on the front axle [N] |
| $F_{sr}$ | Static load on the rear axle [N] |
| $F_{x_{\text{tot}}}$ | Total longitudinal tire force [N] |
| $F_{y\max}$ | Maximum lateral tire force [N] |
| $F_{y_{\text{tot}}}$ | Total lateral tire force [N] |
| $g$ | Acceleration due to gravity [m/s$^2$] |
| $H$ | Hessian matrix [-] |
| $h$ | Height of the vehicle CoM [m] |
| $I_z$ | Vehicle $z$-axis inertia [kgm$^2$] |
| $I_{w_i}$ | Moment of inertia of the wheel at $i = (f, r)$ [kgm$^2$] |
| $I_{w_i}$ | Wheel moment of inertia [kgm$^2$] |
| $J(\xi, u)$ | NMPC cost function [-] |
| $J_N(\xi, u)$ | NMPC terminal cost [-] |
| $k_x$ | Longitudinal load derivative scaling parameter [-] |
| $k_y$ | Lateral load derivative scaling parameter [-] |
| $k_{ax}$ | Longitudinal tire load scaling parameter [-] |
| $k_{ay}$ | Lateral tire load scaling parameter [-] |
| $L$ | Center distance between the vehicle and obstacle [m] |
| $l_f$ | Vehicle distance between the CoM and front axle [m] |

| | |
|---|---|
| $L_o$ | Set of obstacle locations [-] |
| $l_r$ | Vehicle distance between the CoM and rear axle [m] |
| $m$ | Vehicle mass [kg] |
| $M_r$ | Wheel resistance torque [Nm] |
| $M_x$ | Additional yaw moment due to longitudinal forces [Nm] |
| $M_y$ | Additional yaw moment due to lateral forces [Nm] |
| $N$ | Prediction horizon |
| $N_1$ | Initial timestep horizon [-] |
| $n_{\text{arctan}}$ | Arctan scaling paramter |
| $n_{a_{cc}}$ | Accelerator scale factor [-] |
| $n_{sw}$ | Steering rate [-] |
| $P_0, R_v, R_n$ | UKF covariance matrices [-] |
| $P_T$ | Prediction time window [s] |
| $Q_{dist}$ | Distance cost scaling parameter [-] |
| $r_o$ | Radius of a circle fitted to the obstacle [m] |
| $r_v$ | Vehicle radius [m] |
| $r_{w_i}$ | Radius of the wheel at $i = (f, r)$ [m] |
| $r_{w_i}$ | Wheel radius [m] |
| $rb_l$ | Left road boundary location [m] |
| $rb_r$ | Right road boundary location [m] |
| $s$ | Distance along the trajectory [m] |
| $T_f$ | Torque at the front axle [Nm] |
| $T_r$ | Torque at the rear axle [Nm] |
| $T_w$ | Wheel torque [Nm] |
| $T_{b,i,j}$ | Braking torque at axle $i = (f, r)$ and side $j = (l, r)$[-] |
| $u$ | Input space [-] |
| $u_p$ | Input space plant model [-] |
| $u_{UKF}$ | UKF input vector [-] |
| $v$ | Vehicle velocity [m/s] |
| $w$ | Vehicle width [m] |
| $y_{UKF}$ | UKF measurement vector [-] |
| $z$ | NLP state and input vector $(\xi, u)$ [-] |