Real-Time Collision Imminent Steering Using One-Level Nonlinear Model Predictive Control

by

John B. Wurts

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2020

John Benjamin Wurts

jbwurts@umich.edu

ORCID iD: 0000-0002-4049-8945

for me

Acknowledgments

As I was preparing for my Eagle Scout court of honor, I came across the saying "behind every Eagle Scout who believes in himself, is an Eagle Scout mom who believed in him first." As I near completion of my doctoral studies, I now appreciate how the work of doctoral students is empowered by so many along the journey.

First and foremost, I wish to extend my sincerest gratitude to my advisers, Dr. Tulga Ersal and Dr. Jeffrey Stein. It is through their leadership, insight, and patience that allowed me to grow as a researcher, but more broadly, critical thinker. On multiple occasions they have allowed me to take risks in my research, foremost accepting me as a graduate student, but also allowing me to branch far from our expertise along the research path. Without their personal investment and desire to see me succeed, this work herein would not have been possible.

My appreciation extends to my committee members and lab mates. I appreciate the efforts of my committee to bring forward their insight as a means of refining my research, with a special appreciation to Vishnu Desaraju for his efforts to showcase the work herein on a live vehicle. And my appreciation to my lab mates, both early in my studies to support my development and later towards completion to keep me accountable and as an opportunity to continue the cycle.

I also wish to acknowledge Toyota Research Institute for their funding to support my graduate studies. Through their university partnerships dozens of graduate students are able to participate in cutting edge research while pursue advanced degrees. The work herein is not possible without their support.

To Elizabeth, as a personal example of the triumphs of hard work through adversary and challenging times.

To my family, parents Judy and Michael, and siblings Meghan and Andrew, as a steady guiding light to keep me on track and ever growing in life.

Along my journey, I am ever thankful of the personal investments those in my life make in me. If I had to pin-point a single instance that led to this point, it was when my father gifted me a Raspberry Pi for Christmas in 2014. That Raspberry Pi supported my tinkering in electronics, and eventual automation of a remote controlled car. It took many years of playing with that pet project until it grew into the scope of a research project, morphing far from my initial tinkering.

That Christmas I received a few other electronics tinkering sets, most of which did not go anywhere. But without that initial investment in my curiosity and passion, I would not be where I am today. The path to the end goal is not known, but I am forever grateful for the efforts by those in my life to encourage me to continue on.

Table of Contents

List of Tables

List of Figures

Abstract

Automotive active safety features are designed to complement or intervene a human driver's actions in safety critical situations. Existing active safety features, such as adaptive cruise control and lane keep assist, are able to exploit the ever growing sensor and computing capabilities of modern automobiles. An emerging feature, collision imminent steering, is designed to perform an evasive lane change to avoid collision if the vehicle believes collision cannot be avoided by braking alone. This is a challenging maneuver, as the expected highway setting is characterized by high speeds, narrow lane restrictions, and hard safety constraints. To perform such a maneuver, the vehicle may be required to operate at the nonlinear dynamics limits, necessitating advanced control strategies to enforce safety and drivability constraints.

This dissertation presents a one-level nonlinear model predictive controller formulation to perform a collision imminent steering maneuver in a highway setting at high speeds, with direct consideration of safety criteria in the highway environment and the nonlinearities characteristic of such a potentially aggressive maneuver. The controller is cognizant of highway sizing constraints, vehicle handling capability and stability limits, and time latency when calculating the control action. In simulated testing, it is shown the controller can avoid collision by conducting a lane change in roughly half the distance required to avoid collision by braking alone. In preliminary vehicle testing, it is shown the control formulation is compatible with the existing perception pipeline, and prescribed control action can safely perform a lane change at low speed.

Further, the controller must be suitable for real-time implementation and compatible with expected automotive control architecture. Collision imminent steering, and more broadly collision avoidance, control is a computationally challenging problem. At highway speeds, the required time for action is on the order of hundreds of milliseconds, requiring a control formulation capable of operating at tens of Hertz. To this extent, this dissertation investigates the computational expense of such a controller, and presents a framework for designing real-time compatible nonlinear model predictive controllers. Specifically, methods for numerically simulating the predicted vehicle response and response sensitivities are compared, their cross interaction with trajectory optimization strategy are considered, and the resulting mapping to a parallel computing hardware architecture is investigated. The framework systematically evaluates the underlying numerical optimization problem for bottlenecks, from which it provides alternative solutions strategies to achieve real-time performance. As applied to the baseline collision imminent steering controller, the procedure results in an approximate three order of magnitude reduction in compute wall time, supporting real-time performance and enabling preliminary testing on automotive grade hardware.

Chapter 1

Introduction

1.1 Motivation

Automotive related accidents have been declining over the past few decades, in large part due to improvements in passive safety features. These features, including mandatory seat belts and airbags, greatly improve the survivability of accidents, but active safety features reduce fatalities through avoiding accidents all together. Active safety features, such as adaptive cruise control and anti-lock braking systems, are designed to complement or even override a human driver in challenging or dangerous scenarios [1]. Modern automobiles are equipped with ever-growing sensor and compute capabilities, supporting advanced active safety features such as lane departure warning, lane keep assist, and collision imminent braking. As vehicles tend towards higher levels of autonomy, these active safety features address limited scopes of the complex situations autonomous vehicles have to face [2].

One emerging active safety feature is collision imminent steering (CIS), which is designed to swerve and change lanes if the vehicle detects a forward collision cannot be avoided by braking alone. This can be a challenging maneuver to perform, as limited space constraints may require pushing the vehicle to its nonlinear limits of handling, which can be a dangerous operating condition, as vehicle control can quickly be lost [3],

[4]. Thus, advanced controllers are required to properly account for these nonlinearities, balancing the control action against safety constraints and lane change performance.

CIS is a specific case of collision avoidance type maneuvers. Collision avoidance controllers have been studied in varying levels of controller and environmental complexity. At low speeds, system nonlinearities are not significant, reducing the complexity of the required solution [5]. At high speeds, system nonlinearities must be directly accounted for, increasing the complexity of the controller, often at computational speeds prohibitive of real-time performance [6].

In the case of CIS, the maneuver is expected to take place in a highway setting, as the inherent high speed requires a long braking distance. However, a highway setting is challenging from a controller design perspective, because the vehicle nonlinearities must be directly accounted for in the controller, yet high speeds and lane sizing mandate a faster solution time.

To this extent, the overarching goal of this thesis is to address two critical needs. First, design of an optimal controller is needed that accounts for system nonlinearities, appropriately balancing vehicle handling capability against lane change performance. The fitness of a candidate CIS maneuver is not necessarily following a reference trajectory, thus optimality criteria need to be developed specific for a CIS maneuver. Hard safety constraints need to be introduced into the controller; ensuring feasible solutions obey vehicle stability limits, as well as collision avoidance guarantees. The controller needs to be designed with consideration to expected perception capability [7], supporting live sensor data input and reducing dependency on *a priori* information.

Second, the controller needs to be compatible with real-time performance, which requires scrutinizing the computational expense. To achieve real-time performance, four areas need to be studied. First, the computational cost required to numerically approximate the prediction model needs to be studied for different numerical integration techniques. Second, sensitivities of the vehicle prediction trajectory need to be considered, relating to the derivatives of the fitness and feasibility criteria in the numerical optimization problem. Third, different trajectory optimization strategies needs to be studied, taking careful consideration of how these methods map into the computational cost of solving the optimization problem. Fourth, modern parallel hardware architectures needs to be considered, and the required mapping of the trajectory optimization strategy into the parallel framework presented.

To understand the scope of the desired CIS controller, relevant literature in the collision avoidance space is presented next.

1.2 Literature Review on Automotive Collision Avoidance Controllers

Automotive collision avoidance controllers are typically structured as one of two architecture types: two-level and one-level strategies [8]. In a two-level architecture, the task of path planning and path following are separated, with each task handled by a different controller or strategy. In contrast, a one-level architecture combines the task of path planning with path following, resulting in a control formulation to solve for both simultaneously in agreement.

Consider first a two-level strategy. Two-level architectures are comparatively computationally easier, as the two independent tasks of path planning and path tracking are both significantly reduced in complexity. Early implementations of two-level architectures focused on achieving collision avoidance through trajectory generation. For example, potential energy field methods can simulate a populated environment where obstacles are regions of high potential energy [9]. From a provided input of a topographical start and terminal position, an initial straight line is morphed, minimizing the trajectory's potential energy through the environment, thus avoiding obstacles represented as a weighted high energy region. The resulting reference trajectory is passed to a lower level controller, such as a PID based controller, to guide the vehicle through the environment [10]. In these cases, obstacle avoidance is not natively addressed by the reference trajectory; rather, these methods rely on the path following controller for obstacle avoidance.

There are two primary drawbacks to using potential energy methods to generate a reference trajectory. First, the appropriate weightings used to generate the potential energy field are not known *a priori*, as they depend on the scenario. Second, potential energy methods cannot distinguish difficult to navigate trajectories from infeasible trajectories that violate safety constraints. Thus, for safety critical situations, hard safety constraints are preferable [11].

An alternative approach for generating a reference trajectory is to utilize known structure of the environment. For dedicated road applications, the center of a lane can be used as a reference trajectory, with lateral deviation limits established through the edge markings [12]. Alternatively, other optimality criteria can be used for trajectory generation, such as minimum lap time [13], or relative passing orientation to obstacles [14].

For path following applications in structured lane settings, using two-level architectures is a common approach, because lateral deviation limits can be generated with respect to a reference trajectory. In the context of operating within curved roads, the reference trajectory is often set as the lane center line because the lateral deviation limits are well defined as the lane widths [15]. To enforce lane boundary limits, the path following controller constrains the lateral deviation to within $\pm\frac{\text{lane width}}{2}$. Alternatively, controllers have been developed for variable lateral deviation limits defined by splines [16].

A more complex methodology of storing lateral deviation limits also includes obstacle information. The *drivable tube* [17] or *safe corridor* [12] concepts represent the allowable space the vehicle can safely reside in to be collision free and within the road limits. A reference trajectory is first propagated through the drivable tube, and at each discrete integration point of the reference trajectory, the left and right distance to the drivable tube is calculated. Similar to prior controllers, these deviation limits are then stored within the optimization problem to form linear constraints, supporting a quadratic penalty path following controller. For the controller of [17], it is shown to navigate sharp turns while maintaining deviation limits and stability criteria, but the underlying optimality is to follow a trajectory known *a priori*, and is not intended to deviate far from the seeding reference trajectory.

For other applications, alternative methods of generating the reference trajectory have been explored [18, 13, 19, 20], but the enforcement of collision-free path following mirrors the same approach: propagate the reference trajectory through the environment and at each discrete integration point, calculate the allowable deviation limits.

Once a reference trajectory is generated, a separate controller handles path following. However, not all path following controllers are capable of ensuring collision avoidance. For example, PID controllers cannot guarantee the satisfaction of the safety constraints. Specifically, PID controllers cannot control the magnitude or position of state trajectory following error, thus cannot ensure the safety of the host vehicle, especially when the CIS maneuver needs to push the vehicle to its dynamic limits. For example, certain provided reference trajectories can overly excite the vehicle response for a set of PID weights, resulting in vehicle roll over [10].

To address the shortcomings of simple low-level path following controllers, advanced controllers have been implemented to capture system nonlinearities. For example, reachability set based controllers [21], [22] as well as feedback-feedforward controllers [23], [18], [13] can incorporate estimated future state information into current control input and greatly improve tracking error. In [23], a feedback-feedforward controller is developed for a performance race car, where vehicle implementation testing showcases a successful following of the optimal racing line through a race course at high speed. However, it also shows that following a pre-computed optimal reference trajectory does not always push the vehicle to its dynamic limits, as the optimal trajectory changes dynamically as the vehicle deviates from the reference path [24].

Alternatively, model predictive control (MPC) has been considered for the path tracking problem. Using a reference trajectory as input, the MPC problem is structured to best follow the trajectory [15], [12], [25], [26], [27], [28]. Using MPC for path following is advantageous, as complex stability and other constraints can be introduced into the formulation. However, the provided reference trajectory cannot guarantee it fully utilizes the vehicle handling capabilities, which is essential for safety critical maneuvers.

Overall, the core drawbacks from two-level architectures are due to the reference trajectory. While a common approach, two-level architectures have three key limitations.

Limitation 1: drivability of reference trajectory. By splitting trajectory generation from path following, two-level architectures cannot ensure the reference trajectory is drivable. Often, the reference trajectory is generated without regards to the system dynamics, which can be problematic in safety critical applications, which might require operating up to the limits of handling, though not exceeding them. The issues of drivability can be so pronounced that traditional PID controllers can lead to vehicle rollover in certain scenarios [10]. Some approaches supplement the reference trajectory with pseudometrics, such as acceleration limits [29], [30], but this does not alleviate the problem. In fact, these limits can overly restrict reference trajectories, which can again be problematic for safety critical applications where a maneuver requires operating up to the limits of handling.

Limitation 2: fitness dictated by path following metric. A second complication with two-level MPC architectures is the fitness of the path following controller. Various controllers exist to follow a reference trajectory, most often focusing on minimizing the l^2-norm of $x - y$ trajectory deviation. For safety critical applications, the fitness of a maneuver might not be measured by $x - y$ trajectory response. Thus, generating an $x - y$ reference trajectory by some other optimality metric can be difficult, especially when operating at the vehicle limits of handling.

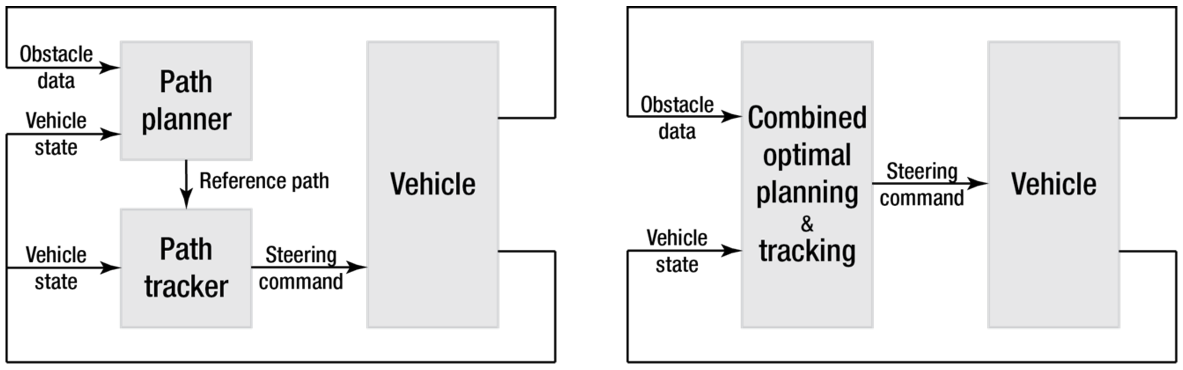
Limitation 3: optimality after deviation. A third complication with two-level MPC architectures occurs when a host vehicle has deviated from a reference path. When a small deviation occurs in closed-loop control implementation, the path following controller attempts to return the host vehicle to the reference trajectory. However, at the vehicle's starting deviated state, the optimal path through the environment may no longer be to return to the previous trajectory. Trying to return to the previous trajectory can push the vehicle above its limits of handling, or not accurately maximize the vehicle's handling capability, both cases resulting in suboptimal performance [24].

The problem of provable safe trajectory generation has been identified, with a few options presented [31]. Alternatively, one-level architectures simultaneously handle the path planning and path tracking problems with exact agreement. While one-level control architectures are a comparatively more complex problem, abstract fitness criteria can be implemented beyond path following. The most common one-level architectures are MPC formulations, although instances of model-free [32] and model reference adaptive control exist [33]. These methods all use some model of the system and extrapolate a predicted response, which is used in the one-level control formulation. To be clear, model-free methods refer to formulations that do not use an explicit model of the system, but rather some generic learned input-output response as a system model.

An example of a two-level versus one-level architecture can be seen in Fig. 1.1. In a two-level architecture, the obstacle information and vehicle state are passed into the path planner. The resulting reference path and vehicle state are passed into the path tracker, which prescribes a steering command to the vehicle. In contrast, for one-level architecture, the obstacle information and vehicle state are passed into the combined path planner and tracker controller, and the resulting steering command sent to the vehicle.

By introducing a one-level architecture, a system constraint is imposed such that the internally generated prediction horizon, which is in exact agreement of a reference trajectory, is guaranteed to be drivable according to the prediction model. This avoids the issue of drivability of a reference trajectory for two-level architectures, which is necessary for safety constraints. Further, if the formulation accurately accounts for and models the system limits, then the controller is allowed to push the vehicle to the defined limits of handling. Combined, these two features allow a CIS controller to ensure the candidate maneuver is drivable, and if necessary, allow the maneuver to push the vehicle to its limits of handling.

One-level MPC architectures have been applied to various driving scenarios already.



(a) two-level controller architecture (b) one-level controller architecture

Figure 1.1: In a two-level control architecture, a separate path planning block passes a reference trajectory to a path tracker, then sending a control command to the plant. In a one-level architecture, a combined path planning and path tracking controller generates a prediction trajectory internally, and the required control command sent to the plant.

For example, one such CIS controller can perform a lane change at low speeds based on intervention threshold [34]. Here, the controller monitors the current trajectory, and monitors the intervention effort required if a CIS were to be performed autonomously. If the autonomous intervention effort exceeds a threshold, the system can blend the human input with the controller or lock out the human, and perform the lane change. By using a different optimality condition compared to path following, the controller is not reliant on a provided state trajectory *a priori*, and can incorporate online sensor information to generate and modify a safer trajectory. In this case, the CIS lane change criteria are formulated based on a lateral displacement threshold at a certain position longitudinally for a straight road. However, this formulation is based on a low speed vehicle, which can be characterized with linear dynamics. As such, this formulation intentionally avoids operating at the vehicle's limits of handling.

To this extent, nonlinear model predictive control can handle nonlinear system response, albeit at a higher computational cost. One-level nonlinear MPC has been applied to various driving scenarios, showcasing complex fitness and feasibility metrics of the application. For example, one-level nonlinear MPC has been used to perform a T-bone mitigation maneuver [35]. In this scenario, a host vehicle believes it will act as the stem in T-bone automotive collision, which is a comparatively severe collision. Instead, the controller attempts to rotate the vehicle 90°, resulting in a side-side collision, thus reducing the severity.

The maneuver is defined by rotating the vehicle 90° in the shortest amount of time. This is a complex problem for a two-level architecture, as the required state trajectory to rotate 90° cannot be known *a priori*, and the time minimization fitness criterion is not well suited for a path following application.

Other one-level nonlinear MPC controllers have been considered in similarly complex conditions. For example, one-level nonlinear MPC is used to avoid vehicle roll-over in open environments [36], or avoid other unstable vehicle regions [37]. In these various applications, one-level nonlinear MPC can handle complex driving environments and nonlinear vehicle responses, but must adjust the controller for the specific vehicle in the specific setting. This often requires formulating the control problem based on the expected perception formulation, leveraging LIDAR data formats [38], moving obstacle estimation [39], or obstacle uncertainty [16], [40], [41], to name a few.

While these one-level nonlinear MPC controllers have been used in unstructured environments, incorporating both lane boundary constraints and obsticle information can be challenging. For example, [42] designs a one-level nonlinear MPC controller to mimic human input while maintaining lateral deviation limits in curved roads. This controller is able to relax a path following optimality condition in favor of mirroring human input as best as safely possible. However, the lateral deviation information is separate from obstacle information, requiring multiple constraints for each obstacle encountered, growing the optimization problem. Additionally, the controller is designed for a vehicle kinematics model, which does not ensure drivability.

Previous collision avoidance controllers have showcased the benefit of a one-level MPC architecture, and for aggressive and complex maneuvers, highlighted the necessity of nonlinear formulations. However, one-level nonlinear MPC controllers, while beneficial for their increased complexity, require a specific development for the candidate application. For these reasons, a one-level nonlinear MPC controller architecture is desirable, but a specific formulation for a candidate vehicle and scenario must be derived. Hence a gap exists where a controller designed for a CIS maneuver operating at the limits of handling does not exist.

While one-level nonlinear MPC controllers can capture complex scenarios, a common side effect to such formulations is increased computational cost. Therefore, various strategies and solutions used to address compute timing are reviewed next.

1.3 Literature Review on Computational Cost of Model Predictive Control

In the control community over the past few decades, MPC has emerged as a competitive solution to various control applications [43]. Fundamentally, MPC is designed to take control action now that is mindful of future control actions and system responses [44]. However, this increase in complexity comes at an increase in computational cost. Over the past few decades as computing power has grown, more complex levels of MPC can be implemented.

The simplest and least computationally expensive form of MPC comes from linear-quadratic regulator (LQR) controllers [45]. The LQR feedback control law is a specific solution to the algebraic Riccati equation. Specifically, LQR solves the optimal control input for a linear system for a quadratic cost function. Depending on the formulation, this special case gives a closed-form solution of the finite or infinite horizon path following problem based on a linear computation of the instantaneous state error. The result is a low computational complexity problem, which can be implemented in real-time on compute limited systems [46].

One drawback to LQR controllers is constraints in both state and input cannot be directly incorporated into the problem. Instead, the full linear MPC problem must be solved online, growing the computational cost. While still in the linear system domain, linear state and input constraints can be incorporated, and mapped into a quadratic programming problem efficiently [47]. Despite the increase in computational cost, modern compute systems can still achieve real-time performance. Applications such as automotive engine timing control [48], engine emission control [49], and quadcopter control [50], to name a few, can incorporate linear MPC solutions online in real-time.

Specific to automotive obstacle avoidance, linear MPC architectures have been implemented. Using a pre-generated reference path through an environment, linear MPC systems have been shown to avoid obstacles while still obeying lane boundary and other state and control limits [9], [12]. Due to the linear nature of the problem, these applications can still achieve real time performance, even on modern low-power hardware architectures [51], [15].

The key computational benefit to solving these linear MPC problems is in exploiting the linear time-invariant system dynamics, and solving the resulting quadratic programming efficiently [52], [53]. Implementing quadratic programs are desirable, because these can be solved within a predetermined tolerance in a fixed number of iterations [54], [55]. Further, this result is shown to map onto deterministic hardware, such as field-programmable gate arrays, which allows solving the control problem in a deterministic wall time [56].

However, these implementations rely on two key assumptions, namely, (i) a linear system model, or a system that can be well controlled using a linearized model, and (ii) a reference trajectory, from which the quadratic penalty can be generated. The first limitation can be addressed through nonlinear MPC, and the second through the designation of a one-level MPC instead of a two-level one. As discussed prior in Sec. 1.2, linear MPC formulations cannot support operating at the nonlinear limits of handling necessary for CIS controllers, and for based on the criteria of a CIS maneuver, an optimal reference trajectory is not available *a priori*.

In the case of nonlinear MPC, the nonlinear system dynamics and/or nonlinear objective functions and constraints can be incorporated, which typically break down the mapping to quadratic programming solutions [57], [58]. Instead, nonlinear numerical optimization must be employed, which can come at a significant computational penalty.

Solving the full nonlinear MPC problem in real time is challenging, even on modern compute architectures [59]. While researchers have explored various automotive applications for nonlinear MPC, the solutions are often too computationally expensive to achieve real-time [60], [38], [16], [61]. In order to reduce the compute cost, the underlying nonlinear MPC is reduced in scale. For example, the resulting nonlinear numerical optimization problem cost can be reduced by detuning the control frequency [11], linearizing the design point to map to a quadratic programming problem [25], reducing the system dynamic nonlinearities by mapping tire forces as control variables [17], or converging on feasible but sub-optimal solutions [62], among other options.

The second limitation, availability of a reference trajectory *a priori*, is addressed by one-level MPC, but incurs a higher computational cost, because the task of generating a trajectory online increases the problem size and complexity. If the controller's fitness is not characterized by a path following metric, a new objective function must be incorporated, and likely incurs a higher computational cost than a quadratic error penalty.

While researchers are attentive on the compute time of nonlinear MPC systems, the subtleties that are required to support compute scaling and achieve real-time performance are often masked by modifications to the MPC structure and not discussed. Hence, there is an opportunity to study factors beyond the MPC fitness and feasability formulation to address the computational cost. Specifically, how the differential equations governing the system trajectory are solved, various trajectory optimization strategies, and how these map onto modern computing hardware architectures can provide insight in achieving real-time performance. Altogether, balancing these aspects of the MPC problem could lend to a net computational speed up, beyond high level adjustments to the MPC formulation. However, the literature does not provide a framework for a comprehensive balance of these components.

Fundamentally, one-level nonlinear MPC controllers will always have a higher computational cost than linear path following controllers. However, for complex controller objectives, the merits of an advanced controller formulation have been identified and deemed necessary. Yet, for advanced nonlinear MPC controllers to be implemented, they must achieve real-time capability. Thus, it is proposed to scrutinize the underlying computational expense of such controllers and present various solutions to achieve real-time compatibility.

1.4 Research Objectives

The task of automotive obstacle avoidance is an extensive and ongoing research area in the control community. There exist many controller formulations to address various and diverse obstacle avoidance scenarios. These controllers can handle variations in environment structure, spanning from well-defined paved road settings to unstructured open fields, as well as varying levels of vehicle excitation. However, these controllers are often structured for a specific application, resulting in a gap where existing controllers do not fit the intended CIS maneuver. To this extent, this dissertation seeks to present a controller formulation that captures the relevant highway setting, while allowing the controller to operate up to, but not exceeding, the vehicle nonlinear limits of handling.

While the existing nonlinear MPC controllers can theoretically address their intended use case, they are typically incompatible with real-time performance. For feasible implementation on a passenger vehicle, a CIS controller must be able to solve the underlying nonlinear numerical optimization problem within real-time targets. While some nonlinear MPC controllers handle real-time constraints by simplifying or reducing the complexity of the MPC fitness and feasibility formulations, this approach correspondingly reduces the controller performance. To maintain the maximum performance possible, this dissertation seeks to investigate the computational expense of nonlinear MPC controllers, provide insight on various numeric approximation strategies, trajectory optimization structures, and hardware interfacing, and showcase a net optimal formulation that is compatible with real-time performance.

Based on these two challenges, the research objectives of this dissertation are as follows.

- Design a nonlinear one-level MPC controller to perform the intended CIS maneuver in a provably safe manner, even at the limits of handling. To achieve this, the controller requires novel hard safety constraint formulations specific to the intended passenger vehicle in a highway setting. The controller formulation must be compatible with a drivable tube concept to capture both curved lane boundary information and scattered obstacle data, provided by an expected perception stack. Additionally, a unique safety metric must be derived to define the fitness of such a CIS maneuver.

- Using the controller, investigate the window of opportunity, representing obstacle distances where the vehicle cannot avoid a collision by braking alone, but can safely perform an evasive lane change. Showcase the feasibility of performing such a CIS maneuver in a curved environment for both inside and outside lane changes, as well as a straight road environment.

- Introduce an adaptive controller formulation to improve the controller's robustness to uncertainty in the coefficient of friction. Analyze the baseline controller performance to benchmark the closed-loop stability to discrepancies in plant-prediction model coefficient of friction, and by extension, benchmark the adaptive formulation to showcase the improved controller performance to expected uncertainties.

- Propose a framework for the design of real-time compatible nonlinear MPC controllers, focusing on the computational cost of solving the underlying nonlinear numerical optimization problem. The framework includes addressing the cost of numerically simulating the prediction trajectory and sensitivities, trade-offs in the trajectory optimization architecture, and provides a mapping of the controller software design to parallel hardware implementation. The resulting design framework strives for real-time computing performance without reducing controller fidelity by the fitness and feasibility metrics.

1.5 Dissertation Organization

The remainder of the dissertation is organized as follows.

Chapter 2 focuses on the optimal control problem formulation to represent the CIS maneuver. Sec. 2.1 presents the plant model as well as highway environment. Sec. 2.2 presents the prediction model used in MPC, as well as derives constraints for the CIS maneuver. Sec. 2.4 presents the numeric optimization problem, and provides insight on how it is solved. Sec. 2.5 presents various CIS maneuvers in curved roads. A special straight road scenario and controller formulation is presented in Sec. 2.6. Sec. 2.7 presents a preliminary analysis to real world effects, as well as showcases an adaptive formulation.

Chapter 3 focuses on the computational cost of solving nonlinear MPC problems, using the CIS maneuver as the prime example. Sec. 3.1 presents various methods of numerically simulating the differential equations governing the prediction model, and recommends an approach on establishing integration resolution. Sec. 3.2 presents analytic derivatives of explicit integration techniques to improve the convergence rate of numeric optimization. Sec. 3.3 presents various methods of trajectory optimization, showcasing trade-offs of different approaches. Sec. 3.4 presents considerations necessary when mapping to parallel hardware, and showcases a GPU accelerated implementation of the CIS maneuver.

Chapter 4 concludes this dissertation. Sec. 4.1 summarizes the contributions made in this dissertation to the field of CIS controller formulation and computational cost of nonlinear MPC controllers. Sec. 4.2 lists existing and under review conference publications, journal publications, and US patent publications. Sec. 4.3 makes recommendations on what aspects of the CIS formulation and computational cost can be addressed to further improve the controller formulation.

Appendix A discusses what can be made publicly available on preliminary live vehicle testing. This includes a high level description of the vehicle interfacing and test conditions, as well as findings from low speed testing.

Chapter 2

CIS Controller Formulation

Similar to existing state-of-the-art obstacle avoidance and safety critical controllers [35], [34], [63], [37], [42], [11], a one-level architecture is desirable for a CIS controller. And like for the other controllers, fitness of such a maneuver is not defined by a path following metric, and such a reference trajectory is not available. While a potential drawback to one-level architectures, the fitness and feasibility metrics used to evaluate a candidate CIS maneuver must be uniquely derived.

For a candidate CIS maneuver to be considered safe, it must obey three key constraints. First, the host vehicle must leave the starting lane when passing the obstacle. Second, the host vehicle must remain within the outer lane boundary through the maneuver. Third, the host vehicle must remain stable throughout the maneuver and settle in the next lane at the end of the prediction horizon. Within this chapter, these three high level descriptors of a safe CIS maneuver are mapped into fitness and feasability criteria to generate the CIS controller. An example of a CIS maneuver can be seen in Fig. 2.1, shown in a 2 lane straight road setting including the safe braking distance and safe steering distance.

The remaining sections of this chapter are organized as follows. Sec. 2.1 introduces a geometrically simulated highway setting and the numeric plant model used in simulations. Sec. 2.2 introduces the prediction model and maps the CIS criteria into

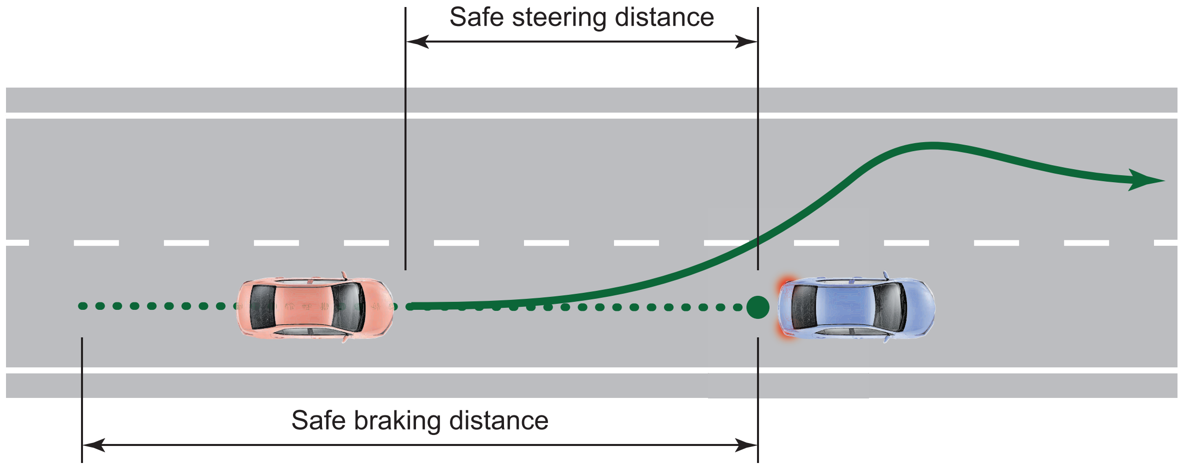


Figure 2.1: A topographical view of a successful lane change. The host vehicle starts at the left side of the image and travels to the right. The host vehicle clears the starting lane boundary before passing the obstacle, stays within the second lane outer boundary, and settles in the next lane at the end of the maneuver.

fitness and feasibility metrics. Sec. 2.4 formulates the optimal control problem. Sec. 2.5 showcases the controller in a few different CIS scenarios. Sec. 2.6 introduces a different CIS controller formulation for straight roads and contains numeric simulations. Sec. 2.7 highlights robustness testing and introduces an adaptable formulation.

2.1 Candidate CIS Environment and Plant Model

2.1.1 Geometrically Generated Highway Environment

A CIS maneuver is intended to take place if the vehicle detects it cannot avoid collision by braking alone; hence a highway environment is the most likely scenario due to the high speed operating condition. However, the scale of a highway setting makes a CIS maneuver challenging due to the relatively narrow lane widths. To design the CIS controller, a geometrically generated highway environment is generated, taking into account typical highway spacing and operating conditions.

The maximum highway speed in the United States of America varies state by state, reaching 85 MPH in Texas, though most states peak around 70 MPH [64]. Herein, the simulated highway considers a host vehicle traveling at $u_0 = 35$ m/s, representing about 78 MPH or 126 KPH.

Allowable road curvature is based on a combination of design speed, expected traffic load, and sight distance, among other factors. For a 78 MPH speed limit, the tightest turn radius allowable per US building regulations is $1,500$ m [65]. However, a curvature of $r_{\text{turn}} = 500$ m is considered to emphasize the effects of road curvature and to generate a more extreme maneuver.

The candidate section has three lanes with the center of the middle lane following a constant right hand curve of r_{turn}. The lane width is set at the minimum allowable width of $w_{\text{lane}} = 3.7$ m [66], again to generate a challenging scenario. The host vehicle starts in the center lane with a steady state trajectory to follow the road curvature.

To motivate a CIS maneuver, there is a stopped obstacle identified in the center lane at some distance d_{obstacle} ahead. A separate logic system would identify if the vehicle is capable of avoiding a collision by braking alone, seeding the high level decision of initiating a CIS maneuver or not. Due to the high speeds and sharp road curvature, the limit braking distance can be high. For example, a $0.8g$ deceleration for the described scenario requires approximately 79 m of travel. However, this is the most aggressive braking possible, and human drivers prefer to decelerate significantly slower [65]; braking at $0.4g$ requires 169 m.

For the numeric simulations, an obstacle distance of $d_{\text{obstacle}} = 47$ m is chosen as this obstacle distance is too short to avoid collision by braking alone, but, as will be shown, is sufficiently large to perform an evasive lane change maneuver. Because the host and obstacle are both in the center lane, there are two types of CIS maneuvers that can take place: an outside CIS representing a lane change to the left, and an inside CIS representing lane change to the right.

Consider first an outside CIS into the left lane. From CIS criterion (1), the maneuver is considered safe if the vehicle resides in the center or left lane, and from criterion (2), the maneuver is considered safe if the vehicle remains exclusively in the left lane. This information is well captured in the *drivable tube* [17], [67] or *safe corridor* [12] concepts.

The drivable tube, referred here forward, represents the topographical area the host vehicle can reside within without causing collision or leaving the lane boundaries. CIS criteria (1) and (2) can be directly embedded in the drivable tube by adjusting the edges of the drivable tube.

As discussed Appendix A, the drivable tube is readily available from the host vehicle's perception stack pipeline. The drivable tube can be generated onboard, leveraging sensors such as vision based cameras, LIDAR, radar, and high definition reference maps, employing techniques such as road segmentation and localization [7]. The provided drivable tube is expected to capture both the lane boundary information as well as obstacle information. By designing the CIS controller to stay within the drivable tube, this directly embeds the first two CIS criteria and is compatible with realistic vehicle expectations. Additionally, by directly using the drivable tube concept in the controller formulation, the limitations of generating deviation limits through a reference trajectory as discussed in Chapter 1 are avoided.

A key characteristic of the drivable tube is the edges of the tube do not intersect, and the tube does not branch into multiple tubes. Thus, for the inside and outside CIS maneuvers, there are two unique drivable tubes. The drivable tube is generated geometrically for the numerical simulations herein.

The high level safety criteria of metrics (1) and (2) are defined with respect to the topographic position of the vehicle in the environment. That is, these safety criteria are not concerned with the vehicle rotation, steering angles, side slip, etc. when traversing the environment. Due to the restrictive nature of the lane sizing, it is desirable to set the drivable space as large as possible. For an outside CIS maneuver into the left lane, the allowable space is the entire starting center lane and left lane prior to passing the obstacle, then exclusively the left lane thereafter.

This space represents the allowable region the host vehicle can reside within, but is not directly compatible with the CIS controller due how the prediction model is handled. As discussed in Sec. 2.2, the prediction model used by the controller is a 3 degree of freedom (3DoF) bicycle model which only considers the vehicle's center of gravity (CG). To generate the drivable tube in a format useful for the controller, the safe space is reduced by the vehicle half width, $\frac{w_{\text{vehicle}}}{2}$.

An additional restriction is applied to address corner clipping due to vehicle rotation. Because the drivable tube concept is not concerned with the vehicle rotation, it is theoretically possible for the CG to be within the buffer of $\frac{w_{\text{vehicle}}}{2}$ yet one of the corners to clip beyond the lane boundary.

There are different approaches to handling vehicle corner clipping, such as overlapping circles [42], or expressly monitoring each corner [20]. However, testing of the algorithm by implementing a safety buffer $\sigma = 0.5$ m sufficiently restricts the edge boundary to avoid corner clipping. While it is desirable to grow the drivable space as large as possible, more complex methods increase the complexity of the underlying optimization problem.

For an outside lane change, the right drivable tube edge follows a radius of $r_{\text{turn}} - \frac{w_{\text{lane}}}{2} + \frac{w_{\text{vehicle}}}{2} + \sigma$ prior to crossing the obstacle, then follows $r_{\text{turn}} + \frac{w_{\text{lane}}}{2} + \frac{w_{\text{vehicle}}}{2} + \sigma$ thereafter. The left edge follows $r_{\text{turn}} - \frac{3w_{\text{lane}}}{2} - \frac{w_{\text{vehicle}}}{2} - \sigma$ throughout.

For an inside lane change, the left edge follows a radius of $r_{\text{turn}} + \frac{w_{\text{lane}}}{2} - \frac{w_{\text{vehicle}}}{2} - \sigma$ before the obstacle, and $r_{\text{turn}} - \frac{w_{\text{lane}}}{2} - \frac{w_{\text{vehicle}}}{2} - \sigma$ thereafter. The right edge follows $r_{\text{turn}} - \frac{3w_{\text{lane}}}{2} - \frac{w_{\text{vehicle}}}{2} - \sigma$ throughout.

The drivable tube is stored as a sequence of matched point pairs, where each pair consists for the $[x, y]$ position of the left and right tube edge. Consecutive pairs in the sequence march along the road to form the drivable tube. The four points contained in sequential pairs form a parallelogram, where the after edge of on parallelogram forms the fore edge of the next. The controller expects the drivable tube to be stored in this sequence of connected parallelograms and will solve the optimization problem to keep the predicted trajectory within the edge.

The lengths of the parallelograms are flexible depending on the external system used to establish them. Parallelograms' lengths do not need to be consistent, but parallelograms of zero length should be avoided. Shorter parallelogram lengths allow for a more accurate road description, but have a higher memory requirement for MPC

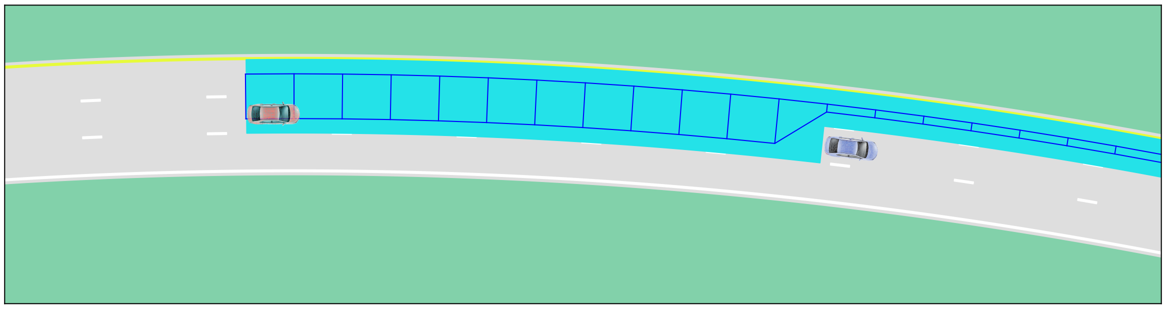


Figure 2.2: The host vehicle, red, encounters the stationary vehicle, blue. For the outside lane change CIS, the safe area for the plant model is shown in teal, and drivable tube for the 3DoF prediction model is shown in the blue parallelograms. The vehicle is allowed to use the entire center and left lane prior to passing the obstacle, then restricted to left lane after passing.

solving. The scenario herein uses matched pairs spaced approximately 5 m down the road. Immediately before the obstacle, the parallelogram is smoothed by blending the consecutive parallelograms, preventing a zero length parallelogram that can lead to numerical difficulty. An example drivable tube constructed from the parallelogram, as well as the safe allowable space for the host vehicle, can be seen in Fig. 2.2.

2.1.2 Host Vehicle Plant Model

The host vehicle is modeled as a luxury sedan, as this class of vehicles is most likely to feature the latest technologies. One such feature is active rear steering, which allows the rear wheels to be steered by an onboard computer independent of driver input and front wheel manipulation. The MPC controller is designed to leverage active rear wheel steering, if available, as leveraging rear wheel steering has been shown to improve vehicle performance over conventional front only steering [68].

The host vehicle is numerically simulated as a 14 degree of freedom (14DoF) dynamics model [69]. The model is validated against CarSim's F-class sedan [70], which is comparable to a BMW 7 series, Audi A8, or Mercedes S class. Ref. [69] contain the detailed equations of motion, with Table 2.1 listing parameters used. While vehicle manufacturers often publish some parameters, such as vehicle mass and weight dis-

Table 2.1: 14DoF Plant Parameters

Parameter	Symbol	Value
Vehicle sprung mass	m_{sprung}	1820 kg
Per wheel unsprung mass	m_{unsprung}	50 kg
Weight distribution	-	51.4/48.6 F/R
Wheel base	-	3.2 m
Vehicle width	w_v	1.9 m
Vehicle track width	l_c	1.6 m
Roll moment of inertia	$i_x x$	1023.8 kg m^2
Pitch moment of inertia	$i_y y$	3567.2 kg m^2
Yaw moment of inertia	$i_z z$	4095.0 kg m^2
Strut height	z_{strut}	0.590 m
Strut stiffness	k_{strut}	83000 N/m
Strut damping	d_{strut}	1896.1 N/$\frac{m}{s}$
Wheel stiffness	k_{wheel}	278000 N/m
Wheel radius	r_{wheel}	0.353 m

The host vehicle is modeled after a 2018 BMW 740*i*. While some parameters are published by the manufacturer, others can be estimated based on normalized ratios for sedans.

tribution, other parameters are not available and are estimated by normalized trends [71].

The plant model uses nonlinear Pacejka tire forces [72], given in (2.1), with parameters given in Table 2.2. (2.1) uses the lateral slip ratio $\frac{V_x}{V_y}$, which must be calculated at each of the tire contact points in each tire's frame. Tire manufacturers seldom, if ever, provided tire property coefficients or performance metrics. Instead, these tire parameters are calculated to provide a peak tire force of $0.8g$ at approximately 12° slip and 10% force relaxation at high slip angles.

$$\begin{aligned}
F_y &= \mu F_z \sigma_y \\
\sigma_y &= -\sin(C\ \arctan(B\frac{V_y}{V_x})) \\
V_x &= u\ \cos(\delta) + (v + \omega l)\sin(\delta) \\
V_y &= -u\ \sin(\delta) + (v + \omega l)\cos(\delta)
\end{aligned} \tag{2.1}$$

Table 2.2: Pacejka Tire Properties

Tire Parameter	**Symbol**	**Value**
Coefficient of Friction	μ	0.8
Tire Property	B	13
Tire Property	C	1.285
Tire Longitudinal Velocity	V_x	
Tire Lateral Velocity	V_y	
Vertical Tire Force	F_z	

The vertical tire force, F_z, is calculated based on the instantaneous strut deflection, relating to the spring force, and the strut vertical velocity, relating to the damper force. The higher fidelity plant model includes suspension dynamics, thus the vertical tire force requires a dynamic analysis as opposed to a static equilibrium analysis.

For numerical simulation purposes, the starting steady state trajectory is established through a separate simulation. In the example of a curved road environment with a given road curvature and design velocity, the remaining states in the 14DoF model can be established. In the case of for front and rear steering capable vehicles, solving for the steady state vector results in a continuum of solutions in the front and rear steering angle. As such, the steady state is fixed at $\delta_r = 0$, as this solution also satisfies front only steering architectures.

Other vehicle models can be used as well. Options such as directly interfacing into CarSim, ADAMS vehicle simulation, higher or lower degree of freedom numeric models, or models trained on empirical data are all viable options. From Sec. 2.7, the difference in plant and prediction model fidelity and parameterization results in some amount of model discrepancy, which can be addressed in controller formulation.

With the drivable tube and plant model established, the controller formulation can be introduced.

2.2 Prediction Model

MPC controllers leverage some model of a plant to predict the future response for a candidate control trajectory. While the prediction model used by the controller might be a full order model of the plant, this can be computationally prohibitive. Instead, prediction models used in MPC are often reduced order models that can still make sufficiently accurate future state predictions in a computationally efficient manner.

The 3DoF bicycle model has been shown to sufficiently accurate in high speed obstacle avoidance applications [73], provided the obstacle is not excessively wide [3]. This is because the bicycle model does not inherently capture vehicle roll; wide obstacles require prolonged turning, which, over time, can excite large vehicle roll in the plant, which can lead to excessive deviation between the plant response and the prediction model.

The prediction model used for the intended CIS controller is a single track 3DoF bicycle model with active independent front and rear wheels. The 3DoF bicycle model states, control inputs, and equations of motion are given as follows.

$$\mathbf{x} = \begin{bmatrix} \text{global x position [m]} \\ \text{global y position [m]} \\ \text{vehicle yaw [rad]} \\ \text{longitudinal velocity [m/s]} \\ \text{lateral velocity [m/s]} \\ \text{yaw rate [rad/s]} \\ \text{front steering angle [rad]} \\ \text{rear steering angle [rad]} \end{bmatrix} = \begin{bmatrix} x \\ y \\ \psi \\ u \\ v \\ \omega \\ \delta_f \\ \delta_r \end{bmatrix} \tag{2.2}$$

$$\mathbf{u} = \begin{bmatrix} \text{front steering rate [rad/s]} \\ \text{rear steering rate [rad/s]} \end{bmatrix} = \begin{bmatrix} \dot{\delta}_f \\ \dot{\delta}_r \end{bmatrix} \tag{2.3}$$

$$\frac{d\mathbf{x}}{dt} = \begin{bmatrix} u\ \cos(\psi) - v\ \sin(\psi) \\ u\ \sin(\psi) + v\ \cos(\psi) \\ \omega \\ 0 \\ -u\ \omega + \frac{F_{y,f}\cos(\delta_f) + F_{y,r}\cos(\delta_r)}{m} \\ \frac{F_{y,f}\cos(\delta_f) l_f - F_{y,r}\cos(\delta_r) l_r}{I_{zz}} \\ \dot{\delta}_f \\ \dot{\delta}_r \end{bmatrix} \tag{2.4}$$

In (2.4), the time derivative of longitudinal velocity is set to zero. This is because both the prediction and plant models lock the longitudinal velocity. Per US highway testing standards, the double lane change maneuver, also known as "moose avoidance maneuver", does not allow acceleration or braking [74]. In numerical simulations, longitudinal acceleration due to yawing under non-zero lateral velocity has been negligible. The longitudinal velocity is retained in the system dynamics as a reference in computing the tire slip. However, modern optimizers often reduce this locked state from the control problem.

The prediction model also uses the same Pacejka tire forces as the plant model, defined by (2.1) and Table 2.2. In initial simulations, the prediction tire model uses the same coefficients as the plant, but model discrepancies in tire parameters are investigated in Sec. 2.7. For the single track bicycle model, the tire forces are calculated as a lump front wheel force at the front axle location and a lump rear force at the rear axle location. The parameters used in the prediction mode are listed in Table 2.3.

In the next section, the fitness and feasibility metrics of the CIS criteria are mapped into numerical constraints for use in the nonlinear optimization representing the controller.

Table 2.3: 3DoF Model Parameters

Parameter	**Symbol**	**Value**
Vehicle mass	m	2020 kg
Weight distribution	-	51.4/48.6 F/R
Wheel base	-	3.2 m
Vehicle width	w_v	1.9 m
Front wheel to CG distance	l_f	1.56 m
Rear wheel to CG distance	l_r	1.64 m
Front wheel vertical force	$F_{z,f}$	1038
Rear wheel vertical force	$F_{z,r}$	982
Yaw moment of inertia	I_{zz}	4095.0 $\text{kg} \cdot \text{m}^2$

The upper table contains values published by the manufacturer. The lower table contains values derived as needed for 3DoF prediction model.

2.3 Curved Road CIS Fitness and Feasibility Formulation

To design a controller to perform a CIS maneuver, the three CIS criteria must be translated into numeric constraints. In the next subsections, the feasibility metrics are derived, as well as constraints on the vehicle, and fitness metric introduced.

2.3.1 Boundary Constraints

In Subsec. 2.1.1, the benefit of the drivable tube is highlighted as it incorporates both lane boundary information and obstacle information into one data structure. By adjusting the drivable tube to follow the allowable space, the drivable tube can enforce CIS criteria one and two. The CIS controller is then tasked with finding a steering sequence to keep the vehicle within the drivable tube.

Using vector algebra, the relevant parallelogram can quickly be identified, and corresponding left and right boundary constraints evaluated. Consider Fig. 2.3 showing an integration state located inside a parallelogram.

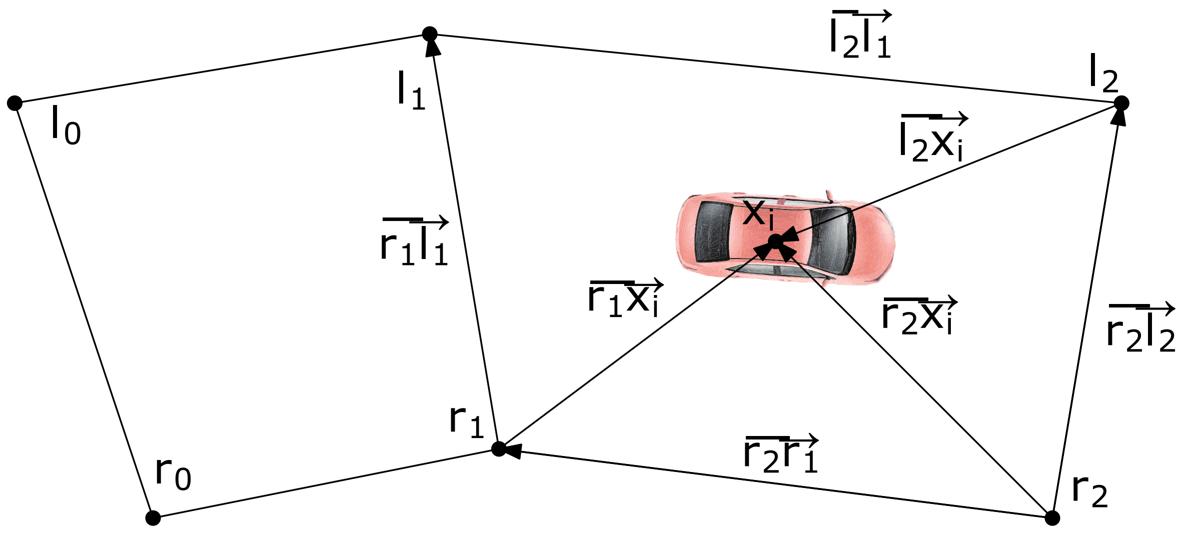

$\overrightarrow{l_2 l_1}$
l_2
l_1
l_0
$\overrightarrow{l_2 x_i}$
x_i
$\overrightarrow{r_1 l_1}$
$\overrightarrow{r_1 x_i}$
$\overrightarrow{r_2 x_i}$
$\overrightarrow{r_2 l_2}$
r_1
$\overrightarrow{r_2 r_1}$
r_0
r_2

Figure 2.3: An integration state $\mathbf{x}_i$ is shown as residing inside a parallelogram. Using various vector cross products, the algorithm can identify when an integration state has left one parallelogram and traversed into the next. Once the active parallelogram is identified, the right and left boundary constraints are evaluated, again with vector logic.

As the MPC controller integrates the state forward in time, the algorithm must identify which parallelograms in the drivable tube are applicable to that state. To find the active parallelogram, the vector cross product of the vehicle position with the fore parallelogram boundary is calculated.

Consider the matched pair $(\mathbf{r}_k, \mathbf{l}_k$, representing the $(k+1)^{th}$ matched pair in the drivable tube sequence and the for parallelogram boundary of the k^{th} parallelogram. For the i^{th} integration state in the prediction horizon, the vehicle's position vector relative to the right boundary in the matched pair is given by $\overrightarrow{\mathbf{r}_k\mathbf{x}_i}$. To find the active k^{th} parallelogram, k is increased until the fore boundary cross product with the position vector crosses zero, which can be calculated numerically by $(\overrightarrow{\mathbf{r}_{k-1}\mathbf{x}_i} \times \overrightarrow{\mathbf{r}_{k-1}\mathbf{l}_{k-1}})(\overrightarrow{\mathbf{r}_k\mathbf{x}_i} \times \overrightarrow{\mathbf{r}_k\mathbf{l}_k}) < 0$. This fast vector algebra allows efficient scaling to the GPU hardware used to simulate the vehicle and evaluate the fitness and feasibility criteria, to be discussed in Chapter 3.

With the active parallelogram k identified, the left and right lane boundaries are evaluated. The left lane boundary constraint is evaluated by the cross product of the position vector with left edge vector, $\overrightarrow{\mathbf{x}_i\mathbf{l}_k} \times \overrightarrow{\mathbf{l}_k\mathbf{l}_{k-1}}$. This constraint is formed such that if the cross product is less than zero, the vehicle integration state is to the right of the left lane boundary, which is valid. It is also important to note the validity of the constraint is based on the sign. This allows the valid limits of the constraint, being in reference to 0, to be established at the start of the MPC problem and avoid having to first seed a trajectory.

Likewise, the right lane boundary is generated by the cross product of the right edge vector with the position vector, $\overrightarrow{\mathbf{r}_k\mathbf{r}_{k-1}} \times \overrightarrow{\mathbf{x}_i\mathbf{r}_k}$. Here, the order is swapped to maintain a cross product of less than or equal to zero to be valid in convention with constraint formulation. Together, the left and right lane boundary constraints form (2.5) and (2.6).

$$d_l = \overrightarrow{\mathbf{l}_k\mathbf{x}_i} \times \overrightarrow{\mathbf{l}_k\mathbf{l}_{k-1}} \leq 0 \tag{2.5}$$

$$d_r = \overrightarrow{\mathbf{r}_k\mathbf{r}_{k-1}} \times \overrightarrow{\mathbf{r}_k\mathbf{x}_i} \leq 0 \tag{2.6}$$

Note, the constraint is formed as at least C^2 smooth. This helps with solving the numerical optimization problem using gradient based optimization, which is necessary for real-time performance.

Further, the left and right constraints are evaluated based on the integration state's instantaneous position in the environment, in contrast to other methods that use an initial seeding through a reference trajectory for lateral displacement limits. This ensures the left and right constraints are relevant at the converged optimal trajectory, as well as all the intermediate trajectories, not just the first candidate seeding trajectory.

2.3.2 Vehicle Stability: Tire Slip

The third CIS criterion addresses vehicle stability by requiring the host vehicle remains stable throughout the maneuver and settle in the next lane at the end of the prediction horizon. The first description, remaining stable, is enforced by ensuring the vehicle

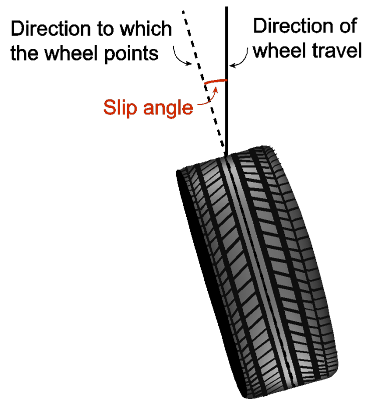

Direction to which the wheel points
Direction of wheel travel
Slip angle

Figure 2.4: The tire slip angle is the difference in the tire velocity vector and alignment vector. It is through this discrepancy that tire forces are generated.

retains controllability throughout. Various methods and techniques on vehicle stability have been studied such as the Milliken Moment Method [75], tire lift off [76], and vehicle envelope control [17], [67]. The presented controller ensures vehicle stability by establishing bounds on the tire slip angle and restricting the vehicle to within these stable limits.

Tire slip, denoted $s = \frac{v_y}{v_x}$, is the ratio of lateral velocity to longitudinal velocity. This can also be expressed as the tire slip angle; representing the angle between the tire velocity vector and the direction the tire is rolling. It is through some amount of tire slip that the tire rubber compound deforms as the contact patch, resulting in a tire force between the road surface and the vehicle. The tire slip angle can be seen in Fig. 2.4.

Depending on the tire properties used in (2.1), the tire lateral force may relax beyond the peak slip angle. This is an unsafe driving condition because the tire forces are completely or nearly insensitive to steering angle, thus the controller has little to no control authority over the vehicle response, thus control authority can be quickly lost and mid-maneuver adjustments cannot reliably be made when operating in this regime [36]. Additionally, tire relaxation at high slip causes gradient inversion within the optimization problem, where the gradient information indicates the tire force is actually reduced by increasing tire slip. This causes local minima and is challenging

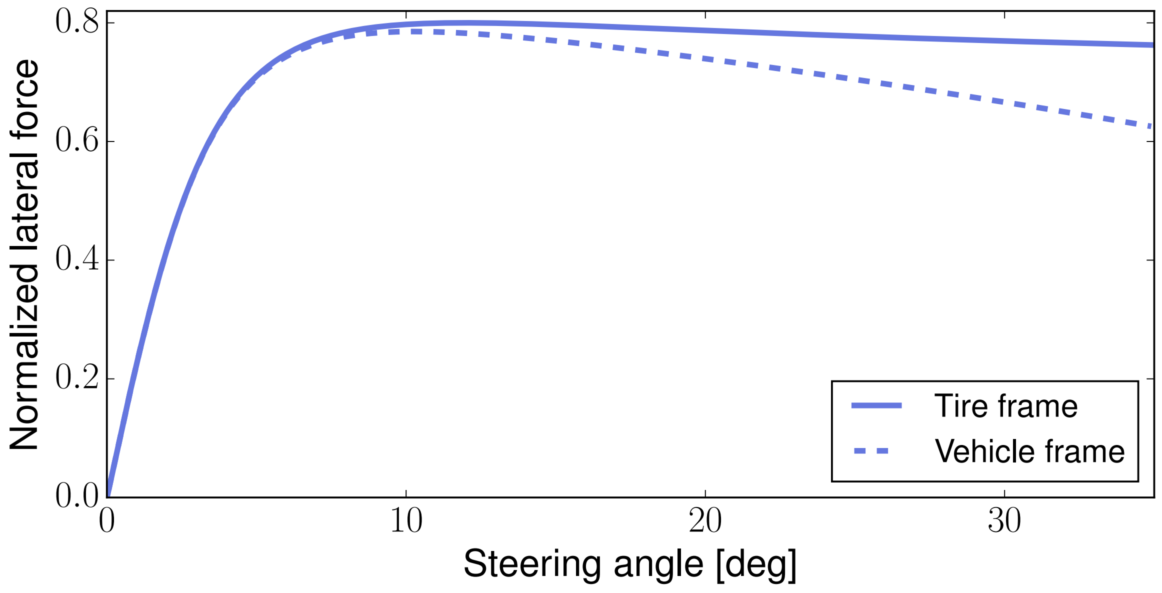


Figure 2.5: The lateral force in the tire frame and vehicle frame is plotted in solid blue and dashed blue, respectively. In this scenario, the vehicle is traveling straight ahead, thus the steering angle matches exactly the slip angle. Due to the coordinate transformation, at high steering angles the vehicle lateral force decays faster than the tire lateral force.

for the optimization problem to converge

For the parameters chosen in Table 2.2, the peak tire force occurs around 12° slip with 10% relaxation above that. However, due to the coordinate transform between the tire frame and vehicle frame, the maximum lateral force is not necessarily at 12°. When the vehicle is traveling purely longitudinally, the peak tire force occurs around 10° tire slip. Consider Fig. 2.5 plotting the lateral force in the tire frame and in the vehicle frame when the vehicle is traveling longitudinally. If the vehicle has zero lateral velocity and yaw rate, then the steering angle and slip angle match exactly.

The lateral force mismatch between the vehicle frame and tire frame is exacerbated when lateral velocity is introduced. Lateral velocity causes an initial tire slip angle that can be either constructive or destructive with the steering angle, as seen in Fig. 2.6, plotting the lateral forces at different lateral velocities.

In Fig. 2.6, the difference in steering angle for peak lateral force in the vehicle frame

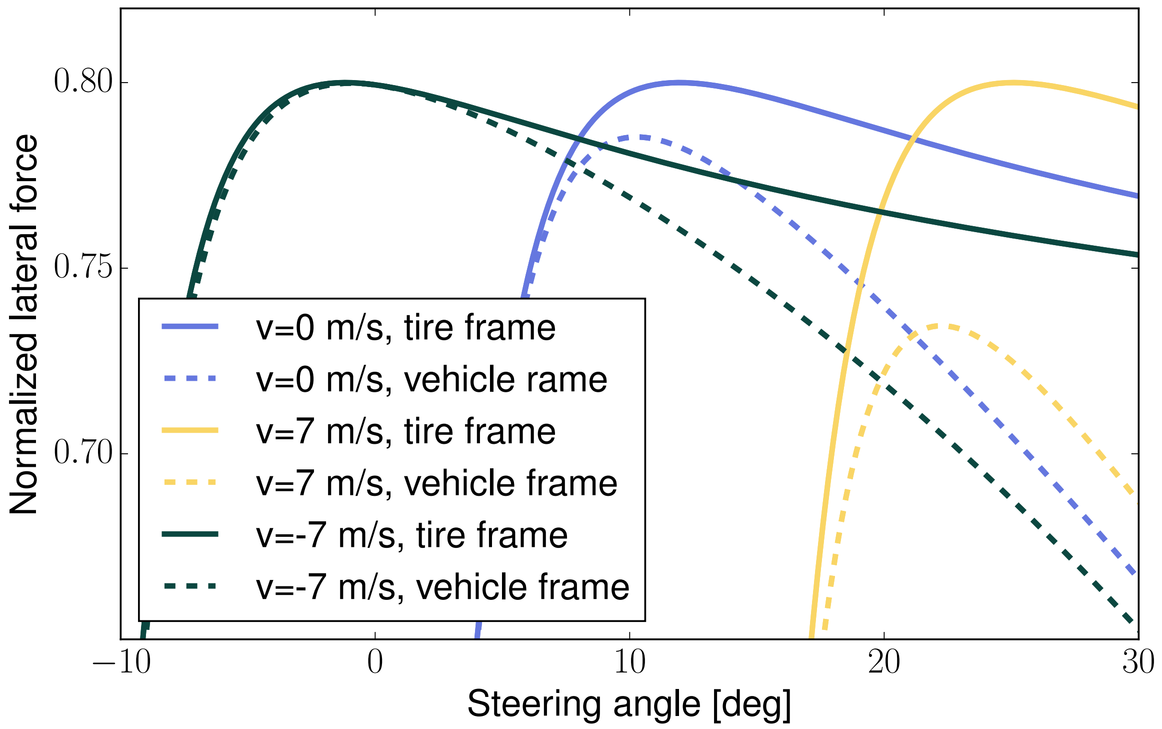


Figure 2.6: The same response of Fig. 2.5 are plotted, but for variations in lateral velocity. For this example, angular velocity is zero, but angular velocity introduces tire slip to the same effect lateral velocity does.

versus tire frame is not consistent for different lateral velocities, nor is the difference in magnitude of the peak lateral force in the vehicle frame versus tire frame consistent. Further, the neutral steering angle for zero lateral force is not at a fixed steering angle. Thus, applying constraints to the steering angle does not ensure vehicle stability.

Fundamentally, the stable steering region is where $\frac{d|F_{y,\text{veh. frame}}|}{d\delta} > 0$. However, the stability constraint in this form, and subsequently calculating sensitivities, is computationally expensive. As a conservative alternative, limiting the peak tire slip to a set α_{peak} has restricted the slip angle to the stable region in practice. α_{peak} can be established *a priori* though offline simulations of the expected or experience operating states. For the CIS maneuver and chosen tire coefficients, $\alpha_{\text{peak}} = 8^{\circ}$ allows the controller to operate within 98% of the peak available tire force. Subsequently, a trajectory is considered stable if the tire slip never exceeds α_{peak}.

This constraint is formulated on the front and rear wheels as follows and is enforced at every integration state in the prediction trajectory.

$$\alpha_f = |\delta_f - arctan(\frac{v + \omega l_f}{u})| \leq \alpha_{\text{peak}} \tag{2.7}$$

$$\alpha_r = |\delta_r - arctan(\frac{v - \omega l_r}{u})| \leq \alpha_{\text{peak}} \tag{2.8}$$

For the tires used in the bicycle model, the front and rear tires have the same properties. This is not always the case, but the controller can enforce different $\alpha_{\text{peak, front}}$ and $\alpha_{\text{peak, rear}}$ constraints for the front and rear tires. Further, by forming a separate front and rear stability constraint, the numeric optimizer has better control of addressing the feasibility criteria as the gradient information pertaining to the front slip and rear slip is independent.

The tire slip only forms half of the vehicle stability consideration. Next, the terminal stability is developed.

2.3.3 Terminal State Constraint

The latter component of the third CIS criteria is the vehicle must settle in the desired lane at the end of the prediction horizon. This ensures the MPC controller sees the entire maneuver though prior to taking action, thus preventing starting a CIS maneuver that later becomes infeasible in closed loop as the prediction horizon recedes.

The simpler forms of CIS perform a single lane change maneuver. For the curved road example, this implies the vehicle settles in the left lane for the outside lane change, or the right lane for the inside lane change. Alternatively, a more complected CIS maneuvers of double lane change is simulated, where the host vehicle settles back in the original lane.

The terminal state, $\mathbf{x}_{\text{stable}}$, is derived from road information. For the geometrically generated sample case, the terminal state would be a right hand turn following a radius of $r_{\text{lane}} = r_{\text{turn}} + w_{\text{lane}}$ for the outside lane change, and $r_{\text{lane}} = r_{\text{turn}} - w_{\text{lane}}$ for the inside lane change. Similar to the initial state, the terminal stable state for the new radius is calculated by evaluating the steady state conditions from (2.4).

One challenging element of terminal state constraint is establishing *a priori* the location of final state in the prediction horizon. While it is easy to describe the desired position as centered in the next lane, it is difficult to state how far down the lane the final state is, and correspondingly, how much the vehicle needs to be rotated relative to its starting position. For this implementation, the terminal (x_t, y_t) position is set based on radius from the curve center (x_c, y_x), not explicitly road position. The terminal vehicle rotation angle is set based on (x_t, y_t) position, and calculated such that the vehicle velocity is tangent to the road center.

In practice, implementing the terminal position would require some form of estimate for road curvature at the end of the maneuver, which supports a look-up table or alternative estimate for the stable states. Alternatively, both in practice and in numerical simulation the drivable space boundaries can be artificially contracted to mandate the vehicle is in the center of the lane far in the prediction horizon.

It is not essential that the terminal position constraints are as accurate as possible. The terminal position constraints are enforced far in the prediction horizon where the vehicle has the most control authority and ability to correct in closed-loop implementation.

The terminal position (x_t, y_t) and rotation ψ_t are structured as follows.

$$(x_t - x_c)^2 + (y_t - y_c)^2 = r_{\text{lane}}^2 \tag{2.9}$$

$$arctan(\frac{y_t - y_c}{x_t - x_c}) = \frac{\pi}{2} - arctan(\frac{v_t}{y_0}) \tag{2.10}$$

The complete terminal state is structured as follows.

$$\mathbf{x}_{\text{stable}} = \begin{bmatrix} x_t & y_t & \psi_t & u_0 & v_t & \omega_t & \delta_{f,0} & 0 \end{bmatrix}^T \tag{2.11}$$

The inclusion of the terminal state constraint addresses recursive feasibility considerations. If the initial maneuver obeys the terminal state constraint, the next iteration in the receding horizon of closed-loop MPC obeys the terminal state constraint as well, with the final control input as the zero vector. While this is valid in the context of the prediction model, it does not guarantee recursive feasibility subject to plant-prediction model mismatch.

2.3.4 Vehicle Physical Limits

The last set of constraints on the MPC problem is to capture the physical limits of the vehicle. Specifically, the front and rear wheels are both steering angle and steering rate limited.

For this vehicle, the front wheels have a simulated maximum steering angle limit of $\pm 35°$ and a maximum rate limit of ± 70 deg/s. For rear wheel steering capable vehicles, the rear wheels typically do not have as strong control authority, and are comparatively reduced. For this vehicle, a simulated limitation on the rear wheel angle is set at $\pm 10°$

and rate limited to ± 35 deg/s. To model a front-only steering vehicle, the rear wheels' angle limit can be set at $0°$.

Like the other constraints, the steering rate limits are enforced at every integration point in the prediction horizon. Because the optimization problem is structured such that the design variables are the steering rates, the physical rate limitations are enforced as bounds on the design variables.

Additionally, the steering angle is a linear function of the design variables, and for certain numerical optimizers, it might be handled differently. While it does not change the structure of the numerical optimization problem presented here, this subtle difference should be considered for simulation timing purposes.

Hitherto, all the constraints for the optimization problem are built. Any candidate trajectory that obeys all these constraints is considered feasible. Next, the objective function is formulated to determine the optimal trajectory.

2.3.5 Objective Function: Minimum Slip

As discussed in Sec. 1.2, many of the obstacle avoidance algorithms are designed based on path following formulations. In the context of CIS, following a predefined reference path does not necessarily correlate to the safest trajectory. Instead, it is proposed to minimize the largest tire slip. By minimizing the peak tire slip in the prediction horizon, this maximizes the additional available tire force, which maximizes the control authority available to make mid-maneuver corrections. Additionally, by minimizing the peak tire slip, and correlating to minimizing the peak tire force, the resulting CIS can be argued to be the least invasive on the occupants.

Alternatively, a minimum distance formulation is presented in Sec. 2.6. While leaving the starting lane in the shortest distance can provide an informative metric for how late to intervene, it might not be the best in practice. If the vehicle decides it will intervene, it can be beneficial to intervene right away instead of delaying.

If the vehicle detects it cannot brake in time and must perform a lane change, one option is to still perform the minimum distance lane change, thus maximizing the available gap between when leaving the lane and the obstacle. However, this would be an overly aggressive maneuver, as the minimum distance lane change will always load the vehicle tires to the maximum allowable force, to be see in Sec. 2.6.

Minimizing a maximum function can be difficult in gradient based optimization because the maximum element in the aggregate can change index discretely, resulting in discrete changes to the gradient. Thus, the Kreisselmeier-Steinhauser (KS) constraint aggregation function is used to minimize the peak tire slip in a gradient smooth fashion.

2.3.6 KS Function

The KS function was proposed as a means of creating a single performance metric to represent a group of performance indicators [77]. At a high level, the KS function represents a smooth near-infinity norm, which is beneficial in gradient-based optimization as it mathematically represents a smooth maximal-like function. The discrete KS function is introduced as follows [78], [79].

$$\mathrm{KS}(\mathbf{g}, \rho) = \frac{1}{\rho} \ln\Big(\sum_{i=1}^{n} e^{\rho g_i} \Big) \tag{2.12}$$

In (2.12), $\mathbf{g}$ represents a vector of constraints, with g_i being the i^{th} constraint value. For the CIS MPC formulation, g_i represents the constraint value at the i^{th} time integration step, $\mathbf{g}$ represents the constraint vector over the prediction horizon, and ρ represents the constraint aggregation curvature parameter.

While the KS function is introduced here as a means to minimize the peak tire slip, constraint aggregation techniques are beneficial for other constraints as well because they reduce the size of the numerical optimization problem. For example, enforcing the front tire slip constraint at n integration points in the prediction trajectory would require n constraints, which can be reduced to 1 aggregated constraint. Each constraint derived from the CIS criteria is collected into its own KS aggregation metric. This reduces m feasibility constraints over n integration states from mn constraints in the optimization problem to m KS constraints.

To leverage the constraint aggregation technique for the objective function and other constraints, consider the following properties of the KS function. Specifically, there are a few key properties that are desirable for safety constraint aggregation [80]. Researchers have shown that the KS function is a conservative aggregation, and as ρ becomes large the aggregation approaches the maximum value [78]. These are numerically represented as follows.

$$\mathrm{KS}(\mathbf{g}, \rho) \geq \max(g_i)\ \forall i \in [1, n] \tag{2.13a}$$

$$\lim_{\rho \to \infty} \mathrm{KS}(\mathbf{g}, \rho) = \max(g_i)\ \forall i \in [1, n] \tag{2.13b}$$

$$\max(g_i) \leq \mathrm{KS}(\mathbf{g}, \rho) \leq \max(g_i) + \frac{\ln(n)}{\rho} \tag{2.13c}$$

The significance of the conservative aggregation can be seen from (2.13a); if the KS aggregation obeys the constraint, than each component of $\mathbf{g}$ obeys the constraint as well. In terms of the CIS maneuver, this means if the KS aggregation of the tire slip constraint is satisfied, then each integration point in the prediction horizon will obey the tire slip constraint as well.

While the conservative nature of the KS function is important at a theoretical level, there are practical limitations to using the KS function. Because the KS function is conservative, the KS aggregation will be larger than the maximum of $\mathbf{g}$. For the drivable tube boundary constraint, if $\mathrm{KS}(\mathbf{g}, \rho) = 0$ and ρ is small, then $\max(g_i)$ can be much less than 0 by (2.13c). This can cause the controller to be overly conservative of the drivable tube edge boundary, thus not using the fully allowable space, hence not achieving good CIS performance.

Large values of ρ can cause numerical difficulties depending on the sign of the elements of $\mathbf{g}$. For example, the tire slip constraint has a positive constraint value; thus $\lim_{\rho \to \infty} e^{\rho g_i} = \infty$, which causes numerical difficulties. Using algebraic manipulation, a shifted KS aggregation formulation can be used.

$$\mathrm{KS}(\mathbf{g},\rho,\overline{g}) = \overline{g} + \frac{1}{\rho}\mathrm{ln}\Big(\sum_{i=1}^{n} e^{\rho(g_i-\overline{g})}\Big) \tag{2.14}$$

(2.14) is identical to (2.12), but is numerically robust. By choosing $\overline{g} = \max(g_i)$, the exponent is non-positive, thus avoiding overflow errors. Additionally, the shifted form of (2.14) provides insight into the maximum overestimation seen in (2.13c). If $\overline{g}$ is taken as $\max(g_i)$, the largest feasible value of $(g_i - \overline{g})$ is 0. If $g_i = \max(g_i)\ \forall i \in [1, n]$, then (2.14) is equivalent to $\mathrm{KS}(\mathbf{g}, \rho, \max(g_i)) = \max(g_i) + \frac{1}{\rho}ln(ne^0)$, which reduces to the third property.

However, evaluating $\max(g_i)$ requires storing all values of g_i, iterating through the list for the maximum value, then evaluating the KS aggregation, which can be computationally expensive. Alternatively, as the controller propagates the predicted state trajectory, the controller can store the incremental KS contribution for a fixed $\overline{g}$. Calculating the incremental KS contribution at each time step has the additional benefit that the entire state trajectory does not need to be stored, only the current state at numeric integration.

While choosing $\overline{g}$ is not obvious, choosing $\overline{g}$ as the peak allowable constraint value is a good starting point. If a solution to the numerical optimization problem exists, the starting design point is in the neighborhood of the feasible region, and values of ρ are reasonable, then the nested exponent does not encounter overflow issues and hence retains the aggregate value and gradient information. Further, by setting $\overline{g}$ to the constraint limit means $\overline{g}$ can be set prior to simulating the candidate control trajectory, avoiding having to store every instance of g_i.

Even with setting $\overline{g}$ as the allowable constraint limit there can be numerical overflow issues. Often, the initial control trajectory violates the constraints, thus causing a positive value in the exponent.

Consider a numerical analysis related to computing accuracy standards. Per IEEE 754 standard on single-precision floating-point, the exponent of a single-precision floating-point is an integer on $[-126, 127]$. For the largest exponent of 127, this represents $2^{127} \approx 10^{38}$. While 10^{38} is large, $10^{38} \approx e^{87.5}$ forms an upper bound on the stable values of the nested exponent. Thus, ρ must be scaled to avoid numerical overflow errors, and a method to set this parameter once *a priori* is presented in Sec. 2.4.

Evaluating in double-precision float-point does little to help. Per IEEE 754, the exponent of a double-precision floating point is an integer on $[-1023, 1024]$. Thus, $2^{1024} \approx 10^{308} \approx e^{710}$, which allows a larger upper bound for the nested exponent, but only allows a single order of magnitude larger ρ over single-precision.

Using the instantaneous tire slip at every integration point similar to (2.7) and (2.8), the KS aggregation is generated as follows.

$$\mathrm{KS}(\boldsymbol{\alpha}, \rho_{\text{slip}}) = \frac{1}{\rho}\ln\Big(\sum_{i=1}^{n} e^{\rho_{\text{slip}}\boldsymbol{\alpha}_i}\Big) \tag{2.15}$$

Here, $\boldsymbol{\alpha}$ is a vector containing the front and rear instantaneous tire slip angles at every integration state in the prediction horizon.

For most of the trajectory simulation methods discussed in Chapter 3, the lane boundary and tire slip constraints are enforced through the KS function. For each KS function, a ρ constraint aggregation parameter must be defined *a priori*, and a method to do so is as follows.

2.3.7 ρ Scaling

The parameter ρ should be scaled as large as possible without causing numerical stability issues. The various KS constraints used so far aggregate values of varying magnitude, thus a universal ρ for the optimization is not feasible.

From KS property (2.13c), the overshoot bounding equation can fix ρ for a set allowable percent overshoot. This introduces an allowable overshoot parameter, $f_{\text{overshoot}}$, which is the ratio of how much overestimation though the KS aggregation is allowable to the maximum allowable constraint value.

$$\begin{aligned} f_{\text{overshoot}} &:= \frac{\text{KS}(\mathbf{g}, \rho) - \max(g_i)}{\overline{g}_{\text{allowable}}} \\ \rho &= \frac{\ln(n)}{f_{\text{overshoot}}\ \overline{g}_{\text{allowable}}} \end{aligned} \tag{2.16}$$

Based on an analysis of the largest expected value of g_i and the number of discrete integration points n in the prediction horizon, $f_{\text{overshoot}} = 0.15$ aggregates the constraints while avoiding numerical overflow issues. In simulation, $\rho_{\text{boundary}} = 13$ and $\rho_{\text{front}} = \rho_{\text{rear}} = 264$ has been stable. In the context of the objective function, there is no peak allowable objective value. However, for a feasible solution to exist, the peak tire slip constraint must be obeyed, hence $\rho_{\text{obj}} = 264$ is stable.

While ρ is a tuning parameter to the CIS controller, it is established *a priori* for each constraint. Once it is set offline, it does not change during online use.

With the objective and constraints defined, the numerical optimization problem can be formulated.

2.4 Optimal Control Problem Formulation

The task of finding a control sequence to perform a CIS maneuver is posed as a numerical optimization problem. The design variables, in this case steering angle control rates [81], [82], are adjusted such that by the feasibility criteria, the vehicle response obeys the hard safety constraints. Of the set of feasible trajectories, the best trajectory is the one that minimizes the peak tire slip.

To begin, the instantaneous plant state seeds the starting state for the prediction model. In implementation, the instantaneous 3DoF state is provided by a localization stack. The current CIS controller formulation requires all 8 states in the 3DoF bicycle model in (2.2) to be defined. Some states, like $x-y$ position and steering angle, can be directly measured, but other states, such as lateral velocity, might require estimation. While all states are observable and controllable, the observation task is beyond the scope of the controller, but some insight is provided in Chapter 4.

For numerical simulations herein, the 3DoF states are passed directly from the 14DoF state vector. This bypasses sensor noise and data acquisition timing, but insight on uncertainty is provided in Sec. 2.7. Although the plant states are exactly available to the prediction model, this passing still retains deviation between the prediction model and plant model due to model fidelity.

One complication with the starting state is the controller takes a nontrivial amount of time to solve. The controller is designed for a $t_{\text{update}} = 100$ ms closed-loop update rate to the plant. As a result, the starting state the controller sees is seeded forward 100 ms by the prediction model, denoted the initial state $\mathbf{x}_0$. Additionally, this implies the earliest a control action can be implemented to make a correction is 100 ms into the future.

From the initial state $\mathbf{x}_0$, the control trajectory is numerically integrated forward with a zero order control hold. The zero order control hold represents a constant control input, in this case constant steering rate, for a set block of time and is chosen to simulate closed loop control architectures typically found on distributed automotive control systems. Details on the numerical integration method and trajectory simulation are explored further in Chapter 3. The resulting prediction trajectory is defined by n integration states, each denoted $\mathbf{x}_i$.

For this problem, an MPC time horizon of 3.2 s has been sufficiently long in simulation for the vehicle to complete the lane change maneuver and stabilize in the next lane. While a longer time horizon can be used, this increases the problem size, typically incurring a computational penalty. Identifying the appropriate time horizon for MPC is an ongoing research area in controls research [8].

The time horizon is broken into uniform constant control intervals. In the latest version of the controller, the prediction horizon is broken into 50 ms intervals defined by a constant front and rear control rate, resulting in a total of 64 segments. For the 100 ms update rate, the controller issues two control intervals to the plant every interval.

Combining the objective function and constraints described prior, the nonlinear numerical optimization problem is formulated as follows.

$$\begin{aligned}
\min_{\mathbf{u}} \quad & \mathrm{KS}(\boldsymbol{\alpha}, \rho) & & \\
\text{subject to} \quad & d_{l,i} & \leq 0 & \quad \forall\, i \in [1, n] \\
& d_{r,i} & \leq 0 & \quad \forall\, i \in [1, n] \\
& \alpha_{f,i} & \leq \alpha_{\text{peak}} & \quad \forall\, i \in [1, n] \\
& \alpha_{r,i} & \leq \alpha_{\text{peak}} & \quad \forall\, i \in [1, n] \\
& |\delta_{f,i}| & \leq \delta_{f,\max} & \quad \forall\, i \in [1, n] \\
& |\delta_{r,i}| & \leq \delta_{r,\max} & \quad \forall\, i \in [1, n] \\
& |\dot{\delta}_{f,i}| & \leq \dot{\delta}_{f,\max} & \quad \forall\, i \in [1, n] \\
& |\dot{\delta}_{r,i}| & \leq \dot{\delta}_{r,\max} & \quad \forall\, i \in [1, n] \\
& \mathbf{x}_{\text{terminal}} - \mathbf{x}_{\text{stable}} & = \mathbf{0} &
\end{aligned} \tag{2.17}$$

(2.17) does not explicitly include the equations of motion as part of the optimization problem, as the equations of motion are enforced in the mapping of the control input to prediction state trajectory, discussed further in Sec. 3.1.

(2.17) still retains the tire slips as constraints, even though the objective function is to minimize the peak tire slip. Even though the optimal solution is expected to minimize the peak tire force, retaining these stability constraints in the optimization problem improves the convergence rate as intermediate candidate solutions might be infeasible, and due to the gradient inversion can have trouble converging.

With the numerical optimization problem established, various scenarios are constructed to evaluate the performance of the CIS controller.

2.5 Numeric Simulation of Curved Roads

Solving nonlinear numerical optimization problems is an entire discipline on itself, but (2.17) is implemented in IPOPT [83] for simulation purposes. The simulation hardware is a 2017 HP Omen desktop with an Intel i7-7700k CPU and NVidia GTX 1080 discrete GPU, which, at the time, was a high performance personal desktop machine. Solutions converge in approximately 55 ms wall time, but details on achieving real-time performance are addressed in Chapter 3.

IPOPT is chosen primarily as it is an interior point solver. Interior point solver methods are desirable for this type of application because the optimization problem is tasked with finding a feasible point first, then finding the optimal point. This is important for time sensitive safety critical applications, such as CIS, because it means the optimizer attempts to find a control trajectory that first obeys the hard safety constraints, then iterates on the optimality if time allows. Additionally, IPOPT does not require the initial candidate point, nor subsequent iterations, to be strictly feasible. This ensures the optimizer does not fail during the nonlinear iterations, nor does the discrete change in the drivable tube that occurs when an obstacle enters the lane terminate the iterations. The plant vehicle and environment simulation is modeled in Python, and relevant CIS control information passed into a C implementation of IPOPT.

As discussed in Sec. 2.1, an obstacle is fixed 47 m down the road with the host vehicle traveling at 35 m/s. The prior discussed braking distance of 79 to 169 m dictates a collision cannot be avoided by braking alone, thus requires an evasive lane change. Consider first the outside CIS maneuver.

2.5.1 Outside Lane Change

Consider first an outside lane change, representing a change into the left lane for a right hand turn. Fig. 2.7 shows 4 concurrent plots highlighting different aspects of the maneuver. The first plot shows the x–y trajectory through the lane corridor, second plot shows the front and rear wheel angles, third plot shows the front and rear slip angles, and fourth plot shows the front and rear steering rate commands.

This is a comparatively easy maneuver, because the vehicle begins by relaxing the tires and opening up the radius to the turn. Although the vehicle does turn to the left initially, this is a brief maneuver immediately followed by a long sweeping right turn. The vehicle holds the turn, just leaving the starting lane when passing the obstacle, and then just coming to the edge of the outer lane limit before stabilizing for the end of the maneuver.

This single lane change to the outside reaches a peak tire slip of approximately $4.6°$ slip, representing about 86% of the available tire force. Because the objective is structured to minimize the peak tire slip, there is no penalty for holding the vehicle at that peak slip for prolonged periods of time. As a result, the optimal solution turns left to $+4.6°$, then begins the sweeping turn to the right, during which the vehicle is loaded to $-4.6°$.

There is some deviation from the peak slip later in the trajectory for two reasons. First, the controller acts in the receding horizon, and minimizes the peak slip from that point forward. Even though the vehicle reaches a higher peak slip earlier in the maneuver, the closed-loop solution will only push to the peak slip if necessary. Second, the discrepancy between the 3DoF prediction model and the 14DoF plant model becomes apparent, causing small perturbations throughout the maneuver. The 14DoF plant includes higher order dynamics, such as suspension response, which the controller cannot account for, but does correct for in the closed-loop through feedback.

Additionally, the importance of the computational speed becomes apparent from this maneuver. When the host vehicle identifies the obstacle at some distance into the future, it does not know if a safe CIS maneuver exists, thus does not immediately take action. Instead, the CIS controller begins solving the maneuver, seeding the trajectory as where it thinks the host vehicle will be 100 ms into the future. As a result, there is no control intervention over the first 100 ms, effectively bringing the obstacle 3.5 m closer.

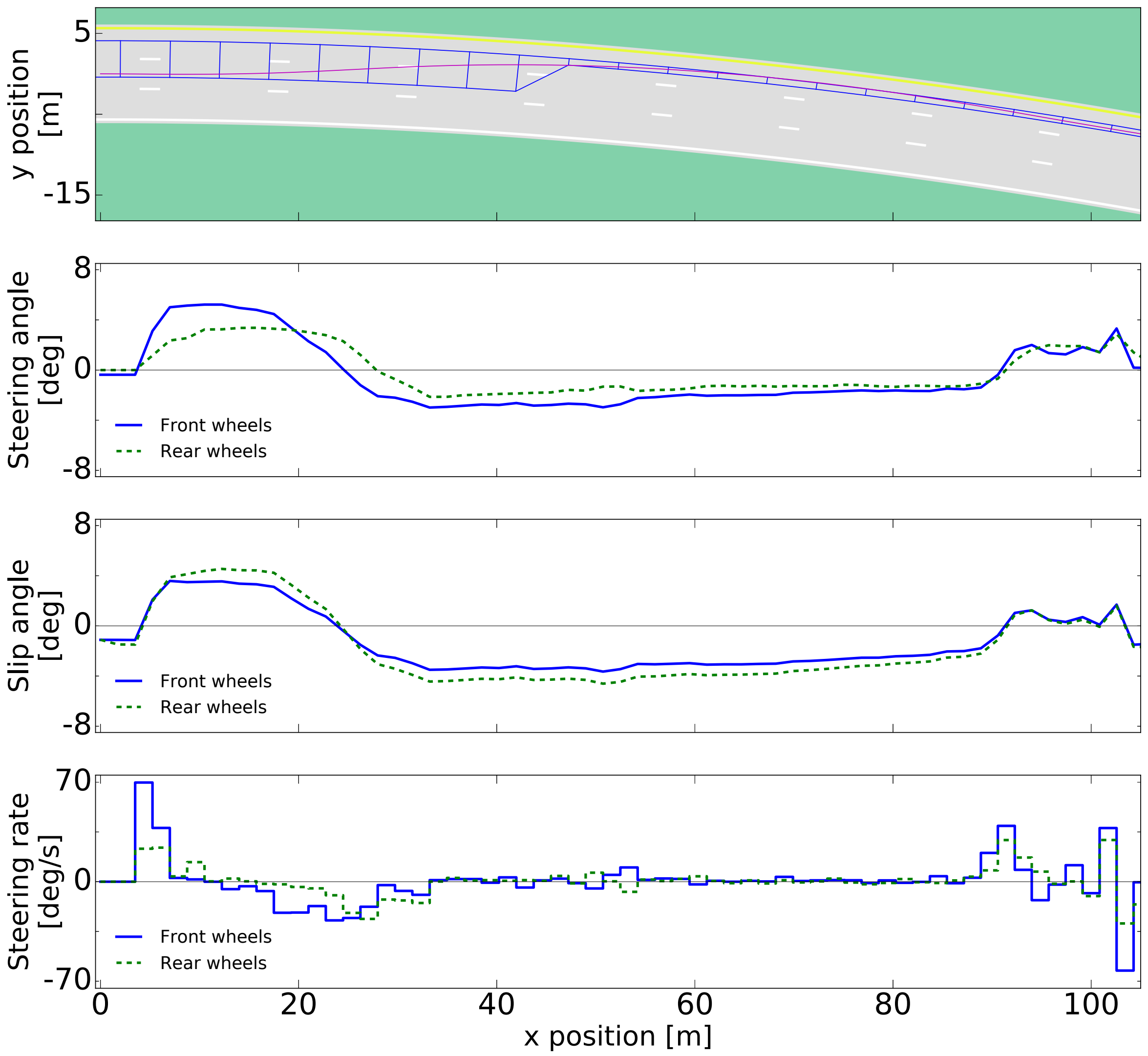


Figure 2.7: An outside lane change CIS maneuver. The vehicle identifies an obstacle 47 m away, and begins a quick left turn followed by a long sweeping right turn into the next lane.

2.5.2 Inside Lane Change

Next, consider an inside lane change, where the vehicle changes into the right lane for a right hand turn. Fig. 2.8 shows the same 4 concurrent plots as the previous example.

This is a comparatively difficult maneuver, because the vehicle must make a more aggressive turn-in than the starting trajectory. The inside CIS maneuver begins with a stronger turn to the right to begin the lane change, holds that tighter turn for a stretch, then makes a left turn to counter steer. Similar to the outside lane change, the vehicle just changes lanes when passing to the inside, and travels to the edge limit of the inside lane change. The maneuver finishes with the vehicle settled in the inside lane at the new steady state terminal trajectory.

One difference with the inside lane change is the counter steering takes place just before crossing the obstacle. This is because the MPC controller can see that the right lane boundary constraint will be active, and must avoid turning too far to the right. This can be difficult for a non-professional human driver, as identifying the perfect moment to initiate the counter steer is not trivial. However, the MPC accounts for the future states, and makes the appropriate corrections in a timely manner.

Additionally, the nature of the necessary counter steer emphasizes the need for a one-level MPC architecture. It would be difficult to identify an optimal drivable $x - y$ reference trajectory *a priori* requiring the perfect counter steer, and would be challenging to develop a controller to safely follow said reference trajectory as it drives at the edges of the drivable tube, requiring the vehicle operates at the limits of handling.

This inside lane change requires a peak tire slip of approximately 7.2° slip, corresponding to approximately 95% of the available tire force. While the CIS controller is able to find a maneuver to perform the inside lane change for the obstacle at 47 m away, this is approaching the minimum distance an inside CIS maneuver can be feasible.

While the outside lane change is less aggressive than the inside lane change, potential obstacle occlusion is not considered in this work, which might prevent ensuring the

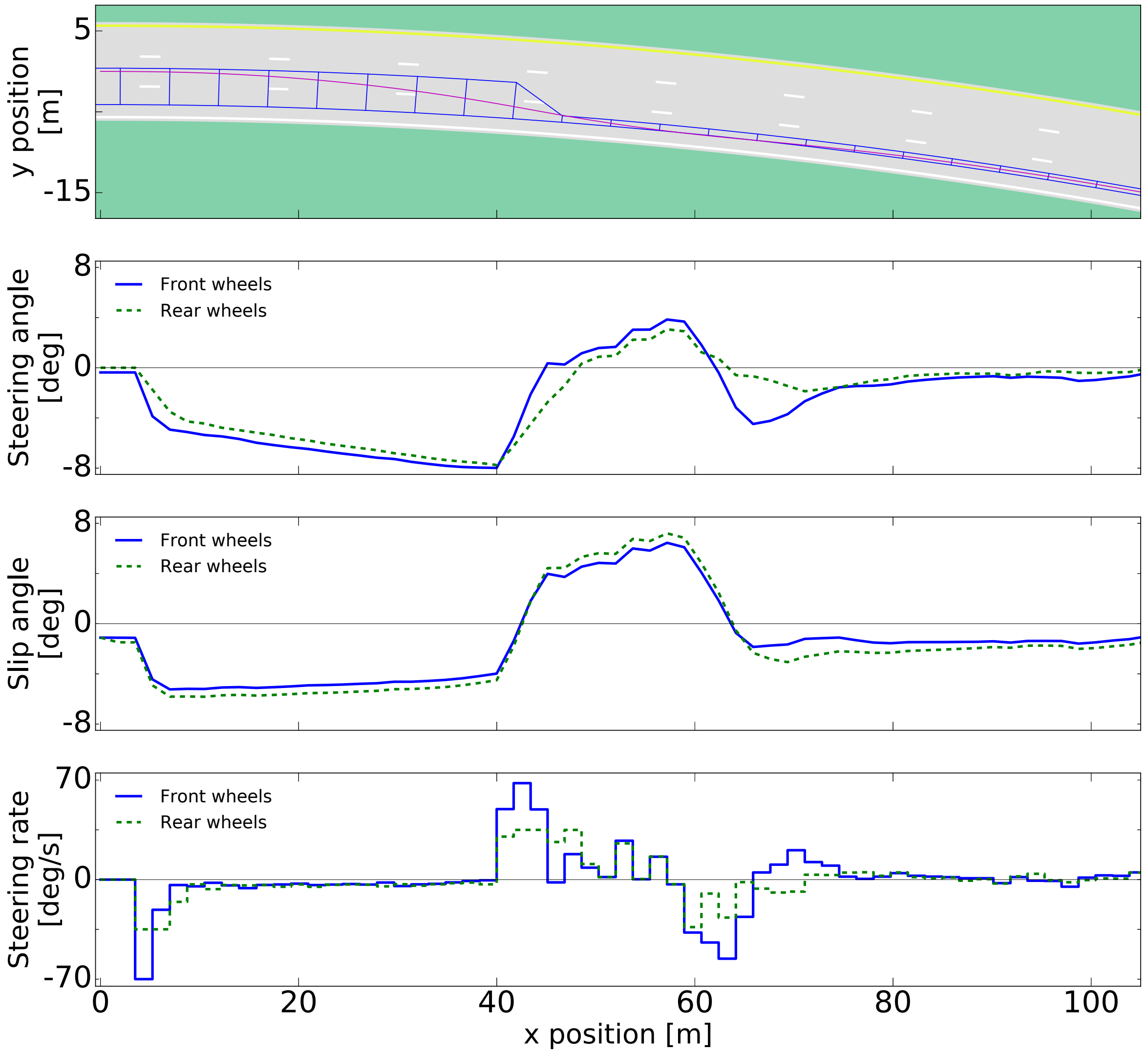


Figure 2.8: For an inside lane change CIS maneuver, the vehicle must quickly further load the tires to a tighter turn, which is comparatively more aggressive. Additionally, just prior to passing the obstacle, the host begins an equally aggressive counter steer to avoid turning too far into the inside lane. In final version, figure 3 and 4 are intended to be side by side at the top of the page for comparison purposes.

outside lane is safe for a lane change.

2.5.3 Outside Double Lane Change

In the previous simulations, the vehicle performs a single lane change, where the terminal position is either the outer or inner lane. Alternatively, a double lane change maneuver can be conceived, where the vehicle must return to the starting lane after passing the obstacle. In this scenario, the center lane is blocked between 57 m and 67 m, and the outside lane is blocked at 97 m. Thus, the host vehicle must leave the starting lane within 57 m, return to the center lane before 97 m but after 67 m, and must settle in the original lane at the end of the trajectory. The resulting four concurrent plots can be seen in Fig. 2.9.

Similar to the outside single lane change, the outside double lane change begins with a quick turn to the left and then a sweeping turn to the right. However, based on the obstacle distance and outer lane restriction at 97 m, the vehicle counter steers comparatively sooner. In contrast to the single lane change, the double lane change does not reach the outer lane boundary, but rather must begin re-entering the starting lane as soon as it passes the obstacle.

The nonlinear MPC is able to properly account for the outer lane restriction, and identifies that the least aggressive maneuver requires an earlier counter steer to allow the vehicle to settle in the starting lane in time. Further, this maneuver does not reach the outer lane boundary, making it more challenging to generate an optimal reference trajectory *a priori* that minimizes peak tire excitation, highlighting the importance of a one-level architecture.

For this example, the outer lane is artificially restricted at 97 m, mandating a double lane change. If this restriction were not in place, and either the outer lane or starting lane are equally acceptable terminal lanes, then a higher level controller would have to evaluate the target terminal lane. This can be challenging for the controller, because based on the obstacle distance and prediction horizon, it cannot be guaranteed there is

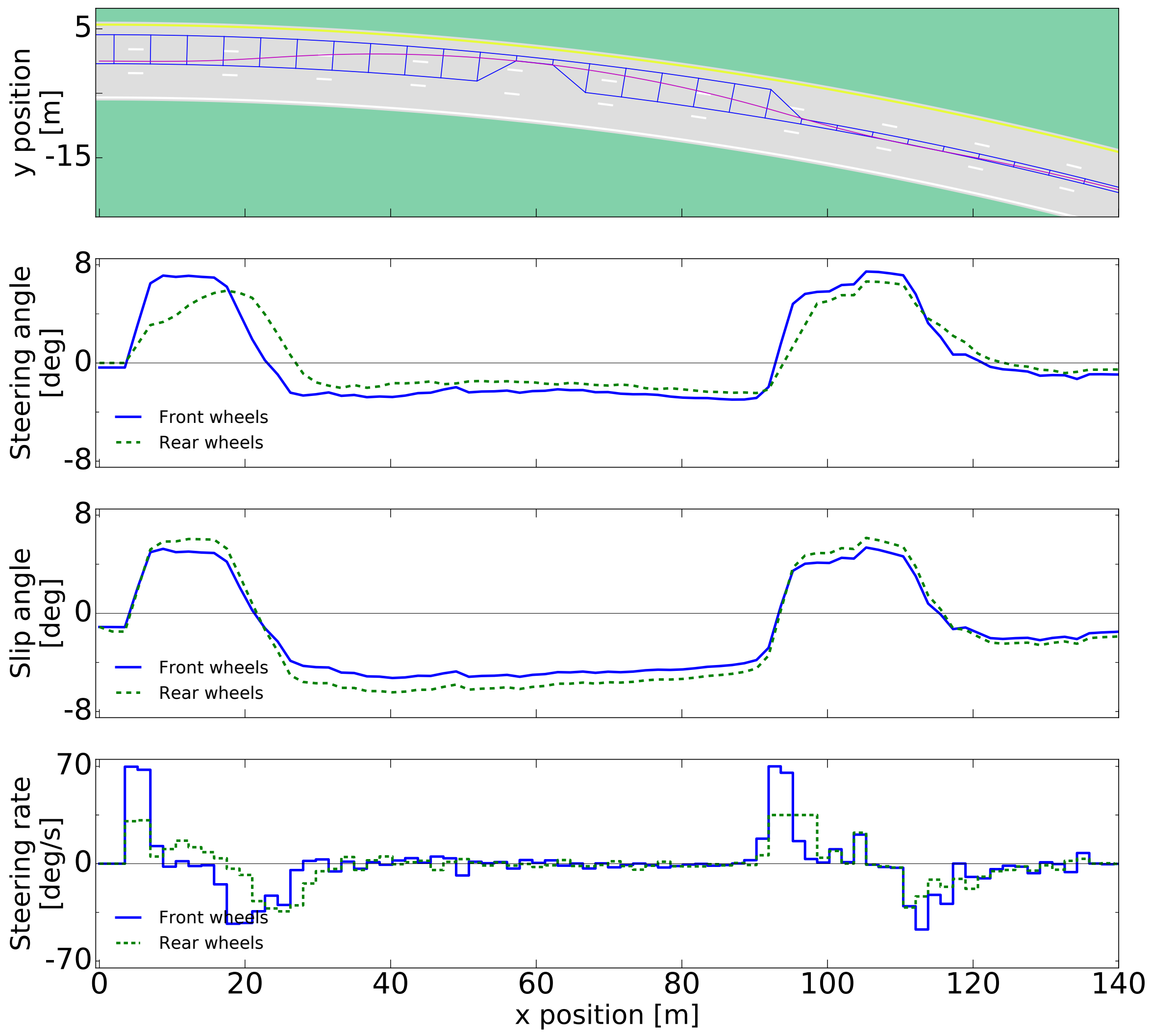


Figure 2.9: An outside double lane change CIS maneuver. The vehicle identifies an obstacle 57 m away, but must find a trajectory that returns to the initial lane on completion.

sufficient travel distance after the obstacle for the host to return into the starting lane. This can be problematic as the existing MPC formulation cannot identify sufficient space to return and can only conclude it is an infeasible problem. Thus, the decision coming from a higher level controller to perform a single lane change versus double lane change must be conscious of this corner case.

Based on these various situations, the CIS controller developed with regards to a drivable tube can adapt to various types of CIS maneuvers. Hence, it is feasible this controller can handle variations in road curvatures, speeds, and other highway factors.

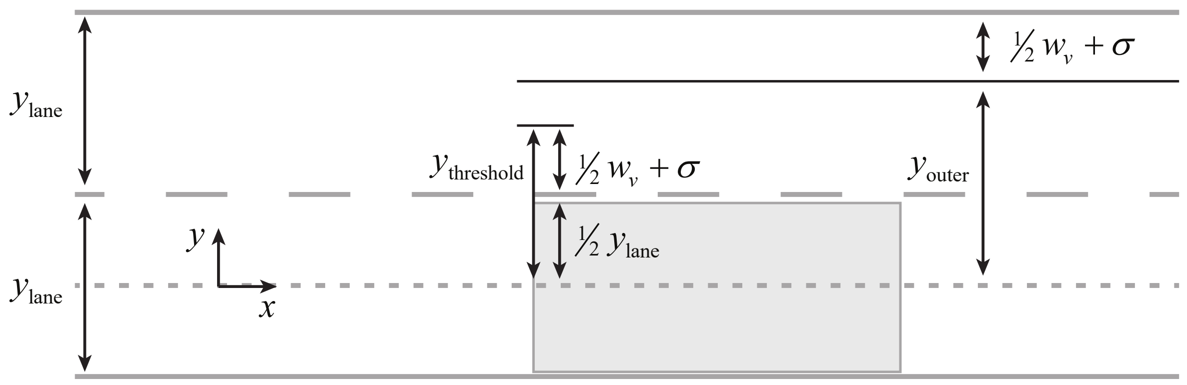
2.6 Straight Road Formulation

2.6.1 Straight Road Environment

One special highway scenario is in straight road settings. In the straight road setting, the effects of road curvature are not present, and certain criteria about the lane change can be exploited in the global reference frame. For specifically, the lane change threshold can be well defined in the global coordinate frame, easily supporting a new formulation.

While the minimum slip formulation discussed prior in Sec. 2.3 is useful for the minimum aggressive maneuver, it does not directly support finding the limits of performance. For that, a minimum distance formulation is presented, which seeks to leave the starting lane in the minimum distance subject to the CIS criteria. This formulation establishes what the latest possible moment for intervention is, and can be useful to help tune when the controller intervenes.

The geometrically simulated straight road environment uses the same $w_{\text{lane}} = 3.7$ m as the curved road, as well as same plant and prediction model. However, by aligning the road longitudinally with the global x axis, the lateral lane change boundary aligns with a lateral displacement of $y = \frac{w_{\text{lane}}}{2}$. Similar to the contraction in the drivable tube, the lateral lane boundaries are adjusted for the vehicle half width and safety buffer.



y_{lane}
$\frac{1}{2} w_v + \sigma$
$y_{\text{threshold}}$
$\frac{1}{2} w_v + \sigma$
y_{outer}
$\frac{1}{2} y_{\text{lane}}$
y
x
y_{lane}

Figure 2.10: Numerical thresholds for the straight road scenario.

As such, a lane change takes place at $y_{\text{threshold}} = \frac{w_{\text{lane}}}{2} + \frac{w_{\text{vehicle}}}{2} + \sigma$, and the outer lane boundary is enforced along $y_{\text{outer}} = \frac{3w_{\text{lane}}}{2} + \frac{w_{\text{vehicle}}}{2} + \sigma$. An example of the straight road setting can be seen in Fig. 2.10.

For this scenario, the host starts centered in the lower lane traveling to the right. The center of the lower lane is set as $y = 0$, and the vehicle must displace in the positive lateral position to make the lane change. Next, this lane change criteria is numerically formulated to support the minimum distance numerical optimization problem.

2.6.2 Minimum Distance Objective Function Formulation

The minimum distance formulation is designed to minimize the x position when the y position satisfies the lane change threshold. This formulation cannot be captured through the drivable tube concept because the tube would have to know the x position *a priori* to contract the edge boundary. By extension, the first CIS criterion is not guaranteed through the drivable tube, rather enforced through the objective function.

Due to the discrete integration states in the prediction trajectory, no point $\mathbf{x}_i$ will perfectly align with $y_i = y_{\text{theshold}}$. However, consider the two consecutive integration points bounding $y_{\text{threshold}}$, where $y_k < y_{\text{threshold}} < y_{k+1}$. The x position when passing the obstacle can be estimated by linear interpolation between the bounding states $\mathbf{x}_k$

and $\mathbf{x}_{k+1}$ as follows.

$$\text{distance}_{\text{fixed offset}}(\mathbf{x}_k, \mathbf{x}_{k+1}) = x_k + \frac{x_{k+1} - x_k}{y_{k+1} - y_k}(y_{\text{threshold}} - y_k) \tag{2.18}$$

(2.18) solves for the x position when the y position crosses $y_{\text{threshold}}$ and is C^2 with respect to $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$. Additionally, this objective function inherently captures the first CIS criteria, as the objective function mandates the y_{theshold} is satisfy, ensuring the vehicle completely changes lanes for the solved x position. Next, the second CIS criterion is constructed.

2.6.3 Outer Lane Boundary Constraint

While the drivable tube enforces the second CIS criterion natively by adjusting the outer tube boundary, the straight road controller requires this constraint to be explicitly enforced. However, by exploiting the straight road boundaries in the global coordinate frame, this constraint is straight forward.

By the second CIS criterion, a candidate maneuver is considered safe if the y position at all discrete integration points in the prediction horizon are less than y_{outer}, as seen in (2.19).

$$y_i \leq y_{\text{outer}} \; \forall \; i \in [1, n] \tag{2.19}$$

This constraint can be incorporated into a KS aggregation similar to the curved road example using a aggregation parameter of $\rho_{y_{\text{lane}}} = 200$. Finally, the third CIS criterion is presented as well.

2.6.4 Straight Road Stability Criteria

The two stability criteria for the straight road case mirror the curved road. In the case of the controllability consideration, the tire slip constraints in (2.7) and (2.8) are enforced the same as curved roads. The terminal state position is similar, but is easier to define in the straight road case.

For the straight road, the settled terminal state leaves the host vehicle traveling straight down the road, with wheels forward and y shifted into the next lane. Similar to the curved road scenario, establishing the x position at the end of the prediction horizon is challenging. However, because there is no curvature, the y and ψ states are well established, which supports the settled state definition as follows.

$$\mathbf{x}_{\text{stable}} = \begin{bmatrix} x & y_{\text{lane}} & 0 & u_0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{2.20}$$

In (2.20), the final x position is defined recursively; per the CIS criteria, the terminal x position is not a consideration, thus has no influence.

The remaining constraints in the optimization problem are dictated by the vehicle model. For the straight road case, the same host vehicle and prediction models are used, thus the same steering angle and steering rate constraints are enforced. Thus, all constraints are formed to design the numeric optimization problem that is the minimum distance CIS controller as follows.

2.6.5 Minimum Distance Optimization Problem Formulation

Combining the constraints specific to the straight road minimum distance formulation and the select stability and vehicle constraints discussed in Sec. 2.3, the numerical optimization problem is posed as follows. The optimization problem does not specifically include the KS aggregation in the formulation, but it is recommended to include it when mapped into IPOPT as this will reduce the optimization problem size, improving convergence speed.

$$\begin{array}{lllll}
\min\limits_{\mathbf{u}} & \text{distance}_{\text{fixed offset}}(\mathbf{x}_k, \mathbf{x}_{k+1}) & & & \\
\text{subject to} & y_i & \leq y_{\text{boundary}} & \forall\ i \in [1, n] & \\
& \alpha_{f,i} & \leq \alpha_{\text{peak}} & \forall\ i \in [1, n] & \\
& \alpha_{r,i} & \leq \alpha_{\text{peak}} & \forall\ i \in [1, n] & \\
& |\delta_{f,i}| & \leq \delta_{f,\max} & \forall\ i \in [1, n] & \qquad (2.21) \\
& |\delta_{r,i}| & \leq \delta_{r,\max} & \forall\ i \in [1, n] & \\
& |\dot{\delta}_{f,i}| & \leq \dot{\delta}_{f,\max} & \forall\ i \in [1, n] & \\
& |\dot{\delta}_{r,i}| & \leq \dot{\delta}_{r,\max} & \forall\ i \in [1, n] & \\
& \mathbf{x}_{\text{terminal}} - \mathbf{x}_{\text{stable}} & = \mathbf{0} & &
\end{array}$$

One drawback to the straight road formulation is the first and second CIS criteria are posed in the global coordinate frame, aligned with the road. This can be challenging for implementation in a vehicle fixed reference frame, as relaxing constraints in the global x position transforms into relaxation in the x and y local coordinate frame. However, this point is not a factor for numerical simulations.

Next, the optimization problem is solved and simulation results reported.

2.6.6 Straight Road Numerical Simulations

For the straight road example, the vehicle velocity is set at $u_0 = 30$ m/s to provide some variation from the previous examples. To motivate the necessity of an evasive lane change, the minimum braking distance is estimated. Because the vehicle is not on a curved road, a component of the available tire force does not need to maintain a radial component, thus braking performance is improved. For the $0.8g$ deceleration allowed through the coefficient of friction, limit braking requires 57.3 m of travel over 3.8 s.

Again, human drivers so not brake as aggressively, which would require additional distance. However, real world effects, such as aerodynamic drag and down force, can improve braking distance. Automotive manufacturers often publish the $60 - 0$ mph braking distance, which the host vehicle is modeled after is listed as 34.4 m. This braking distance represents an average deceleration of about $1.2g$, which for the $30m/s$ example requires 44.6 m travel.

Fig. 2.11 plots the minimum distance lane change in the same 4 subplot manner as prior.

In Fig. 2.11, the first plot shows the $x - y$ trajectory in blue as the vehicle travels down the highway. For front and rear steering limited to $\alpha_{\text{peak}} = 8°$, the vehicle changes lanes in 31.0 m, which is almost half the distance of theoretical limit braking at $0.8g$, and about a two thirds that of $1.2g$. The orange rectangle represents a fully blocked lane and is placed at the x position representing when the host vehicle departs the lane. The blue star represents $y_{\text{threshold}}$.

The second plot shows the front and rear steering angle, as well as vehicle rotation. Both front and rear wheels initialize the maneuver with a turn to the left to begin departing the lane, and then both turn to the right to stay within the lane boundary, and finally steer to settle the vehicle at the end of the trajectory. As the vehicle's lateral velocity and yaw rate grow during the maneuver, the controller makes small adjustments to hold the tires at the peak loading. This can be seen between 8 m and 18 m, where the front slip angle is steady at α_{peak}, but the steering angle slightly grows.

Similar to the curved road example, the straight road case initiates a counter steer prior to clearing the obstacle. This is because if the vehicle held the left turn until leaving the lane, the vehicle would not be able to stay within the lane boundary later in the trajectory. Turning back into the lane at the right moment prior to clearing the obstacle may be counterintuitive for some drivers, especially in an emergency situation at high speeds.

The third plot shows the tire slip throughout the maneuver. The front tires turn left to their peak slip limit to maximize the lateral force in the vehicle frame, and hold the high slip angle until turning to the right and holding the peak slip again. While

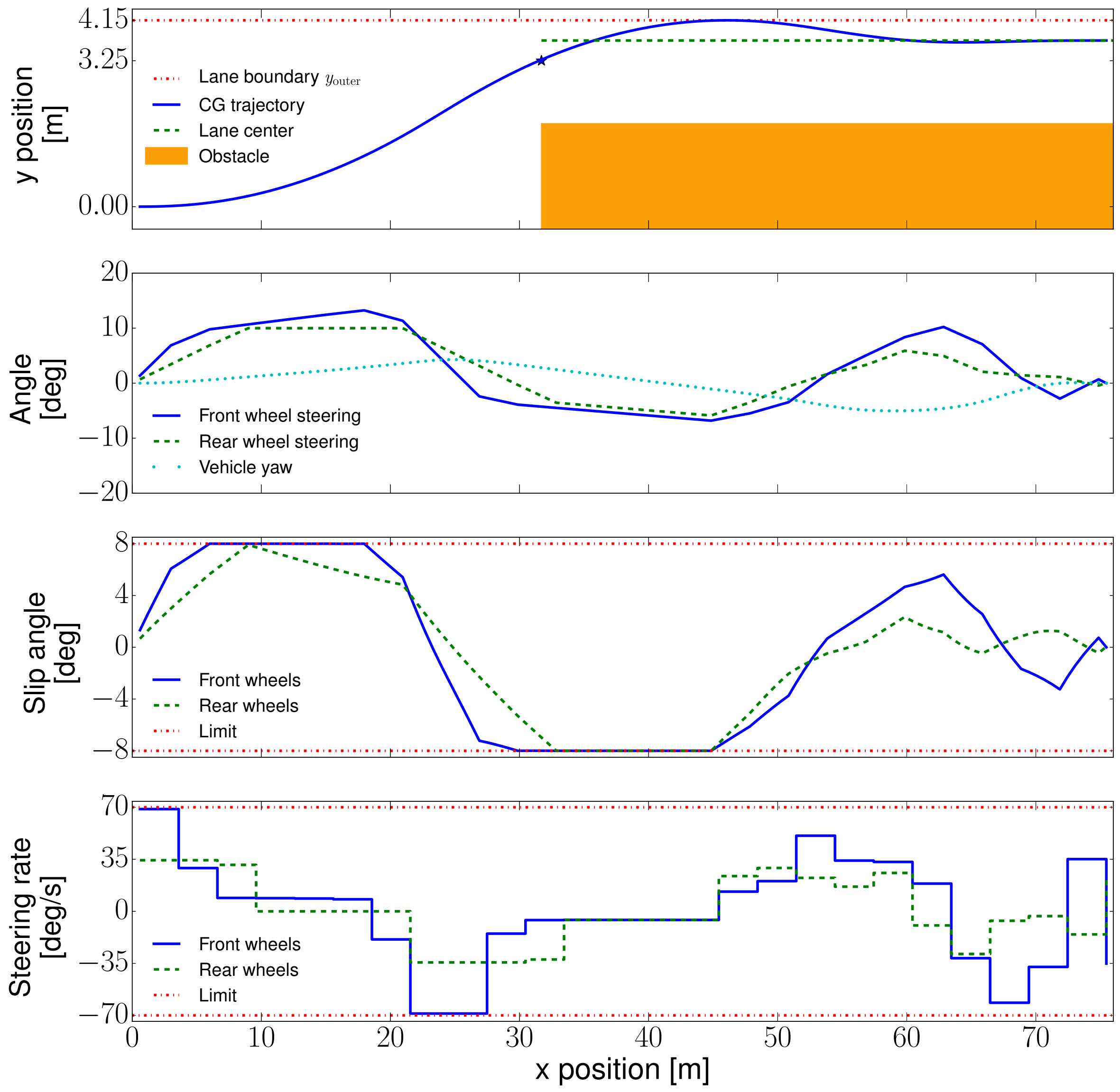


Figure 2.11: The four concurrent plots, plotted against x position of the maneuver, show the various states and control inputs during the aggressive lane change.

holding the high slip, the front wheels make small steering adjustments to maintain the high slip as vehicle lateral velocity and yaw rate develop.

The rear tires attempt to turn to their peak slip limit, as well, but are steering angle and rate limited in doing so. As a result, they reach the peak steering angle, before turning back to the right at the peak steering angle again. Because the rear steering mechanism's steering and steering rate constraints are active, it is feasible that an improved rear steering mechanism can improve the CIS performance.

The fourth plot shows the steering rate throughout the maneuver, but without the 100 ms timing delay. For the minimum distance formulation, the distance is calculated as if the vehicle gets to act immediately to provide the theoretical best performance.

To change lanes in the minimum distance, the controller will always push the vehicle to its allowable limits of handling. For this simulation, α_{peak} is set based on the stability limits discussed in Sec. 2.3.2. In the next section, α_{peak} is adjusted manually to provide insight on the controller performance.

2.6.7 Minimum Distance Performance Versus Maneuver Aggression

To minimize the lane change distance, the CIS controller will always push the vehicle to the allowable limits of handling. By manually adjusting what these limits are, as defined by α_{peak}, the trade-off in lane change distance performance versus vehicle performance can be investigated.

By manually varying the allowable maximum slip angle α_{peak} in (2.21), the distance to perform a lane change for a prescribed level of vehicle excitation is resolved. Fig. 2.12 plots the minimum lane change distance against peak allowable slip for both front and rear steering, and front only steering architectures. Per Sec. 2.3.4, front only steering is numerically achieved by locking rear wheel deflection as $\delta_{\text{rear}} = 0°$.

Based on the Pareto fronts of Fig. 2.12, the windows of opportunity to initiate a CIS maneuver are sizable, even for low slip angles. Only at very low slip angles, i.e.

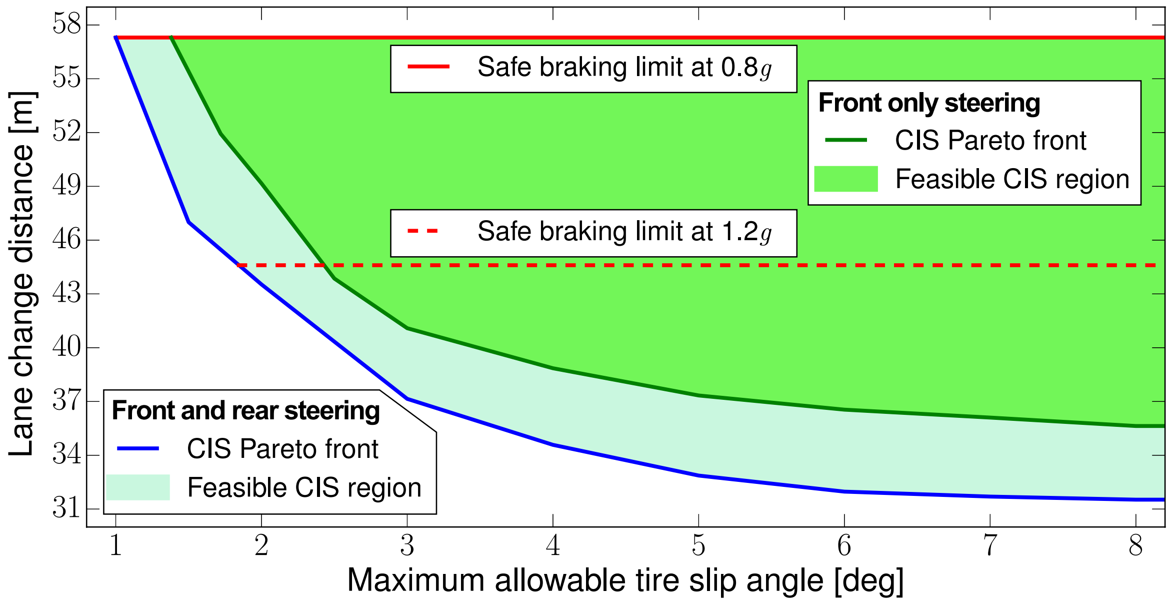


Figure 2.12: Increasing the allowable slip angle improves the effectiveness of the CIS system, but with diminishing returns.

less than 2° peak slip, do the theoretical limit braking distances match the required CIS distance. This implies if the vehicle identifies the obstacle quickly and can solve the CIS maneuver in a time efficient manner; an evasive lane change can be performed with small tire excitation.

While the minimum distance CIS formulation provides a theoretical bound on the shortest lane change distance, waiting until this theoretical limit distance to initiate CIS is not an advisable control strategy in practice due to plant-prediction model mismatch inherent to MPC [60]. Thus, the minimum slip formulation presented with the drivable tube concept is preferable for implementation.

2.7 Robustness Analysis and Adaptive Formulation

2.7.1 Literature Review on Adaptive Automotive Controllers

MPC controllers employ a prediction model to accurately represent the plant system response. For nonlinear MPC formulations operating at the limits of handling, it is critical to accurately capture the system response as incorrect control actions can leave the system in an unrecoverable state [84], [85]. For the CIS controller, it is essential to capture the interactions between the tire and road properly, and parametric uncertainties in the tire model can significantly impact the safety and performance of MPC [86].

Various tire models have been developed taking different parameters as input. Slip-based models such as the Pacejka tire formula [87], [88] and exponential tire formula [89] use the localized tire slip to estimate tire force and are widely used for on-road applications. In these models, the most critical parameter to accurately model the tire force is the coefficient of friction. The high sensitivity of tire response to the coefficient of friction can be problematic for safety critical applications, as path-tracking controllers can require within 2% accuracy in the coefficient of friction [24]. While it is infeasible to obtain this accuracy *a priori*, adaptive controllers with an observer can greatly improve closed-loop controller response.

One such observer considers both time-domain and frequency-domain vehicle responses to extract the best estimate of coefficient of friction [90]. However, in this example, the tire force uses a linear response curve, which limits the accurate range of slip angles, and thus the application is limited to a linear system dynamics model.

Researchers have also used sensor fusion to estimate the slip angle and shown that if the slip angle is used in conjunction with a fixed tire force model, the controller has poor robustness when tire parameters change; but a linear adaptive tire force model, with the coefficient of friction as an added state, greatly improves stability [91]. However, this method is still limited to linear tire forces operating far from the nonlinear region.

In the context of nonlinear tire force curves, particle filters can estimate both the instantaneous tire force, as well as tire force curve in the neighborhood of the slip angle [92]. In this example, a particle filter is used to best fit the instantaneous linear tire response from a set of precomputed curves, and the resulting curve is used for inference in the nonlinear tire response region. Further, it is shown that the particle filter is able to outperform extended Kalman filters due to the breakdown of the Gaussian noise assumption. In the presented examples, the particle filter does not retain a Gaussian uncertainty while converging, which causes difficulty for the extended Kalman filter to converge. During numerical simulations, the particle filter can identify discrete changes in the environment, and estimate the full tire response online [93]. The presented particle filter estimates of the coefficient of friction within 1% of the truth with a settling time of up to 5 s. For some applications this settling time is sufficient, but it is too slow for time critical applications such as CIS. Further, the validation cases were at low speed with soft safety constraints, which are difficult to extrapolate to safety critical maneuvers at high speed.

For the intended CIS application, it is vital to estimate both the coefficient of friction accurately, and quickly. Thus, it is proposed to incorporate an unscented Kalman filter (UKF) as this observer can estimate the coefficient of friction when operating in the nonlinear tire regime in a time efficient manner.

2.7.2 Adaptive Controller Design

One desirable aspect to the minimum slip formulation is it maximizes the peak additional tire force to make mid-maneuver corrections. While the one-level MPC formulation inherently introduces some stability performance, because the control action is resolved based on the instantaneous state, excessive discrepancies between the plant and prediction models can lead to constraint failure.

The ultimate goal of the prediction model is to accurately estimate the corresponding plant state trajectory for a candidate control trajectory. As mentioned, the Pacejka coefficients used in Table 2.2 are seldom, if ever, published. However, research has shown these coefficients can be reliably estimated from controlled lab testing and correlate well to road data [94]. However, the parameters describing the road surface, such as roughness, have a large influence on peak tire response and cannot be extrapolated from separate testing.

As a result, rubber stiffness and relaxation parameters B and C can be well estimated, but coefficient of friction μ is not as reliable. For this reason, the adaptive formulation is concerned with uncertainty in the coefficient of friction.

The error from plant-prediction model mismatch is characterized in two forms. In the first form, when the MPC applies a control command to the plant for a closed-loop iteration, in the next closed-loop iteration the plant is at a different state than what the prediction model expected it to be. This is denoted as the *experienced error*. For the second form of error, for a candidate control trajectory, there is a difference between the future state trajectories of the prediction model and the open-loop plant. This is denoted as the *prediction error* and is the focus of the adaptive formulation.

The prediction error is further compounded by the time latency of solving the MPC formulation. As mentioned, solving the optimal control problem (2.17) requires a nontrivial amount of time, which is expected to be about t_{update}, thus the starting state set as the predicted state at $t_i + t_{\text{update}}$. This compounds the prediction error, because during the time gap of t_{update}, the controller cannot change the plant's control action. Additionally, as t_{update} grows longer, the deviation of where the plant will be at t_{i+1} and where the controller thinks the plant will be at $t_i + t_{\text{update}}$ grows.

Fig. 2.13 illustrates this issue on an example maneuver. There are four theoretical trajectories shown: the plant trajectory and the three prediction trajectories solved at sequential closed-loop iterations. In this example, the vehicle is making a right turn with the prediction model using a higher coefficient of friction than the plant.

While Fig. 2.13 only illustrates the error in the topographical displacement in $x-y$, the predicted state and plant state may deviate in the other components, too. For the example CIS maneuver, this means the lateral velocity and yaw rate, both having a high influence on the tire response, can deviate and further cause discrepancies between the plant and prediction models.

To help address the errors produced by differences in the plant and prediction model, a UKF observer is introduced to estimate the plant's coefficient of friction. Recall that prior art identified the uncertainty in the nonlinear tire response for a particle filter estimator as non-Gaussian [92]. For the considered CIS maneuver, nonlinear Pacejka

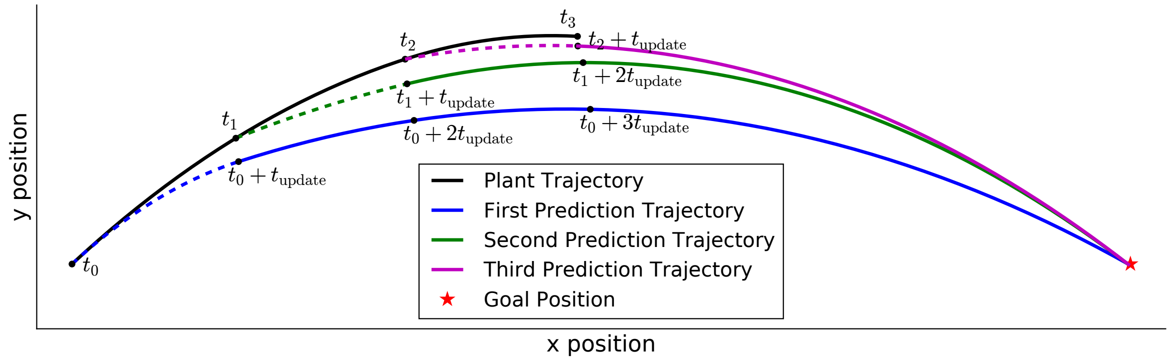


Figure 2.13: Three iterations of an example closed-loop MPC controller are shown. The vehicle starts at t_0, but extrapolates the starting state by the blue dotted line, because its control command will not be applied until t_{update}. The controller's prediction of the plant's path until the goal is shown in solid line. In the second iteration, the MPC sees the plant at a different state than expected based on the previous solution. It takes the most current information and iterates the process as shown in green and red for the subsequent iterations.

tire forces are used, and the estimator needs to identify the coefficient of friction for a range of slip angles with nonlinear response. The benefit of the UKF filter is that by propagating many sigma points, the estimator can better capture the nonlinear response in tire force due to the noise in the vehicle states and coefficient of friction compared to an extended Kalman Filter, while converging faster than particle filters.

The UKF is focused on identifying the coefficient of friction, which is a linear scaling of the tire force based on the Pacejka tire formula in Eq. (2.1). However, because of the noise in the measured states and forward propagation of the nonlinear system dynamics, the estimated tire slip is a nonlinear transformation, resulting in nonlinear mapping to the tire force. In actuality, the UKF compares the required tire force to achieve the next measurement against the belief in the system dynamics, outputting the linear scaling, by way of coefficient of friction, to best agree the two beliefs. The UKF feedback block diagram can be seen in Fig. 2.14.

The exact details of the UKF are not significant for the results of the controller, just that the prediction model adjusts. Details on the UKF can be found in [95] for an off-road application. For the work herein, the UKF is applied to an on-road scenario by

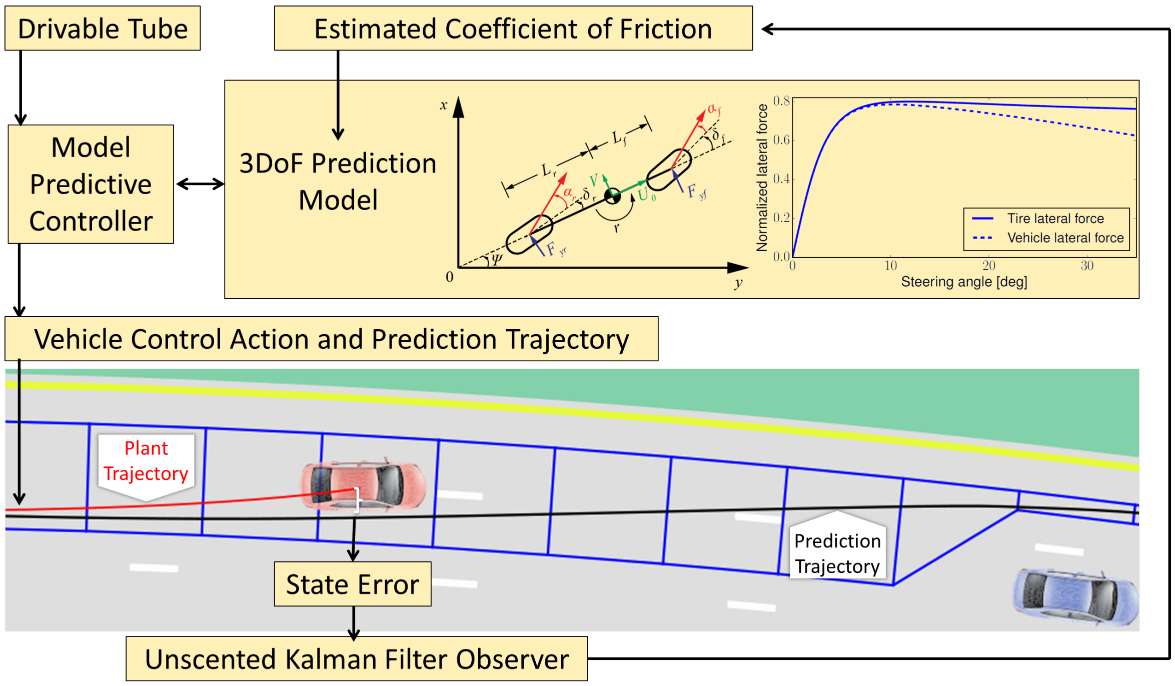


Figure 2.14: The UKF observer resides outside the MPC controller, taking in sensor information and outputting updated tire response parameters.

estimating the coefficient of friction instead of the sinkage coefficient. State uncertainty used in the estimator is the same as [95] and consistent with commercially available automotive sensors.

The UKF observer runs at a prediction rate at 100 Hz. At t_0, the MPC is launched to solve for $t_0 + t_{\text{update}}$ onward using the best guess of the coefficient of friction at t_0. Between t_0 and t_1, the UKF attempts to improve the estimate for the coefficient of friction. At t_1, the second closed-loop iteration is launched using the best guess for coefficient of friction based on the plant data between t_0 and t_1. As the UKF attempts to converge on the coefficient of friction during the closed-loop iterations, the adaptive controller has a latency of t_{update} before the MPC is able to change the control trajectory to account for the improved coefficient of friction estimate. The error due to latency can only be improved by reducing the length of t_{update}, which is a function of the controller and independent of the observer. This highlights the necessity of a fast settling observer to produce an accurate prediction model quickly, which is demonstrated further in Sec. 2.7.3 in simulation.

For the UKF to run at 100 Hz, relevant data must be available at 100 Hz. While some sensors, such as IMUs, can support sampling rates over 1 kHz, other state estimates, such as a localization from LIDAR data, could be at frequencies as low as 1 Hz. While data fusion strategies can handle different data rates, the work here uses 100 Hz sensor information for all states.

In general, the effect of update rate on the convergence of nonlinear Kalman filters is not deterministic. At higher frequencies, the observer can incorporate more sensor estimates in a given time. However, at lower update rates, the system dynamics are given more time to evolve, which can provide more insight if the system model has high belief. In general, simulating the UKF between 100 and 1000 Hz has not affected CIS controller performance.

With the observer in place, the closed-loop adaptive CIS controller is compared to the nonadaptive performance.

2.7.3 Numerical Simulations of Adaptive MPC Controller

Using the adaptive prediction model in closed loop, (2.17) is re-solved for the same environmental settings, but for the front only steering architecture. Fig. 2.15 shows a similar four subplots, but also includes a fifth subplot showing the estimated coefficient of friction. The plant has a uniform coefficient of friction of 0.6 throughout the maneuver, but the UKF starts the simulation with an initial coefficient of friction guess of 0.8. In general, this is a challenging maneuver because the MPC must perform an aggressive lane change and at the same time is experiencing a discrete jump decrease in the coefficient of friction.

A key aspect to the observer is the settling time required to estimate the coefficient of friction. The maneuver begins with the observer believing the coefficient of friction is 0.8; hence, the first closed-loop iteration is solved with a higher coefficient of friction than the plant. This can be problematic, because the prediction model expects the

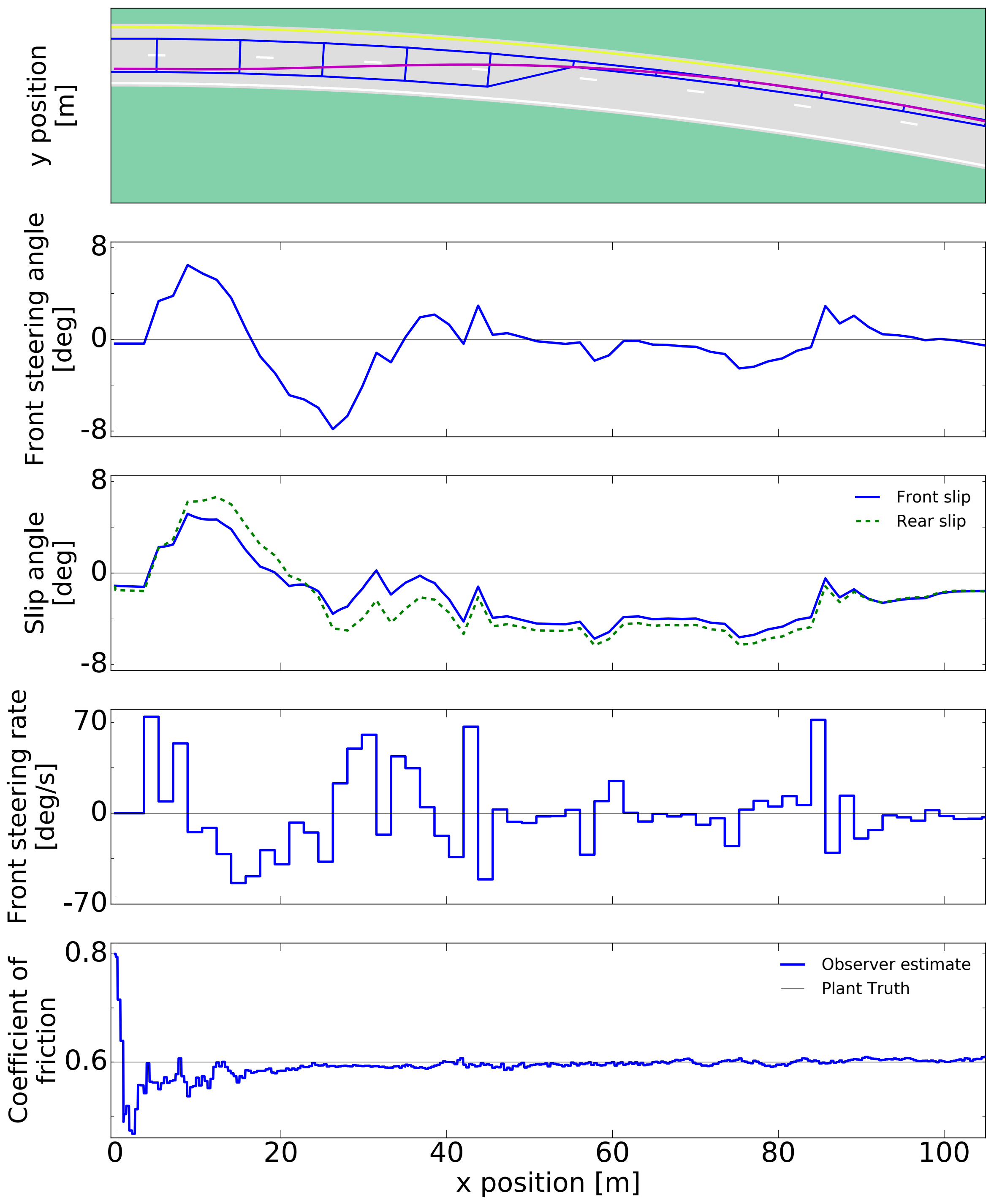


Figure 2.15: Five concurrent plots are shown showing various states of the adaptive MPC performing a CIS maneuver. By improving the estimate for the coefficient of friction online, the adaptive MPC can update the prediction model to more accurately represent the plant, thus avoiding an unrecoverable maneuver later in the trajectory.

plant to be more responsive than it actually is, hence the solved control trajectory under actuates. During the first 400 ms of simulation, the observer rapidly changes its estimate of the coefficient of friction, closing in on the plant truth of 0.6. At the second iteration, the prediction model is simulated with a coefficient of friction of 0.54, which is still a 10% error from the plant, but offers a significant reduction from the 33% error in the first iteration.

The UKF observer converges to within 5% of the coefficient of friction within 400 ms. Compared to the state of the art reviewed in Sec. 1.2, where observers can require order seconds to stabilize, this is a significant improvement. Because the UKF can quickly trend towards the plant coefficient of friction, and ultimately settle in a shorter amount of time, the need for a more aggressive control command is recognized earlier in the iterations, and the controller can apply said command to the plant sooner in the maneuver. By taking an action earlier in the maneuver, the actual peak tire slip experienced is reduced and avoids saturation.

Later in the trajectory the effect of plant-prediction model mismatch becomes apparent. Around 80 m into the maneuver, the controller must rapidly prescribe controller adjustments, even though the coefficient of friction is well estimated. Similar to the nonadaptive case, the suspension dynamics have a sizeable effect on the plant response [96], [97], thus the prediction model does not have sufficient fidelity to capture the plant as accurately. While this does not result in failure, it requires sudden correction for an otherwise non-exciting point in the maneuver.

One potential drawback to the presented UKF is the estimate for coefficient of friction can overshoot the constant plant truth, as seen at the start of the maneuver. There are two causes for this overshoot. First, determining the coefficient of friction is still susceptible to the identifiability concerns recognized in the art [90]. At the start of the maneuver, both the front and rear tires are at a low slip angle and the controller is unloading the tires as it turns to the left, resulting in little excitation of the vehicle dynamics. At low vehicle excitation, random sensor noise results in a large variance in tire slip angle, which correlates to a large variance in coefficient of friction. Due to the low excitation, the UKF has difficulty converging at the start of the maneuver. After approximately 20 m of travel, the vehicle has a persistently active tire slip where the coefficient of friction can be observable, and the UKF converges rapidly.

Compounding the identifiability concerns, the second cause for overshoot is due to the process noise covariance matrix, which is unknown for the nonlinear 3DoF prediction model, and is treated as tuning weights based on simulations. However, it is more important for the UKF to estimate in the neighborhood of the plant coefficient of friction quickly than to settle on the exact value. In general, faster settling times means earlier iterations of the closed-loop controller have a more accurate prediction model, which means earlier iterations can take appropriate action to set the vehicle up for success later in the maneuver. The tunable parameters used in the UKF are not purely focused on settling time, but rather balance transient performance against sensor noise, resulting in the initial overshoot.

Later in the maneuver there is still some variation in the coefficient of friction. This comes from the constant sensor noise in the vehicle state, as well as system dynamics discrepancies in the high fidelity plant model and bicycle prediction model.

Prior in Sec. 2.7 it was discussed that the plant-prediction model mismatch error can be addressed by the nature of closed loop iterations alone, provided the mismatch error is small. Fig. 2.16 shows a batch analysis for various plant coefficients of friction and nonadaptive prediction coefficients of friction. For the nonadaptive controller, there exists a green band roughly along the diagonal. For points on the diagonal, the prediction model has the same coefficient of friction as the plant, but is still subject to state error due to the 3DoF prediction model versus 14DoF plant model fidelity. For points near the diagonal, the prediction model response is sufficiently close to the plant response that the closed loop control can handle the experienced error without changing the prediction model. However, there are two segmented regions of failure experienced in the lower right corner and upper left corner.

In the lower right corner, the prediction model has a lower coefficient of friction than the plant, meaning the plant is more responsive than the prediction model. This

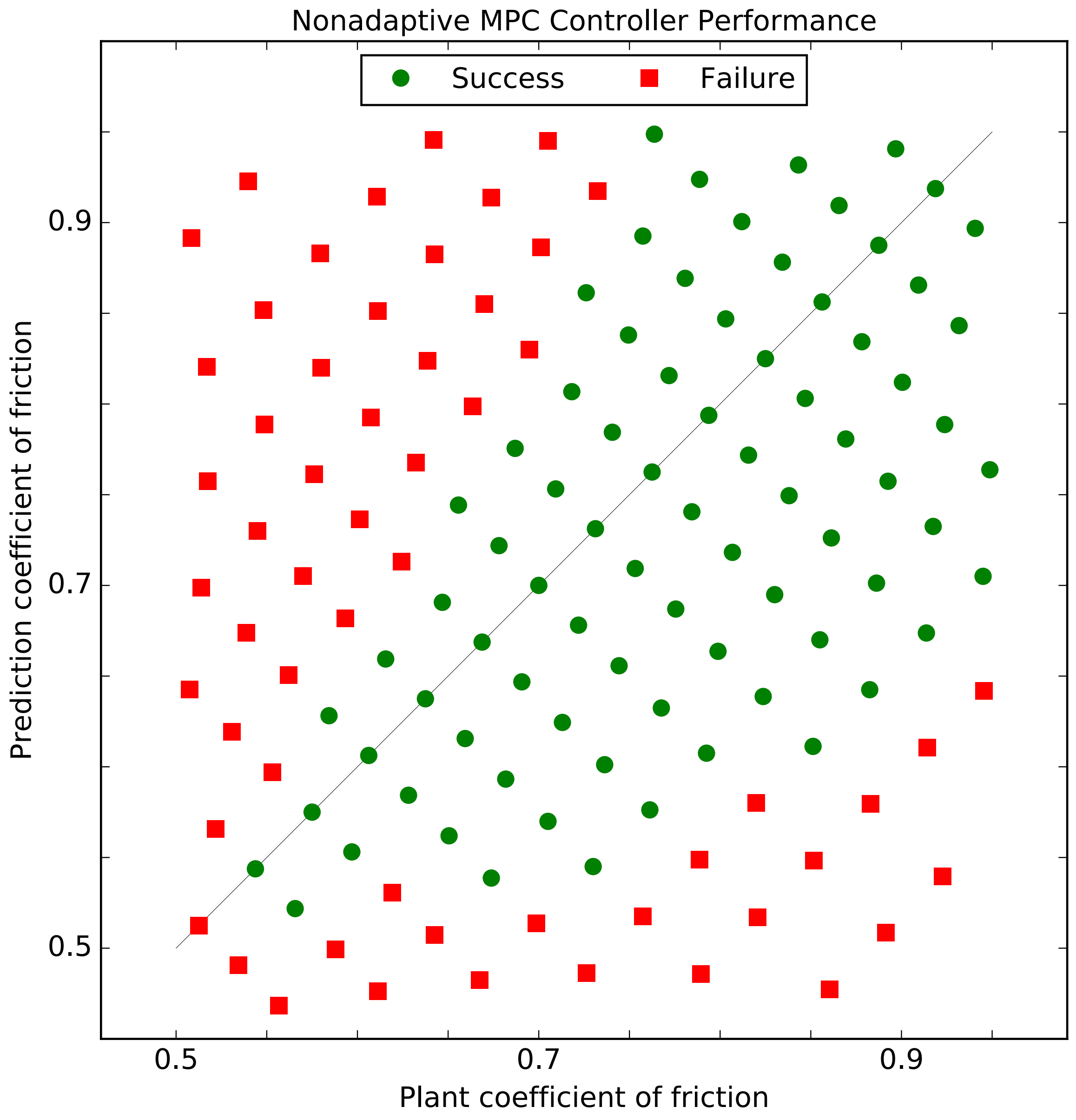


Figure 2.16: Various combinations of the coefficient of friction for the plant model and prediction model are analyzed. Red squares indicate at some point in the maneuver, the plant violated hard safety constraints, resulting in an unsuccessful CIS maneuver. Green circles indicate the closed loop nature of the controller is sufficient to account for the small deviation in plant and prediction model mismatch.

can cause failure, because when the controller is attempting to turn back into the lane to avoid overshooting the outer lane boundary, the plant turns in too much and does not completely leave the starting lane when passing the obstacle, violating a hard safety constraint of the CIS maneuver.

In the upper left corner, the prediction model has a higher coefficient of friction than the plant, implying the plant is less responsive than the prediction model. Such as in Fig. 2.13, a less responsive plant drifts to the outer lane easier and helps leave the starting lane. However, beyond the obstacle, the plant continues to drift to the outside, eventually violating the outer lane boundary constraint, resulting in failure.

Fig. 2.16 only considers the plant to a coefficient of friction lower limit of 0.5. This is because 0.52 is approximately the lower limit, below which no feasible CIS maneuver exists, even if the prediction model has perfect knowledge of the plant.

For comparison, the same batch analysis is also conducted for the adaptive controller using the UKF observer.

Fig. 2.17 shows a similar successful region along the diagonal, and two segmented failure regions in the bottom right and upper left. However, the adaptive controller's success region expands farther from the diagonal than the nonadaptive controller, showing improved performance for a larger discrepancy in the plant and prediction models' coefficient of friction. For example, the nonadaptive controller is able to perform a successful CIS for only up to a 25% error in the coefficient of friction, whereas the adaptive controller performs successfully for as a large as a 55% error. Given that the plant coefficient of friction is not known *a priori*, the improved robustness of an adaptive controller with a fast converging observer suggests the method is more robust to real world effects and should be included at implementation.

The adaptive controller's success boundary is not as well defined as the nonadaptive controller. Below a coefficient of friction of 0.52, the controller cannot conclude a feasible solution exists. Due to the effect of the sensor noise on the estimate in the UKF, certain coefficient of friction pairs result in Eq. (2.17) being infeasible on some repetitions of the simulation. This noise causes a blurred edge boundary, with some

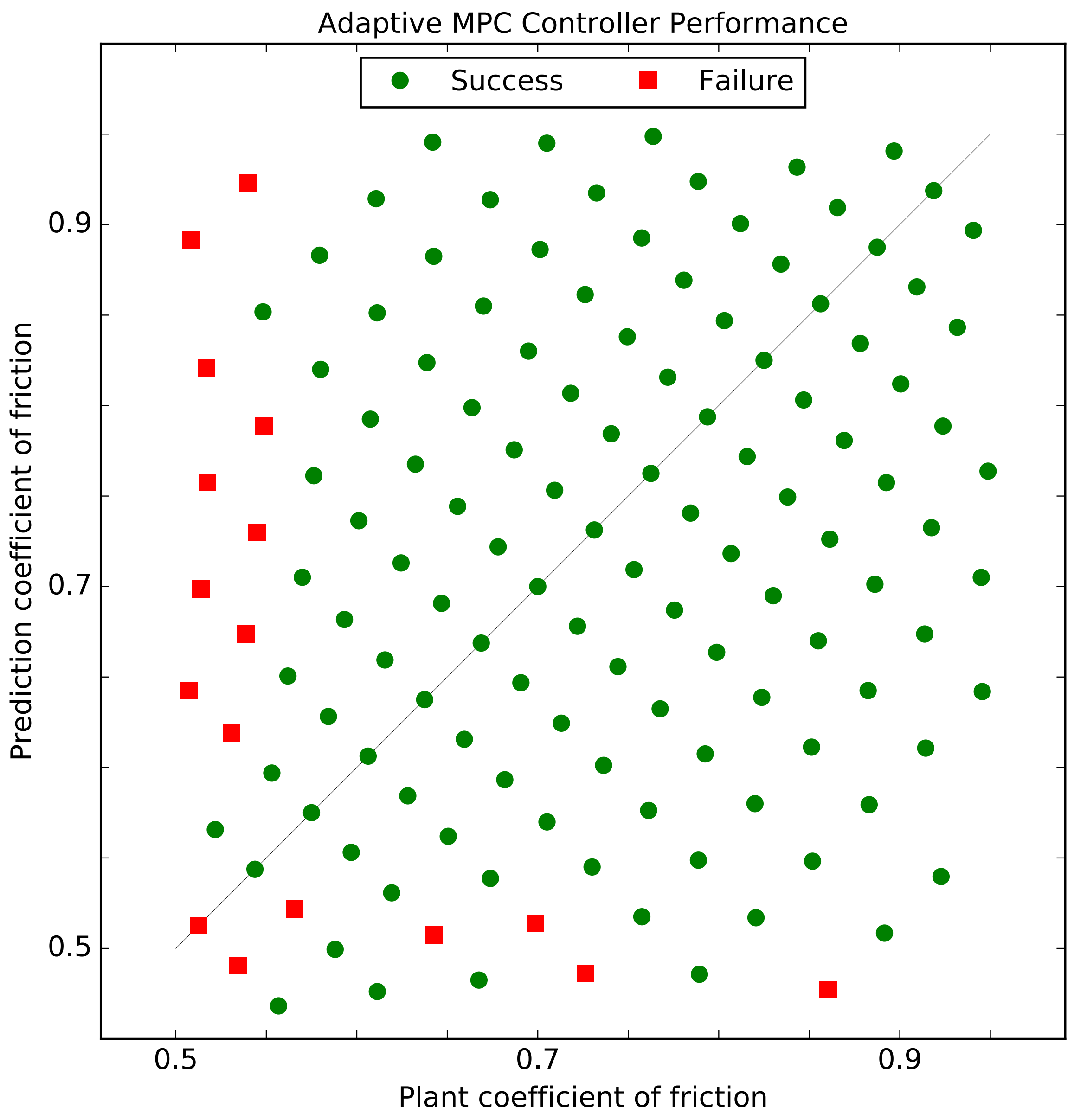


Figure 2.17: Various combinations of the coefficient of friction for the plant model and prediction model are analyzed. The band of green circles extends further than the nonadaptive case, showing the adaptive MPC can handle a larger error in the initial prediction coefficient of friction.

edge points that are successful in the nonadaptive case to be sometimes unsuccessful in the adaptive one.

Both controllers experience failure at low plant-prediction coefficients of friction. This is because the chosen highway scenario defined by the vehicle speed, obstacle distance, road curvature, etc., inherently pushes the vehicle to its limits of handling, which is reduced by the coefficient of friction.

To recap the prior few sections, a nonlinear one-level MPC controller is presented to perform an evasive lane change maneuver by some optimality condition. The controller is primarily designed to incorporate a drivable tube concept, embedding the CIS safety criteria. The controller formulation is shown to be adaptive to different highway scenarios, supporting flexibility to expected real world conditions. Additionally, an adaptive controller formulation is introduced to support plant-prediction model mismatch through the coefficient of friction. The adaptive controller improves performance in uncertain road conditions, further improving the feasibility of implementation.

However, a major barrier to implementing the proposed controller is being able to solve the nonlinear optimization problem in real-time. In the next chapter, the computational cost of nonlinear MPC is investigated, and a framework for real-time performance introduced.

Chapter 3

Computational Cost of Nonlinear MPC Controllers

The optimal control problem to determine a CIS maneuver is posed as a numerical optimization problem in Chapter 2. The computational cost of computing the maneuver depends on convergence properties of the underlying numerical optimization problem. While the specific nonlinear optimizer, in this case IPOPT, is taken as an off the shelf solution, how the optimal control problem is mapped into the numerical optimization problem has a significant effect on the solution time. In the most primitive form, the CIS controller converged to solutions on the order of tens of seconds. For the $t_{\text{update}} = 100$ ms target, this is computationally prohibitive in terms of real-time performance.

To achieve real-time execution, the underlying computational cost of solving the nonlinear optimization problem is scrutinized. While the intended application is the CIS controller from Chapter 2, this dissertation presents the computational framework in a broad nonlinear MPC context. Specific to within the nonlinear MPC problem, the numerical integration scheme is discussed, trajectory optimization structure investigated, and parallel computing considered. To begin, the numerical integration procedure is discussed next.

3.1 Computational Cost: Integration Time Step

At a high level, the CIS optimal control problem is to find a control sequence that steers the vehicle from its starting state, through the drivable space, and into the next lane. This forms an initial value problem (IVP), which is well studied in the literature [8]. A common approach to solving these methods is the shooting method.

In the shooting method, the initial state is propagated in time by applying the relevant control trajectory at the appropriate time steps. The system dynamics used in (2.4), and more generally systems considered in modern applications, have no closed form solution. Thus, numerical methods must be used to approximate the system response. The earliest implementation of solving (2.17) uses Forward Euler integration at a significant time penalty. Consider Fig. 3.1 plotting a computer profiling of the solve time of a Forward Euler single shooting solution, as well as a candidate major iteration. The profiling shows this implementation spends approximately 90% of the wall computational time is spent evaluating the system dynamics for the finite difference sensitivity approximation.

Given this bottleneck, there are two approaches to reducing the computational cost: either compute the system dynamics faster, or reduce the number of times required to compute the system dynamics. Accelerating the computational time of the underlying system dynamics is both application and hardware specific; there is no blanket solution to reducing this computational cost. However, reducing the number of times the system dynamics must be computed can be addressed through various trajectory simulation strategies.

As mentioned, the shooting method is one option for trajectory simulation. However, the inclusion of the terminal state constraint introduces an implicit constant, which changes the IVP to a boundary value problem (BVP), which is similarly well studied in the literature [98].

Two common numerical method families of solving BVP are shooting methods and collocation methods [99]. Similar to IVP simulations, BVP shooting methods propagate

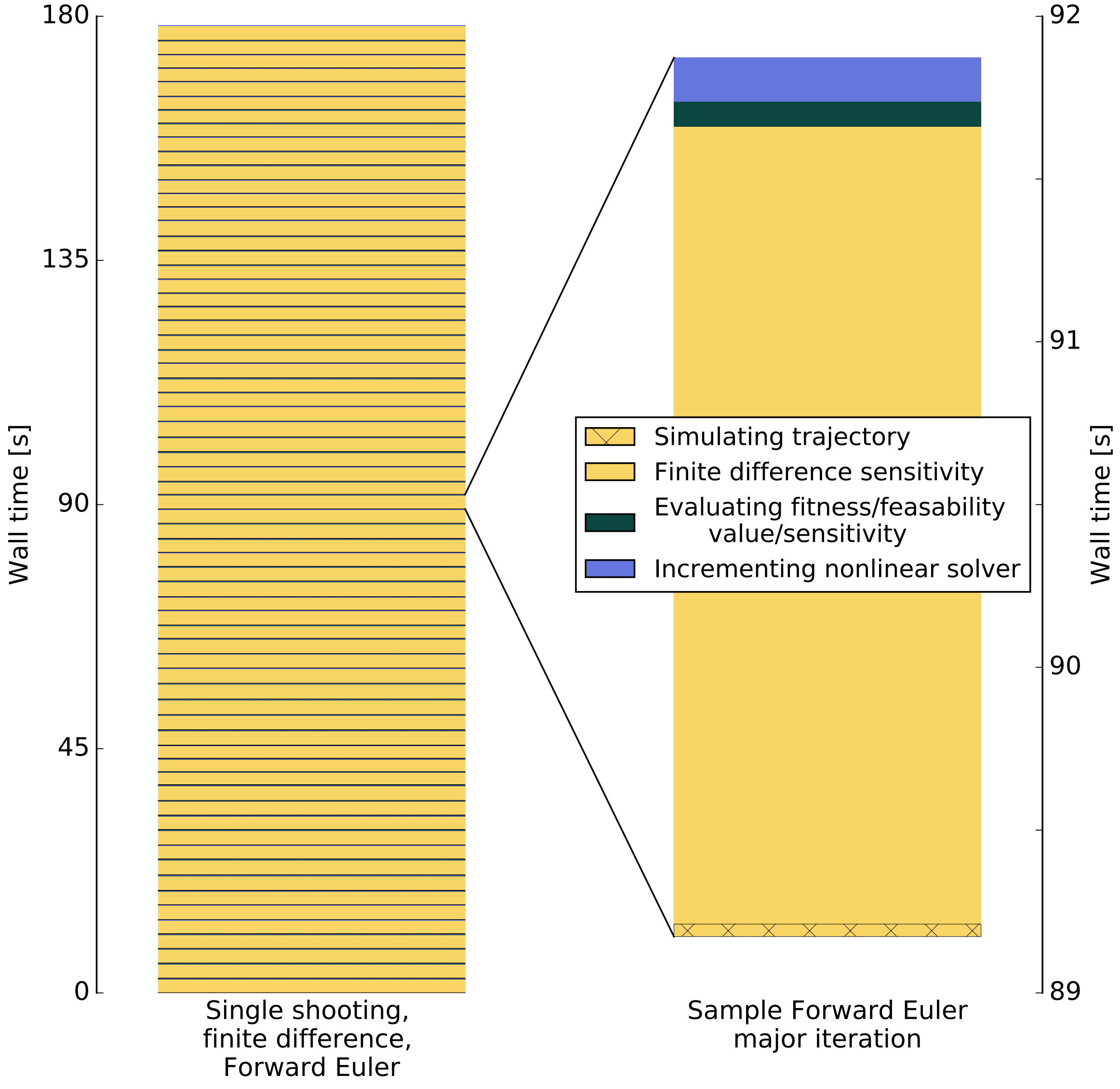


Figure 3.1: In the most rudimentary form, the single shooting, Forward Euler, finite difference sensitivity implementation of the CIS controller is profiled. The left plot shows the approximate 70 major iterations the nonlinear optimizer required to converge, and the right shows the breakdown of a candidate major iteration. The vast majority of the wall time is spent calculating the finite differences for the derivatives, as this requires simulating the trajectory 64 times.

an initial system state through the control trajectory, and conclude convergence when the open loop simulation obeys the terminal state constraint.

In collocation methods, the system response and control trajectories are approximated by some basis function through the time domain, with the system dynamics imposed at the collocation points. The joint system response and control trajectory is consistent when all collocation points simultaneously obey the system dynamic constraints. Using different basis functions and intervals distinguishes different families of collocation methods.

Dynamic programming, a third family, is a procedure to reduce the original BVP into a set of sub-problems through a recursive reduction. By reducing the single comparatively large BVP into multiple consecutive sub-BVPs, the sub-BVPs are comparatively easier to solve, and when stitched together, form a solution to the original BVP. However, dynamic programming suffers from the curse of dimensionality as all possible system state changes for all possible control inputs must be represented within the dynamic program. Thus, dynamic programming quickly becomes computationally infeasible for problems with a large state space, such as in the 3DoF bicycle model.

In the context of reducing the required number of system dynamics evaluations, the distinguishing difference between the numerical methods is the integration accuracy. Taylor series analysis of numerical methods gives the method accuracy order though the truncation error, and for a fixed time step, higher order methods are expected to have a smaller integration error compared to lower order.

By extension, for a fixed integration accuracy, a higher order method is expected to be able to use a larger time step than a lower order method. The trade-off is then higher order methods have fewer integration segments, but are more complex requiring system dynamic evaluations at intermediate corrector points. Thus, to minimize the number of system dynamic evaluations, the proper integration method must balance time step length against integration complexity.

Alternatively, embedded methods are designed to monitor the integration accuracy online, and will reduce the time step as necessary to address a predefined integration resolution. Embedded methods, such as Heun-Euler or Runge-Kutta-Fehlberg, use two integration methods of different order for the same time step [100]. In the case of Heun-Euler, the integration step is calculated once using Forward Euler and again using Heun's Method. If the difference in propagated state is above a resolution threshold, the time step is reduced and method repeated until converged.

By monitoring the integration error and adjusting the time step accordingly, embedded methods can reliably integrate a system to a given tolerance. However, the nature of adjusting the time step during the method introduces some complications for MPC problems. For example, when one integration time step is reduced, it shifts all future discrete integration points backwards in time. This causes challenges for gradient-based optimization solvers because the state sensitivities change discretely. Additionally, if the number of time steps change during the trajectory simulation, it means the computational cost of simulating a fixed prediction horizon is not deterministic. For these reasons, it is advantageous to use a fixed time step.

Establishing the time step *a priori* for an arbitrary nonlinear system is difficult, especially if no integration resolution is provided. Thus, in this work it is proposed to use some established or measurable uncertainty of the system to provide bounds on the integration error. The motivation is that integrating the system dynamics to a resolution finer than the uncertainty in the system parameters provides frivolous insight.

Typically, uncertainties in system parameters, control input, and sensor noise, etc., are modeled as a Gaussian distribution, which can be propagated through the nominal linearized system dynamics to give a transformed Gaussian uncertainty of future states. This breaks down in nonlinear system, even when linearized, if the uncertainties are sufficiently large. Alternatively, bounds can be generated *a priori* using Monte Carlo or other importance sampling methods. By simulating various uncertainties through time, bounds on the state error at each discrete integration point are generated. Using these bounds in state variation, the largest allowable time step for a candidate numerical method is established, and the computational effort for various methods compared.

For the case study CIS problem, uncertainty in the state trajectory is most strongly affected by uncertainty in the coefficient of friction. Advanced path-tracking controllers can require within 2% accuracy of the coefficient of friction to maintain stability, which forms an upper and lower bound for this case study [24].

The system truth is taken as a fine resolution integration of the nominal parameters for the prediction model. The prediction model is used as truth in favor of the plant model, because the prediction model is the only system the MPC has knowledge of, and if the prediction model is of insufficient complexity, then it is possible no integration time step can accurately capture the plant. In the chosen scenarios, ten filtered random control trajectories are applied to random initial states, and the system dynamics are integrated for the various methods using a decreasing integration time step until the trajectory is inside the bounds. The use of ten candidate maneuvers sufficiently covers the design region, yet does not incur excessive computational error. In general, the number of candidate maneuvers must sufficiently explore the design region comparable to the intended application, requiring more samples when systems are subject to multiple sources of uncertainty, forming uncertainty in multiple dimensions. The control trajectories are random in value, but resulting trajectories are filtered to ensure the vehicle dynamics are excited but not excessive and enter inaccurately modeled system regions.

A candidate trajectory, as well as $\pm 2\%$ bounds of road coefficient of friction are shown in Fig. 3.2. The initial yaw rotation is normalized to align the trajectory about the x axis.

From the vehicle architecture in Sec. 2.1.2, the MPC control inputs are set as piecewise constant steering rates for a duration of $t_{\text{update}} = 100$ ms. To appropriately handle the discrete jump in control rate, the maximum integration length is $t_{\text{interval}} = 100$ ms. To find the appropriate control length for a given integration method, the time step is reduced by t_{interval}/k for integer increments of k to divide the interval uniformly. The time step is converged when for a given k, the integration resolution for all candidate control trajectories are inside the uncertainty bounds of the coefficient of friction.

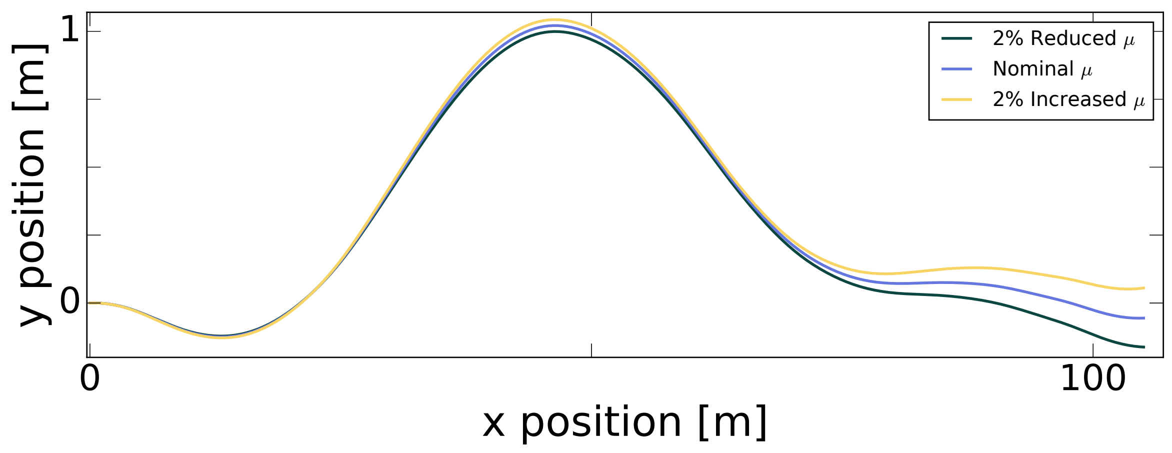


Figure 3.2: The nominal trajectory in blue, as well as +2% coefficient of friction in yellow and −2% coefficient of friction in green. While the state bounds are most obvious in $x - y$ in the figure, the bounds are enforced for all system states.

Recall in Chapter 2 the control commands are issued as two consecutive 50 ms intervals. In Sec. 3.4.2, it is established the most computationally efficient architecture evaluates in 50 ms intervals. The length t_{update} is still retained here to establish the longest allowable time step.

Of the 10 integration methods considered, 6 are explicit and 4 are implicit. The implicit integration schemes are converged within a state tolerance of 10^{-3}. The required time steps for the various methods to achieve the desired accuracy are listed in Table 3.1.

Of the considered methods, linear multistep methods are not included. This is because multistep methods typically employ a similar integration scheme as higher order explicit integration methods, but include the intermediate corrector points as integration points in the solution. One benefit of multistep methods is that by tracking the intermediate stages directly in the solution, the CIS criteria can also be enforced at these points. For the CIS controller, the additional resolution does not improve the performance of the controller, and the additional memory requirements and problem size is detrimental to computational speed.

The tested collocation methods are based on the Gauss-Lobatto quadrature [101].

Table 3.1: Integration Scheme Resolutions

Integration Scheme	**Integration Type**	**Time Step [ms]**
Forward Euler	Explicit	1.41
Heun's Method	Explicit	2.63
Explicit Midpoint Method	Explicit	2.44
Runge-Kutta Fourth-Order	Explicit	50.00
Runge-Kutta 3/8	Explicit	50.00
Backward Euler	Implicit	1.54
Crank Nicolson	Implicit	16.67
3 Point Collocation	Implicit	20.00
4 Point Collocation	Implicit	50.00
5 Point Collocation	Implicit	100.00

Gauss-Lobatto, as opposed to Radau or Laguerre-Gauss, quadrature is chosen based on the consistency constraint enforced at the ends of the intervals.

For this example system and control length t_{interval}, higher order collocation beyond 5 points provides excessive accuracy for the computational cost; but these methods might be useful to other applications. For that reason, higher order collocation based methods, such as spectral and pseudo-spectral, are not considered for this CIS formulation. These higher order methods are often used in trajectory optimization due to the incorporation of the state trajectory into the optimization problem, which is discussed in Sec. 3.3.

As expected, within explicit and implicit methods, the higher order methods allow for a longer time step, and hence fewer integration points in the prediction horizon. However, these higher order methods achieve improved accuracy by evaluating and incorporating the system dynamics at intermediate points in the integration step.

For example, Runge-Kutta Fourth Order (RK4) evaluates the system dynamics at the start of the interval and three corrector points, resulting in a total of four system dynamics evaluations. Yet, based on the previous assumption that solution time can be reduced by minimizing the number of system dynamics evaluations, RK4 is the most efficient considered explicit integration method in Table 3.1. The advantage over

Runge-Kutta 3/8 is apparent in Sec. 3.4 when memory requirements are discussed, along with higher order Runge-Kutta methods.

It is also seen within implicit methods that higher order schemes are more accurate, but distinguishing the advantages of implicit versus explicit are not as clear. While the converged 5 point collocation only requires five evaluations of the system dynamics, the nature of implicit methods require many iterations to converge, with each iteration requiring multiple system dynamics evaluations. Advanced implementations of these implicit methods, such as spectral integration, incorporate the implicit convergence as a part of the optimization problem as opposed to open loop integration of the control trajectory. This improves the convergence rate of the implicit constraints, and hence reduces system dynamics evaluations, but increases the complexity of the optimization problem. For the candidate CIS maneuver, Sec. 3.3 shows implicit schemes excessively grow the optimization problem, thus explicit schemes are favorable.

Based on these observations, RK4 integration is recommended for the CIS controller because it is the most efficient explicit integration method, on par with 4 point orthogonal collocation, and avoids increased complexity in the optimization problem.

Fig. 3.3 compares the wall time of a candidate major iteration for the Forward Euler and RK4 methods. Both methods use finite differences for derivatives, which still require the vast majority of the wall time. However, because RK4 can simulate the trajectory approximately 8 times faster than Forward Euler, both the trajectory simulation and gradient calculation is reduced.

In the next section, gradients of the MPC prediction trajectory are presented for RK4 integration.

3.2 Analytic Derivatives of RK4 Explicit Integration

In the previous section, different integration methods are compared to establish the longest allowable time step to support the same integration accuracy. For the presented CIS application, explicit RK4 integration is chosen as it can minimize the number of

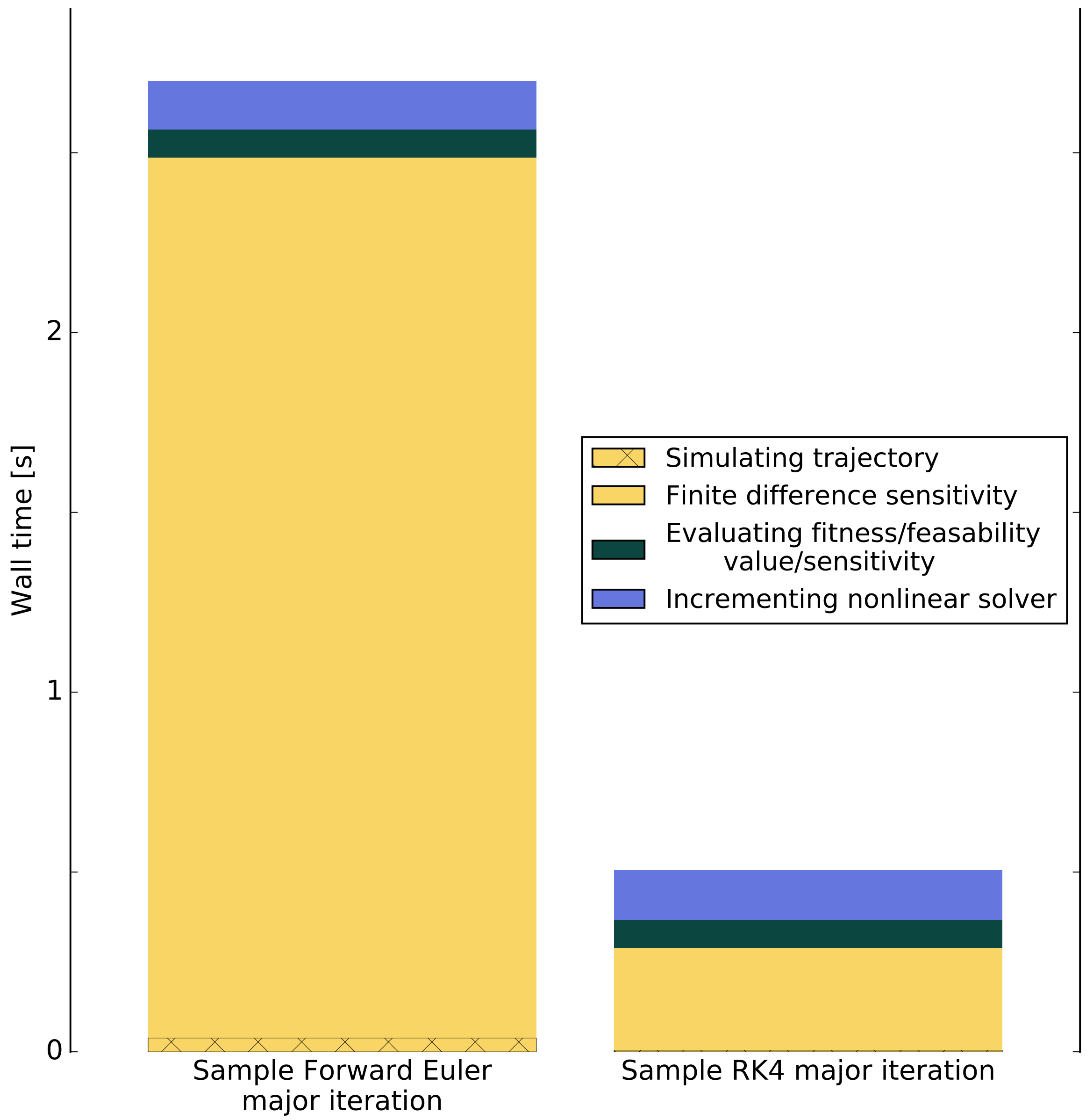


Figure 3.3: The wall time of a candidate major iteration for Forward Euler and RK4 are compared.

system dynamics evaluations for an established normalized integration accuracy.

Within the MPC problem, the controller must map a candidate control trajectory to a corresponding state trajectory, and then evaluate the state trajectory for fitness and feasibility. While the details of evaluating the fitness and feasibility are application dependent, the mapping of the control trajectory to the state trajectory is achieved numerically through the integration.

Numeric optimizers can be categorized as gradient-based or gradient-free. In gradient-based optimizers, a single candidate design point is incremented based on local gradient and curvature to improve the solution. In gradient-free optimizers, multiple design points are morphed based on different design criteria to transition the set of design points into the neighborhood of the optimum before concluding convergence. Gradient-free approaches, such as particle swarm and genetic algorithms, are not dependent on local gradient or curvature, thus not as susceptible to gradient inversion or local minimum. However, gradient-free optimizers can take orders of magnitude longer time to converge and require orders of magnitude more function evaluations compared to gradient-based optimizers [102]. To improve solution time, it is favored to use gradient-based optimization.

By name, gradient-based optimization requires sensitivities of the objective function and constraints to the design variables. For MPC applications, this can be expressed through the chain rule as follows.

$$\frac{dJ}{d\mathbf{u}} = \frac{dJ}{d\mathbf{X}}\frac{d\mathbf{X}}{d\mathbf{u}} \tag{3.1}$$

$$\frac{d\mathbf{g}}{d\mathbf{u}} = \frac{d\mathbf{G}}{d\mathbf{X}}\frac{d\mathbf{X}}{d\mathbf{u}} \tag{3.2}$$

Here, $\mathbf{u}$ is a vector of control inputs, $\mathbf{X}$ is a vector containing the state $\mathbf{x}$ at each of the n integration points, $J(\mathbf{X})$ is the objective function, and $\mathbf{g}(\mathbf{X})$ is a vector of the constraints of the MPC.

The details of evaluating $\frac{dJ}{d\mathbf{X}}$ and $\frac{d\mathbf{G}}{d\mathbf{X}}$ are specific to each MPC problem and there is not a generic strategy to computationally improve this step. Further, of the various MPC applications referred to in Chapter 1, evaluating J, $\mathbf{G}$, $\frac{dJ}{d\mathbf{X}}$, and $\frac{d\mathbf{G}}{d\mathbf{X}}$ is not the most computationally expensive step. In the context of two-level MPC, evaluating J and $\frac{dJ}{d\mathbf{X}}$ results in a quadratic penalty calculation, and linear sensitivity problem, respectively, and is of low computation cost. Specific to the one-level CIS case study, these steps account for 10% to 20% of the time spent evaluating a candidate trajectory. Rather, evaluating $\mathbf{X}$ and $\frac{d\mathbf{X}}{d\mathbf{u}}$ is the most computationally expensive step. Efficient integration schemes to evaluate $\mathbf{X}$ is discussed prior in Sec. 3.1, but it can be computationally beneficial to calculate $\mathbf{X}$ and $\frac{d\mathbf{X}}{d\mathbf{u}}$ concurrently.

There are a few options to evaluate $\frac{d\mathbf{X}}{d\mathbf{u}}$ with differing levels of accuracy and computational effort. In the simplest form, finite differences can be used to approximate the sensitivity of the state trajectory to previous control inputs. However, finite difference sensitivity approximation requires simulating the state trajectory for each of the control inputs, resulting in many system dynamics evaluations. Additionally, the appropriate step size cannot be determined *a priori*, resulting in numerically inaccurate approximations, to be seen in Fig. 3.4.

The complex finite difference approximation uses a step in the complex plane, which can avoid round-off error from subtraction between small numbers [103]. However, this approach requires evaluating the system dynamics, and possibly fitness and feasibility, using a complex value as input, which is not always possible. Further, the complex step still requires $r + 1$ function evaluations to calculate the sensitivities to r inputs.

Alternatively, exact derivatives can be calculated using automatic differentiation of computer simulations [104], [105]. Consider a computer program to perform a calculation using a sequence of trigonometric, exponential, algebraic, etc. functions. In automatic differentiation, a secondary computer program iterates through the primary simulation and calculates the sensitivities of all intermediate stored values to all previous stored values [106]. This produces the sensitivity of all outputs of the primary simulation to both all intermediate calculated values and all initial inputs and values.

While automatic differentiation avoids requiring the user to explicitly program the derivatives, the computational procedure, memory requirements, and sparsity of the derivatives are not necessarily optimum [107]. For example, using neural networks for the system dynamics can require hundreds of thousands of nodes. Automatic differentiation would then generate and retain a hundreds of thousands by hundreds of thousands sensitivity matrix, which can exceed memory allocation limits.

Additionally, automatic differentiation blindly calculates sensitivities and does not inherently recognize a sensitivity value of zero from a sensitivity value as sparse zero. This can result in retaining and calculating many internal sensitivities that do not affect the final sensitivities or sensitivities of interest.

Exact derivatives can also be calculated analytically, but can be challenging and error prone for users to derive and implement. However, for performance critical applications, analytic derivatives are the most computationally and memory efficient. Researchers have shown the family of explicit Runge-Kutta integration can be analytically differentiated [108]. However, this high-level proof does not showcase the sparsity of the sensitivities, nor does it incorporate computer memory considerations. These shortcomings are addressed herein.

The CIS controller in Chapter 2 uses constant control rates over the integration step. Correspondingly, simplifications to the RK4 sensitivities are made using a zero-order control hold as follows.

$$\begin{aligned} \mathbf{x}_{i+1} &= \mathbf{x}_i + \frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4) \\ \mathbf{k}_j &= f(\boldsymbol{\xi}_j, \mathbf{u}_i); \quad \forall j \in [1, 4] \end{aligned} \tag{3.3}$$

where f is the system dynamics of the prediction model and

$$\begin{aligned}
\boldsymbol{\xi}_1 &= \mathbf{x}_i \\
\boldsymbol{\xi}_2 &= \mathbf{x}_i + \frac{\Delta t}{2}\mathbf{k}_1 \\
\boldsymbol{\xi}_3 &= \mathbf{x}_i + \frac{\Delta t}{2}\mathbf{k}_2 \\
\boldsymbol{\xi}_4 &= \mathbf{x}_i + \Delta t \ \mathbf{k}_3
\end{aligned}$$

Calculating the state derivatives in terms of the intermediate $\boldsymbol{\xi}_j$ results in

$$\begin{aligned}
\frac{d\mathbf{x}_{i+1}}{d\mathbf{x}_i} &= \frac{d\mathbf{x}_i}{d\mathbf{x}_i} + \frac{d}{d\mathbf{x}_i}\Big(\frac{\Delta t}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)\Big) \\
&= \mathbf{I} + \frac{\Delta t}{6}\Big(\frac{d\mathbf{k}_1}{d\mathbf{x}_i} + 2\frac{d\mathbf{k}_2}{d\mathbf{x}_i} + 2\frac{d\mathbf{k}_3}{d\mathbf{x}_i} + \frac{d\mathbf{k}_4}{d\mathbf{x}_i}\Big)
\end{aligned} \tag{3.4}$$

where $\mathbf{I}$ is the identity matrix of appropriate size and

$$\begin{aligned}
\frac{d\mathbf{k}_1}{d\mathbf{x}_i} &= \frac{df(\boldsymbol{\xi}_1,\mathbf{u}_i)}{d\boldsymbol{\xi}_1}\frac{d\boldsymbol{\xi}_1}{d\mathbf{x}_i} \qquad (3.5a)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_1,\mathbf{u}_i}(\mathbf{I})\\
\frac{d\mathbf{k}_2}{d\mathbf{x}_i} &= \frac{df(\boldsymbol{\xi}_2,\mathbf{u}_i)}{d\boldsymbol{\xi}_2}\frac{d\boldsymbol{\xi}_2}{d\mathbf{x}_i} \qquad (3.5b)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_2,\mathbf{u}_i}\frac{d}{d\mathbf{x}_i}\big(\mathbf{x}_i+\frac{\Delta t}{2}\mathbf{k}_1\big)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_2,\mathbf{u}_i}\big(\mathbf{I}+\frac{\Delta t}{2}\frac{d\mathbf{k}_1}{d\mathbf{x}_i}\big)\\
\frac{d\mathbf{k}_3}{d\mathbf{x}_i} &= \frac{df(\boldsymbol{\xi}_3,\mathbf{u}_i)}{d\boldsymbol{\xi}_3}\frac{d\boldsymbol{\xi}_3}{d\mathbf{x}_i} \qquad (3.5c)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_3,\mathbf{u}_i}\frac{d}{d\mathbf{x}_i}\big(\mathbf{x}_i+\frac{\Delta t}{2}\mathbf{k}_2\big)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_3,\mathbf{u}_i}\big(\mathbf{I}+\frac{\Delta t}{2}\frac{d\mathbf{k}_2}{d\mathbf{x}_i}\big)\\
\frac{d\mathbf{k}_4}{d\mathbf{x}_i} &= \frac{df(\boldsymbol{\xi}_4,\mathbf{u}_i)}{d\boldsymbol{\xi}_4}\frac{d\boldsymbol{\xi}_4}{d\mathbf{x}_i} \qquad (3.5d)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_4,\mathbf{u}_i}\frac{d}{d\mathbf{x}_i}\big(\mathbf{x}_i+\Delta t\ \mathbf{k}_3\big)\\
&= \frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_4,\mathbf{u}_i}\big(\mathbf{I}+\Delta t\ \frac{d\mathbf{k}_3}{d\mathbf{x}_i}\big)
\end{aligned}$$

Note, expressions $\frac{dk_j}{d\mathbf{x}_i}$ for $j = 2, 3, 4$ require $\frac{dk_{j-1}}{d\mathbf{x}_i}$, which is calculated in the previous step.

Similarly, the sensitivity of the state $\mathbf{x}_{i+1}$ to the control input $\mathbf{u}_i$ is calculated as follows.

$$\begin{aligned}
\frac{d\mathbf{x}_{i+1}}{d\mathbf{u}_i} &= \frac{d\mathbf{x}_i}{d\mathbf{u}_i}+\frac{d}{d\mathbf{u}_i}\Big(\frac{\Delta t}{6}(\mathbf{k}_1+2\mathbf{k}_2+2\mathbf{k}_3+\mathbf{k}_4)\Big) \qquad (3.6)\\
&= \frac{\Delta t}{6}\Big(\frac{d\mathbf{k}_1}{d\mathbf{u}_i}+2\frac{d\mathbf{k}_2}{d\mathbf{u}_i}+2\frac{d\mathbf{k}_3}{d\mathbf{u}_i}+\frac{d\mathbf{k}_4}{d\mathbf{u}_i}\Big)
\end{aligned}$$

where

$$
\begin{aligned}
\frac{d\mathbf{k}_1}{d\mathbf{u}_i} &= \frac{df(\boldsymbol{\xi}_1, \mathbf{u}_i)}{d\boldsymbol{\xi}_1}\frac{d\boldsymbol{\xi}_1}{d\mathbf{u}_i} + \frac{df(\boldsymbol{\xi}_1, \mathbf{u}_i)}{d\mathbf{u}_i} \qquad (3.7a)\\
&= \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_1, \mathbf{u}_i}(\mathbf{0}) + \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{u}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_1, \mathbf{u}_i}\\
&= \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{u}}\Big|_{\mathbf{x}_i, \mathbf{u}_i}\\
\frac{d\mathbf{k}_2}{d\mathbf{u}_i} &= \frac{df(\boldsymbol{\xi}_2, \mathbf{u}_i)}{d\boldsymbol{\xi}_2}\frac{d\boldsymbol{\xi}_2}{d\mathbf{u}_i} + \frac{df(\boldsymbol{\xi}_2, \mathbf{u}_i)}{d\mathbf{u}_i} \qquad (3.7b)\\
&= \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_2, \mathbf{u}_i}\left(\frac{\Delta t}{2}\frac{d\mathbf{k}_1}{d\mathbf{u}_i}\right) + \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{u}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_2, \mathbf{u}_i}\\
\frac{d\mathbf{k}_3}{d\mathbf{u}_i} &= \frac{df(\boldsymbol{\xi}_3, \mathbf{u}_i)}{d\boldsymbol{\xi}_3}\frac{d\boldsymbol{\xi}_3}{d\mathbf{u}_i} + \frac{df(\boldsymbol{\xi}_3, \mathbf{u}_i)}{d\mathbf{u}_i} \qquad (3.7c)\\
&= \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_3, \mathbf{u}_i}\left(\frac{\Delta t}{2}\frac{d\mathbf{k}_2}{d\mathbf{u}_i}\right) + \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{u}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_3, \mathbf{u}_i}\\
\frac{d\mathbf{k}_4}{d\mathbf{u}_i} &= \frac{df(\boldsymbol{\xi}_4, \mathbf{u}_i)}{d\boldsymbol{\xi}_4}\frac{d\boldsymbol{\xi}_4}{d\mathbf{u}_i} + \frac{df(\boldsymbol{\xi}_4, \mathbf{u}_i)}{d\mathbf{u}_i} \qquad (3.7d)\\
&= \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{x}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_4, \mathbf{u}_i}\left(\Delta t\ \frac{d\mathbf{k}_3}{d\mathbf{u}_i}\right) + \frac{df(\mathbf{x}, \mathbf{u})}{d\mathbf{u}}\Big|_{\mathbf{x}=\boldsymbol{\xi}_4, \mathbf{u}_i}
\end{aligned}
$$

By (3.3)-(3.7), the sensitivity of the state at the next explicit integration point to the current state and current input is calculated. Using the chain rule, the sensitivity is carried forward through the prediction horizon to calculate the sensitivity of all future states to all inputs and past states as follows.

$$\frac{d\mathbf{x}_{k+i}}{d\mathbf{u}_k} = \frac{d\mathbf{x}_{k+i}}{d\mathbf{x}_{k+i-1}} \cdots \frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_k} \qquad (3.8)$$

$$
\begin{aligned}
\frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_0} &= \frac{d\mathbf{x}_{k+1}}{d\mathbf{x}_k}\frac{d\mathbf{x}_k}{d\mathbf{u}_0} \\
\frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_1} &= \frac{d\mathbf{x}_{k+1}}{d\mathbf{x}_k}\frac{d\mathbf{x}_k}{d\mathbf{u}_1} \\
&\vdots \\
\frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_{k-1}} &= \frac{d\mathbf{x}_{k+1}}{d\mathbf{x}_k}\frac{d\mathbf{x}_k}{d\mathbf{u}_{k-1}} \\
\frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_k} &= \frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_k} \\
\frac{d\mathbf{x}_{k+1}}{d\mathbf{u}_{k+i}} &= \mathbf{0} \ \ \forall \ \ i \geq 1
\end{aligned}
$$

The sensitivities $\frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{x}}$ and $\frac{df(\mathbf{x},\mathbf{u})}{d\mathbf{u}}$ are based on the prediction model system dynamics. Based on the 3DoF bicycle model system dynamics in (2.4), the sparsity of certain states becomes apparent. This becomes even more significant in the Sec. 3.3 when discussed in the context of collocation and introducing the state trajectory into the optimization problem.

Using analytic derivatives to calculate the state trajectory sensitivity to the control trajectory has a couple benefits. First, the computational time is significantly reduced compared to numeric estimation, such as finite difference or complex finite differences. Second, analytic derivatives are more accurate compared to finite differences, which improves the convergence of the nonlinear optimizer.

To visualize the improved accuracy over finite differences, consider the error of finite differences. Fig. 3.4 plots the error in terminal y position for a candidate lane change maneuver for different control inputs in the prediction horizon.

The difficulty with finite differences can be seen in Fig. 3.4. As the step size decreases, the accuracy improves to a limit. Beyond this limit, the accuracy is reduced due to round-off error in the finite difference. Identifying this optimum step size is further challenging, because it is different for each variable, and for each candidate control trajectory.

To quantify the improvement of analytic derivatives over finite differences, the CIS maneuver is solved using both methods. In each simulation, the total wall time as

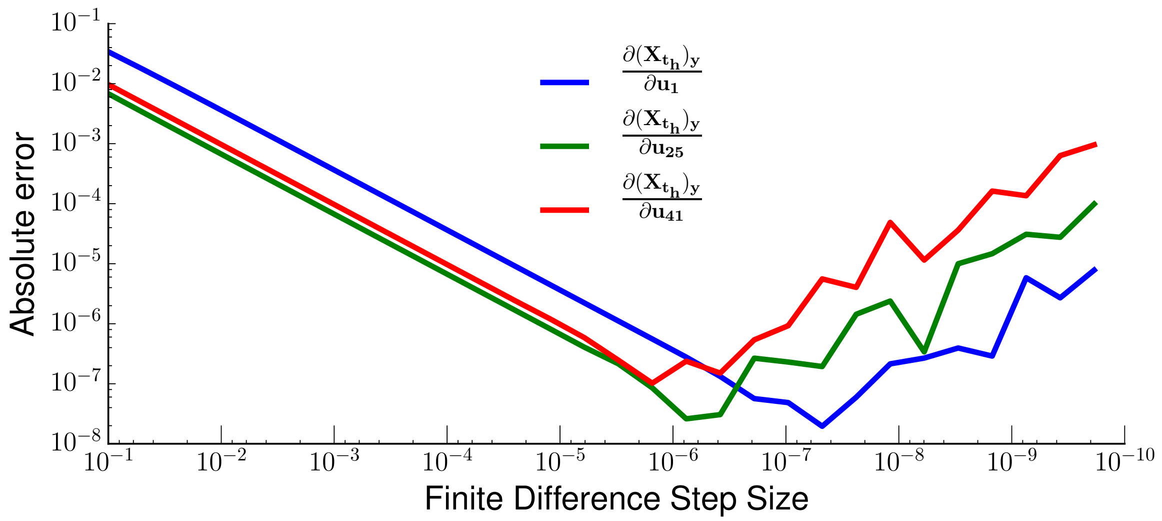


Figure 3.4: The absolute error of the terminal y position based on finite difference approximation for different control inputs in the prediction horizon. The optimal step size cannot be established *a priori*, which results in either too small a step size and inaccurate gradients, or too large a step size and numeric round-off errors.

Table 3.2: Optimization Summary for Candidate CIS Maneuver Using RK4 Single Shooting

Parameter	**Analytic Derivatives**	**Finite Difference**
Solution time [s]	1.46	20.08
Major iterations	56	67
Number of function calls	63	85
Number of sensitivity calls	60	76

well as number of function and sensitivity calls is monitored. The finite differences are solved with a step size of 10^{-4}, as this safely avoids the numeric round-off issues seen in Fig 3.4. Note, this analysis uses single shooting trajectory optimization and does not employ the advancements discussed later in Sec. 3.3. The results of the two experiments are summarized in Table 3.2.

Specific to the CIS controller, the sparsity of the sensitivity of the next integration state $\mathbf{x}_{i+1}$ to the current state $\mathbf{x}_i$ and control input $\mathbf{u}$ for the 3DoF vehicle model can be examined. For the candidate CIS maneuver, the longitudinal velocity is locked, implying that while u is an element in the state vector, the derivative of u_{i+1} to other states is $\mathbf{0}$, and sensitivity of other states to u_i is 0. With this modification, (2.4) is updated and used for the state consistency sensitivity.

Figs. 3.5a - 3.5c show the state-state sensitivity for Forward Euler, RK4, and 5 point collocation. This is not an exact comparison, as Table 3.1 shows the required time steps to achieve a desired resolution. To simulate a 100 ms time step, a single 5 point collocation integration could be used, or two RK4 steps, or about 71 Forward Euler steps.

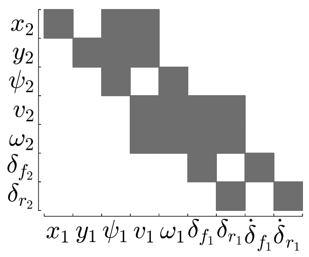
Consider first the Forward Euler sensitivity in Fig 3.5a. The first two states, $[x, y]$, do not affect the system dynamics, but do have an influence on the next state through the identity matrix. However, if the coefficient of friction changed with road position, then the tire forces would be sensitive to the $[x, y]$ position, and the Jacobian would be adjusted accordingly. Because Forward Euler only takes the system dynamics at the starting point, the tire forces do not directly affect the next $[x, y, \psi]$ states, rather only the time derivatives at the next integration point.

In contrast, the RK4 integration uses multiple intermediate corrector points in the integration step. As a result, the $[x, y, \psi]$ states are fully sensitive to the time derivatives, as the $[v, \omega, \delta_f, \delta_r, \dot{\delta}_f \dot{\delta}_r]$ all affect the tire forces.

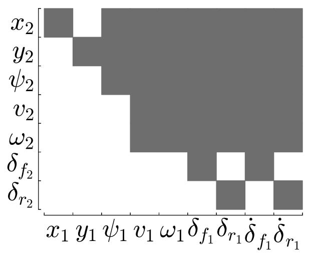
From the perspective of a single integration step, Forward Euler is a more efficient integration method than RK4. However, because Forward Euler requires a significantly finer integration resolution, and hence significantly more integration steps resulting in significantly more segments, the Forward Euler integration is not as computationally efficient.

In the context of the 5 point collocation integration, the nature of the implicit integration scheme is apparent. 5 point Gauss-Lobatto collocation requires implicitly solving three intermediary states as well as the terminal state. Because these must be solved implicitly, there is significant cross sensitivity within the interval, which makes the state consistency constraint comparatively Jacobian dense.

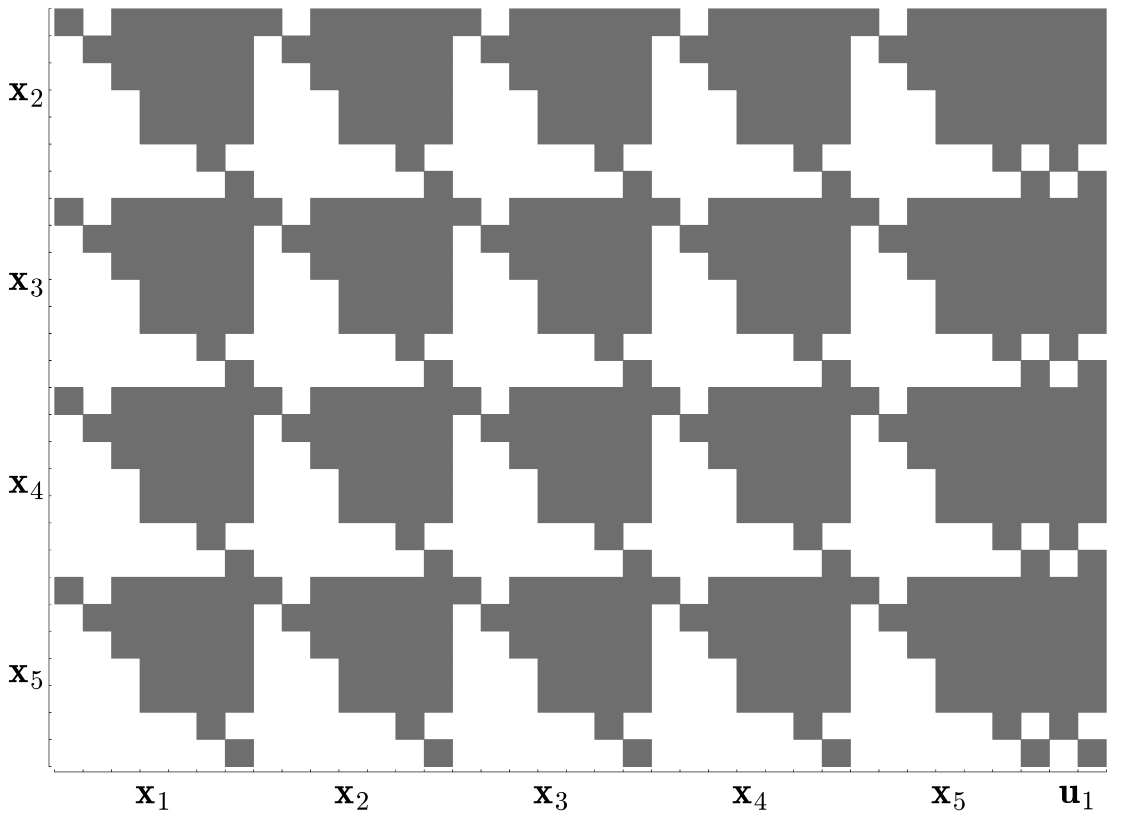
In Sec. 3.1, the implicit states have been solved internal to the integration scheme for comparison in time step length. In practice, these internal states would be mapped



(a) State-state sensitivity for Forward Euler



(b) State-state sensitivity for RK4



(c) State-state sensitivity for 5 point collocation

Figure 3.5: The state-state sensitivities for various integration methods. The implicit nature of collocation becomes apparent from the third subplot: obeying the consistency constraints at the collocation points are fully sensitive to all internal states, as well as the control input.

into the optimization problem to allow the implicit constraints of the integration scheme to be solved within the optimization problem. While this would allow the implicit scheme to converge faster, the additional states and dense sensitivity excessively grows the optimization problem.

In contrast, the intermediate corrector states used in RK4 are generated explicitly when validating the integration scheme and are not part of the optimization problem. Thus, RK4 offers an appropriate balance of complexity and accuracy to facilitate larger integration time steps, but not too complex to incur computational penalties at the optimization problem stage.

By exploiting the sparsity of the RK4 integration, the computational requirement to calculate the sensitivities is reduced, as well as memory dependency. Both factors are important to leverage to achieve real-time performance, showcased in in Sec. 3.4

Based on the comparison between analytic derivatives and finite differences, the two core advantages of analytic derivatives are evident. The improved accuracy of analytic derivatives allows the optimization problem to converge in fewer iterations, which requires fewer function evaluations and sensitivity calls. Based on the profiling discussed in Sec. 3.1, evaluating the system dynamics and derivatives is the most computationally expensive stage; hence reducing the number of function calls and sensitives improves wall time. Further, analytic derivatives are an order of magnitude faster than finite differences, which again reduces wall time.

Fig. 3.6 shows the wall time to solve the CIS controller using a single shooting, analytic derivatives, RK4 based CIS controller. Compared to the simple implementation depicted in Fig. 3.1, analytic derivatives converge in fewer major iterations, and within each major iteration, the gradient calculation time is improved.

A secondary improvement in gradient accuracy comes from the Hessian. In the prior optimization profilings, both analytic and finite difference based optimization use a numerically approximated Hessian generated from the sensitivities. As a result, the finite difference solution has a compounding error in the Hessian, whereas the analytic derivatives have exact gradients and a single numeric error step in the Hessian.

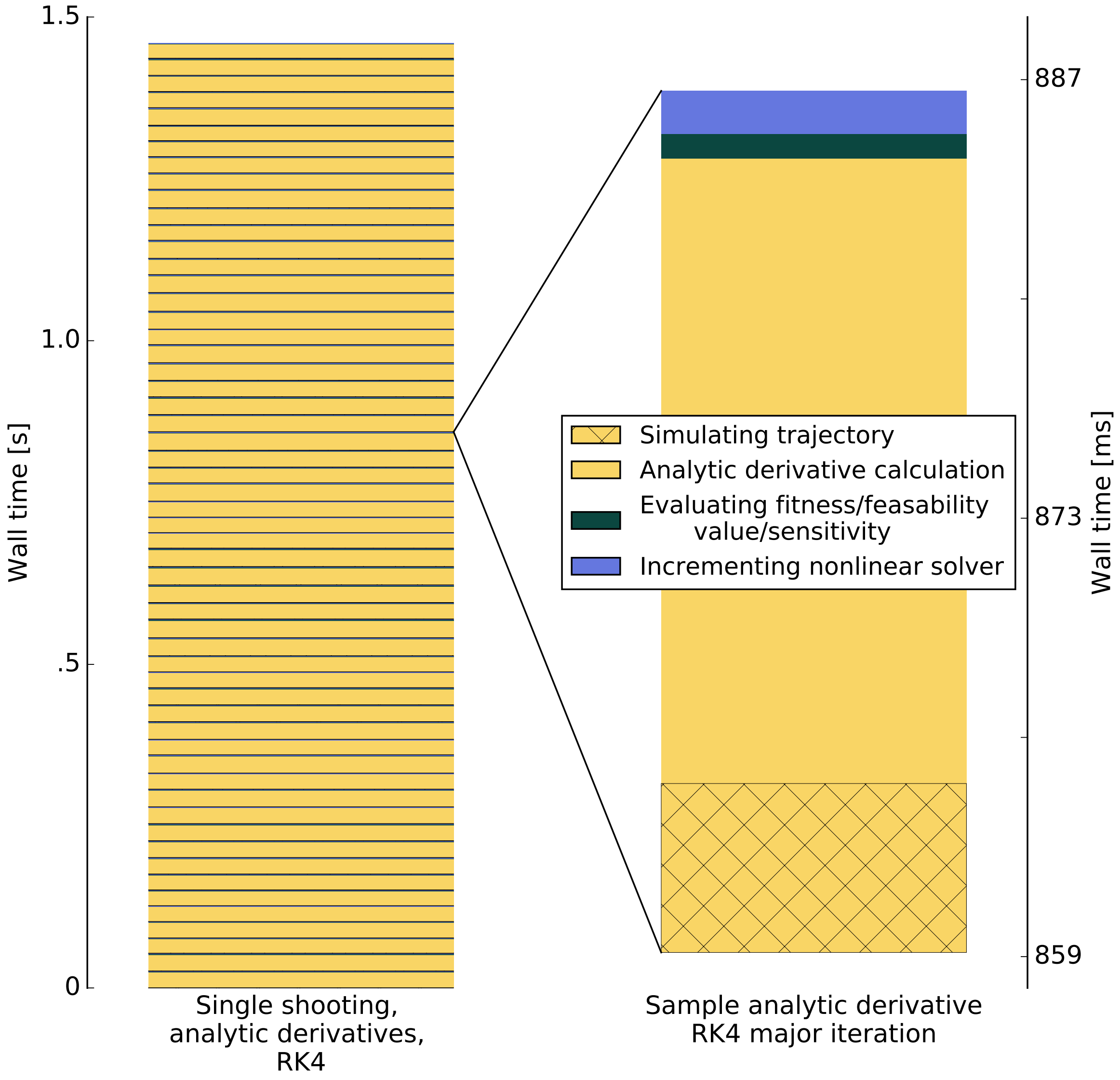


Figure 3.6: The wall time to solve the single shooting, analytic derivatives, RK4 based CIS controller, and a candidate major iteration are plotted.

It is feasible to derive an analytic Hessian of the prediction state trajectory to the control trajectory, again leveraging the sparsity structure [109]. However, Hessian matrices are typically significantly larger, containing approximately squared the number of elements as the Jacobian, depending on the sparsity. Numerical optimizers do not always update or adjust the Hessian at every major iteration, thus it is not clear that analytically calculating the Hessian is advantageous over an occasional numerically approximated Hessian.

Overall, introducing analytic derivatives is an order of magnitude faster wall time improvement over finite difference, and an additional approximately 20% faster convergence rate within the optimization problem. Further, analytic derivatives can exploit the sparsity of the Jacobian, which reduces the solve time for the optimizer's linear sub-problem in between each major iteration [110] [111] [112], providing an advantage over automatic differentiation. However, the current wall time of approximately 1.5 s is still prohibitive of real-time performance, requiring modifications to the trajectory optimization structure. In the next section, various trajectory optimization strategies are considered, within which the sparsity of the Jacobian is examined in depth.

3.3 Trajectory Optimization Structure

In the previous section, the benefits of analytic derivatives over finite differences are shown in an example CIS optimization. However, the experiment uses a single shooting trajectory optimization, which is not necessarily the most efficient approach.

In single shooting trajectory simulation, the state trajectory is found by numerically integrating the system dynamics according to the control trajectory. The optimization problem is then structured such that for the given starting state, the control trajectory is adjusted based on some fitness and feasibility criteria until optimally converged. However, single shooting trajectory simulation suffers from high sequential dependency. That is, to find the final state, the previous state must be calculated, which requires the state prior, and so forth, all the way to the initial state. As a result, simulating the state trajectory cannot be effectively parallelized. It might be possible to parallelize evaluating the system dynamics for an arbitrary test case, but the CIS dynamics vector (2.4) is not suitable.

Additionally, single shooting methods suffer from large discrepancies in state sensitivities. The sensitivity of the terminal state to early control inputs can be orders of magnitude larger than the sensitivity to later control inputs. This can be challenging for the numerical optimizer because small changes to the initial control inputs are more effective at addressing the terminal state constraint than adjustments late in the trajectory, which makes the optimization problem more difficult to converge.

In contrast, collocation methods introduce the state trajectory as part of the optimization problem. For these implicit methods, both the control input and integration states are introduced as design variables, but doing so adds an additional consistency constraint that the integration states obey the system dynamics for the applied control input. By directly incorporating the state trajectory as part of the optimization problem, the optimizer has more control of the vehicle response. In the case of single shooting, the state trajectory can only be evaluated by integrating the nonlinear dynamics through the entire maneuver. In contrast, collocation methods can directly modify the candidate state trajectory to address the fitness and feasibility criteria, and then simultaneously make corrections to both the state trajectory and control trajectory to validate the consistency constraints.

Thus, a conundrum arises between balancing the integration method and trajectory simulation method. In Sec. 3.1 it has been established that explicit RK4 integration can integrate the control trajectory with fewer system dynamics evaluations than other explicit and implicit methods considered. However, introducing the state trajectory into the optimization problem reduces the complexity of the problem, allowing the optimizer to converge faster.

To balance these two benefits, multiple shooting trajectory simulation is introduced [113]. In multiple shooting trajectory simulation, the state trajectory is broken into q segments, and the starting state of each segment is introduced into the optimization problem. The first segment's starting state is dictated by the starting state of the plant system, but the starting states of the remaining segments are design variables. Similar to collocation based trajectory simulations, the multiple shooting method introduces a consistency constraint such that the end of one segment must exactly match the start of the next segment. For the last segment, the final state must obey the terminal state constraint by the fitness and feasibility criteria. Traditional single shooting trajectory simulation is a special case of multiple shooting simulation when only 1 segment is used.

The state consistency constraint for an arbitrary integration procedure, F, is expressed as follows.

$$\mathbf{g}_{\text{cons}} = \mathbf{x}_i + F(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}, \mathbf{u}_{i+1}, \Delta t) - \mathbf{x}_{i+1} \forall i \in [1, q-1] \tag{3.9}$$

In addition to the state consistency constraint for each segment, the feasibility constraints are applied to each segment as well. While the MPC constraints in (2.17) are applied to the whole trajectory, q-$\mathbf{g}_{\text{seg}}$ constraints are introduced, with one for each segment. As a result, if each of the q segments obeys the feasibility constraints, then the entire trajectory is feasibly by (2.17). The fitness criteria is dependent on the whole trajectory, thus a single fitness criteria is retained.

Introducing multiple shooting has three core benefits. First, the starting state of each segment is introduced into the optimization problem, directly giving the optimizer partial control of the state trajectory. Second, by reducing the trajectory length into shorter segments, each segment's terminal state sensitivity to control inputs within the segment is more consistent in magnitude, avoiding the compounding error seen in single shooting. Third, the optimizer establishes the starting state for each segment at the same time, thus when evaluating a candidate design point for fitness and feasibility, each segment can be simulated in parallel.

The above third point, allowing parallel simulation, is critical to addressing the computational wall time. While adding direct control of the state trajectory into the

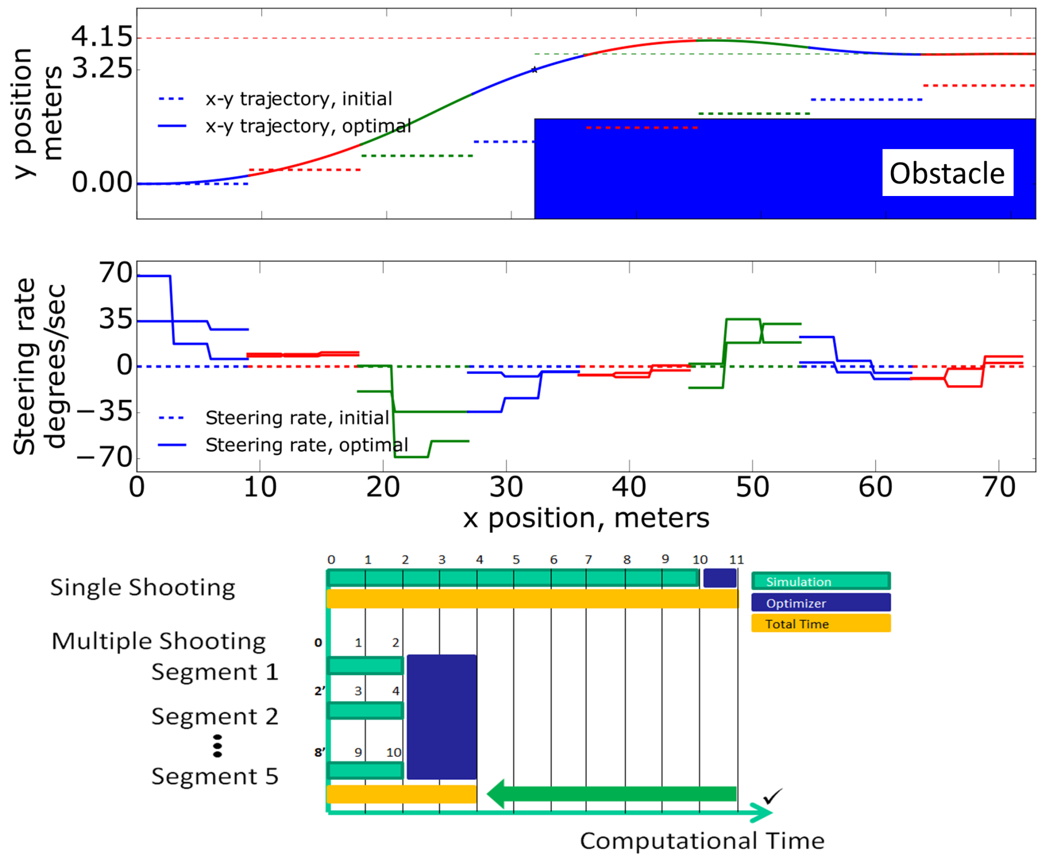


Figure 3.7: In multiple shooting, the sequential dependency of single shooting is reduced by simulating the multiple segments in parallel. However, the introduction of consistency constraints grows the optimization problem, requiring more time to converge on a solution.

optimization problem makes the problem easier to converge, it also grows the optimization problem size. As the number of q segments grows, simulating the trajectory in parallel becomes more efficient, but adds more consistency constraints. An example of this trade-off can be seen in Fig. 3.7 showing multiple shooting for a straight road lane change. Here, the segments can be simulated in parallel for reduced wall time, but the increase in consistency constraints grows the time required to converge the optimization problem.

It is also important to note that in Fig. 3.7, the initial candidate design points is infeasible, as the state trajectory has step discontinuities in y. However, through the

Table 3.3: Solve Time Versus Number of Segments in Multiple Shooting.

Number of Segments	**Optimization Wall Time [s]**
1 segment	1.60
2 segments	0.82
3 segments	0.48
4 segments	0.35
5 segments	0.21
6 segments	0.24
8 segments	0.26
10 segments	0.37
15 segments	0.44
30 segments	0.52

consistency constraints, the converged solution obeys a continuous state trajectory.

While increasing the number of segments improves the parallel computation of the state trajectory by allowing more parallel branch instances, this only decreases the computational wall time to a limit as the optimization time begins to grow. The optimal number of segments in multiple shooting cannot be determined *a priori*. Therefore, tests are performed to seek the optimal balance.

Table 3.3 investigates the number of segments versus simulation time for the CIS controller. Simulating the state trajectory is done in parallel using a custom CUDA system dynamics implementation, to be discussed in Sec. 3.4, and the optimization problem is again solved using IPOPT [83].

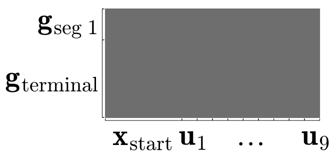
From the simulation times in Table 3.3, it is apparent as the number of segments increases, the wall time initially decreases, then starts to increase. The minimum wall time is around 5 segments of length 6 control inputs each. This trend in wall time showcases the trade-off in efficiency of simulating the state trajectory against growing the optimization problem size.

The effect of varying segment lengths on the optimization problem structure can be seen in Figs. 3.8a - 3.8c. For these figures, a normalized 9 control input trajectory is plotted for comparative purposes between figures. In Fig. 3.8a, the single shooting trajectory Jacobian is plotted. While single shooting trajectory simulation has comparatively fewer dense elements, simulating the entire trajectory cannot be parallelized, thus not as computationally efficient. In Fig 3.8b, a 3 segment multiple shooting Jacobian is plotted, with each segment containing 3 control inputs. While there are more dense elements than the single shooting, the state trajectory is calculated in parallel across three independent threads. Finally, in Fig 3.8c, a 9 segment multiple shooting Jacobian is plotted, with each segment containing a single control input. In Figs. 3.8b and 3.8c, the impact of the consistency constraint (3.9) can be seen in the off diagonal grey blocks, representing a negative identity matrix.

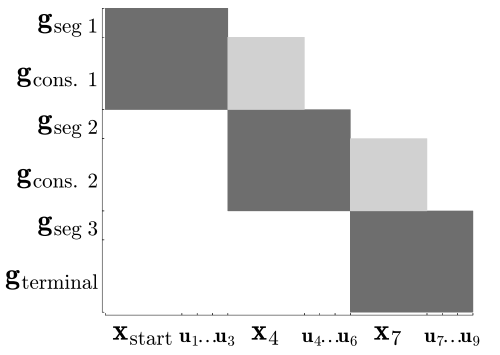
The diagonal Jacobian structure in Fig. 3.8c has a subtle difference compared to the 3 segment formulations. In the 9 segment formulation, each segment has a control input of length 1, suggesting the structure of $\mathbf{x}_i$ $\forall i \in [1, 9]$ defines the entire state trajectory $\mathbf{X}$. In this special case, the fitness and feasibility of the MPC problem can entirely be evaluated by the provided state trajectory, and the control trajectory is only used to validate the consistency constraints. This can be seen in the Jacobian structure where the constraints $\mathbf{g}_{\text{seg}}$ are only sensitive to the states, and the consistency constraints $\mathbf{g}_{\text{cons}}$ are sensitive to $\mathbf{x}_i$ and $\mathbf{u}_i$.

This special case of multiple shooting allows for a unique formulation in evaluating the $\mathbf{g}_{\text{seg}}$ and $\mathbf{g}_{\text{cons}}$ constraints in parallel. Specifically, each segment can be evaluated in parallel, and within each segment, evaluating $\mathbf{g}_{\text{seg}}$ and $\mathbf{g}_{\text{cons}}$ can be done in parallel. Additionally, when there is only one state in the segment, there is no need to implement a constraint aggregation technique; rather, the single calculated value is used directly.

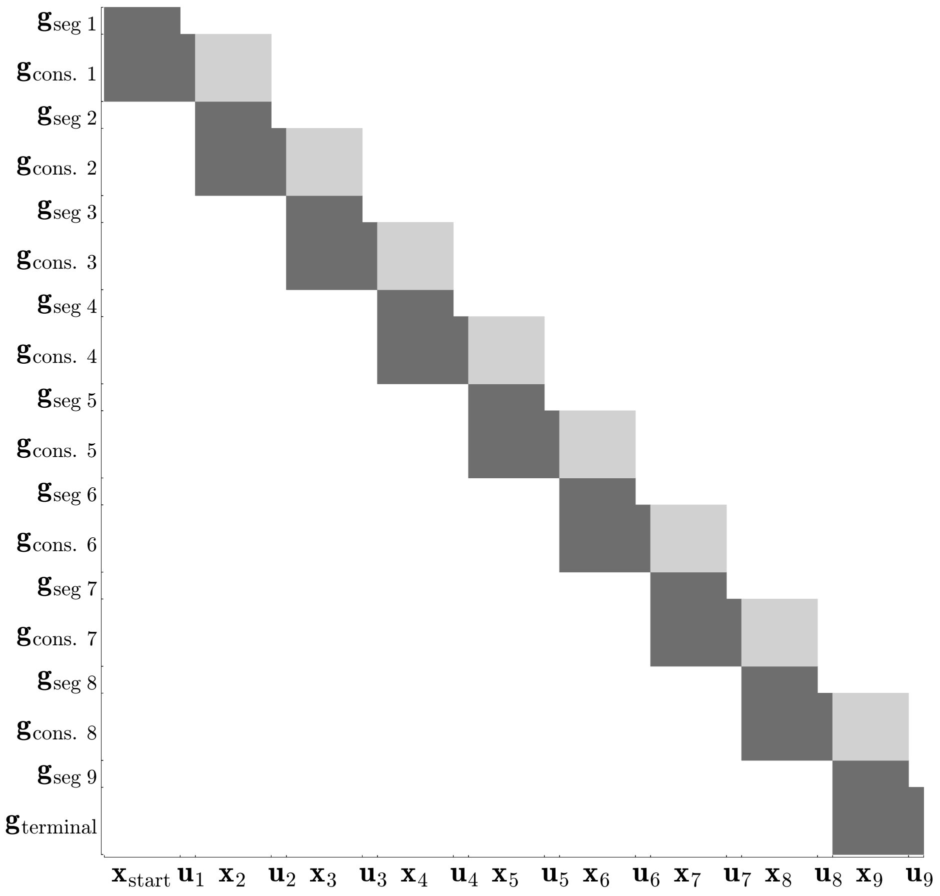
By evaluating the segment feasibility and consistency constraints in parallel, reducing the constraint aggregation complexity, and refining the Jacobian structure, the n-segment multiple shooting can be re-evaluated to exploit these benefits, which is not seen in Table 3.3. Further, this special formulation can more efficiently utilize the parallel hardware, as explored in the next section.


$\mathbf{g}_{\text{seg 1}}$
$\mathbf{g}_{\text{terminal}}$
$\mathbf{x}_{\text{start}}$ $\mathbf{u}_1$ … $\mathbf{u}_9$

(a) Single Shooting Jacobian


$\mathbf{g}_{\text{seg 1}}$
$\mathbf{g}_{\text{cons. 1}}$
$\mathbf{g}_{\text{seg 2}}$
$\mathbf{g}_{\text{cons. 2}}$
$\mathbf{g}_{\text{seg 3}}$
$\mathbf{g}_{\text{terminal}}$
$\mathbf{x}_{\text{start}}$ $\mathbf{u}_1..\mathbf{u}_3$ $\mathbf{x}_4$ $\mathbf{u}_4..\mathbf{u}_6$ $\mathbf{x}_7$ $\mathbf{u}_7..\mathbf{u}_9$

(b) 3 Segment Multiple Shooting Jacobian


$\mathbf{g}_{\text{seg 1}}$
$\mathbf{g}_{\text{cons. 1}}$
$\mathbf{g}_{\text{seg 2}}$
$\mathbf{g}_{\text{cons. 2}}$
$\mathbf{g}_{\text{seg 3}}$
$\mathbf{g}_{\text{cons. 3}}$
$\mathbf{g}_{\text{seg 4}}$
$\mathbf{g}_{\text{cons. 4}}$
$\mathbf{g}_{\text{seg 5}}$
$\mathbf{g}_{\text{cons. 5}}$
$\mathbf{g}_{\text{seg 6}}$
$\mathbf{g}_{\text{cons. 6}}$
$\mathbf{g}_{\text{seg 7}}$
$\mathbf{g}_{\text{cons. 7}}$
$\mathbf{g}_{\text{seg 8}}$
$\mathbf{g}_{\text{cons. 8}}$
$\mathbf{g}_{\text{seg 9}}$
$\mathbf{g}_{\text{terminal}}$
$\mathbf{x}_{\text{start}}$ $\mathbf{u}_1$ $\mathbf{x}_2$ $\mathbf{u}_2$ $\mathbf{x}_3$ $\mathbf{u}_3$ $\mathbf{x}_4$ $\mathbf{u}_4$ $\mathbf{x}_5$ $\mathbf{u}_5$ $\mathbf{x}_6$ $\mathbf{u}_6$ $\mathbf{x}_7$ $\mathbf{u}_7$ $\mathbf{x}_8$ $\mathbf{u}_8$ $\mathbf{x}_9$ $\mathbf{u}_9$

(c) 9 Segment Multiple Shooting Jacobian

Figure 3.8: Various multiple shooting structures. Dark grey blocks represent a fully dense Jacobian structure for an arbitrary system dynamics. Light grey blocks represent the negative identity matrix.

3.4 Parallel Hardware: GPU Acceleration through CUDA

NVidia's CUDA programming language allows for custom programming to run on NVidia based GPU hardware [114]. At a silicon level, GPU architectures are designed for simplistic massively parallel applications, in contrast to CPUs, which are optimized for complex serial tasks. While GPUs can support higher floating point operations per second (FLOPS), the application must directly support and map to massively parallel hardware [115], [116], [117].

At a high level, the CUDA programming language mimics the C programming language, but carries certain intrinsic variables to support parallel programming. The core concept with parallel computing is all threads evaluate the same code, but due to the intrinsic variables unique to each thread, the relevant data that is operated on can vary. For example, the instruction `float x_position = x_states[theadIdx.x]` can be used to locally load the vehicle x position into that thread's register. As all threads load their local vehicle state, the same system dynamics are evaluated embarrassingly parallel across all threads.

A fundamental difference between GPU and CPU architectures is in the scaling of thread count for problems. For example, CPUs commonly support 4 to 16 threads [118], whereas NVidia GPUs support parallel execution of order hundreds to thousands of threads [119]. If multiple shooting trajectory simulation is mapped into the parallel thread architecture efficiently, this allows a GPU to simulate all segments of the multiple shooting in parallel.

To map the multiple shooting trajectory into GPU hardware requires an additional step in understanding how GPU threads are evaluated in parallel. For improved memory and computational efficiency, CUDA leverages thread blocks. Thread blocks are a collection of threads and within a thread block, threads can access common shared memory and synchronize within themselves. By extension, thread blocks cannot efficiently share memory or synchronize with other thread blocks, and incur a comparatively significant computational penalty when required to do so. One option of simulating the multiple shooting trajectory optimization in parallel is to assign each block to its own segment.

3.4.1 Block Parallel Multiple Shooting

When designing a CUDA parallelized application, the number of thread blocks and number of threads within each block must be explicitly stated. In the block parallel implementation, there are q blocks for the q segments, and sufficient threads within each block to simulate the segment. Because the thread blocks are evaluated in parallel, each segment of the multiple shooting is evaluated for fitness and feasibility in parallel.

To map the optimization problem to a parallel hardware, the design variables and constraint vector are carefully structured. Consider a generic multiple shooting structure of q segments with r control inputs within each segment. The design variables can be ordered as $[\mathbf{x}_{\text{start}}, \mathbf{u}_1, ..., \mathbf{u}_r, \mathbf{x}_2, \mathbf{u}_{r+1}, ..., \mathbf{u}_{2r}, ..., \mathbf{x}_q, ..., \mathbf{u}_{qr}]$. Using this format, $(\texttt{blockIdx.x})_i$ knows it represents segment i, and can access the starting state at index $i(\mathbf{x} + r\,\mathbf{u})$, with the following $r\,\mathbf{u}$ indices corresponding to the control inputs. Additionally, the constraint vector is structured as $[\mathbf{g}_{\text{seg }1}, \mathbf{g}_{\text{cons. }1}, ..., \mathbf{g}_{\text{seg q}}, \mathbf{g}_{\text{terminal}}]$. This allows segment i to store the feasibility constraints starting at index $i(\mathbf{g}_{\text{seg}} + \mathbf{g}_{\text{cons}})$ and state consistency constraints starting at $i(\mathbf{g}_{\text{seg}} + \mathbf{g}_{\text{cons}}) + \mathbf{g}_{\text{seg}}$.

Additionally, CUDA support atomic functions, which allow independent threads to access and modify a common variable without causing conflicts between competing threads. Recall the objective function in (2.17) is to minimize the largest tire slip experienced in the trajectory. This would require first calculating the peak tire slip within each segment, then synchronizing and comparing across segments. To avoid the computational penalty of synchronizing blocks, independent blocks can increment the KS function of slip through atomic functions, hence still minimizing the peak tire slip experienced in the maneuver.

To calculate the sensitivities of the q-$\mathbf{g}_{\text{seg}}$ constraints, analytic derivatives are implemented in parallel within each segment using appropriate indexing. When implementing the analytic derivatives for the specific system dynamics, the sparsity of the Jacobian should be retained when indexing the sensitivity vectors, as this reduces the amount of data required to be generated and transferred from the GPU. Exploiting the sparsity of the Jacobian reduces the memory size, and significantly accelerates sparse linear algebra solvers used in IPOPT.

One bottleneck to this block parallelization remains, namely, the system dynamics. Although the blocks are evaluated in parallel, within each block the system dynamics cannot be evaluated in parallel. Instead, a single thread calculates the system dynamics at the current state, and then broadcasts the results to the other threads within block. The remaining linear algebra can be evaluated in parallel, such as sensitivities to the design variables and linear algebra steps required in Sec. 3.2.

Avoiding this thread bottleneck is critical to maximizing the performance of the GPU hardware. GPUs are able to achieve high FLOPS by fully leveraging all threads at all times. By having to restrict the blocks to a single thread for system dynamics, which as discussed is the largest computational penalty in the optimization, the computational efficiency of the GPU is significantly reduced.

To avoid this bottleneck, it is possible to implement a thread parallel multiple shooting as discussed next.

3.4.2 Thread Parallel Multiple Shooting

The thread parallel multiple shooting is specific to the highest segment count multiple shooting, where for n integration states in the prediction horizon there are n segments. In thread parallel multiple shooting, each thread is structured to evaluate a single integration point for fitness and feasibility. While these threads do get coupled into thread blocks, the mapping does not leverage the benefits of a thread block. However, by massively parallelizing the segments at the thread level, the bottleneck seen in block parallelization can be reduced by keeping all threads active as discussed next.

A core limitation to the size and scope of the threads and thread blocks is the available memory. While there is sufficient shared memory to implement the block parallelization structure, the thread parallelization can be limited by the per thread memory restrictions. The results showcased in this paper are run on an NVidia Pascal microarchitecture, which supports up to 255 32-bit registers per thread [114]. For thread parallelization, this means the entire RK4 integration step, as well as fitness and feasibility evaluations and sensitivities, must be optimized for 255 registers.

It is feasible that higher order Runge-Kutta integration schemes could be more efficient. Fifth order and higher Runge-Kutta schemes exist, but these methods exceed memory availability. In RK4, the states at the intermediate corrector points, as well as sensitivities, must be fully contained within the per thread register limit. However, the RK4 Butcher tableau has diagonal structure [120], meaning each corrector point is a calculated exclusively by the prior integration point. As a result, the intermediate corrector points can be over written, reducing the maximum number of registers used at any one time. For this reason, RK4 outperforms RK 3/8, as RK 3/8 method retains all prior corrector points when calculating the next intermediate point, as well as final integration point.

For the 3DoF bicycle model, retaining these calculations and sensitives exceed the register count for fifth order and above Runge-Kutta methods, which incurs significant memory access penalties. However, it is feasible that for other applications with simpler system dynamics higher order explicit methods could be implemented, or in the case of applications with more complicated system dynamics only second or third order explicit methods compatible with the available memory.

Using the thread parallelized RK4 implementation, with all the other computational improvements discussed, achieves 55 ms convergence time for the nonlinear CIS problem considered. However, the tested implementation uses a slightly different control structure of constant 50 ms control rates, instead of 100 ms, as this allows each of the n segments defined by its own $[\mathbf{x}, \mathbf{u}]$. It is possible to implement constraints within

IPOPT as $\mathbf{u}_{2i} = \mathbf{u}_{2i+1} \forall 2i + 1 \leq n$, which would require each the control action $\mathbf{u}$ to be equal within each pair.

By following the procedure laid out in previous sections, the final joint software-hardware solution thus achieved the target loop timing and is compatible for real-time performance. In Fig. 3.9, the wall time for IPOPT to converge, as well as a candidate major iteration, are plotted. Within the major iteration, the parallel computing nature is apparent as each thread can simulate the trajectory, sensitivity, and fitness/feasibility metrics in parallel, reducing net elapsed time. However, the increased size of the optimization problem becomes apparent as a larger ratio of time is spent within IPOPT iterating the candidate design point.

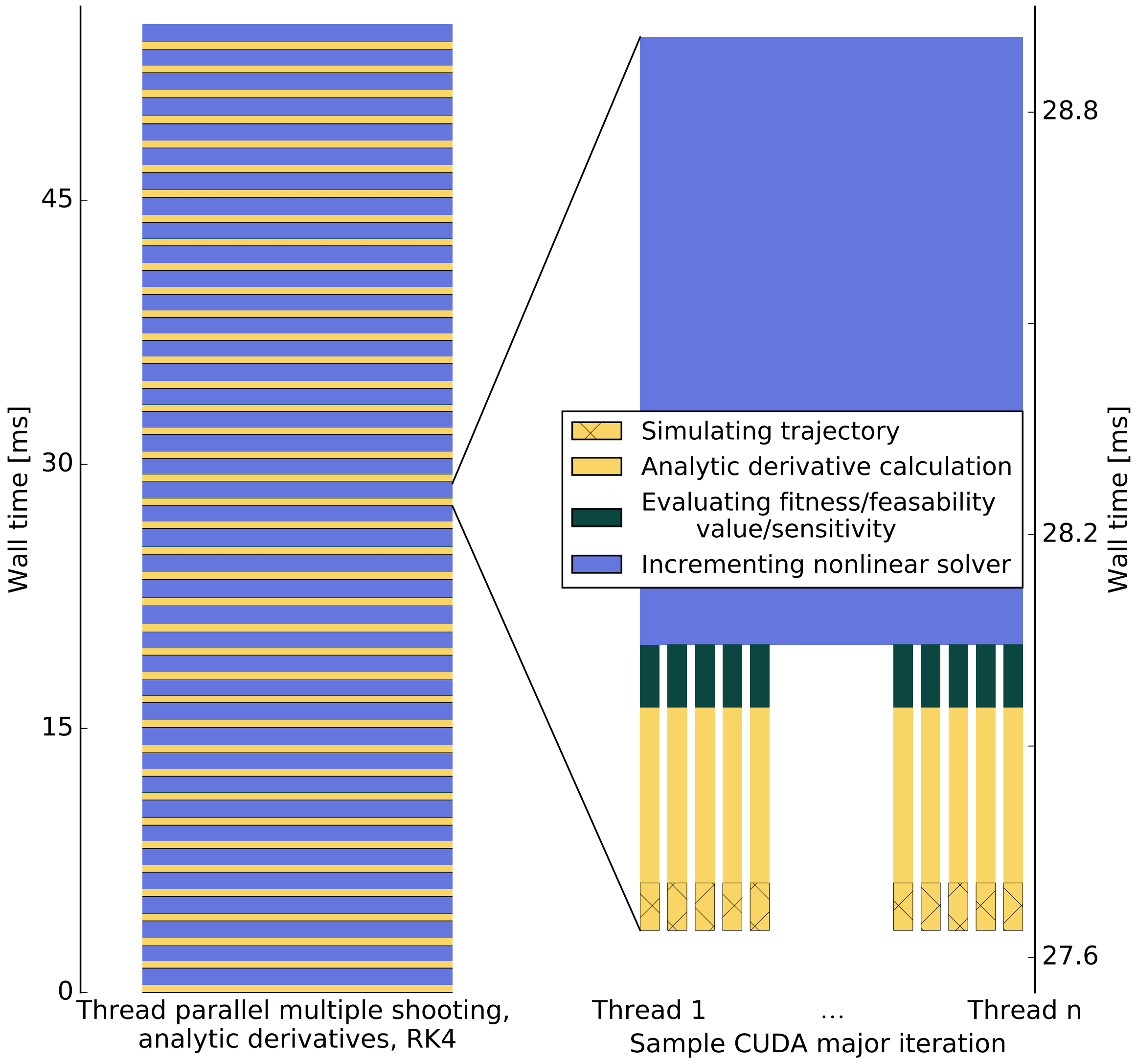


Figure 3.9: The elapsed time for the thread parallel, analytic derivative, RK4 based CIS controller solution is plotted, as well as a candidate major iteration.

Chapter 4

Summary and Outlook

This dissertation presents a CIS controller to perform the evasive lane change maneuver in safety critical situations. As part of the underlying optimization formulation, feasible solutions are guaranteed to obey hard safety constraints, ensuring the prescribed control actions are provably safe. Additionally, the controller is developed to be compatible with an evaluation test vehicle, requiring holistic assumptions on the available perception system and compute power. The resulting formulation is experimentally evaluated in a simplified test case, green-lighting further testing. The contributions made during development are further highlighted as follows.

4.1 Dissertation Contributions

While pursuing the research objectives discussed in Sec. 1.4, the following salient contributions to the field of automotive collision avoidance controllers are made.

4.1.1 One-level Nonlinear MPC Controller Design

The one-level nonlinear MPC CIS controller developed is capable of safely performing an evasive lane change maneuver, even at the limits of handling if needed. While other controllers address collision avoidance for their intended application, these controllers are not designed for a jointly one-level, nonlinear, MPC formulation with hard safety constraints, even extending to the nonlinear limits of handling, for the expected highway scenario. Often, obstacle avoidance controllers are structured as a two-level architecture, tasked with separate path generation and path following controllers. In the context of a safety critical application that might require operating at the vehicle limits of handling, it is not feasible to design a reference trajectory that maximizes the vehicle handling capabilities without exceeding them. Further, it is challenging to design a path following controller with hard safety guarantees while operating at the limits of handling.

To this extent, the safety considerations of an evasive lane change, as well as vehicle stability guarantees, are formulated as hard constraints in a one-level architecture. The one-level architecture directly maps the control actions to a predicted vehicle response trajectory, and adjusts the control actions to ensure the response obeys hard safety constraints. Further, this nonlinear formulation accounts for nonlinearities in the safety constraints, as well as vehicle and tire models. While computationally challenging, the reduced order nonlinear prediction model sufficiently describe the higher order plant model, successfully allowing evasive lane changes at the limits of handling in simulation tests.

4.1.2 Window of Feasibility for CIS

For the numerically simulated cases, it is shown the CIS controller can perform an evasive lane change in roughly half the distance required for limit braking. Hence, if for some reason the host vehicle detects a forward collision cannot be avoided by braking alone, potentially due to a third party collision ahead or a vehicle unsafely pulling into the host's lane, it is possible that an evasive lane change could safely avoid the collision. Further, if the obstacle is identified far away and controller solved quickly, the lane change can take place at low vehicle excitation. But, if the obstacle is near, the formulation is capable of identifying an aggressive lane change is necessary and taking action. Although this insight into controller performance is important to benchmark its capability, a similar CIS controller is developed for implementation.

4.1.3 Minimum Slip Controller Formulation

While the hard safety constraints of the one-level architecture ensure feasibility, the fitness of a candidate CIS maneuver is not based on a path following metric; thus, a novel slip minimization formulation is introduced. In the context of vehicle dynamics, tire slip is the reference parameter that generates tire force; minimizing the tire slip minimizes the tire force. The maximum force the tires are allowed to produce is limited by the coefficient of friction, which depends on the road surface and tire rubber interaction.

By minimizing the tire force needed for the maneuver, this formulation maximizes the remaining available tire force. This is desirable for two reasons. First, the minimum tire force solution can be argued that it is the least intrusive on occupants, thus the most comfortable maneuver. Second, by maximizing the additional available tire force, the resulting maneuver maximizes the additional available control authority to make corrections mid maneuver if disturbances happen. As a result, the inherent plant-prediction model mismatch due to model fidelity can be addressed, and the CIS maneuver safely completed.

4.1.4 Adaptive CIS Controller Formulation

While the combined minimum slip fitness metric and closed-loop nature of the controller implementation provide some level of stability, failure can still occur if the prediction model does not accurately represent the plant system. Literature has identified most of the parameters dictating the vehicle dynamics can be well estimated and extrapolated, with the key exception being coefficient of friction. Coefficient of friction is heavily dependent on the road surface condition, which cannot necessarily be known *a priori*.

Thus, it is proposed to use a UKF observer to estimate the coefficient of friction in closed loop. Through simulation testing, it is shown the UKF quickly identifies the plant system coefficient of friction, allowing subsequent MPC solutions to use an improved prediction model. The fact the UKF adjusts for the discrepancy in coefficient of friction quickly is critical to the performance of the adaptive formulation. If the plant system operates at a lower coefficient of friction than is believed by the prediction model, early identification of the plant-prediction model mismatch gives the CIS controller comparatively more time to adjust the control trajectory prior to constraint failure. This prevents tire saturation, which often results in violating lane boundary constraints. If the plant system operates at a higher coefficient of friction, the adaptive controller formulation recognizes the plant system is more responsive than believed and does not prescribe as aggressive a maneuver. In simulation testing, the adaptive controller formulation can maintain hard safety constraint feasibility for an initial coefficient of friction discrepancy of up to 55% between the plant and prediction models, compared to the baseline case maintaining up to 25% discrepancy.

4.1.5 Real-time Controller Performance

Arguably the most challenging hurdle to implementing such complex nonlinear MPC controllers is achieving real-time performance. While many controllers have been implemented for vehicle control in different controller architectures, they are often for a low computationally complex application. Yet, for some applications like evasive lane changes at the limits of handling, retaining the problem complexity is essential to ensuring safe operation. While computing power has been steadily improving for the past few decades, it is not foreseeable for computing power to grow the three orders of magnitude required to implement the original CIS implementation. To this extent, a framework for achieving real-time performance of the CIS controller, and more broadly nonlinear MPC controllers, is presented.

Through profiling of the controller, it is identified that evaluating the system dynamics is the most computationally expensive stage. This is because for most nonlinear systems of interest, there is no closed-form solution of the model, thus numerical integration is required [121]. To map the control inputs to the system response trajectory, system dynamics must be evaluated multiple times. Additionally, to calculate the sensitivity of the system response to control inputs, the original finite difference sensitivity calculation requires simulating many prediction trajectories. While it is infeasible to provide a blanket solution to improving the wall time of evaluating the system dynamics for all nonlinear system, the framework is intended to minimize the number of times the system dynamics need to be evaluated, hence reducing computational cost.

To begin, various numerical integration techniques are compared to establish the longest allowable time step, hence fewest number of integration points. The numerical integration methods are benchmarked off a derived or implied system uncertainty so as to keep the integration resolution consistent with expected system response. This analysis is mindful of both the computational cost and memory dependencies of the numerical integration methods.

Second, the trajectory optimization structure is discussed, balancing the resulting optimization problem size with optimization problem complexity. While some implicit trajectory simulation methods, such as collocation and the family of spectral methods, are often implemented for these complex nonlinear applications, multiple shooting is introduced and compared. The core advantage of collocation methods is the introduction and control of the state trajectory into the optimization problem, and it is identified multiple shooting trajectory simulation can do the same. For the intended CIS controller, it is concluded multiple shooting with explicit integration is the proper balance of integration time step against controller complexity, though the framework supports other conclusions for different applications.

Finally, the mapping of the optimization problem to modern compute hardware is discussed. While computing power is steadily growing, applications require careful mapping onto the underlying hardware to fully leverage the available compute resources. To this extent, two formulations of the multiple shooting trajectory simulation are presented for a GPU co-processor architecture. While both formulations leverage parallel computing hardware, the block parallelization structure achieves 7× speed up and thread parallelization structure achieves 26× speed up over the serial CPU implementation.

Without modification to the fitness and feasibility metrics of the nonlinear MPC controller, the framework for designing the trajectory simulation architecture achieves the necessary three orders of magnitude speed up for real-time compatibility. Additionally, it is shown real-time performance is not achieved by only addressing the software side of the problem, such as though numerical integration or collocation trajectory simulation, or the hardware side, such as parallel threads or exotic co-processors, but instead requires a balanced software-hardware solution.

Additionally, preliminary experiments on test vehicle confirm the impenetrability as part of a full stack of modern autonomous driving architecture. These low speed tests validate the assumptions the controller is built on, ranging from the expected perception data structures to available automotive compute capabilities. Although the low speed tests intentionally do not push the vehicle to the limits of handling, they serve as a stepping stone to validate the controller is working as intend within the vehicle architecture, green-lighting higher speed tests.

4.2 List of Publications

As a result of the efforts leading up to this dissertation, the above contributions have been recognized in conference presentations, journal publications, and patent submissions, denoted below.

4.2.1 Conference Presentations

1. J. Wurts, J. L. Stein, and T. Ersal, "Collision Imminent Steering Using Nonlinear Model Predictive Control," in *2018 Annual American Control Conference (ACC)*. 2018, pp. 4772-4777 [122]

2. J. Wurts, J. L. Stein, and T. Ersal, “Increasing Computational Speed of Nonlinear Model Predictive Control Using Analytic Gradients of the Explicit Integration Scheme with Application to Collision Imminent Steering,” in *2018 Conference on Control Technology and Applications (CCTA)*. 2018, pp. 1026–1031. [123]

3. J. Wurts, J. L. Stein, and T. Ersal, “Minimum Slip Collision Imminent Steering in Curved Roads Using Nonlinear Model Predictive Control,” in *2019 American Control Conference (ACC)*. 2019, pp. 3975–3980 [124]

4. J. Wurts, J. Dallas, J. L. Stein, and T. Ersal, "Adaptive Nonlinear Model Predictive Control for Collision Imminent Steering with Uncertain Coefficient of Friction", in *2020 American Control Conference (ACC)*. 2020, *to appear* [125]

5. J. Dallas, J. Wurts, J. L. Stein, and T. Ersal, "Contingent Nonlinear Model Predictive Control for Collision Imminent Steering in Uncertain Environments", in *21st International Federation of Automatic Control World Congress*. 2020, *to appear* [126]

4.2.2 Journal Publications

1. J. Wurts, J. L. Stein, and T. Ersal, "Collision Imminent Steering at High Speed Using Nonlinear Model Predictive Control," in *IEEE Transactions on Vehicle Technology*. 2020, *accepted, to appear*[127]

2. J. Wurts, J. L. Stein, and T. Ersal, "Collision Imminent Steering in Curved Roads Using Nonlinear Model Predictive Control," in *IEEE Transactions on Vehicle Technology*. 2020, *under review*[128]

3. J. Wurts, J. L. Stein, and T. Ersal, "Design for Real-Time Nonlinear Model Predictive Control with Application to Collision Imminent Steering," in *IEEE Transactions on Control Systems Technology*. 2020, *under review*[129]

4.2.3 United States Patent and Trademark Submission

1. J. L. Stein, T. Ersal, and J. Wurts, "Collision Imminent Steering Control System and Methods," US Patent App. 15/971,318 [130]

2. J. L. Stein, T. Ersal, and J. Wurts, "Lane Change Maneuvers with Minimized Tire Slip," US Patent App. 16/452,936 [131]

3. J. Dallas, J. L. Stein, T. Ersal, and J. Wurts, "Contingent Model Predictive Control Incorporating Online Estimation of Nominal and Uncertain Parameters", University of Michigan Invention Disclosure, *under patent consideration*

4.3 Future Work

The contributions herein support a CIS controller compatible with real-time performance. While the controller design and performance framework supports the intended application, these results can expand to support more advanced implementations and applications. Some potential considerations are highlighted below.

4.3.1 Shared Control Formulation

In the current form, the envisioned automotive CIS system is intended to lock out a human driver and prescribe the CIS controller's commands to the vehicle. From a human driver's perspective, this can be concerning, because such as system does not necessarily have the human's trust that the actions are safe. However, it is feasible to pose the CIS control problem in a shared control setting.

In a shared control setting, there are two sources of control action: one from a human and a second solved internal to the controller. The controller then takes as input the human's control action and, by some metric, blends the control commands to issue a single action to the system.

Shared control architectures have been well studied, especially in the field of vehicle control [132], [133]. These shared control architectures leverage the steering wheel for haptic feedback as a means of encouraging the driver to take what the controller believes to be a better control action.

The proposed CIS controller can employ a similar design by acting as overwatch to the human driver. In the envisioned architecture, the CIS controller will take as input the human's current control action. The optimization problem will then be structured to find a control sequence such that the first control action equals the driver's current action subject to the remaining feasibility constraints in (2.17).

In this formulation, the controller will identify if the driver's control action could perform a CIS maneuver, but will only intervene if the human is not acting quickly or aggressively enough. This avoids completely locking the human out of control of the vehicle, and even gives the human as much control authority as the controller identifies as feasible.

The resulting control trajectory can then be used to assist the driver, either as visual reference for intended trajectory or through a similar haptic feedback.

4.3.2 Robust Versus Adaptive Controller Formulations

To perform the intended CIS maneuver, the algorithm must be cognizant of the instantaneous environment conditions and vehicle handling capabilities. This is not trivial, as the algorithm relies on sensor data, perceptual processes, mathematical system models, and stored reference data, all of which are prone to varying levels of uncertainty. Thus, balancing the controller performance under these uncertainties is nontrivial. If the algorithm is overly conservative in action, then the controller's *believed* capability of obstacle avoidance is reduced. If the algorithm is overly confident in its data and models, then the controller's *actual* capability is reduced. Either case would result in safety issues that are in reality avoidable. Thus, there is an opportunity to incorporate the various sources of uncertainty directly into the MPC controller, balancing instantaneous uncertainty with safety critical performance.

In Sec. 2.7, the controller is made resilient to uncertainty in the coefficient of friction by introducing an adaptive control formulation. Of the prediction model parameters, the prediction trajectory is most sensitive to coefficient of friction, yet the coefficient of friction cannot be known *a priori* and can change during operation. However, tracking the uncertainty in coefficient of friction shows a low frequency response, which is well captured in an adaptive formulation. However, uncertainties with frequency content too high for adaptive formulations, such as sensor noise, require robust formulations [134].

Thus, a blended adaptive robust controller formulation can account for uncertainty and noise through multiple sources. By properly incorporating such uncertainty into the controller, the controller performance can be provably improved without excess risk [135].

Often, uncertain parameters are modeled as independent Gaussian distributions. However, traditional estimation approaches and optimization formulations used for modern perception pipelines can also produce uncertainty models. Thus, there is an opportunity to not only incorporate the process noise into the controller formulation, but the exact noise model can be included in the nonlinear controller as well.

While a vastly computationally complex problem, a both adaptive and robust one-level nonlinear MPC formulation promises significant performance improvements subject to various sources of uncertainty.

4.3.3 Parallel Implementation of Nonlinear Numerical Optimizer

The optimal control formulation in (2.17) is solved using IPOPT as an off the shelf nonlinear optimizer. While the trajectory simulation strategy employs parallel computing hardware, IPOPT does not natively do so as well.

There are two opportunities to leverage parallel computing within the optimizer.

One option is through the speed up of the linear algebra steps. GPUs are inherently good at linear algebra subroutines because these stages are computationally heavy, but have little sequential dependencies. Linear algebra subroutines such as matrix-matrix multiplication and matrix inversion can greatly benefit from being run on GPUs over CPUs.

However, there is an additional opportunity to fundamentally redesign the numerical optimization procedure to leverage parallel computing, which is not specific to IPOPT. Typically numerical optimization is a sequential dependency heavy operation: from a candidate design point, calculate the search direction, solve the line search subproblem, evaluate new candidate point for fitness and feasibility, then repeat. While this procedure cannot be natively parallelized, multiple candidate points can be considered in parallel.

In Sec. 3.4, it was discussed GPU applications can support thousands of threads simultaneously. The GPU acceleration for evaluating fitness and feasibility criteria can support evaluating tens to hundreds of candidate control trajectories simultaneously for no wall time penalty.

The advantages of simulating candidate control points in parallel has been recognized and leveraged in path integral controllers [136] and particle swarm optimization [137]. However, these implementations are based on a statistical analysis and do not benefit from the advantages of gradient-based optimization over gradient-free.

Hence, there is an opportunity for a gradient-based numeric optimizer to consider many candidate points in parallel. For example, during the line search procedure, multiple candidate points can be propagated with slight perturbations in the hopes of identifying at least one point better than the traditional line search method.

The motivation in developing a new optimization procedure addresses the final thread parallel multiple shooting profiling. In the final form, the majority of the wall time was spent inside the numeric optimizer, not evaluating candidate design points. By improving the convergence procedure of the optimizer, the net wall time can be improved.

Appendix

Appendix A

Test Vehicle Implementation

As a proof of concept, the CIS controller developed in Chapter 2 is implemented on a Toyota Research Institute (TRI) test platform. The vehicle is a modified Lexus LS 500h, equipped with additional sensors and hardware as a development platform for self-driving research [138]. Beyond the standard sensing package of the Lexus LS 500h, the test platform features multiple LIDAR, vision based cameras, and radar for perception purposes. Computing hardware is primarily commercially available components consistent with high performance consumer grade products. A press release photo of the development platform as appeared at Consumer Electronics Show 2019 can be seen in Fig. A.1.

As input to the CIS controller, the TRI development platform's perception stack readily provides the drivable tube. Similar to the geometrically drivable tube in Sec. 2.1, the perception stack adjusts the drivable tube to capture both lane boundary and obstacle information. A work flow block diagram of the CIS controller implemented within the vehicle control pipeline can be seen in Fig. A.2.

While the controller is designed to natively interface with the test architecture, performing a CIS maneuver at highway speeds is a dangerous test condition, even in controlled environments. Thus, as a proof of concept, preliminary testing addresses a low speed lane change.

Figure A.1: The Toyota Research Institute test vehicle is a modified Lexus LS 500h with additional sensors for self-driving research.

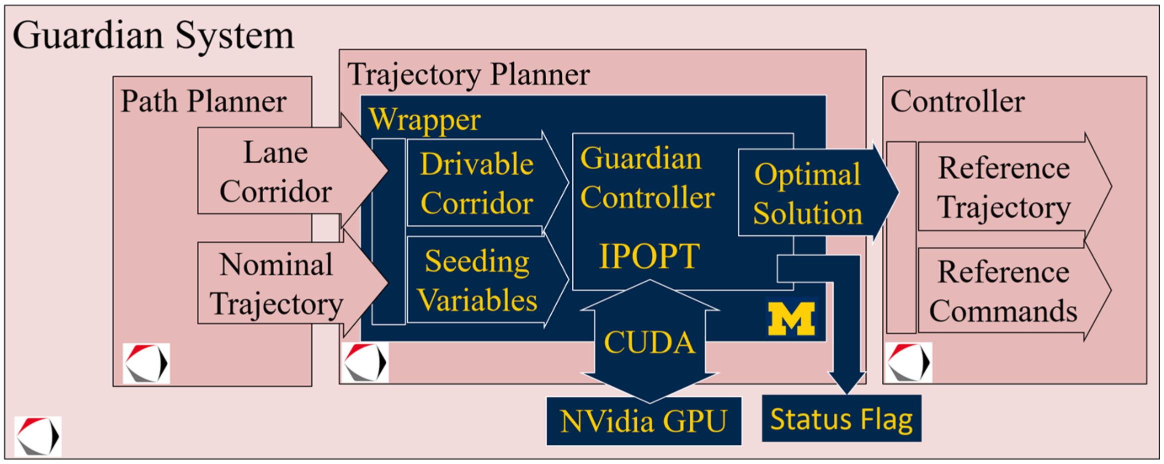


Figure A.2: The CIS controller resides within the vehicle control pipeline, taking input from the perception stack and output control commands to sub-dependencies.

In a low speed lane change, the host vehicle travels approximately 20 MPH in a straight line in its starting lane. A barrel is place far down the road, partially blocking the host's lane. For the barrel distance and vehicle speed, the vehicle could feasibly brake to avoid collision, but a lane change trajectory is mandated.

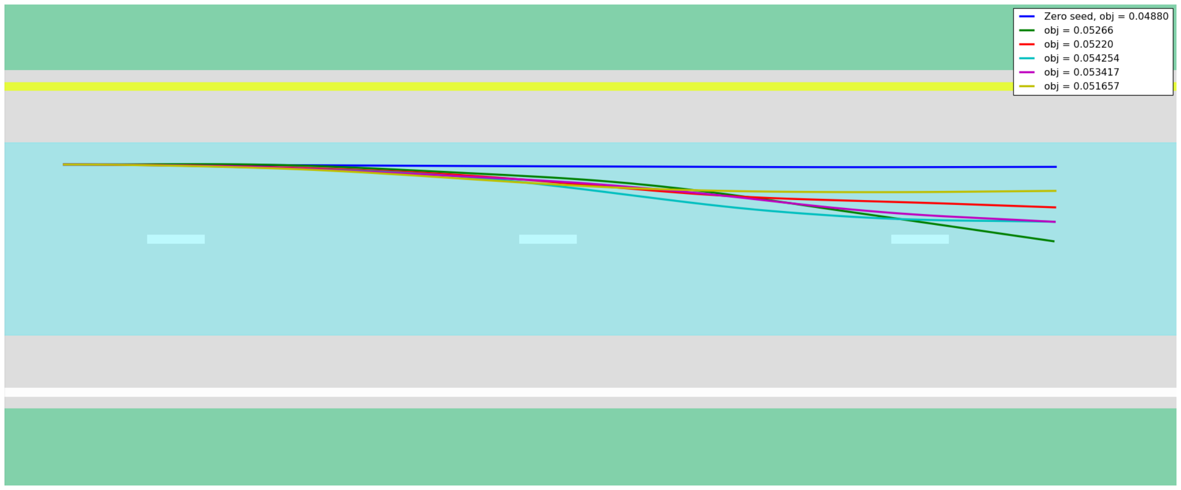
In the test, the lane change takes place well in advance of the barrel at low vehicle excitation. While this result does not showcase the capability of such a complicated nonlinear one-level MPC controller, it establishes the controller formulation is compatible with expected testing hardware as well as control pipeline. Additionally, it suggests slightly higher speed testing can proceed.

However, one insight provided by the low speed test conditions is variations in subsequent prediction trajectories. Recall the CIS controller is tasked with navigating the vehicle through the drivable tube as a feasability metric, with the optimality metric as minimizing peak tire slip. In testing, consecutive iterations showed the prediction trajectories vary in $x - y$ space, best categorized as wiggles. These wiggles are not a wiggle within any one trajectory; instead the wiggles are variations in consecutive prediction horizons where the latter stretch of the prediction trajectories vary.

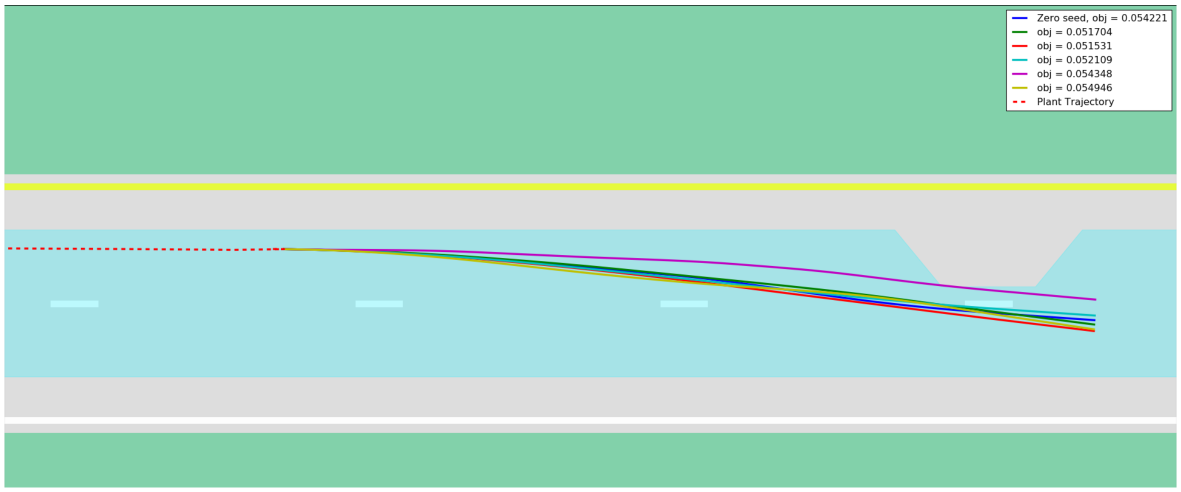
These wiggles become apparent in testing because the prediction horizon is displayed to the test engineer and safety driver. The resulting plant trajectory does not exhibit wiggles or oscillations in state response. However, it is important to convey to the test engineers the controller is operating as intended, requiring an investigation into the source of the wiggles.

In subsequent numerical simulations of the low speed test condition, these trajectory wiggles can be reproduced. Further analysis shows at low speeds and low vehicle excitation, the design space is comparatively flat, meaning small changes to the control rates can induce wiggles in the prediction trajectory, but have little variation on the objective function. These wiggles can be seen in Fig. A.3, showing multiple converged trajectories at different stages of the maneuver.

In Fig. A.3, each of the trajectory plots is a converged solution from IPOPT for a different random starting control trajectory. The initial random control trajectory



(a) Candidate trajectories within the drivable tube when the obstacle is far away.



(b) Candidate trajectories when the obstacle is within the prediction horizon.

Figure A.3: The low speed vehicle test is recreated in numerical simulation in an attempt to recreate the wiggles. The first sub plot shows various trajectories when the obstacle is far away. In the second subplot, the obstacle reduces the drivable tube, but there are still variations between trajectories.

is likely to cause constraint violation, but IPOPT quickly finds a feasible trajectory, staying within the drivable tube. However, the design space is comparatively flat, as the converged trajectories' objective function value is within $\pm 2\%$ other solutions.

Within each prediction trajectory, the variation in $x - y$ position later in the trajectory is apparent. Even though all trajectories obey the drivable tube limits, the absence of obstacle interference leaves the drivable tube comparatively wide. From this simulation, it should be noted the low speed test terminal constraints are difficult to enforce. For the 3.2 s prediction horizon, the low vehicle speed means the longitudinal travel within the prediction trajectory is low. Hence, when the obstacle is far away, the terminal state constraint is not well posed.

Slightly later in the maneuver as the vehicle encounters the obstacle, these variations still persist. Even though all the candidate trajectories obey the drivable tube around the obstacle, the low speed test condition implies low vehicle excitation is required. Again, this makes the design space comparatively flat.

There are many parameters used in nonlinear numerical optimization to establish convergence criteria, as well as iteration procedure. Depending on these tuning parameters, the converged numerical optimization problem can result in slightly different control trajectories, mapping to slightly different prediction trajectories, which can be within a comparable peak tire slip. for the set of parameters used within IPOPT, the candidate trajectories converged on the believed optimum prematurely, as the solutions are still approximately 10% above the global minimum of uniformly zero control action.

In a similar numerical investigation of the high speed CIS, the design space is found to not be comparatively flat. When the obstacle is imminent and an aggressive lane change is performed which comparatively excites the vehicle dynamics, these wiggles do not persist.

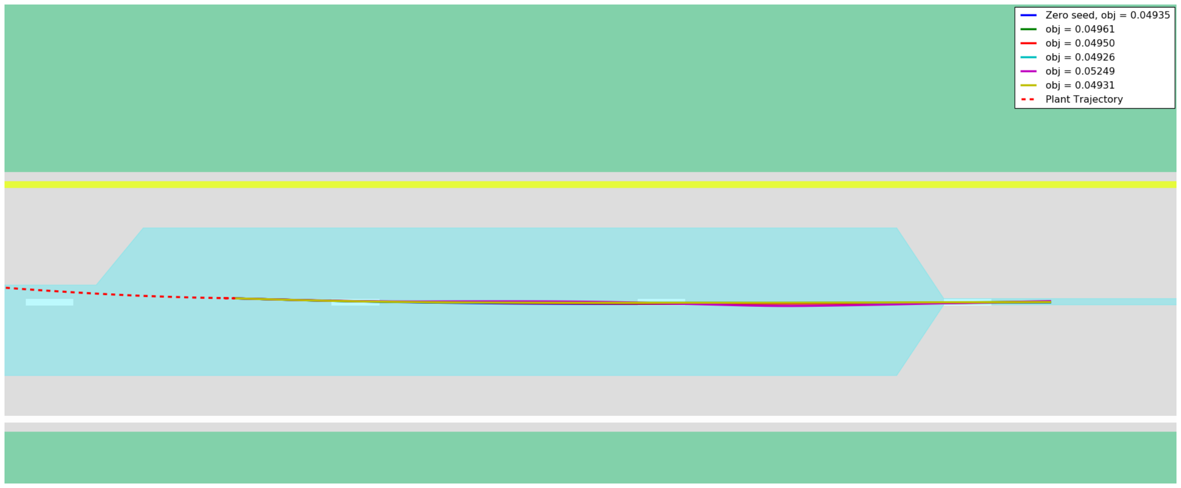
Tightening the convergence parameters of IPOPT might help the converged control trajectories tend towards a singular trajectory, but this increases the computational cost of the optimization as the candidate trajectory needs to be iterated further. For the relatively flat design space of the low speed obstacle avoidance maneuver requiring low vehicle excitation, improving the vehicle control is frivolous.

Yet, these low speed vehicle test results are insightful for two reasons. First, it means the CIS controller reliably finds the optimal control commands to perform an evasive lane change; hence the solution can be tested at high speeds. Second, because the wiggles can be recreated in simulation, the controller is acting as designed and this is not some corner case coming from an outside system or issue with the optimizer. It also implies the low speed lane change is comparatively easy, and traditional two-level path following controllers are well suited for this scenario.

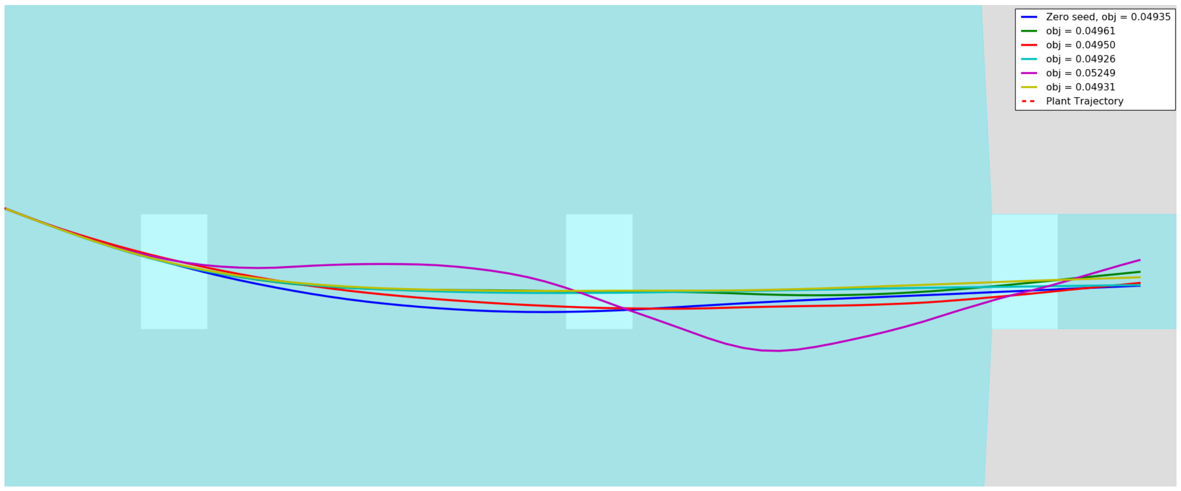
While the wiggles in state trajectory are accounted for and do not lead to instability, they can be concerning from an operator perspective. If the wiggles need to be removed, numerical simulations show narrowing the drivable space, as well as tuning the optimization parameters, can sufficiently restrict the low speed test case, removing wiggles in closed form. Fig. A.4, showing an artificial restriction in the drivable space.

Recall in Sec. 2.3 and 2.6, the terminal constraints are enforced through $\mathbf{x}_{\text{terminal}}$, which was geometrically produced. It is difficult to establish $\mathbf{x}_{\text{terminal}}$ for the low speed test when the obstacle is far away at all points in the closed loop iterations. However, by artificially restricting the drivable tube after the obstacle, the terminal $x-y$ state is restricted. By enforcing the starting state consistency constraint and terminal position constraint, the maximum variation in $x-y$ is reduced, but non-zero. When zoomed in, there is some variations, but these wiggles are on the order of 1 cm, which is not a concern to the safety drivers.

While the low speed test case showed some trajectory wiggles, these wiggles can be recreated in simulation and accounted for, suggesting there is not an anomaly persisting. Further, these wiggles in prediction trajectory are only seen under the hood to the test engineer, not experienced by the driver in the vehicle response. While the test case began at 20 MPH, this proof of concept suggests higher speed tests, such as at 25 to 40 MPH, can proceed.



(a) Artificial restriction in the drivable tube after passing the obstacle.



(b) Zoom in on the candidate trajectories with drivable tube restriction.

Figure A.4: By artificially restricting the drivable tube, a termianl state constraint is effectively introduced in $x - y$ space within the road.

Bibliography

[1] S. Thrun, "Toward robotic cars," *Communications of the ACM*, vol. 53, no. 4, pp. 99–106, 2010.

[2] T. Litman, *Autonomous vehicle implementation predictions.* Victoria Transport Policy Institute Victoria, Canada, 2017.

[3] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A study on model fidelity for model predictive control-based obstacle avoidance in high-speed autonomous ground vehicles," *Vehicle System Dynamics*, vol. 54, no. 11, pp. 1629–1650, 2016.

[4] P. N. Currier, *A method for modeling and prediction of ground vehicle dynamics and stability in autonomous systems.* PhD thesis, Virginia Tech, 2011.

[5] H. J. Kim and Y. Yoon, "Steering method for vehicle and apparatus thereof," May 14 2013. US Patent 8,442,713.

[6] J. Liu, *High-Speed Obstacle Avoidance at the Dynamic Limits for Autonomous Ground Vehicles.* PhD thesis, 2016.

[7] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 176–183, IEEE, 2014.

[8] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[9] J.-M. Park, D.-W. Kim, Y.-S. Yoon, H. J. Kim, and K.-S. Yi, "Obstacle avoidance of autonomous vehicles based on model predictive control," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 223, no. 12, pp. 1499–1516, 2009.

[10] M. A. Abbas, R. Milman, and J. M. Eklund, "Obstacle avoidance in real time with nonlinear model predictive control of autonomous vehicles," *Canadian Journal of Electrical and Computer Engineering*, vol. 40, no. 1, pp. 12–22, 2017.

[11] H. Febbo, J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Moving obstacle avoidance for large, high-speed autonomous ground vehicles," in *American Control Conference*, pp. 5568–5573, IEEE, 2017.

[12] D. Madås, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philippsen, A. Eidehall, and K.-M. Dahlén, "On path planning methods for automotive collision avoidance," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 931–937, IEEE, 2013.

[13] P. A. Theodosis and J. C. Gerdes, "Generating a racing line for an autonomous racecar using professional driving techniques," in *ASME 2011 Dynamic Systems and Control Conference and Bath/ASME Symposium on Fluid Power and Motion Control*, pp. 853–860, American Society of Mechanical Engineers, 2011.

[14] Y. Yoon, T. Choe, Y. Park, and H. J. Kim, "Obstacle avoidance for wheeled robots in unknown environments using model predictive control," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 6792–6797, 2008.

[15] S. Di Cairano, U. Kalabić, and K. Berntorp, "Vehicle tracking control on piecewise-clothoidal trajectories by mpc with guaranteed error bounds," in *Decision and Control (CDC), 2016 IEEE 55th Conference on*, pp. 709–714, IEEE, 2016.

[16] W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus, "Safe nonlinear trajectory generation for parallel autonomy with a dynamic vehicle model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2994–3008, 2017.

[17] M. Brown, J. Funke, S. Erlien, and J. C. Gerdes, "Safe driving envelopes for path tracking in autonomous vehicles," *Control Engineering Practice*, vol. 61, pp. 307–316, 2017.

[18] N. R. Kapania, J. Subosits, and J. C. Gerdes, "A sequential two-step algorithm for fast generation of vehicle racing trajectories," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 9, p. 091005, 2016.

[19] M. Gerdts, S. Karrenberg, B. Müller-Beßler, and G. Stock, "Generating locally optimal trajectories for an automatically driven car," *Optimization and Engineering*, vol. 10, no. 4, p. 439, 2009.

[20] Y. Yoon, J. M. Park, H. J. Kim, and S. Sastry, "Utilizing parallax information for collision avoidance in dynamic environments," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4186–4186, IEEE, 2008.

[21] H. Ahn, K. Berntorp, and S. Di Cairano, "Reachability-based decision making for city driving," in *2018 Annual American Control Conference (ACC)*, pp. 3203–3208, IEEE, 2018.

[22] K. Leung, E. Schmerling, M. Chen, J. Talbot, J. C. Gerdes, and M. Pavone, "On infusing reachability-based safety assurance within probabilistic planning frameworks for human-robot vehicle interactions," *arXiv preprint arXiv:1812.11315*, 2018.

[23] N. R. Kapania and J. C. Gerdes, "Design of a feedback-feedforward steering controller for accurate path tracking and stability at the limits of handling," *Vehicle System Dynamics*, vol. 53, no. 12, pp. 1687–1704, 2015.

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

[25] J. Alsterda, M. Brown, and J. C. Gerdes, "Contingency model predictive control for automated vehicles," in *2019 Annual American Control Conference (ACC)*, pp. 718–722, IEEE, 2019.

[26] Z. Wang, G. Li, H. Jiang, Q. Chen, and H. Zhang, "Collision-free navigation of autonomous vehicles using convex quadratic programming-based model predictive control," *IEEE/ASME Transactions on Mechatronics*, vol. 23, no. 3, pp. 1103–1113, 2018.

[27] Y. Yoon, J. Shin, H. J. Kim, Y. Park, and S. Sastry, "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles," *Control Engineering Practice*, vol. 17, no. 7, pp. 741–750, 2009.

[28] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg, "Receding horizon maneuver generation for automated highway driving," *Control Engineering Practice*, vol. 41, pp. 124–133, 2015.

[29] R. Jafari, S. Zeng, N. Moshchuk, and B. Litkouhi, "Reactive path planning for emergency steering maneuvers on highway roads," in *2017 American Control Conference (ACC)*, pp. 2943–2949, IEEE, 2017.

[30] F. Altché, P. Polack, and A. de La Fortelle, "A simple dynamic model for aggressive, near-limits trajectory planning," *arXiv preprint arXiv:1703.01225*, 2017.

[31] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[32] Z. Hou and S. Jin, *Model free adaptive control: theory and applications.* CRC press, 2013.

[33] D. Zhang and B. Wei, "A review on model reference adaptive control of robotic manipulators," *Annual Reviews in Control*, vol. 43, pp. 188–198, 2017.

[34] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2-4, pp. 190–216, 2010.

[35] I. Chakraborty, P. Tsiotras, and R. S. Diaz, "Time-optimal vehicle posture control to mitigate unavoidable collisions using conventional control inputs," in *American Control Conference*, pp. 2165–2170, 2013.

[36] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A nonlinear model predictive control formulation for obstacle avoidance in high-speed autonomous ground vehicles in unstructured environments," *Vehicle System Dynamics*, pp. 1–30, 2017.

[37] C. E. Beal and J. C. Gerdes, "Model predictive control for vehicle stabilization at the limits of handling," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 4, pp. 1258–1269, 2012.

[38] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, 2017.

[39] H. Febbo, P. Jayakumar, J. L. Stein, and T. Ersal, "Real-time trajectory planning for automated vehicle safety and performance in dynamic environments," *arXiv preprint arXiv:2001.10163*, 2020.

[40] A. Patterson, A. Lakshmanan, and N. Hovakimyan, "Intent-aware probabilistic trajectory estimation for collision prediction with uncertainty quantification," *arXiv preprint arXiv:1904.02765*, 2019.

[41] A. Patterson, A. Gahlawat, and N. Hovakimyan, "Learning probabilistic intersection traffic models for trajectory prediction," *arXiv preprint arXiv:2002.01965*, 2020.

[42] W. Schwarting, J. Alonso-Mora, L. Pauli, S. Karaman, and D. Rus, "Parallel autonomy in automated vehicles: Safe motion generation with minimal intervention," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1928–1935, IEEE, 2017.

[43] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*, vol. 19. Siam, 2010.

[44] E. F. Camacho and C. B. Alba, *Model predictive control.* Springer Science & Business Media, 2013.

[45] H. Kwakernaak and R. Sivan, *Linear optimal control systems*, vol. 1. Wiley-interscience New York, 1972.

[46] H.-S. Juang and K.-Y. Lurrr, "Design and control of a two-wheel self-balancing robot using the arduino microcontroller board," in *2013 10th IEEE International Conference on Control and Automation (ICCA)*, pp. 634–639, IEEE, 2013.

[47] P. TøNdel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit mpc solutions," *Automatica*, vol. 39, no. 3, pp. 489–497, 2003.

[48] S. Di Cairano, D. Yanakiev, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat, "Model predictive idle speed control: Design, analysis, and experimental evaluation," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 1, pp. 84–97, 2011.

[49] H. Li, K. Butts, K. Zaseck, D. Liao-McPherson, and I. Kolmanovsky, "Emissions modeling of a light-duty diesel engine for model-based control design using multi-layer perceptron neural networks," tech. rep., SAE Technical Paper, 2017.

[50] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2520–2525, IEEE, 2011.

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

[52] D. Liao-McPherson, M. Nicotra, and I. Kolmanovsky, "Time distributed sequential quadratic programming for model predictive control: Stability and robustness," *arXiv preprint arXiv:1903.02605*, 2019.

[53] J. T. Betts and W. P. Huffman, "Path-constrained trajectory optimization using sparse sequential quadratic programming," *Journal of Guidance, Control, and Dynamics*, vol. 16, no. 1, pp. 59–68, 1993.

[54] S. Richter, C. N. Jones, and M. Morari, "Computational complexity certification for real-time mpc with input constraints based on the fast gradient method," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2012.

[55] R. Milman and E. J. Davison, "Guaranteed bounds on the performance cost of a fast real-time suboptimal constrained mpc controller," in *IEEE Conference on Decision and Control*, vol. 2, pp. 2035–2040, 2004.

[56] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, 2014.

[57] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[58] L. Grüne and J. Pannek, "Nonlinear model predictive control," in *Nonlinear Model Predictive Control*, pp. 45–69, Springer, 2017.

[59] J. Albersmeyer, D. Beigel, C. Kirches, L. Wirsching, H. G. Bock, and J. P. Schlöder, "Fast nonlinear model predictive control with an application in automotive engineering," in *Nonlinear Model Predictive Control*, pp. 471–480, Springer, 2009.

[60] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Improving the robustness of an mpc-based obstacle avoidance algorithm to parametric uncertainty using worst-case scenarios," *Vehicle System Dynamics*, vol. 57, no. 6, pp. 874–913, 2019.

[61] I. Chakraborty, P. Tsiotras, and J. Lu, "Vehicle posture control through aggressive maneuvering for mitigation of t-bone collisions," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 3264–3269, IEEE, 2011.

[62] M. J. Tenny, J. B. Rawlings, and R. Bindlish, "Feasible real-time nonlinear model predictive control," in *AICHE SYMPOSIUM SERIES*, pp. 433–437, New York; American Institute of Chemical Engineers; 1998, 2002.

[63] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "Combined speed and steering control in high speed autonomous ground vehicles for obstacle avoidance using model predictive control," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 10, pp. 8746–8763, 2017.

[64] D. Solomon, "The struggles of the fastest highway in america," Jul 2017.

[65] U. D. of Transportion, *Speed Concepts: Informational Guide.* Federal Highway Administration, 2009.

[66] A. A. O. S. Highway and T. Off, *A Policy on Geometric Design of Highways and Streets 2001.* American Association of State Highway Transport., 2001.

[67] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Safe driving envelopes for shared control of ground vehicles," *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 831–836, 2013.

[68] A. Alleyne, "A comparison of alternative obstacle avoidance strategies for vehicle control," *Vehicle System Dynamics*, vol. 27, no. 5-6, pp. 371–392, 1997.

[69] T. Shim and C. Ghike, "Understanding the limitations of different vehicle models for roll dynamics studies," *Vehicle system dynamics*, vol. 45, no. 3, pp. 191–216, 2007.

[70] T. D. Gillespie, "Carsim data manual," *Ann Arbor, Michigan: Mechanical Simulation Corporation*, 2004.

[71] G. J. Heydinger, R. A. Bixel, W. R. Garrott, M. Pyne, J. G. Howe, and D. A. Guenther, "Measured vehicle inertial parameters-NHTSA's data through November 1998," tech. rep., SAE Technical Paper, 1999.

[72] E. Bakker, L. Nyborg, and H. B. Pacejka, "Tyre modelling for use in vehicle dynamics studies," tech. rep., SAE Technical Paper, 1987.

[73] J. hwan Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *2013 American control conference*, pp. 188–193, IEEE, 2013.

[74] Y. Peng and X. Yang, "Comparison of various double-lane change manoeuvre specifications," *Vehicle System Dynamics*, vol. 50, no. 7, pp. 1157–1171, 2012.

[75] R. C. Hoffman, J. L. Stein, L. S. Louca, and K. Huh, "Using the milliken moment method and dynamic simulation to evaluate vehicle stability and controllability," in *ASME 2004 International Mechanical Engineering Congress and Exposition*, pp. 173–180, American Society of Mechanical Engineers Digital Collection, 2008.

[76] J. Liu, P. Jayakumar, J. L. Stein, and T. Ersal, "A study on model fidelity for model predictive control based obstacle avoidance in high speed autonomous ground vehicles," *Vehicle System Dynamics*, vol. 54, no. 11, pp. 1629–1650, 2016.

[77] G. Kreisselmeier and R. Steinhauser, "Systematic control design by optimizing a vector performance index," in *Computer aided design of control systems*, pp. 113–117, Elsevier, 1980.

[78] G. J. Kennedy and J. E. Hicken, "Improved constraint-aggregation methods," *Computer Methods in Applied Mechanics and Engineering*, vol. 289, pp. 332–354, 2015.

[79] M. de FV Pereira, I. Kolmanovsky, and C. E. Cesnik, "Model predictive control with constraint aggregation applied to conventional and very flexible aircraft," in *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 431–437, IEEE, 2019.

[80] A. Richards, "Fast model predictive control with soft constraints," *European Journal of Control*, vol. 25, pp. 51–59, 2015.

[81] T. Gu and J. M. Dolan, "On-road motion planning for autonomous vehicles," in *International Conference on Intelligent Robotics and Applications*, pp. 588–597, Springer, 2012.

[82] D. Kogan and R. Murray, "Optimization-based navigation for the darpa grand challenge," in *Conference on Decision and Control (CDC)*, 2006.

[83] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[84] J. Jiang, M. Seaid, M. S. Mohamed, and H. Li, "Inverse algorithm for real-time road roughness estimation for autonomous vehicles," *Archive of Applied Mechanics*, pp. 1–16, 2020.

[85] K. Berntorp, R. Quirynen, T. Uno, and S. Di Cairano, "Trajectory tracking for autonomous vehicles on varying road surfaces by friction-adaptive nonlinear model predictive control," *Vehicle System Dynamics*, pp. 1–21, 2019.

[86] C. R. Carlson and J. C. Gerdes, "Consistent nonlinear estimation of longitudinal tire stiffness and effective radius," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 6, pp. 1010–1020, 2005.

[87] M. G. Bekker, "Mechanics of locomotion and lunar surface vehicle concepts," *Sae Transactions*, pp. 549–569, 1964.

[88] J. Y. Wong, *Theory of ground vehicles*. John Wiley & Sons, 2008.

[89] K. Guo and L. Ren, "A unified semi-empirical tire model with higher accuracy and less parameters," *SAE transactions*, pp. 1513–1520, 1999.

[90] C. Sierra, E. Tseng, A. Jain, and H. Peng, "Cornering stiffness estimation based on vehicle lateral dynamics," *Vehicle System Dynamics*, vol. 44, no. sup1, pp. 24–38, 2006.

[91] G. Baffet, A. Charara, and D. Lechner, "Estimation of vehicle sideslip, tire force and wheel cornering stiffness," *Control Engineering Practice*, vol. 17, no. 11, pp. 1255–1264, 2009.

[92] K. Berntorp and S. Di Cairano, "Tire-stiffness and vehicle-state estimation based on noise-adaptive particle filtering," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 3, pp. 1100–1114, 2018.

[93] K. Berntorp, R. Quirynen, and S. Di Cairano, "Steering of autonomous vehicles based on friction-adaptive nonlinear model-predictive control," in *2019 Annual American Control Conference (ACC)*, pp. 965–970, IEEE, 2019.

[94] F. Braghin, F. Cheli, and E. Sabbioni, "Environmental effects on pacejka's scaling factors," *Vehicle System Dynamics*, vol. 44, no. 7, pp. 547–568, 2006.

[95] J. Dallas, K. Jain, Z. Dong, M. P. Cole, P. Jayakumar, and T. Ersal, "Online terrain estimation for autonomous vehicles on deformable terrains," *arXiv preprint arXiv:1908.00130*, 2019.

[96] N. Cooper, D. Crolla, M. Levesley, and W. Manning, "Integration of active suspension and active driveline to ensure stability while improving vehicle dynamics," tech. rep., SAE Technical Paper, 2005.

[97] E. Esmailzadeh and H. Taghirad, "Active vehicle suspensions with optimal state-feedback control," *International Journal of Modelling and Simulation*, vol. 18, no. 3, pp. 228–238, 1998.

[98] H. B. Keller, *Numerical methods for two-point boundary-value problems*. Courier Dover Publications, 2018.

[99] L. T. Biegler, "An overview of simultaneous strategies for dynamic optimization," *Chemical Engineering and Processing: Process Intensification*, vol. 46, no. 11, pp. 1043–1053, 2007.

[100] E. Fehlberg, "Low-order classical runge-kutta formulas with stepsize control and their application to some heat transfer problems," 1969.

[101] J. D. Hedengren, R. A. Shishavan, K. M. Powell, and T. F. Edgar, "Nonlinear modeling, estimation and predictive control in apmonitor," *Computers & Chemical Engineering*, vol. 70, pp. 133–148, 2014.

[102] K. Schittkowski, C. Zillober, and R. Zotemantel, "Numerical comparison of nonlinear programming algorithms for structural optimization," *Structural Optimization*, vol. 7, no. 1-2, pp. 1–19, 1994.

[103] J. Kim, D. G. Bates, and I. Postlethwaite, "Nonlinear robust performance analysis using complex-step gradient approximation," *Automatica*, vol. 42, no. 1, pp. 177–182, 2006.

[104] A. Griewank *et al.*, "On automatic differentiation," *Mathematical Programming: recent developments and applications*, vol. 6, no. 6, pp. 83–107, 1989.

[105] J. Andersson, J. Åkesson, and M. Diehl, "Casadi: A symbolic package for automatic differentiation and optimal control," in *Recent advances in algorithmic differentiation*, pp. 297–307, Springer, 2012.

[106] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, vol. 105. Siam, 2008.

[107] J. T. Betts and W. P. Huffman, "Exploiting sparsity in the direct transcription method for optimal control," *Computational Optimization and Applications*, vol. 14, no. 2, pp. 179–201, 1999.

[108] A. Walther, "Automatic differentiation of explicit runge-kutta methods for optimal control," *Computational Optimization and Applications*, vol. 36, no. 1, pp. 83–108, 2007.

[109] A. H. Gebremedhin, A. Tarafdar, A. Pothen, and A. Walther, "Efficient computation of sparse hessians using coloring and automatic differentiation," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 209–223, 2009.

[110] Y. Saad, *Iterative methods for sparse linear systems*, vol. 82. siam, 2003.

[111] C. C. Paige and M. A. Saunders, "Lsqr: An algorithm for sparse linear equations and sparse least squares," *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, no. 1, pp. 43–71, 1982.

[112] Y. Saad and B. Suchomel, "Arms: An algebraic recursive multilevel solver for general sparse linear systems," *Numerical linear algebra with applications*, vol. 9, no. 5, pp. 359–378, 2002.

[113] H. G. Bock and K.-J. Plitt, "A multiple shooting algorithm for direct solution of optimal control problems," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 1603–1608, 1984.

[114] C. Nvidia, "Nvidia cuda c programming guide," *Nvidia Corporation*, vol. 120, no. 18, p. 8, 2011.

[115] V. W. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, *et al.*, "Debunking the 100x gpu vs. cpu myth: an evaluation of throughput computing on cpu and gpu," in *Proceedings of the 37th annual international symposium on Computer architecture*, pp. 451–460, 2010.

[116] R. Vuduc, A. Chandramowlishwaran, J. Choi, M. Guney, and A. Shringarpure, "On the limits of gpu acceleration," in *Proceedings of the 2nd USENIX conference on Hot topics in parallelism*, vol. 13, 2010.

[117] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W.-m. W. Hwu, "Optimization principles and application performance evaluation of a multithreaded gpu using cuda," in *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pp. 73–82, 2008.

[118] G. Chrysos, "Intel® xeon phi^{TM} coprocessor-the architecture," *Intel Whitepaper*, vol. 176, 2014.

[119] D. Patterson, "The top 10 innovations in the new nvidia fermi architecture, and the top 3 next challenges," *Nvidia Whitepaper*, vol. 47, 2009.

[120] J. Butcher, "Runge-kutta methods," *Scholarpedia*, vol. 2, no. 9, p. 3147, 2007.

[121] M. Gerdts, *Optimal control of ODEs and DAEs.* Walter de Gruyter, 2011.

[122] J. Wurts, J. L. Stein, and T. Ersal, "Collision imminent steering using nonlinear model predictive control," in *2018 Annual American Control Conference (ACC)*, pp. 4772–4777, IEEE, 2018.

[123] J. Wurts, J. L. Stein, and T. Ersal, "Increasing computational speed of nonlinear model predictive control using analytic gradients of the explicit integration scheme with application to collision imminent steering," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, pp. 1026–1031, IEEE, 2018.

[124] J. Wurts, J. L. Stein, and T. Ersal, "Minimum slip collision imminent steering in curved roads using nonlinear model predictive control," in *2019 American Control Conference (ACC)*, pp. 3975–3980, IEEE, 2019.

[125] J. Wurts, J. L. Stein, and T. Ersal, "Adaptive nonlinear model predictive control for collision imminent steering with uncertain coefficient of friction," in *2020 American Control Conference (ACC)*, p. accepted, IEEE, 2020.

[126] J. Dallas, J. Wurts, J. L. Stein, and T. Ersal, "Contingent nonlinear model predictive control for collision imminent steering in uncertain environments," in *21st International Federation of Automatic Control World Congress*, p. accepted, IFAC, 2020.

[127] J. Wurts, J. L. Stein, and T. Ersal, "Collision imminent steering at high speed using nonlinear model predictive control," *Transactions on Vehicle Technology, accepted*, 2018.

[128] J. Wurts, J. L. Stein, and T. Ersal, "Collision imminent steering in curved roads using one-level nonlinear model predictive control," *Transactions on Vehicle Technology, under review*, 2020.

[129] J. Wurts, J. L. Stein, and T. Ersal, "Design for real-time nonlinear model predictive control with application to collision imminent steering," *Transactions on Control Systems Technology, under review*, 2020.

[130] J. L. Stein, T. Ersal, and J. Wurts, "Collision imminent steering control systems and methods," Nov. 7 2019. US Patent App. 15/971,318.

[131] J. L. Stein, T. Ersal, and J. Wurts, "Lane change maneuvers with minimized tire slip." US Patent App. 16/452,936.

[132] M. Steele and R. B. Gillespie, "Shared control between human and machine: Using a haptic steering wheel to aid in land vehicle guidance," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 45, pp. 1671–1675, SAGE Publications Sage CA: Los Angeles, CA, 2001.

[133] P. Griffiths and R. B. Gillespie, "Shared control between human and machine: Haptic display of automation during manual control of vehicle heading," in *12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS'04. Proceedings.*, pp. 358–366, IEEE, 2004.

[134] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Vehicle System Dynamics*, vol. 52, no. 6, pp. 802–823, 2014.

[135] I. R. Dunning, *Advances in robust and adaptive optimization: algorithms, software, and insights.* PhD thesis, Massachusetts Institute of Technology, 2016.

[136] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *IEEE International Conference on Robotics and Automation*, pp. 1433–1440, 2016.

[137] Y. Zhou and Y. Tan, "Gpu-based parallel particle swarm optimization," in *2009 IEEE Congress on Evolutionary Computation*, pp. 1493–1500, IEEE, 2009.

[138] "Toyota research institute rolls out p4 automated driving test vehicle at ces," *Toyota USA Newsroom*, Jun 2019.