Eindhoven University of Technology

MASTER

Application of Optimization Methods to Quasi Steady-State Lap Time Simulation

Themen, S.C. (Bas)

Award date:
2024

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 10. Mai. 2025

Department of Mechanical Engineering

MSc Thesis Report

Application of Optimization Methods to Quasi Steady-State Lap Time Simulation

DC 2023.088

Student:
Bas Themen (1514032) s.c.themen@student.tue.nl

Supervisors:
Dr. ir. Igo Besselink i.j.m.besselink@tue.nl
Girish Radhakrishnan MSc girish.radhakrishnan@optimumg.com

Eindhoven, December 29, 2023

Abstract

Lap time simulation (LTS) algorithms play a pivotal role in refining vehicle setup and enhancing driver training in motorsports. These algorithms effectively mimic a vehicle's performance at its peak capability along a predetermined trajectory or between circuit boundaries. In this study, a quasi steady-state planar single-track vehicle model, featuring an elevated center of gravity, is employed following a predefined racing line.

Initial investigations focus on a steady-state optimization utilizing a gradient-based algorithm. Vehicle inputs such as steering angle and motor torque are designated as design variables, while the vehicle state is determined using a Newton-Raphson algorithm within the constraints function. The findings highlight improved convergence by 10% in comparison to previous research, albeit computational processing time remains considerably lengthy, rendering the implementation in an LTS algorithm impractical.

To address this, an LTS algorithm has subsequently been developed, with a single-step lookahead distance. Various optimization problems have been formulated to identify the optimal vehicle performance at different points on the circuit. Leveraging the problem structure, (semi-)analytical values of all potential minimizers are determined through a reformulation of the GG diagram. The applicable minimizers are then selected by evaluation with the grip and power constraints.

The model, however, reveals two fundamental drawbacks, including intermittent infeasibility of solutions during cornering and a rapid velocity decrease on straight-line segments. Additionally, a vehicle with a higher friction coefficient at the rear tire is unable to fully utilize its grip during cornering. Nonetheless, compared to a point mass model, the revised LTS algorithm effectively accounts for longitudinal load transfer, as well as uneven weight, grip, torque, and downforce distributions between the two axles. A single lap simulation conducted for the Circuit de Barcelona-Catalunya requires 7.7 seconds of computational time.

A comparison with telemetry data from two different vehicles unveils that the LTS algorithm closely mirrors reality at the apices of low- and medium-speed corners. Nonetheless, differences are observed during braking, high-speed corners, and for power-limited conditions.

Keywords: Lap time simulation, vehicle dynamics, optimization, GG diagram

Declaration concerning the TU/e Code of Scientific Conduct

I have read the TU/e Code of Scientific Conduct[i].

In carrying out research, design and educational activities, I shall observe the five central values of scientific integrity, namely: trustworthiness, intellectual honesty, openness, independence and societal responsibility, as well as the norms and principles which follow from them.

Date

24/12/2023 ……..……..

Name

S.C. Themen ……..……..

ID-number

1514032 ……..……..

Signature

……..……..

Submit the signed declaration to the student administration of your department.

[i] See: https://www.tue.nl/en/our-university/about-the-university/organization/integrity/scientific-integrity/
The Netherlands Code of Conduct for Scientific Integrity, endorsed by 6 umbrella organizations, including the VSNU, can be found here also. More information about scientific integrity is published on the websites of TU/e and VSNU

Acknowledgements

What a story this has been. There is a good reason why a Master thesis project should only last six months and not just over two years like my journey. I think the front tire constraint for a standard LMP2 vehicle in the adapted GG diagram describes my emotions pretty well. It seems confusing, but if you look at it from the right perspective you discover the beauty in it. When you first look at it, it raises a lot of questions and it looks like you are not doing the right thing. However, if you turn the image upside down it kind of looks like a heart, or a gnome hat like Igo suggested countless times. This beautifully signifies the moment when everything came together and all the equations just worked. As such I am very proud of myself to have had the perseverance to get through this entire process no matter how many times I was fighting my own body and mind to continue.

Obviously, I could not have done it alone and I really appreciate the support from my supervisors Igo and Girish. Every week I came up with some new interesting things that I first needed to decipher myself and then find a way to make it understandable for them. It was tough to find our common wavelength, but at the end of the day we always both found the correct energy transfer to arrive in the same orbit configuration. There you go, I have effectively used my bachelor in Physics to make a joke. I could make more, but the joke has already collapsed, so actually quantum mechanics forbids this. Igo always had his comments like "Als je de voordeur niet kan gebruiken, moet je misschien de achterdeur vinden" and they helped a lot to figure things out. The feedback to my report was a lot and rough to get through, but it helped massively and assisted me in creating a report that I am very proud of!

Oh Girish, if I added up the time I waited for our meetings to start, I probably would have celebrated my Christmas as a Master of Science. (joking of course!) You were always really supportive and gave me some beautiful words of encouragement, and I am really thankful for that! Also a thank you to OptimumG for allowing me to use data from actual racing cars (how cool is that!!!) and for giving me an assignment that combines my passion for motorsport, vehicle dynamics and optimization.

Luckily, a lot more than just working on my thesis has happened over the past year. I have made giant leaps in understanding myself and how my autism affects my life. It has been a blast to work together on this with Timmo. In the end, it is nice to learn about some cars doing cool stuff, but learning how to live my life and understanding what gives me joy is the most valuable thing I could ask for. So thank you so much for everything, Timmo! I hope to chase after your paizuri koning title soon! Also a big thank you for the support I got with forming my professional life from Marlou, and Marco for giving a flavor of Brabantse gezelligheid (and baldness) to my progress in life.

Queryn, Wouter, Meneer, and the rest of The Diorite Experts, thank you so much for the continued fun and support throughout all these years and for making our UltraHardCore seasons such a success! Rin, I am very grateful to have met you, and here's to many more cosplays and cuddles! Keep slaying, you awesome people in Cosplayers from NL! A big thank you to all my teammates in University Racing Eindhoven over the years, and for allowing me to use some data to compare my LTS results. You better use my research to improve our current model! It's not always been easy, but always thank you for the support and love, papa and mama. But also a thank you to all the people who have hurt me, and allowed me to learn valuable lessons. (I just wish it did not have to be this painful sometimes).

Finally, thank you for reading this, of course! I hope you find the contents of my thesis interesting to read and insightful. Treat yourself, each other and our planet with love, kindness and compassion and have yourself a great day!

Love,
Bas

Contents

1 Introduction **1**
1.1 History of lap time simulation . . . 1
1.2 OptimumG . . . 3
1.3 Research objectives . . . 4
1.4 Outline of the thesis . . . 4

2 Vehicle models **5**
2.1 Vehicle sub-systems . . . 5
2.1.1 Tires . . . 5
2.1.2 Aerodynamics . . . 8
2.1.3 Powertrain & brake model . . . 9
2.2 Planar single-track vehicle model . . . 11

3 Optimization **15**
3.1 General topics in optimization . . . 15
3.1.1 Problem formulation . . . 15
3.1.2 Conditions for optimality . . . 15
3.1.3 Types of optimization algorithms . . . 16
3.1.4 Convergence . . . 16
3.2 Root finding . . . 17
3.2.1 Newton-Raphson . . . 17
3.3 Sequential Quadratic Programming . . . 19
3.4 Summary . . . 20

4 Nested optimization **21**
4.1 Optimization problem formulation . . . 21
4.2 Results comparison with direct optimization . . . 22
4.2.1 Gripping condition . . . 23
4.2.2 Drifting condition . . . 25
4.2.3 Overall verdict . . . 26
4.3 Summary . . . 27

5 Optimization of lap time simulation **28**
5.1 Modeling lap time simulation . . . 28
5.1.1 Model inputs . . . 28
5.1.2 Forward loop . . . 28
5.1.3 Reverse loop . . . 29
5.1.4 Lap time calculation . . . 30
5.2 Usage of optimization theory . . . 30
5.2.1 Forward loop . . . 30
5.2.2 Reverse loop . . . 32
5.3 Determination of minimizers . . . 34
5.3.1 Conventional GG diagram . . . 34
5.3.2 Adapted GG diagram . . . 36
5.3.3 Shape of the constraints . . . 37
5.3.4 Finding potential minimizers . . . 40
5.4 Increasing vehicle model complexity . . . 45

5.5 Summary 47

6 Applications of lap time simulation **48**
6.1 Model deficiencies 48
6.1.1 Solution outside of the feasible domain 48
6.1.2 Rapid loss of velocity 49
6.2 Comparison with conventional LTS 51
6.2.1 Constant radius turn 52
6.2.2 Circuit de Barcelona-Catalunya 55
6.3 Effect of differences in tire friction coefficient 58
6.3.1 Stability 59
6.4 Comparison with real telemetry data 61
6.4.1 LMP2 at Circuit de Barcelona-Catalunya 61
6.4.2 Formula Student at AutoX 64
6.5 Summary 67

7 Conclusions and recommendations **68**

References **70**

A Derivation of constraints **73**
A.1 Front tire grip 73
A.2 Rear tire grip 73

B Derivation of minimizers **75**

1 Introduction

In the competitive environment that is motorsports, the one performance metric that never lies is the stopwatch. Race cars have many different subsystems that can all be configured individually. This myriad of options leads to trillions of potential vehicle setups [1]. With a finite amount of time to evaluate setup changes, finding the optimal vehicle setup for a specific circuit is nearly impossible. This leads to the introduction of lap time simulation (LTS), computer simulations that predict the behavior of a vehicle on a circuit. These computer simulations can be done when physical testing is not an option. Advantages are that vehicle setups are quickly changed by adjusting the vehicle parameters, and there is no risk of equipment damage or even injury to a human driver.

1.1 History of lap time simulation

The origin of lap time simulation traces back to the mid 1950s and was introduced by the Mercedes Formula One team [2]. Circuits were mathematically dissected and the velocity and required gear were calculated at each point of the track. These models made a significant improvement to driver coaching, which gave a huge advantage over other constructors. At the start of the 1980s, the use of lap time simulation algorithms became commonplace and the algorithms featured increasingly more complex vehicle models. An example is RCSIM developed by Milliken and Milliken [2], which was released in 1982. RCSIM features a single-track model that includes longitudinal load transfer, tire load sensitivity, and brake balance. On a race track, the path is optimized such that at every turn the vehicle starts and ends on the outside of the track and hits the apex of the turn in between these points. The vehicle model has been expanded to a four-wheel model in subsequent releases, and features slip and camber angles, some suspension detail and engine modelling. The model consists of a very complex and non-linear model, which is solved with numerous layers of iterative solvers.

Gadola, et al. [3] implemented a vehicle model that includes a non-linear tire model, lateral and longitudinal load transfer and a linear chassis model. The racing line is created using a genetic algorithm. Tracks are split into sections from apex to apex. The speed at the transition between acceleration and braking is determined through an iterative process. The model is found to be reliable, but improvements can still be made to the gearshift strategy and the optimum cornering line. Similarly, Siegler, et al. [4] use a Newton-Raphson approach to find the maximum forward velocity to negotiate the corner minimum path radius at all times. A steady-state, quasi steady-state (QSS) and transient strategy are implemented in the model and compared for a Formula SAE car navigating a 180 degree hairpin. It is concluded that there is no great difference in overall time between the three strategies. However, the transient strategy is found to be slightly more accurate than the other two.

Brayshaw and Harrison [5] use a gradient-based optimization strategy to solve multiple subsequent optimization problems featuring a quasi steady-state vehicle model. By finding the maximum possible velocity, longitudinal acceleration and lateral acceleration through the optimization problem, a GG diagram can be formed. The lap time simulation algorithm uses this GG diagram by interpolating the results at each apex of the circuit. Between the apices, the acceleration and deceleration profiles are calculated, which leads to the velocity profile. The quasi steady-state method is found to be a good approximation of the optimized transient solution. Moreover, computational times are significantly reduced from hours to minutes.

In the aforementioned works, the racing line is always computed or available before the simulation of the vehicle model. In 1992, Fujioka and Kimura [6] pioneered the use of optimal control strategies in the lap time simulation problem. The minimum maneuver time is chosen as objective, which is optimized by choosing the optimal racing line. Casanova [7] developed a ground breaking method in the field of minimum maneuver time problems in 2000. Optimal control is used to solve a non-linear programming problem, which returns the lateral and longitudinal controls of the vehicle that yield the minimum time required to traverse a lap of a circuit. This work results in satisfactory accuracy, robustness and computational efficiency. However, the approach requires long computational times in the order of days. In 2008, Kelly [8] built on the model of Casanova, and added a thermodynamic tire model to evaluate dynamic tire temperature effects on lap time and vehicle stability.

Velenis and Tsiotras [9] found a robust and computationally efficient method, that semi-analytically solves a minimum maneuver time problem for a specified path. The vehicle is modeled as a point mass and the control is limited by the maximum possible longitudinal and lateral acceleration. Emphasis is put on the proof of optimality of the method and the computational cost, that is kept minimal for online implementations in autonomous vehicles.

Timings and Cole [10] created a lap time simulation algorithm using model predictive control in 2013. Key changes to the modeling approach are the change of objective to maximizing the distance traveled along the track centerline as a different expression for the minimum time problem and taking a linearization of a non-linear vehicle model, the track, and vehicle trajectories that allows for the formulation of a convex optimization problem. Compared to a non-convex non-linear programming solution, the control actions are non-optimal at certain points, but a significant improvement in computational efficiency is achieved.

In 2021, Massaro and Limebeer [11] released a paper summarizing the history of minimum lap time simulation with emphasis on the state-of-the-art optimal control problems. Major contributions include:

- The powertrain is modeled as a relationship between the tire longitudinal force and vehicle speed [12].
- Introduction of transients in a QSS model based on a dynamic response in an iterative process [13].
- Control of a hybrid electric vehicle in the minimum lap time simulation framework [14].
- Changes in aerodynamics based on vehicle movement [15].
- Influences of tire wear based on a thermodynamic model [16].
- A three-dimensional road model [17].
- Location-dependent road-tire friction coefficients [18].
- A minimum lap time simulation model that considers motorcycles by including a roll degree of freedom in the vehicle model [19].

Reference [11] also describes the fundamentals of the optimal control problem, different solution methods, road modelling and reconstruction, vehicle positioning, and vehicle models in detail.

1.2 OptimumG

OptimumG was founded in 1997 in Denver, USA. The company is active in the automotive and motorsports branches, focusing on applied vehicle dynamics and data acquisition. OptimumG offers custom software packages, knowledge transfer through seminars, and consultancy services. OptimumG aims to assist their customers with better understanding their vehicle, which may lead to improved performance.

The software packages developed by OptimumG are:

- OptimumLap. A lap time simulation tool that uses a point mass vehicle model in quasi steady-state.
- OptimumKinematics. A tool for suspension design based on a multi-body dynamics approach.
- OptimumTire. A software package for reading, fitting, and exporting tire data based on various empirical models.
- OptimumDynamics. Vehicle dynamics simulation software using a quasi steady-state approach.

As explained earlier, lap time simulation is a crucial tool for understanding and optimizing vehicle performance. Hence, detailed knowledge on such algorithms are of utmost importance to give a competitive edge over other competitors. This makes this research attractive for OptimumG to improve their knowledge on lap time simulation algorithms. In return, OptimumG offered guidance, and provided vehicle data and telemetry from their partners in the ELMS racing category.

Recently, OptimumG has been active in the field of vehicle performance optimization through research in collaboration with Eindhoven University of Technology. Radhakrishnan, et al. [20] created time efficient steady-state analysis and frequency responses by using a Newton iteration and vehicle model linearization respectively. With a case study, it was concluded that a four-wheel model can be developed that allows for efficient computations of steady-state, linearization, and time-domain simulations.

In [21], Beckers investigated the optimal driving conditions and vehicle setup of an LMP2 car in steady-state cornering for different corner radii using gradient-based optimization. In general, two optimal solutions are found, which correspond to the operating region of the tires. Increasing the vehicle model complexity introduces challenges that affect the rate of convergence. Themen [22] developed a discrete PID motor torque controller that verifies the existence of the stable solution in [21] for a single-track model. Moreover, regions where the vehicle is torque limited have been identified.

In [23], Calorio investigated optimal control based lap time simulation for a planar four-wheel vehicle model. Vehicle inputs, vehicle trajectory and vehicle setup are optimized, and it is found that the optimal vehicle trajectory changes depending on the vehicle setup. However, the model is found to be computationally expensive as a simulation around an oval style track takes around 45 minutes to complete.

1.3 Research objectives

This research aims to enhance the complexity of the vehicle models compared to conventional point mass LTS algorithms while maintaining computational efficiency. The associated research questions are:

- Is it possible to find analytical solutions for the optimal vehicle inputs?
- What changes need to be made to a conventional LTS model to incorporate more complex vehicle models?
- What is the optimal optimization framework for lap time simulation?
- What is a favorable trade off between computational efficiency and model exactness?

It is of utmost importance that the algorithm is computationally efficient, such that it can be implemented in commercial software. Moreover, the vehicle model shall contain important physical phenomena for race car performance. As observed in the aforementioned history of lap time simulation, a more detailed vehicle model comes at the cost of high computational efficiency. This means that a trade off should be found between the level of complexity and the calculation time.

In this research, it was decided to focus on a quasi steady-state lap time simulation algorithm with a predefined path. Many of these algorithms require iterative solvers [2][3][4], which take multiple iterations to be solved. Instead, this work focuses on a single-step lookahead distance, and uses optimization theory to redefine the forward and reverse loops of conventional quasi steady-state point mass LTS algorithms. By using an adapted version of the GG diagram and the structure of the optimization problem, (semi-)analytical solutions of the optimal driving conditions can be found at each iteration through visual inspection for a single-track vehicle model with an elevated center of gravity. This should allow for very fast computational times, even if an elaborate vehicle model is implemented. Furthermore, the algorithm in [21] is changed to only optimize vehicle inputs and calculate the vehicle state during the optimization through a root solving algorithm in order to test whether it can be a candidate for implementation in a lap time simulation algorithm.

1.4 Outline of the thesis

In this work, first the vehicle models and optimization theory are discussed in Chapter 2 and Chapter 3 respectively. A nested optimization approach is investigated in Chapter 4. The relevant theory behind lap time simulation and the implementation of optimization theory are discussed in Chapter 5, and a discussion of some of its applications is given in Chapter 6. Finally, the findings of this work are concluded in Chapter 7.

2 Vehicle models

Vehicle models are at the center of any analysis in vehicle dynamics. Mathematical models can be formulated for all kinds of vehicle sub-systems to analyze the motion of the vehicle on different levels. The motion is described by Newton's and Euler's laws of motion from classical mechanics.

First, some important vehicle sub-systems are introduced and their contribution to the vehicle motion is described. This is followed by the general equations of motion of the vehicle body and their steady-state simplifications. Two similar vehicle models are formulated that will be used in Chapter 4 and Chapter 5.

2.1 Vehicle sub-systems

Decomposition of the parts that make up a vehicle is crucial to understanding the overall motion of the vehicle. In this way, global parameters like center of gravity velocities can be linked to the local forces within a component. For the vehicle models the sub-systems tires, aerodynamics and powertrain will be discussed.

2.1.1 Tires

The tires are a very important element of the vehicle as the contact patches of each tire are the only places in which the vehicle is in direct contact with the external world, in this case the road. The magnitude of the forces exerted by a tire on the road depends on the slip of the tire, the load on the tire, the roughness of the road surface and many other parameters.

Two types of slip may be defined:

- Slip ratio SR: A relation between the amount of slip a tire experiences relative to its forward motion.
- Slip angle SA: The angle between the travel direction of the wheel and the direction in which the wheel points.

Slip ratio

The slip velocity V_{sx} between the tire and the road can be seen in Figure 2.1. This velocity can be decomposed into the translational part given by the longitudinal motion of the wheel center V_x and the rotational part given by the wheel rotational speed ω and the effective rolling radius r_E. The slip velocity is defined as follows:

$$V_{sx} = V_x - r_E\omega \tag{2.1}$$

From this, the slip ratio may be defined

$$\mathrm{SR} = -\frac{V_{sx}}{V_x} = -\frac{V_x - r_E\omega}{V_x} \tag{2.2}$$

In reality, the effective rolling radius depends on the vertical force. To avoid this extra complexity, the effective rolling radius is taken to be constant throughout this work, and is assumed to have the same value as the loaded radius r_L, which also is load dependent in reality but taken to be constant.

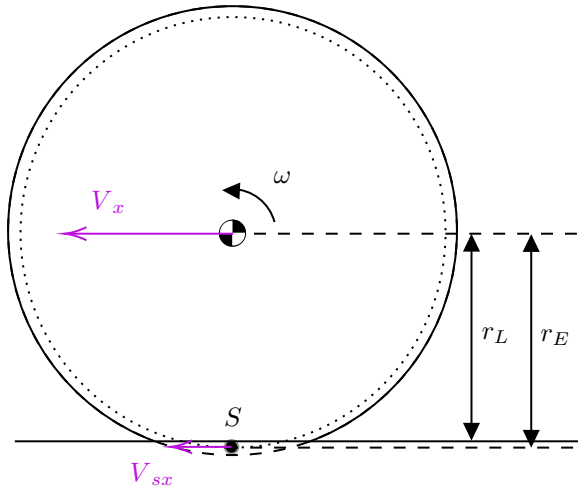


Figure 2.1: Slip ratio of a wheel.

Slip angle

Under the assumption that the tire does not deflect or rotate with respect to the wheel rim, it will have the same velocity at the contact patch as at the wheel center. This is visualized in Figure 2.2. From this, the slip angle SA can be determined:

$$\mathrm{SA} = \tan^{-1}\left(\frac{V_{sy}}{V_x}\right) \tag{2.3}$$

Here, the ISO sign convention is used for the slip angle.

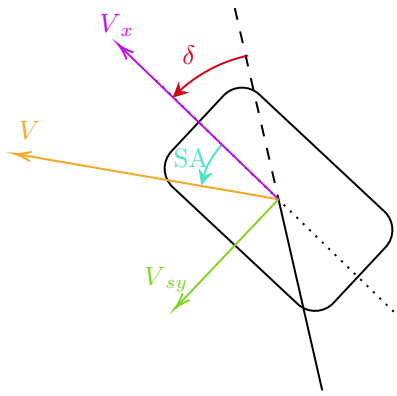


Figure 2.2: Slip angle of a wheel.

Maximum tire grip

The maximum amount of grip an object can have on a flat frictional surface F_s is defined by the normal force F_N that the surface exerts on the object and is scaled by a parameter called the friction coefficient μ. Since the object rests on the surface, the normal force is equivalent to the vertical force F_z on the object, as illustrated in Figure 2.3.

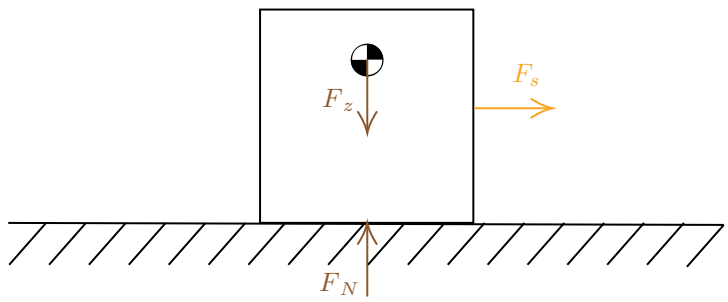


Figure 2.3: Sliding force acting on a mass on a frictional surface.

This principle also holds for tires on a road and gives an upper bound to the maximum achievable longitudinal and lateral forces, F_x and F_y respectively, of the tires. This upper bound takes the following (simplified) mathematical form:

$$\sqrt{F_x^2 + F_y^2} = \mu F_z \tag{2.4}$$

This limit is commonly referred to as the friction circle. In this simplification, it is assumed that the friction coefficient is identical in both longitudinal and lateral direction, although that is not the case in practice. Different friction coefficients in different directions cause the grip limit to be shaped more like an ellipse. Moreover, the friction coefficient linearly decreases with the applied vertical force, as can be seen in Figure 2.4.

A change in vertical force, for example due to aerodynamics at high velocity, will decrease the friction coefficient between the surface and the tire. The decrease in friction coefficient causes the build up of the friction force to decrease with increasing vertical force, resulting in a decline of tire grip increase.

Magic Formula

A model for determining the tire forces is the Magic Formula [24]. The Magic Formula is an empirical model that fits data from tire measurements to a set of mathematical expressions. The Magic Formula 5.2 can be described as follows:

$$[F_x, F_y, M_z] = \mathrm{MF5_2}\,(\mathrm{SR}, \mathrm{SA}, \gamma, F_z, \mathrm{TM}) \tag{2.5}$$

The outputs of the function are the longitudinal force F_x, the lateral force F_y and the self-aligning moment M_z. The inputs of the function are the slip ratio SR, the slip angle SA, the inclination angle of the tire γ, the vertical force on the tire F_z and the Magic Formula coefficients of the tire model TM. For a full definition of the Magic Formula 5.2, the reader is referred to [24].

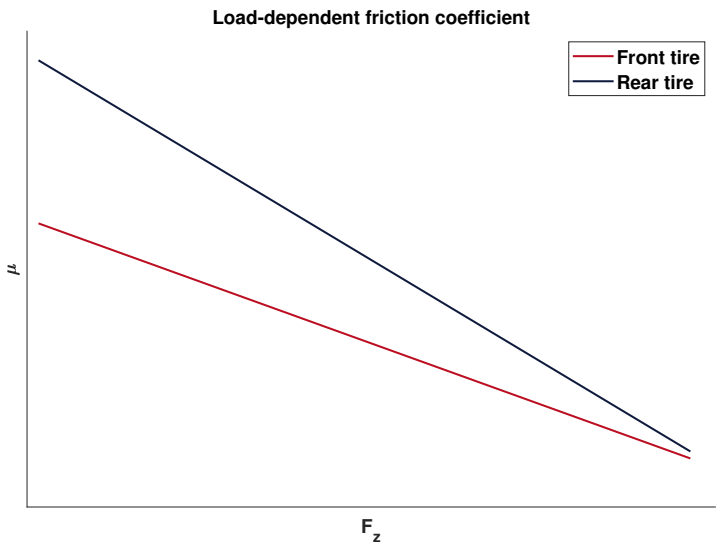


Figure 2.4: Load-dependent friction coefficients. The tire data is taken from an LMP2 car front and rear tire. For confidentiality reasons, only the general trend is shown.

2.1.2 Aerodynamics

Drag

The shape of a vehicle interacts with the medium it passes through. While driving, the vehicle pushes the air to flow around its body. This phenomenon is called drag, and resists the motion of the vehicle. The drag force F_D is a function of some vehicle parameters, the density of the dispersed medium and the velocity of the vehicle:

$$F_D = \frac{1}{2} C_D \rho A V^2 = k_D V^2 \tag{2.6}$$

where C_D is the drag coefficient of the vehicle, ρ is the density of the dispersed medium and A is the frontal area of the vehicle. The drag coefficient is a measure for the slipperiness or efficiency that the vehicle can traverse the medium. In reality, the drag coefficient varies with the angle of attack, but for simplicity it was assumed that it is a constant. Likewise, the air density is assumed to be constant around the entire track. Usually, the constant terms are combined into one constant factor k_D. Since, the drag force quadratically depends on the velocity of the vehicle, significantly more tractive effort is required to maintain a constant high velocity.

Downforce

Race cars are often equipped with winglike appendices, as seen in Figure 2.5. These wings direct the airflow around the vehicle in such a way that a pressure gradient forms, which results in an increase in vertical force on the vehicle. This principle is the same phenomenon as lift in aviation, but works in the opposite direction and is therefore more commonly referred to as downforce in motorsports.

Figure 2.5: Shape of an LMP2 car. Taken from [25].

The mathematical formulation of downforce is similar to that of the drag force, except that the drag coefficient C_D is exchanged for the lift coefficient C_L:

$$F_L = \frac{1}{2} C_L \rho A V^2 = k_L V^2 \tag{2.7}$$

The lift coefficient indicates the efficiency of the vehicle body to generate downforce with its wings. Like the drag coefficient, the lift coefficient is dependent on the angle of attack of the vehicle, but for simplicity it is taken to be constant. A rule of thumb for vehicles with wings and no ground effect is that the lift coefficient is about three times larger than the drag coefficient. Usually, the constant terms are combined into one constant factor k_L. Due to the quadratic dependency on the velocity, the downforce, and thus the vertical force on the tires, will rapidly increase at high velocity. This means, that the available tire grip, as given by (2.4), will also rapidly increase. However, due to the load sensitivity of the tires, the friction coefficient decreases, as shown in Figure 2.4. Therefore, the increase in lateral force is not quadratically dependent on velocity.

2.1.3 Powertrain & brake model

Wheel dynamics

Wheel dynamics describes the rotation of a wheel around the lateral axis through the wheel center. The required torque from the powertrain or brake system M_D is related to the longitudinal force of the tire in the contact patch F_x and the moment caused by the rolling resistance M_{rr}. This is illustrated in Figure 2.6.

Due to the rotation of the tire, tire rubber continuously moves through the contact patch, which causes internal damping and hysteresis. This effect dissipates energy and longitudinally skews the contact pressure of the tire, which moves the application of the vertical force forward. This is modelled as a rolling resistance moment M_{rr}, which equals

$$M_{rr} = -r_L f_{rr} F_z \operatorname{sgn}(\omega) \tag{2.8}$$

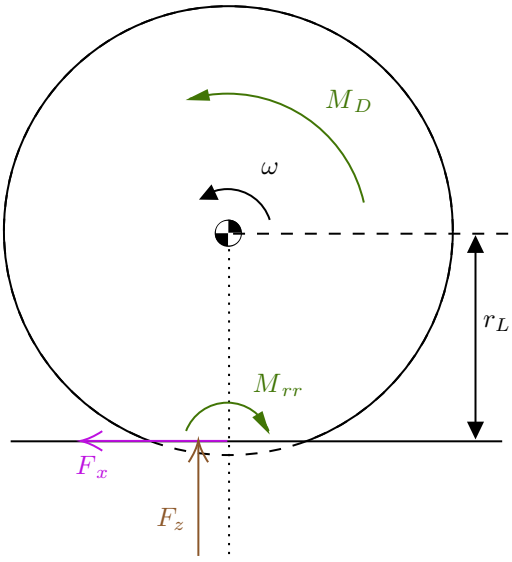


Figure 2.6: Schematic of wheel dynamics.

where f_{rr} is the rolling resistance coefficient, which is a measure for the intensity of the rolling resistance and it depends on the type of road surface being driven on. Moreover, M_{rr} includes a minus sign and the sign function depending on the wheel rotation ω. Using this, the rolling resistance moment will always work opposite to the direction the wheel is rolling.

The equation of motion for wheel dynamics equals

$$I_p\dot{\omega} = M_D - r_L F_x - M_{rr} \tag{2.9}$$

where I_p is the moment of inertia of the tire, wheel rim, brake disk, etc. In steady-state, the left-hand side of the equation is equal to zero, so the torque from the drive system is directly connected to the force in the contact patch. When $M_D \geq 0$, the wheel is being driven by the powertrain and when $M_D < 0$, the wheel is being braked by the brake system. In this model, M_D is directly used as a simulation input.

Constant power model

The simplest possible powertrain model is one where, the engine generates a constant power output P. The resulting propulsion force acting on the chassis' CoG F_M is then given by

$$F_M = \frac{P}{V} \tag{2.10}$$

The dependency on the velocity V indicates that for higher velocities, the propulsion force is reduced. However, this is not the sole effect on the amount the vehicle can accelerate, as this also depends on the drag force on the vehicle body F_D and the rolling resistance on the tires F_{rr}. Therefore, the power limit of the vehicle is as follows:

$$ma_x = F_M - F_D - F_{rr} \tag{2.11}$$

Torque distributions

A method of distributing the engine/brake power over the front and rear axle is by using torque/brake distributions, p_D and p_B, respectively. In a single-track vehicle model, the torque distribution describes the distribution of resulting longitudinal force on the tires of each axle and is expressed as a percentage. When $p_D = 0\%$, all engine torque is provided to the rear axle, i.e. the vehicle is a rear-wheel drive. Likewise, the vehicle is front-wheel driven, when $p_D = 100\%$. When using drive/brake torques, the torque/brake distribution can be calculated as follows:

$$p_V = \frac{M_{D_1}}{M_{D_1} + M_{D_2}} \tag{2.12}$$

where p_V is defined as the torque distribution for $M_D \geq 0$ and p_V is defined as the brake distribution for $M_D \leq 0$.

2.2 Planar single-track vehicle model

The single-track vehicle model is a simplified representation of the dynamics of a four-wheeled vehicle, where the tires and suspension of an axle are lumped into one equivalent tire. A major limitation of this model is the omission of lateral load transfer between the left and right tires on each axle. For the vehicle model used in Chapter 4, the direction of movement may be at an angular offset compared to its chassis orientation, the front wheel can be steered, and engine/brake power is transferred through wheel dynamics. For the vehicle model used in Chapter 5, the elevated center of gravity causes longitudinal load transfer and pitch dynamics, and the tractive forces on the front and rear wheel are distributed by the torque/brake distribution. In both instances, the vehicle is operated on a flat surface, meaning that no changes in elevation or road banking are taken into account. A schematic overview of the single-track model can be found in Figure 2.7 (top view) and Figure 2.8 (side view).

The vehicle has three translational degree of freedom (longitudinal motion, lateral motion, and vertical motion), and two rotational degree of freedom (pitch motion, and yaw motion). The velocities of the chassis in the longitudinal, lateral, and yaw direction are referred to as u, v, and r, respectively. The equivalent tires are separated by a distance of L from each other in the longitudinal direction, which is called the wheelbase. In between the tires, the center of gravity (CoG) is located at a distance a and b from the front and rear tire respectively. In the lateral direction, the CoG is located at a distance R from the center of the corner, or instantaneous center of zero velocity called the cornering radius. In steady-state conditions, the vehicle will perform circular motion around this point. The inverse of the cornering radius κ is called the curvature and by ISO sign conventions, a positive curvature corresponds to a left-hand turn, whereas a negative curvature corresponds to a right-hand turn. In case of an elevated CoG, the CoG is located at a distance h above the road in the vertical direction. The center of pressure (CoP) at which the aerodynamic forces act is located, relative to the CoG, at a distance c in the longitudinal direction and a distance d in the vertical direction.

At a time instance, the vehicle moves with respect to the circular trajectory set out by the curvature. This movement may be non-tangential to the trajectory as the velocity vector can have an angular offset β with respect to the vehicle plane of symmetry, which is called the chassis slip angle. The same holds for the front and rear tires, where the angular offsets of the velocity vectors are given by the slip angles SA. Both tires generate forces in the longitudinal direction F_x and lateral direction F_y and a moment in the yaw direction M_z. This is as a result of the

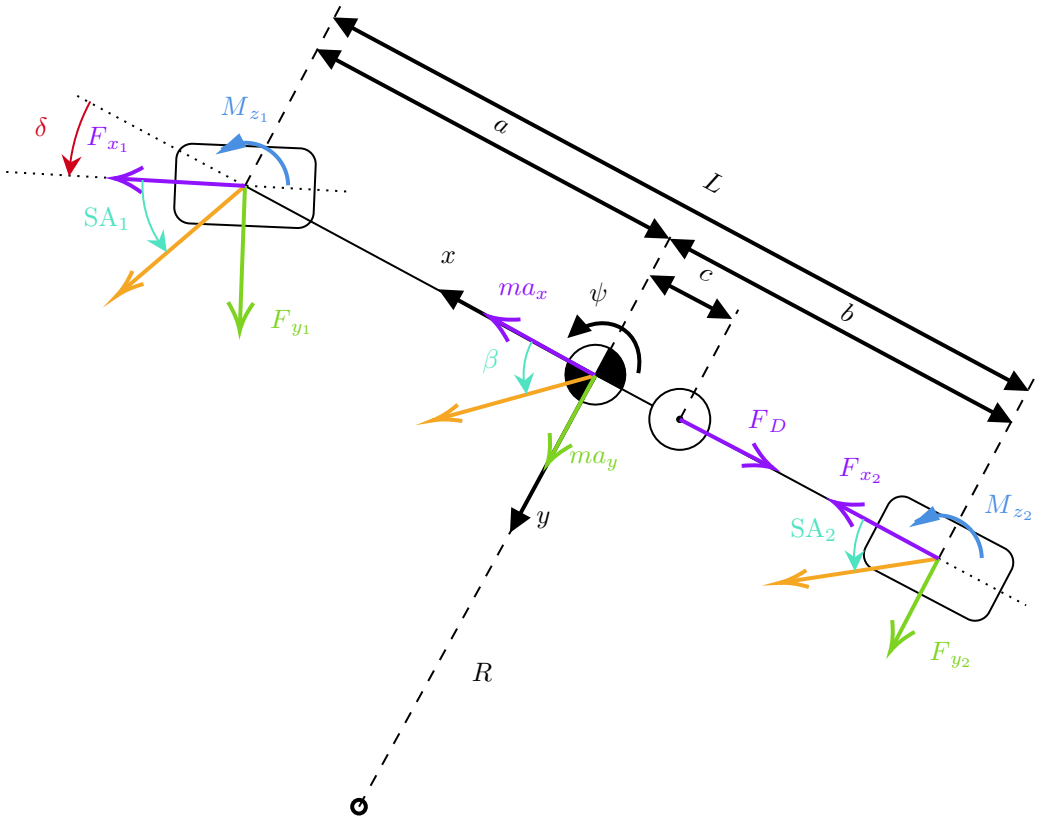


Figure 2.7: Planar single-track vehicle model - top view. ISO sign convention is used.

vertical force F_z pushing each tire into the road caused by the gravity on the vehicle body mg, the aerodynamic downforce F_L, and load transfer due to the acceleration/deceleration, and the drag force F_D. Moreover, the front tire can be steered in the yaw direction with a steering angle δ.

The equations of motion can be set up for both translational and rotational degrees of freedom with the use of force and moment balances. They are as follows:

$$m\left(\dot{u} - vr\right) = F_{x_1}\cos\left(\delta\right) - F_{y_1}\sin\left(\delta\right) + F_{x_2} - F_D \tag{2.13}$$

$$m\left(\dot{v} + ur\right) = F_{y_1}\cos\left(\delta\right) + F_{x_1}\sin\left(\delta\right) + F_{y_2} \tag{2.14}$$

$$m\ddot{z} = F_{z_1} + F_{z_2} - mg - F_L \tag{2.15}$$

$$I_{yy}\ddot{\theta} = -hF_{x_1} - aF_{z_1} + bF_{z_2} - hF_{x_2} - dF_D - cF_L \tag{2.16}$$

$$I_{zz}\dot{r} = a\left[F_{x_1}\cos\left(\delta\right) + F_{x_1}\sin\left(\delta\right)\right] - bF_{y_2} + M_{z_1} + M_{z_2} \tag{2.17}$$

Equation (2.16) is only used for the vehicle model in Chapter 5. Therefore, a simplification is given with no steering input.

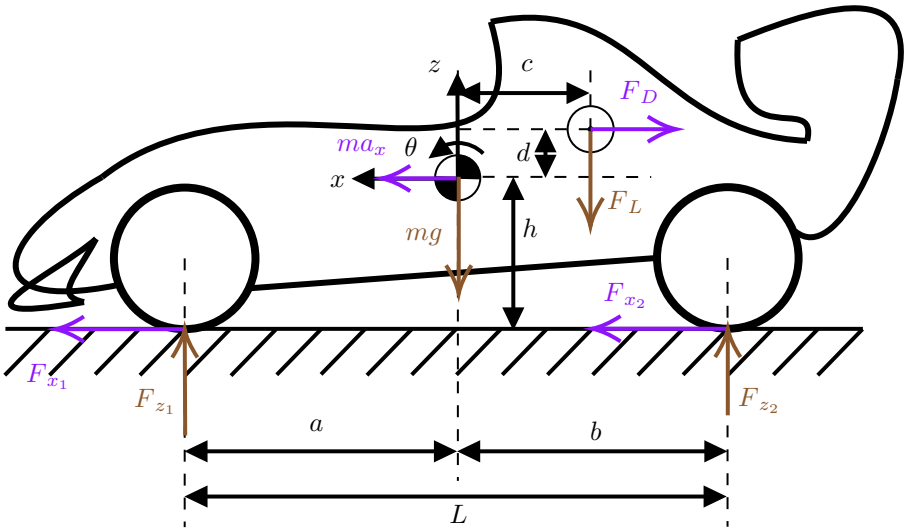


Figure 2.8: Planar single-track vehicle model - side view.

Steady-state cornering

In this work, steady-state conditions are assumed, i.e. there is no change in acceleration. This means that $\dot{u}$ $\dot{v}$, $\ddot{z}$, $\ddot{\theta}$, and $\dot{r}$ are all constant. The latter four accelerations are always assumed to be equal to zero. Motion in the lateral direction is described as pure circular motion, which means that the lateral acceleration $a_y = ur$ is equal to the centripetal acceleration:

$$a_y = V^2\kappa \tag{2.18}$$

where V is the resultant velocity of the CoG, which is equal to the Euclidean norm of the longitudinal and lateral velocity. Finally, the longitudinal motion $a_x = \dot{u} - vr$ will have different values for the models in Chapter 4 and Chapter 5. In Chapter 4, the vehicle drives with a constant forward velocity, meaning that $\dot{u} = 0$ and $a_x = -vr$. In Chapter 5, it is assumed that the vehicle moves along its longitudinal axis, meaning that $v = 0$ and $a_x = \dot{u}$.

With this information, the vertical forces, and for Chapter 5 the tire forces, can be derived. Using the approximation of small steering angles, the vertical forces are derived from a combination of the longitudinal, vertical and pitch equilibria:

$$F_{z_1} = \frac{b}{L}mg - \frac{h}{L}ma_x - \frac{h+d}{L}F_D + \frac{b-c}{L}F_L \tag{2.19}$$

$$F_{z_2} = \frac{a}{L}mg + \frac{h}{L}ma_x + \frac{h+d}{L}F_D + \frac{a+c}{L}F_L \tag{2.20}$$

Both vertical forces are comprised of four terms: static load due to gravity on the vehicle body, load transfer due to acceleration of the vehicle body, load transfer due to the application of the drag force at the CoP, and aerodynamic downforce. With these equations, two important quantities can be defined: the weight distribution ($W_d = \frac{b}{L}$) and the downforce distribution

($D_d = \frac{b-c}{L}$), representing the fraction of the vertical force that is applied to the front wheel. Note that for the rear wheel these coefficients will be $1-W_d$ and $1-D_d$. These distributions are given in percentages, where 100% means that all the weight/downforce is located at the front wheel, and 0% means that all the weight/downforce is located at the rear wheel.

Since these vertical forces are defined for the equivalent wheel, one must divide the vertical force by a factor two when using the Magic Formula. The output forces are then again multiplied by a factor two to obtain the tire forces of the equivalent tire.

For the vehicle model in Chapter 5, the distribution of lateral tire forces is determined using the lateral and yaw equilibria. No steering angle and self-aligning moments are considered as these are not a part of the vehicle model.

$$F_{y_1} = \frac{b}{L} m a_y = W_d m a_y \tag{2.21}$$

$$F_{y_2} = \frac{a}{L} m a_y = (1 - W_d)\, m a_y \tag{2.22}$$

Again, the weight distribution appears in the distribution of forces between the front and rear tire. Finally, the longitudinal force distribution is as described in Section 2.1.3.

$$F_{x_1} = p_V \left(m a_x + F_D\right) \tag{2.23}$$

$$F_{x_2} = (1 - p_V)\left(m a_x + F_D\right) \tag{2.24}$$

where p_V is equal to torque distribution p_D if the longitudinal force of the CoG exceeds the opposing aerodynamic drag force $m a_x \geq -F_D$. Otherwise, this is equal to the brake distribution p_B.

3 Optimization

Optimization is the theory of finding the optimal solution to a problem that may be constrained by boundaries. The optimal solution is found by minimizing an objective value by changing some parameters of the design. Optimization theory is at the foreground for the research discussed in Chapter 4 and Chapter 5. First, optimization theory is discussed in general, followed by a description of the Newton-Raphson and Sequential Quadratic Programming algorithms, which are common optimization algorithms for root solving and gradient-based optimization respectively.

3.1 General topics in optimization

3.1.1 Problem formulation

An optimization problem takes the following standard form:

$$\begin{aligned} & \underline{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T \\ \min_{\underline{x}} \quad & f(\underline{x}) \\ \text{s.t.} \quad & \underline{g}(\underline{x}) \leq \underline{0} \\ & \underline{h}(\underline{x}) = \underline{0} \\ & \underline{lb} \leq \underline{x} \leq \underline{ub} \end{aligned} \tag{3.1}$$

where $\underline{x}$ is an $N \times 1$ vector called the design variables, f is a scalar quantity called the objective function, $\underline{g}$ is an $M_g \times 1$ vector called the inequality constraints and $\underline{h}$ is an $M_h \times 1$ vector called the equality constraints. N, M_g and M_h are integers representing the amount of design variables and constraints of the system. $\underline{lb}$ and $\underline{ub}$ represent inequality constraints on the design variables called the lower and upper bound respectively. The goal of optimization is to find the values of $\underline{x}$ such that the value of f is minimized, while the constraints $\underline{g}$ and $\underline{h}$, and the design variable boundaries are satisfied. When a constraint is equal to zero, it is called active.

3.1.2 Conditions for optimality

The Karush-Kuhn-Tucker (KKT) conditions have to be met for a solution to the optimization problem, called the minimizer. They are formulated as follows:

Solution $\underline{x}^*$ can only be a minimizer to the optimization problem if, for $\underline{\lambda}_g^* \geq \underline{0}$ and $\underline{\lambda}_h^* \neq \underline{0}$, the following conditions hold:

$$\nabla f(\underline{x}^*) + \underline{\lambda}_g^{*T} \nabla \underline{g}(\underline{x}^*) + \underline{\lambda}_h^{*T} \nabla \underline{h}(\underline{x}^*) = \underline{0}^T \tag{3.2}$$

$$\underline{g}(\underline{x}^*) \leq \underline{0} \tag{3.3}$$

$$\underline{h}(\underline{x}^*) = \underline{0} \tag{3.4}$$

$$\underline{\lambda}_g^{*T} \underline{g}(\underline{x}^*) = \underline{0} \tag{3.5}$$

The vectors $\underline{\lambda}_g$ and $\underline{\lambda}_h$ are called Lagrange multipliers and represent the contributions of the active constraint to the optimal point. These conditions consist of three parts. (3.2) requires the gradient of the Lagrangian $L(\underline{x}) = f(\underline{x}) + \underline{\lambda}_g^T \underline{g}(\underline{x}) + \underline{\lambda}_h^T \underline{h}(\underline{x})$ to be equal to zero at the point $\underline{x}^*$. This means that at this point the Lagrangian is either a minimum, a maximum, or a saddle point. Equations (3.3) and (3.4) require the inequality constraints and the equality constraints, respectively, to be satisfied. Finally, (3.5) requires the active inequality constraints to be treated

as equality constraints by a correct choice of $\underline{\lambda}_g^*$.

However, a solution that satisfies these conditions may still not be a minimizer to the optimization problem due to the ambiguity of (3.2). By itself, the KKT conditions may find a maximum or saddle point in the objective function, which is not optimal. Therefore, another sufficient condition needs to be introduced to ensure optimality. This condition requires for solution $\underline{x}^*$ to be a minimizer to the optimization problem that the Hessian of the Lagrangian $\nabla^2_{\underline{x},\underline{x}} L\left(\underline{x}^*, \underline{\lambda}_g^*, \underline{\lambda}_h^*\right)$ is positive-definite on the subspace tangent to the active constraints. This means that any perturbation of the solution $\delta\underline{x}$ will lead to an increase of the objective score:

$$\delta f\left(\underline{x}^*\right) = \delta\underline{x}^T \nabla^2_{\underline{x},\underline{x}} L\left(\underline{x}^*, \underline{\lambda}_g^*, \underline{\lambda}_h^*\right) \delta\underline{x} > 0 \tag{3.6}$$

3.1.3 Types of optimization algorithms

There are three methods of solving mathematical optimization problems: gradient-based optimization, gradient-free optimization and hybrid optimization, which is a combination of gradient-based and gradient-free optimization. Gradient-based optimization algorithms are generally faster as they use derivative information to arrive at a solution, but may not always lead to global convergence. Gradient-free optimization uses alternative methods like pattern searching or genetics to find solutions. This method is slower, but is more robust for ensuring global convergence [26]. Hybrid optimization uses a gradient-free method to find the neighborhood of the global minimum, and then switches to a gradient-based method to find the minimum. This method performs worse in terms of computational time and global convergence than the respective strong points of the other algorithms, but this method may still be useful for some applications. In this work, gradient-based optimization was used and is described in more detail in later sections.

3.1.4 Convergence

The efficacy and drawbacks of gradient-based optimization algorithms stem from the use of gradients. An analytical, numerical or automatic gradient calculation is relatively fast since it immediately proposes a search direction and can be used to determine the step length, which improves the computational efficiency of the algorithm. However, the proposed search direction also proves to be a pitfall if the initial guess of the design variables is located in the neighborhood of a local minimum. In such a case, the algorithm will satisfy the KKT conditions and ensure that the Hessian of the Lagrangian is positive-definite in the neighborhood of the local minimum for small perturbations $\delta\underline{x}$. However, it is not checked whether the sufficient condition (3.6) holds for any arbitrary perturbation $\delta\underline{x}$. An example of such a situation is displayed in Figure 3.1.

Choosing an initial guess within the "valley" of the local minimum will lead to an optimal value corresponding to that local minimum, rather than the global optimum. Therefore, when selecting the initial guess of a design variable for a gradient-based optimization algorithm, caution must be exercised to avoid settling for a local minimum. To mitigate this issue, it is advisable to give careful consideration to the selection of the initial guess. One possible strategy is to (randomly) sample the initial guess, although this approach may be impractical for highly non-monotonic functions, requiring a large number of samples. In such cases, gradient-free optimization algorithms may be preferable.

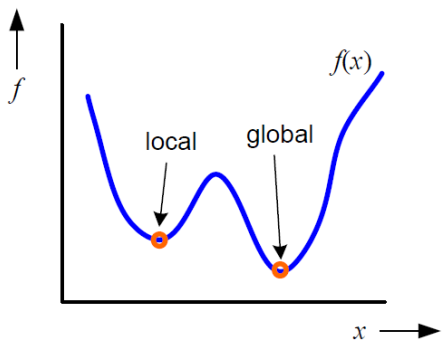


Figure 3.1: A 1D objective function with multiple local minima. Taken from [27].

3.2 Root finding

The objective of root finding algorithms is to find a state vector $\underline{s}^*$ such that the $M_F \times 1$ function $\underline{F}(\underline{s})$ obeys the following equation:

$$\underline{F}(\underline{s}^*) = \underline{0} \tag{3.7}$$

There are various methods to tackle this problem as, for example, the bracketing method, interpolation, iterative methods, or a combination of these methods. In this research, an iterative method called the Newton-Raphson method is used.

3.2.1 Newton-Raphson

The Newton-Raphson method corrects the value of the state vector $\underline{s}$ in an iterative manner using the partial derivatives of the function and the residual error during each iteration. An initial guess of the state $\underline{s}_0$ must be provided. For non-linear functions $\underline{F}(\underline{s})$, this means that the algorithm will find a single solution close to the initial guess, while potentially multiple solutions exist [28].

The algorithm

The definition of the Newton-Raphson algorithm in this thesis is taken from [29]. It is formulated as:

$$\underline{s}_{k+1} = \underline{s}_k + \alpha \left(-\underline{J}(\underline{s}_k)^{-1} \underline{\varepsilon}(\underline{s}_k)\right) \tag{3.8}$$

This equation describes the calculation of the state $\underline{s}$ at iteration $k+1$ of the algorithm. The value of the state at the previous iteration $\underline{s}_k$ is altered with the help of a corrective term. This term consists of the residual error $\underline{\varepsilon}(\underline{s}_k)$, the inverse of the Jacobian $\underline{J}(\underline{s}_k)$ and a step size control parameter α. The residual error is the value of the evaluated function at the previous iteration:

$$\underline{\varepsilon}(\underline{s}_k) = \underline{F}(\underline{s}_k) \tag{3.9}$$

The Jacobian consists of the partial derivatives of the evaluated function to the states and is evaluated at the state of the previous iteration:

$$\underline{J}(\underline{s}_k) = \left.\frac{\partial \underline{F}(\underline{s})}{\partial \underline{s}}\right|_{\underline{s}=\underline{s}_k} \tag{3.10}$$

The Jacobian is a crucial part of the algorithm as it determines the search direction for the next iteration. Since an analytical expression of the Jacobian is often not available in complex dynamical systems, the Jacobian may be approximated through numerical differentiation. This may for example be done with the forward difference method, given by

$$\underline{J}\left(\underline{s}_k\right) \approx \frac{\underline{F}\left(\underline{s}_k + \underline{h}\right) - \underline{F}\left(\underline{s}_k\right)}{\underline{h}} \tag{3.11}$$

Here, $\underline{h}$ is the provided step size of the state. This step size must be chosen carefully as very small values will lead to numerical inaccuracies, and very large values will result in systematic errors. The forward difference method is less accurate than, for example, the central difference method, since it considers only a single extra function evaluation for each element of the state vector, and the higher order terms of the approximation are omitted [30]. However, in optimization it is usually preferred to use the forward difference method as it requires half the function evaluations, which improves the computational efficiency.

Step size regulation

The combination of the inverse Jacobian and the residual errors is sufficient to ensure convergence [31]. However, the step size regulator α is added to improve the rate of convergence. α is defined as follows:

$$\alpha = \min\left(1, \gamma \left\|\underline{\varepsilon}\left(\underline{s}_k\right)\right\|^{-p}\right) \tag{3.12}$$

The step size regulator is inversely proportional to the Euclidean norm of the residual error at the previous iteration raised to a power p. This ensures that if the error is large, meaning that the point is far away from a solution, smaller steps are taken. As the solution gets closer to the stationary point, the step size increases. In order to avoid oscillatory behavior around the solution, the step size regulator is limited to an upper bound of 1. The constant γ is defined with the initial guess of the state

$$\gamma = \gamma_0 \left\|\underline{\varepsilon}\left(\underline{s}_0\right)\right\|^{p} \tag{3.13}$$

where the parameters γ_0 and p are predefined constants. The scaling factor is a positive parameter with $\gamma_0 \ll 1$. The step size p is typically chosen to be 2.

Premature termination conditions

The definition of the step size regulator may pose a problem if the residual error at the current iteration is significantly larger than the residual error of the initial guess. In this case, the scaling factor would approach zero and the algorithm would stall. For such occurrences, a termination condition is defined to avoid extreme computational times. The algorithm is terminated if the error of the current iteration is larger than the initial error by a factor ξ_1.

Another possible issue is that the Jacobian may become singular. This means that a change in one or more states does not result in a different function value, which also causes the algorithm to stall. The associated safety measure checks, whether the determinant of the Jacobian is below a tolerance ξ_2. If this is the case, the Jacobian is (near-)singular and the inverse cannot be accurately computed, and subsequently the algorithm is terminated.

Finally, a maximum number of iterations k_{max} is specified for which the algorithm may run. If the number of iterations exceeds this upper bound, the algorithm is terminated. In all three

cases, the algorithm does not converge. Mathematically, these premature termination conditions are denoted as follows:

$$\|\underline{\varepsilon}(\underline{s}_k)\| > \xi_1 \|\underline{\varepsilon}(\underline{s}_0)\| \tag{3.14}$$

$$\det(J(\underline{s}_k)) < \xi_2 \tag{3.15}$$

$$k > k_{max} \tag{3.16}$$

Successful termination condition

The termination condition that specifies convergence of the algorithm uses the Euclidean norm of the residual errors. When the residual errors are small, the associated state is in the vicinity of the solution $\underline{s}^*$. In a numerical method, this is specified with an error tolerance ϵ. The solution will not be exactly zero, but the number of significant digits that are correct are specified with the error tolerance. Therefore, sufficiently small error tolerances yield a solution with a desired precision. In this case, the algorithm converges. In mathematical terms, the successful termination condition is as follows:

$$\|\underline{\varepsilon}(\underline{s}_k)\| < \epsilon \tag{3.17}$$

3.3 Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is a mathematical optimization algorithm that is commonly used to solve non-linear optimization problems. It is an iterative algorithm that works by building a quadratic approximation of the objective function at the current iterate through a Taylor series, and then solving a subproblem that minimizes this approximation subject to the constraints of the original problem.

For a problem that only includes equality constraints, the following quadratic program is solved at each iteration:

$$\begin{aligned} & \delta\underline{x} = \begin{bmatrix}\delta x_1 & \delta x_2 & \cdots & \delta x_N\end{bmatrix}^T \\ \min_{\delta\underline{x}} \quad & f_k + \nabla f_k^T \delta\underline{x} + \frac{1}{2}\delta\underline{x}^T \nabla^2_{\underline{x},\underline{x}} L_k \delta\underline{x} \\ \text{s.t.} \quad & \underline{h}_k + \nabla \underline{h}_k \delta\underline{x} = \underline{0} \end{aligned} \tag{3.18}$$

The objective function is quadratically approximated, and a linear approximation is made of the equality constraints for $\underline{x}_{k+1} = \underline{x}_k + \delta\underline{x}$ around the point $\underline{x}_k$. If the KKT conditions (3.2)-(3.3), and the sufficient condition (3.6) hold, the unique solution to this quadratic program satisfies [31]

$$\nabla^2_{\underline{x},\underline{x}} L_k \delta\underline{x}_k + \nabla f_k - \nabla \underline{h}_k^T \underline{\lambda}_{h,k+1} = \underline{0} \tag{3.19}$$

$$\nabla \underline{h}_k \delta\underline{x}_k + \underline{h}_k = \underline{0} \tag{3.20}$$

The solutions to this problem are then used to calculate the value of the design variables and Lagrange multipliers at the new iteration. A common approach is to use Newton iterations. A simplified version of (3.8) is used here:

$$\begin{bmatrix}\underline{x}_{k+1} \\ \underline{\lambda}_{h,k+1}\end{bmatrix} = \begin{bmatrix}\underline{x}_k \\ \underline{\lambda}_{h,k}\end{bmatrix} + \begin{bmatrix}\delta\underline{x} \\ \delta\underline{\lambda}_h\end{bmatrix} \tag{3.21}$$

where the difference between subsequent iterations $\delta\underline{x}$ and $\delta\underline{\lambda}_h$ can be described through the Jacobian of a vector composed of KKT conditions (3.19) and (3.20), resulting in the following

problem:

$$\begin{bmatrix} \nabla^2_{\underline{x},\underline{x}} L_k & -\nabla \underline{h}_k^T \\ \nabla \underline{h}_k & 0 \end{bmatrix} \begin{bmatrix} \delta \underline{x} \\ \delta \underline{\lambda}_h \end{bmatrix} = \begin{bmatrix} -\nabla f_k + \nabla \underline{h}_k^T \underline{\lambda}_{h,k} \\ -\underline{h}_k \end{bmatrix} \tag{3.22}$$

By subtracting $\nabla \underline{h}_k^T \underline{\lambda}_{h,k}$ from the first equation, one arrives at the following result:

$$\begin{bmatrix} \nabla^2_{\underline{x},\underline{x}} L_k & -\nabla \underline{h}_k^T \\ \nabla \underline{h}_k & 0 \end{bmatrix} \begin{bmatrix} \delta \underline{x} \\ \underline{\lambda}_{h,k+1} \end{bmatrix} = \begin{bmatrix} -\nabla f_k \\ -\underline{h}_k \end{bmatrix} \tag{3.23}$$

For a non-singular matrix, the change of the design vector $\delta \underline{x}$ and the Lagrange multipliers at the next iteration $\lambda_{h,k+1}$ can be found. The design variables at the next iteration are found by using the solution of $\delta \underline{x}$ in (3.21). The resulting solution is used as the next iterate. The process is repeated until convergence is achieved.

One can include inequality constraints by adding a linear approximation of the inequality constraints to the quadratic subproblem (3.18) and to the Newton iterations (3.21). The Lagrange multipliers at the next iteration $\underline{\lambda}_{g,k+1}$ are then found as a solution to the quadratic subproblem.

SQP has been shown to be very effective for solving a wide range of non-linear optimization problems. One of the advantages of SQP is that it can handle both equality and inequality constraints. In addition, it is able to exploit the structure of the problem by using the derivative information of the objective function and the constraints. The resulting quadratic subproblem can be solved very efficiently to achieve faster convergence than other optimization algorithms [31].

One of the challenges of SQP is that it can be computationally expensive, particularly for large-scale problems since the first and second order derivatives have to be determined. However, advances in computing technology and optimization algorithms, e.g. the approximation of the Hessian, have made it possible to apply SQP to increasingly more complex problems.

Overall, SQP is a powerful and versatile optimization algorithm that has been widely used in scientific and engineering applications. Its ability to handle both equality and inequality constraints, as well as its ability to exploit the structure of the problem, make it a valuable tool for solving non-linear optimization problems.

3.4 Summary

In this chapter, optimization theory has been discussed. Solutions to an optimization problem must satisfy the KKT conditions and the sufficient condition on the Hessian of the Lagrangian. A distinction is made between gradient-based algorithms that use derivative information, and gradient-free algorithms that use alternative methods like evolution. The issues with global convergence of gradient-based algorithms is elaborated on, and a solution is proposed to bypass this drawback. Finally, the root finding and gradient-based algorithms, Newton-Raphson and Sequential Quadratic Programming respectively, are explained.

4 Nested optimization

In [21], Beckers describes a gradient-based optimization routine for different vehicle models in steady-state cornering conditions. This research successfully found optimal solutions for the driving inputs and some vehicle parameters. However, this model also optimizes vehicle states, such as the chassis slip angle β. In this section, a method is proposed that combines gradient-based optimization and root finding algorithms that mitigates the necessity for optimizing vehicle states. With this change, the aim is to evaluate whether this method can be incorporated into a lap time simulation algorithm.

4.1 Optimization problem formulation

The optimization problem formulation is split up into two parts: a gradient-based optimization of the vehicle inputs and a root finding algorithm for solving the equations of motion. The gradient-based optimization problem formulation takes the form of (3.1) and is as follows:

$$\begin{aligned} & \underline{x} = \begin{bmatrix} \delta & M_{D_1} & M_{D_2} \end{bmatrix}^T, \\ \min_{\underline{x}} \quad & f(\underline{x}) = -V, \\ \text{s.t.} \quad & h(\underline{x}) = \frac{V}{r} - R = 0. \end{aligned} \tag{4.1}$$

It is desired to maximize the resultant velocity of the vehicle by optimizing the vehicle inputs (steering angle δ and motor torque M_D on each wheel). As the state is calculated inside the objective function, the only constraint of this optimization is the requirement that the vehicle should drive on a corner with a constant prescribed radius R. Beckers [21] requires that some of the constraints are the steady-state conditions of the vehicle. However, in this algorithm, steady-state is ensured by using a root finding algorithm.

For every step of the gradient-based optimization, the following root finding algorithm is solved:

$$\begin{aligned} & \underline{s} = \begin{bmatrix} u & v & r & \omega_1 & \omega_2 \end{bmatrix}^T, \\ \text{s.t.} \quad & F_1(\underline{s}) = F_{x_1}\cos\delta + F_{x_2} - F_{y_1}\sin\delta + mvr - F_D = 0, \\ & F_2(\underline{s}) = F_{y_1}\cos\delta + F_{y_2} + F_{x_1}\sin\delta - mur = 0, \\ & F_3(\underline{s}) = a\left(F_{y_1}\cos\delta + F_{x_1}\sin\delta\right) - bF_{y_2} + M_{z_1} + M_{z_2} = 0, \\ & F_4(\underline{s}) = M_{D_1} - F_{x_1}r_{L_1} + M_{rr_1} = 0, \\ & F_5(\underline{s}) = M_{D_2} - F_{x_2}r_{L_2} + M_{rr_2} = 0. \end{aligned} \tag{4.2}$$

The design vector consists of the longitudinal velocity u, lateral velocity v, and yaw velocity r of the center of gravity, and the rotational velocities of each wheel ω. The non-linear equations that should be solved are the steady-state variants of the longitudinal (2.13), lateral (2.14), yaw (2.17), and wheel dynamics (2.9) equations of motion. The states of the vehicle model are iteratively determined such that the steady-state conditions of the vehicle are satisfied. From these states, the objective and constraint values of the gradient-based optimization problem are calculated. In case the Newton-Raphson algorithm reaches a premature termination condition (3.14)-(3.16), the objective value is set to zero, which is the highest physically possible solution. Even though this creates a discontinuity in the objective function, this method turns out to be more reliable than passing incorrect data to the algorithm and will ensure that the gradient-based optimization steers away from spurious points.

4.2 Results comparison with direct optimization

A comparison is made between the routine in [21], from now on called "direct optimization" with a design vector of $\underline{x} = \begin{bmatrix} V & \beta & \delta & \mathrm{SR}_1 & \mathrm{SR}_2 \end{bmatrix}^T$, and the nested optimization with the design vector consisting of only vehicle inputs $\underline{x} = \begin{bmatrix} \delta & M_{D_1} & M_{D_2} \end{bmatrix}^T$. Both optimization methods are different formulations of the same problem: finding the maximum resultant velocity in steady-state for a constant radius corner. Neither method will have any additional constraints besides the steady-state equations (direct optimization) and the corner radius constraint (nested optimization).

With the direct optimization, two main solutions are found, the so called gripping and drifting conditions. The gripping condition refers to a solution, where both tires operate in the slip angle region before the peak of the lateral force called the gripping region. Since the LMP2 rear tire allows for high performance combined slip behavior, also a drifting condition was found. In this condition the front tire is in the gripping region, whereas the rear tire operates at a slip angle that is larger than the peak slip angle called the drifting region.

The Newton-Raphson algorithm in the nested optimization requires an initial guess of the vehicle state. This is chosen to be a kinematic movement, similar to the initial guess chosen in [28]:

$$\underline{s}_0 = \begin{bmatrix} u_0 & 0 & r_0 & \frac{u_0 \cos\delta + a r_0 \sin\delta}{r_{L_1}} & \frac{u_0}{r_{L_2}} \end{bmatrix}^T \tag{4.3}$$

where $u_0 = \sqrt{\mu g R}$ and $r_0 = \frac{u_0}{R}$. This corresponds to a point mass vehicle, without downforce, that exhibits pure centripetal motion at the grip limit of the tire (2.4). This should allow the algorithm to find at least the same gripping condition as was found with the direct optimization.

The Newton-Raphson algorithm is configured with the parameters listed in Table 4.1.

Table 4.1: Parameters of the Newton-Raphson algorithm.

Parameter	**Value**
Numerical differentiation	
$\underline{h}$	$\begin{bmatrix} 10^{-2}\ [\mathrm{m/s}] & 10^{-4}\ [\mathrm{m/s}] & 10^{-4}\ [\mathrm{rad/s}] & 10^{-2}\ [\mathrm{rad/s}] & 10^{-2}\ [\mathrm{rad/s}] \end{bmatrix}^T$
Step size regulation	
p	2 [-]
γ_0	0.03 [-]
Termination conditions	
ξ_1	1.2 [-]
ξ_2	10^{-6} [-]
k_{max}	300 [-]
ϵ	10^{-3} [-]

The optimization framework in use is the MATLAB function `fmincon` [32], which is configured with the settings displayed in Table 4.2.

Table 4.2: Settings of the optimization problem.

Setting	**Value**
`fmincon` settings	
Algorithm	SQP
Finite difference step size	10^{-4}
Optimality tolerance	10^{-3}
Function tolerance	10^{-6}
Step tolerance	10^{-6}
Constraint tolerance	10^{-6} (Direct), 10^{-1} (Nested)
Design vector boundaries	
δ	$[-30, 30]$ [deg]
M_{D_1} & M_{D_2}	$[0, 5000]$ [Nm]

The build-in SQP algorithm of the MATLAB function `fmincon` follows the algorithm described in Section 3.3. However, changes have been made to improve convergence and computational time [32]. For the initial conditions random linear samples are generated using Latin hypercube sampling [33].

The vehicle parameters can be found in Table 4.3. The Magic Formula parameters of the front and rear tire are omitted for confidentiality reasons.

Table 4.3: List of vehicle parameters.

Parameter	**Description**	**Value**
	Mass	
m	Vehicle mass	995 [kg]
g	Gravitational acceleration	9.81 [m/s^2]
W_d	Weight distribution	47.58 [%]
L	Wheelbase	2.98 [m]
	Aerodynamics	
k_D	Drag coefficient	0.6099 [N/(m^2/s^2)]
k_L	Lift coefficient	3.3067 [N/(m^2/s^2)]
D_d	Downforce distribution	48.80 [%]
	Wheels	
f_{rr_1} & f_{rr_2}	Rolling resistance coefficient	0.02 [-]
r_{L_1} & r_{L_2}	Loaded radius	0.353 [m]

4.2.1 Gripping condition

A comparison of the direct and nested optimization results can be found in Figure 4.1.

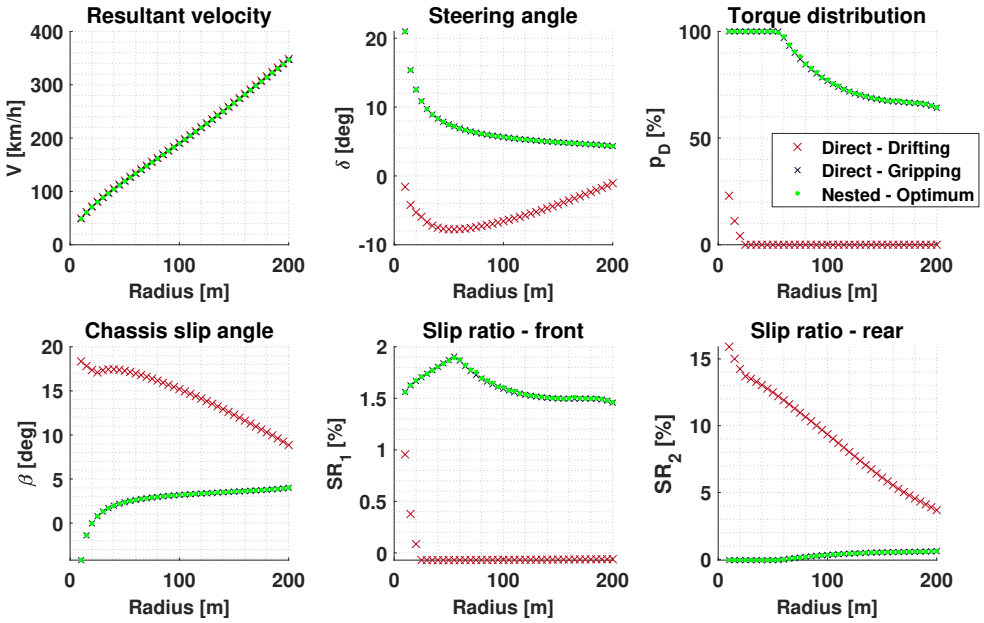


Figure 4.1: Comparison between direct and nested optimization. Direct optimization design variables.

As can be seen in Figure 4.1, the optimum found by the nested optimization is very similar to the gripping condition found by the direct optimization. However, the slightly more optimal drifting condition is not found with the current nested optimization. This is likely due to the choice of initial conditions for the Newton-Raphson algorithm given in (4.3). Since this state defines a kinematic driving condition, any negative values for the steering angle, which are required for a drifting condition as seen in Figure 4.1, will result in a steady-state condition for a vehicle turning in the opposite direction. Because of this the steering angle is changed into positive values associated with the local gripping domain.

Two very important key performance indicators (KPI) of the algorithm performance are the (average) computational time per corner radius and the success rate of finding an optimum. The computational time is device specific, but can be compared if optimizations are run under similar conditions on the same device. The success rate of the algorithm does not depend on the type of device that is used. It specifies the percentage of initial guesses that result in finding an optimum. These KPIs are displayed in Figure 4.2.

Compared to the direct optimization, the nested optimization is severely penalized in terms of computational time, which is slower by a factor 10 on average, as shown in Table 4.4. This is to be expected as the nested optimization combines gradient-based optimization and root finding, which is computationally more demanding than the single gradient-based optimization in the direct optimization. However, the success rate is higher, so the nested optimization is able to find an optimum more often. This means that the nested optimization requires less initial conditions than the direct optimization to have the same likelihood of finding an optimal solution, which in return will improve the overall computational time. Further improvements can also be made if the initial conditions for the design variables and the vehicle states are chosen

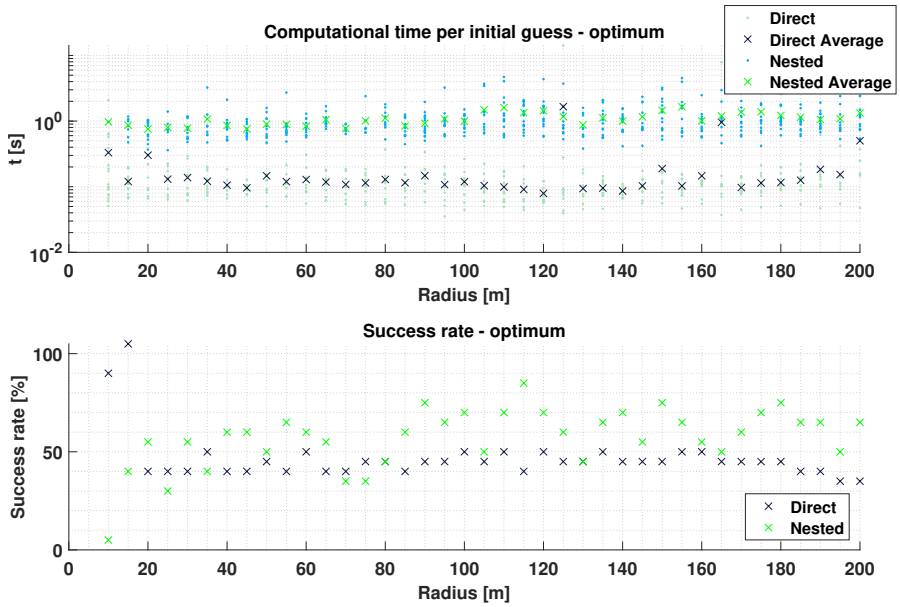


Figure 4.2: Comparison between direct and nested optimization. Performance of the optimization strategies for only the optima.

with the knowledge of the discussion presented in [21], instead of a random selection of points.

Table 4.4: Comparison of the KPIs of the algorithm performance between the direct and nested optimization.

Condition	Average comp. time per initial guess [s]	Average success rate [%]
Direct optimization		
General	0.200	46.7
Nested optimization		
Gripping	1.067	57.1
Drifting	1.237	25.3

4.2.2 Drifting condition

The nested optimization algorithm is also tested on its ability to find the drifting condition. This is done by taking the initial guess of the vehicle state $\underline{s}_0$ to be the drifting condition found with the direct optimization for each radius. The results for the nested optimization are displayed in Figure 4.3.

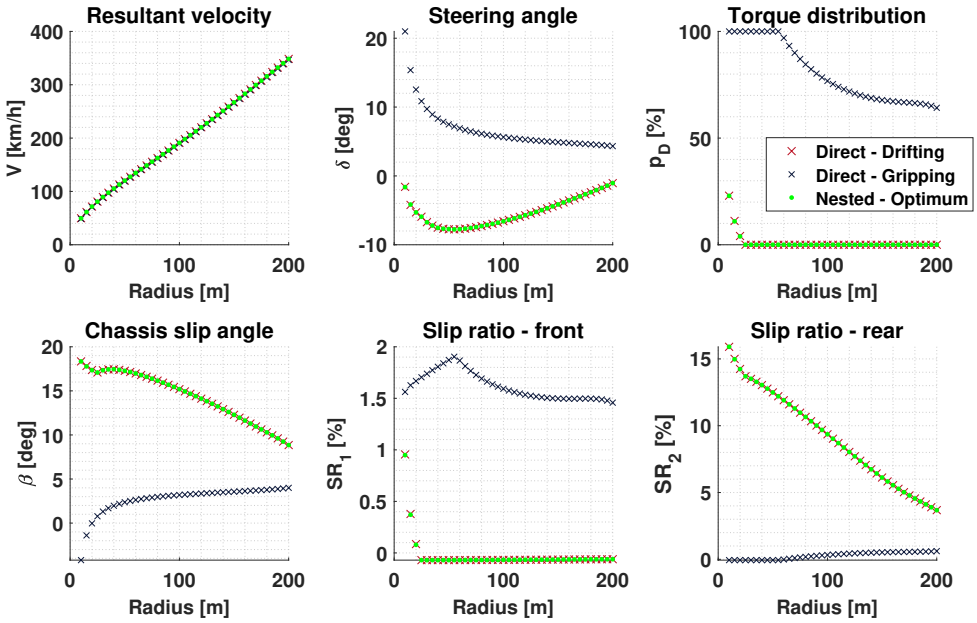


Figure 4.3: Comparison between direct and nested optimization. All solutions to the direct and nested optimization algorithms. The initial guess to the Newton-Raphson algorithm is the drifting condition found with the direct optimization.

The trend line appears to be similar to that of the drifting condition found with the direct optimization. The performance of the nested and direct optimization algorithms for the drifting condition is compared in Figure 4.4.

The computational times are comparable for both gripping and drifting conditions found with the nested optimization, as can be seen in Table 4.4. However, the success rate has gone down significantly and is worse than the success rate of the direct optimization, even though the provided initial condition of the Newton-Raphson algorithm is the final solution. This condition, however, has a very narrow operating window for the vehicle inputs. For most random vehicle inputs, the solution to the steady-state equations is a gripping condition. As the initial conditions to the gradient-based algorithm are randomly sampled through Latin hybercube sampling, the Newton-Raphson algorithm will have a tendency to result in a gripping condition even if it is initialized with a drifting solution.

A likely improvement would, again, be a better choice of the initial guess for both the gradient-based algorithm and a less deterministic initial solution to the root solver.

4.2.3 Overall verdict

The introduction of the root finding algorithm in the objective function and the constraint creates discontinuities in the problem structure. As a result, the numerical calculations of the derivative do not follow a predefined curve and may become inaccurate. This negatively influences both the convergence and the computational time of the algorithm. A recommendation would be to use a gradient-free optimization algorithm as framework instead, similar to the recom-

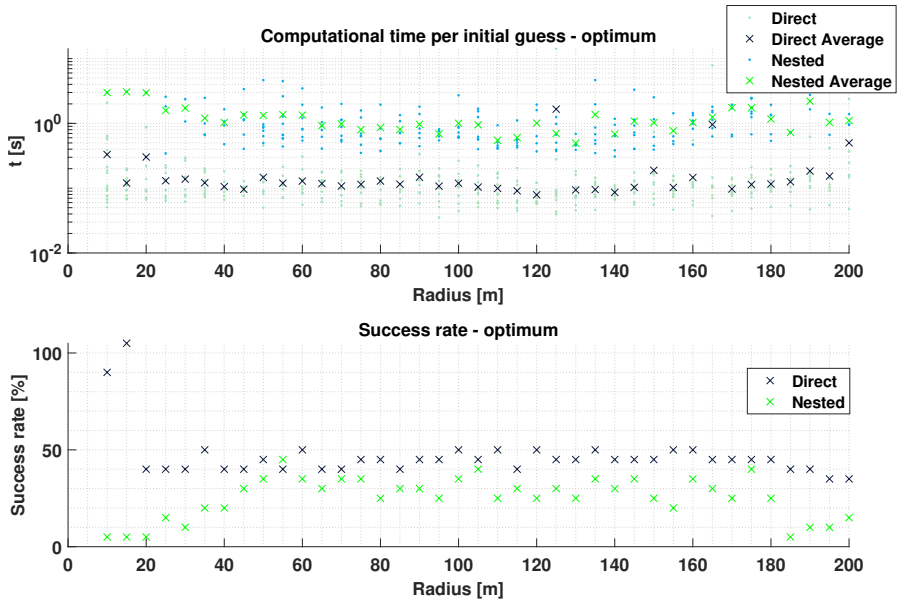


Figure 4.4: Comparison between direct and nested optimization. Performance of the optimization strategies for only the optima. Initial guess to the Newton-Raphson algorithm is the drifting condition found with the direct optimization.

mendation made in [21]. The gradient-free algorithms do not suffer from non-continuity of the optimization problems, and therefore might not experience the same issues with computational time and convergence. Due to the increased computational times and inconsistent convergence, the methodology was deemed to be unsuitable for integration in a lap time simulation algorithm.

4.3 Summary

A nested optimization algorithm has been created to optimize vehicle inputs during steady-state cornering. The algorithm consists of an outer loop, a gradient-based method that maximizes the resultant velocity by computing the optimal steering angle δ and motor torques M_D, and an inner loop, a Newton-Raphson algorithm that ensures that the vehicle is in steady-state conditions. This was compared to the direct optimization method described in [21].

The nested optimization was able to find similar gripping and drifting solutions, albeit for different initial conditions of the Newton-Raphson algorithm. Both gripping and drifting solutions are computationally slower by a factor of 10 for one outer loop iteration compared to the direct optimization. However, the gripping condition of the nested optimization yields a better success rate at finding the optimum compared to the direct optimization with an average success rate of 57.1%. The drifting condition results in a lower average success rate of 25.3%. A recommendation to improve the use of the nested optimization is to make better definitions for the initial conditions of the inner and outer loop. It was chosen not to implement the nested optimization method in a lap time simulation algorithm.

5 Optimization of lap time simulation

Lap time simulation (LTS) is a widely used tool in motorsports to aid vehicle setup and driver training. The conventional steady-state LTS algorithms are computationally fast for quick iterations with different vehicle parameters, but lack the description of important physical phenomena that occur while driving a vehicle at high velocity [4] [5]. In this section, a conventional LTS algorithm is improved by increasing the vehicle model complexity and incorporating the usage of optimization theory. The updated algorithm is described as well as the required optimization problems and a description of the potential minimizers through visual inspection of the adapted GG diagram. Finally, some recommendations are given for further increasing the vehicle model complexity.

5.1 Modeling lap time simulation

Lap time simulation can be split into two parts: the forward loop and the reverse loop. The forward loop defines the general shape of the velocity curve throughout the track, and the reverse loop corrects the acceleration. The profiles are generated using simple standard equations of straight-line motion, and the vehicle is assumed to drive through the track in quasi steady-state. This allows for a very fast calculation of the solution.

5.1.1 Model inputs

The LTS algorithm requires vehicle parameters and a track layout. The vehicle parameters are the constant parameters defined in Chapter 2 and will remain constant during the entire simulation. The track layout has a spatially discretized road curvature κ, which is the inverse of the corner radius R, consisting of N data points. This curvature defines the fixed racing line that the driver will follow in the LTS. To avoid inaccuracies, typically a track discretization of $\Delta s \leq 1$ m is used.

5.1.2 Forward loop

One can calculate the velocity of the vehicle on the track in the corners and under acceleration onto a straight through the forward loop method. The forward loop iterates from an initial velocity chosen at the starting line through the track until it reaches the finish line. At each iteration, it is checked whether the tires are able to generate the required forces to achieve the lateral acceleration based on the proposed velocity at that point. This means that the combination of longitudinal force F_x and lateral force F_y must not exceed the bounds of the friction ellipse given in (2.4).

In the case where the proposed velocity V_i creates a lateral acceleration $a_{y,i}$ that is infeasible for the tire, the velocity and lateral acceleration are reduced, such that the tire is at its grip limit. This ensures that the available lateral grip of the tire is used to its full potential through the corner.

The limiting factor in the longitudinal direction is determined next, which is either the tire or the powertrain. This is done by comparing the remaining longitudinal tire force through the friction ellipse with the available tractive force determined by the powertrain. The smaller of the two will be the tractive force at that point of the track, and will subsequently result in the longitudinal acceleration $a_{x,i}$.

With the (corrected) velocity V_i and the calculated longitudinal acceleration $a_{x,i}$ of the current iteration, the predicted velocity of the next iteration V_{i+1} can be calculated. This is done through the following equation

$$V_{i+1} = \sqrt{V_i^2 + 2\Delta s a_{x,i}} \tag{5.1}$$

This is derived from the standard displacement ($s_{i+1} = s_i + V_i\Delta t + \frac{1}{2}a_{x,i}\Delta t^2$) and velocity ($V_{i+1} = V_i + a_{x,i}\Delta t$) formulas at iterations i and $i+1$. The same steps are then executed for each track data point, until the finish line is reached.

5.1.3 Reverse loop

In the forward loop, instant deceleration is possible at any point, as can be seen in Figure 5.1. However, a vehicle is also grip limited during braking, which means that the calculations inside a braking zone need to be corrected. This is done in the reverse loop, which iterates through the circuit starting from the finish line and reverses towards the start line. At any point, the longitudinal acceleration is corrected to ensure that consistency between the velocity and acceleration is maintained. Therefore, the corrected longitudinal acceleration of the reverse loop can both be positive or negative.

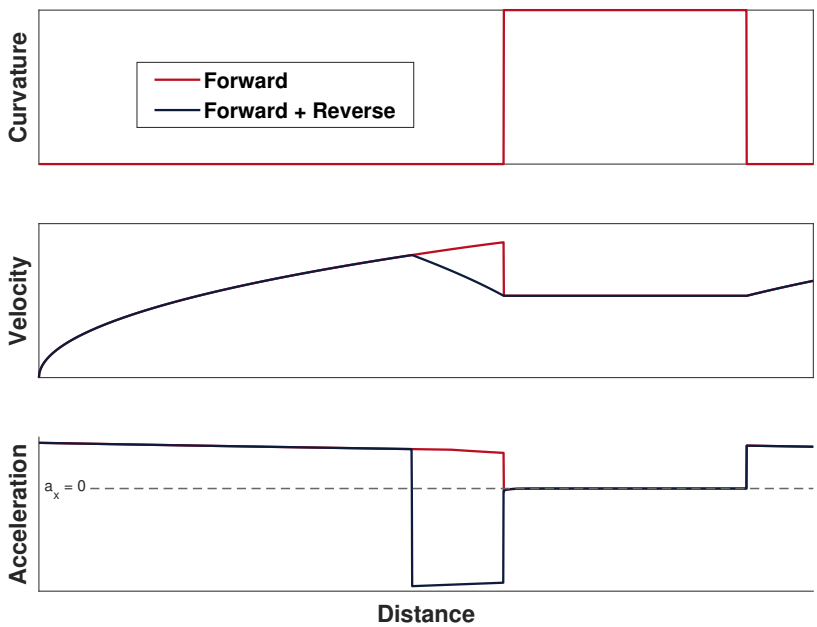


Figure 5.1: Contribution of forward and reverse loop to the LTS process. The forward loop creates the shape of the velocity through the track, and the reverse loop ensures consistency between the velocity and longitudinal acceleration. This was created with the model described in [34].

At each iteration of the reverse loop, the maximum possible longitudinal force for deceleration is determined based on the available grip from the friction ellipse (2.4), which is then used to

calculate the longitudinal acceleration $a_{x,i-1}$ at iteration $i-1$. The proposed maximum possible velocity is then given by a rewritten version of (5.1).

$$V_{i-1} = \sqrt{V_i^2 - 2\Delta s a_{x,i-1}} \tag{5.2}$$

If the value of V_{i-1} that is calculated in the forward loop exceeds this threshold, the vehicle is not able to decelerate as quickly as the forward loop suggests. The velocity is then limited to the threshold value and the lateral acceleration is corrected using

$$a_{y,i-1} = V_{i-1}^2 \kappa_{i-1}. \tag{5.3}$$

If the velocity found in the forward loop does not exceed the threshold, the value of $a_{x,i-1}$ calculated in the reverse loop is invalid and is corrected to be:

$$a_{x,i-1} = \frac{1}{2\Delta s}\left(V_i^2 - V_{i-1}^2\right) \tag{5.4}$$

which, again, follows from (5.1). If V_{i-1} from the forward loop is smaller than V_i calculated in the previous iteration, the $a_{x,i-1}$ will be positive and the vehicle will accelerate. Otherwise, $a_{x,i-1} < 0$ and the vehicle will decelerate.

5.1.4 Lap time calculation

Between every data point, the time taken to travel the fixed distance Δs may be calculated with the average velocity between points, because the longitudinal acceleration is assumed to be constant between subsequent points. The total lap time is the summation of all of these sections:

$$t_{lap} = \sum_{i=1}^{N-1} \frac{\Delta s}{\frac{1}{2}\left(V_{i+1} + V_i\right)} \tag{5.5}$$

5.2 Usage of optimization theory

A drawback of conventional LTS algorithms is the usage of relatively simple point mass vehicle models. These vehicle models allow for a minimal computational effort, but do not take some physical phenomena like load transfer into account that are crucial in capturing the actual behavior of a vehicle pushing its velocity to the limit. To allow for more complex vehicle models in the LTS algorithm, this work uses optimization theory to determine the optimal velocity V and longitudinal acceleration a_x at each point of the track based on the track curvature κ. It was chosen to extend the vehicle model to a single-track model with an elevated center of gravity as this includes longitudinal load transfer and considers unequal weight, grip, torque, and downforce distributions between the front and rear axle. The powertrain model remains a constant power model.

5.2.1 Forward loop

The forward loop remains mostly the same in comparison to the conventional algorithm. The only change is that the assignment of the optimal velocity and longitudinal acceleration at each point is replaced by two optimization problems. A flow chart of the new algorithm is presented in Figure 5.2.

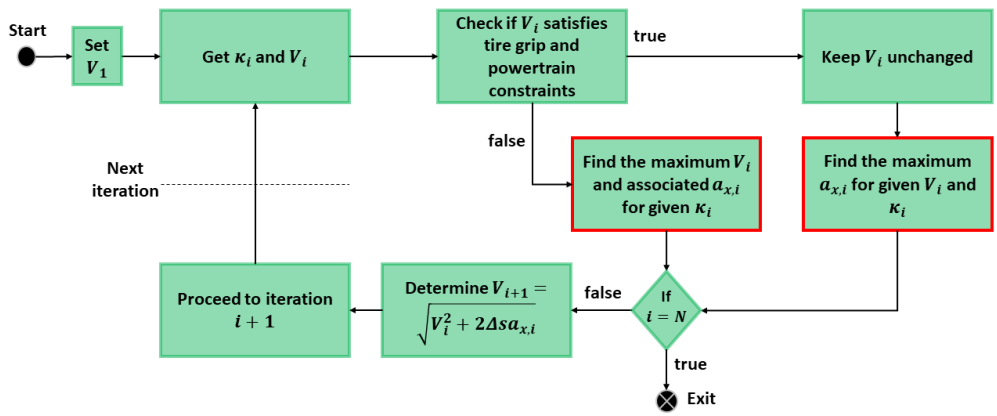


Figure 5.2: Flow chart of the forward loop process. The boxes with a red outline have a corresponding optimization problem.

One starts by setting an initial condition for the velocity in the first iteration. At iteration i, the curvature κ_i is known and the velocity V_i was calculated in iteration $i-1$. The maximum possible velocity at this point of the track is determined. This is equivalent to finding the minimizers of the second optimization problem, which will be covered in Section 5.3. The maximum velocity shall always satisfy the grip limit of both tires and the power limit. If the estimation of V_i from iteration $i-1$ is lower than the maximum possible velocity, the vehicle is assumed to be accelerating in the longitudinal direction. As a result, one tries to find the maximum longitudinal acceleration $a_{x,i}$ such that one of the tires or the powertrain are used to its full potential for the given V_i and κ_i. The equivalent optimization problem is as follows:

$$\begin{aligned} & x = a_{x,i} \\ \min_{x} \quad & f(x) = -a_{x,i}^2 \\ \text{s.t.} \quad & g_1(x) = F_{x_{1,i}}^2 + F_{y_{1,i}}^2 - \left(\mu_1 F_{z_{1,i}}\right)^2 \leq 0 \\ & g_2(x) = F_{x_{2,i}}^2 + F_{y_{2,i}}^2 - \left(\mu_2 F_{z_{2,i}}\right)^2 \leq 0 \\ & g_3(x) = m a_{x,i} - F_{M,i} + F_{D,i} + F_{rr,i} \leq 0 \\ & 0 \leq a_{x,i} < \infty \end{aligned} \tag{5.6}$$

The optimization problem has one design variable ($a_{x,i}$) and tries to find the maximum acceleration, which is equivalent to finding the minimum of the acceleration multiplied by -1, such that the equations for the grip limit of the front and rear tires given by (2.4) (g_1 & g_2) and the power limit of the powertrain given by (2.11) (g_3) are satisfied. The determination of the minimizer of this optimization problem is discussed in Section 5.3.

If the estimation of V_i exceeds the calculated maximum velocity at that point, both the velocity V_i and longitudinal acceleration $a_{x,i}$ have to be adapted such that the velocity is still as high as

possible. This is done in the following equivalent optimization problem:

$$
\begin{aligned}
& \underline{x} = \begin{bmatrix} V_i & a_{x,i} \end{bmatrix}^T \\
\min_{\underline{x}} \quad & f(\underline{x}) = -V_i^2 \\
\text{s.t.} \quad & g_1(\underline{x}) = F_{x_{1,i}}^2 + F_{y_{1,i}}^2 - \left(\mu_1 F_{z_{1,i}}\right)^2 \leq 0 \\
& g_2(\underline{x}) = F_{x_{2,i}}^2 + F_{y_{2,i}}^2 - \left(\mu_2 F_{z_{2,i}}\right)^2 \leq 0 \\
& g_3(\underline{x}) = m a_{x,i} - F_{M,i} + F_{D,i} + F_{rr,i} \leq 0 \\
& 0 \leq V_i < \infty \\
& 0 \leq a_{x,i} < \infty
\end{aligned}
\tag{5.7}
$$

In this optimization problem, two design variables (V_i and $a_{x,i}$) are determined such that the velocity is maximized, or the velocity times -1 is minimized, and the equations for the grip limit of both tires given by (2.4) (g_1 & g_2) and the power limit of the powertrain given by (2.11) (g_3) are not violated. The determination of the minimizer of this optimization problem is discussed in Section 5.3.

After solving one of these optimization problems, one has the optimal velocity V_i and longitudinal acceleration $a_{x,i}$ at iteration i. Subsequently, if the algorithm has not reached the finish line, the velocity at iteration $i+1$ is calculated with (5.1) and one proceeds to the next iteration.

5.2.2 Reverse loop

For more complex vehicle models that include longitudinal load transfer, the correction of the longitudinal acceleration becomes ever more important as this changes the amount of available friction force that can be developed by the tires, which results in a change in maximum possible velocity. This means that the reverse loop algorithm is slightly modified compared to the conventional algorithm. The optimal braking performance is determined through an optimization problem. The other case, where the longitudinal acceleration is corrected to match the subsequent velocities is no longer valid. The correction of $a_{x,i-1}$ through (5.4) will change the developed friction force, and might sometimes create an infeasible solution when on the limit of grip. Therefore, this was also changed to an optimization problem. A flow chart of this process is depicted in Figure 5.3.

One starts by taking the velocity at the final data point V_N calculated by the forward loop, and will move backwards throughout the track until the starting line is reached. At iteration i one knows the curvature of the track at the previous data point κ_{i-1}, and inherits the value of the velocity at the current data point V_i, which was calculated in the previous iteration $i+1$. With this information, the possible combinations of velocity and longitudinal acceleration at iteration $i-1$ will be calculated using the following two equivalent optimization problems (5.8) and (5.9) for braking and traction, respectively. The optimization problem for braking is as follows:

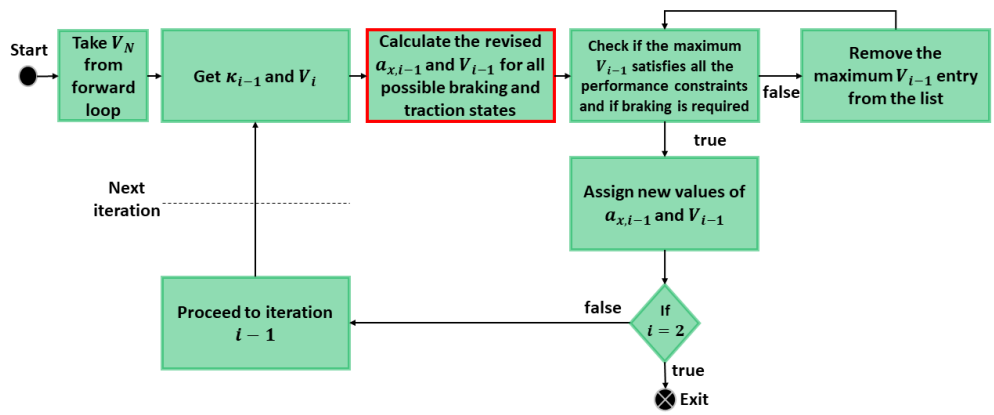


Figure 5.3: Flow chart of the reverse loop process. The box with a red outline has a corresponding optimization problem.

$$
\begin{aligned}
& \underline{x} = \begin{bmatrix} V_{i-1} & a_{x,i-1} \end{bmatrix}^T \\
\min_{\underline{x}} \quad & f(\underline{x}) = -V_{i-1}^2 \\
\text{s.t.} \quad & g_1(\underline{x}) = F_{x_{1,i-1}}^2 + F_{y_{1,i-1}}^2 - \left(\mu_1 F_{z_{1,i-1}}\right)^2 \leq 0 \\
& g_2(\underline{x}) = F_{x_{2,i-1}}^2 + F_{y_{2,i-1}}^2 - \left(\mu_2 F_{z_{2,i-1}}\right)^2 \leq 0 \\
& g_3(\underline{x}) = m a_{x,i-1} + F_{D,i-1} \leq 0 \\
& h_1(\underline{x}) = V_{i-1}^2 - V_i^2 + 2\Delta s a_{x,i-1} = 0
\end{aligned}
\tag{5.8}
$$

In this optimization problem, V_{i-1} and $a_{x,i-1}$ are chosen such that the velocity at iteration $i-1$ is maximized, or the velocity multiplied by -1 is minimized. In the meantime, at iteration $i-1$ the solution must also adhere to the grip limits of both tires defined in (2.4) (g_1 & g_2). Since one is braking, the power limit is not included in this optimization problem and the deceleration must be lower than the deceleration caused by the aerodynamic drag force (g_3). Moreover, an additional equality constraint is added that maintains the consistency between the velocity and acceleration of iterations $i-1$ and i (h_1), which is given in (5.2). The possible minimizers to this optimization problem are discussed in Section 5.3.

The optimization problem for traction is as follows:

$$
\begin{aligned}
& \underline{x} = \begin{bmatrix} V_{i-1} & a_{x,i-1} \end{bmatrix}^T \\
\min_{\underline{x}} \quad & f(\underline{x}) = -V_{i-1}^2 \\
\text{s.t.} \quad & g_1(\underline{x}) = F_{x_{1,i-1}}^2 + F_{y_{1,i-1}}^2 - \left(\mu_1 F_{z_{1,i-1}}\right)^2 \leq 0 \\
& g_2(\underline{x}) = F_{x_{2,i-1}}^2 + F_{y_{2,i-1}}^2 - \left(\mu_2 F_{z_{2,i-1}}\right)^2 \leq 0 \\
& g_3(\underline{x}) = m a_{x,i-1} - F_{M,i-1} + F_{D,i-1} + F_{rr,i-1} \leq 0 \\
& g_4(\underline{x}) = -m a_{x,i-1} - F_{D,i-1} \leq 0 \\
& h_1(\underline{x}) = V_{i-1}^2 - V_i^2 + 2\Delta s a_{x,i-1} = 0 \\
& h_2(\underline{x}) = g_1(\underline{x}), g_2(\underline{x}), g_3(\underline{x}) = 0
\end{aligned}
\tag{5.9}
$$

One calculates the design variables V_{i-1} and $a_{x,i-1}$ as to maximize the velocity at iteration $i-1$, which is equivalent to minimizing the velocity times -1. This problem is subjected to the equations for maximum tire grip of the front and rear tire given by (2.4) (g_1 & g_2), and the equation of maximum power usage by the powertrain given in (2.11) (g_3). Moreover, the total longitudinal acceleration must be larger than the deceleration due to drag, but may still be negative (g_4). In addition, (5.2) is included as an equality constraint to ensure consistency of the velocity and acceleration between iterations $i-1$ and i (h_1). Depending on which performance constraint g_1, g_2, or g_3 intersects equality constraint h_1 without violating any of the other constraints, that constraint is chosen to become active through equality constraint h_2. This ensures that the vehicle remains on the performance limit.

Next up, the possible solutions to the above optimization problems are then sorted from maximum to minimum velocity. This means, that, due to constraint h_1, the two braking solutions will always be listed first, followed by the four solutions for traction. Iteratively, the entry with the maximum velocity in the set is then used to verify if all constraints are satisfied, and, in case of braking, whether the calculated velocity at iteration $i-1$ is lower than the velocity found in the same iteration in the forward loop. If the entry violates one of these conditions, it is removed from the set and the next entry with the new highest velocity is evaluated. This process is repeated until a candidate is found that passes all the tests. The values of V_{i-1} and $a_{x,i-1}$ of this candidate are then stored. Unless the process has reached the start line at which it terminates, the algorithm will continue to the next iteration $i-1$.

5.3 Determination of minimizers

Analytical expressions for the minimizer to the optimization problem will improve computational efficiency. An analytical solution only requires a single function evaluation, and is therefore a more favorable option than for example iteration, which requires multiple function calls. Using the shape of the optimization problem, all possible minimizers of each optimization problem will be investigated. To aid this, the GG diagram will be explained and modified to better suit the optimization problem.

5.3.1 Conventional GG diagram

In motorsports, the GG diagram is an essential tool for assessing the vehicle's performance [35]. The GG diagram is a two-dimensional scatter plot with the lateral acceleration a_y on the horizontal axis and the longitudinal acceleration a_x on the vertical axis. An example of a conventional

GG diagram is given in Figure 5.4. The figure consists of measurement data taken from race car telemetry and the theoretical performance constraints for a constant velocity.

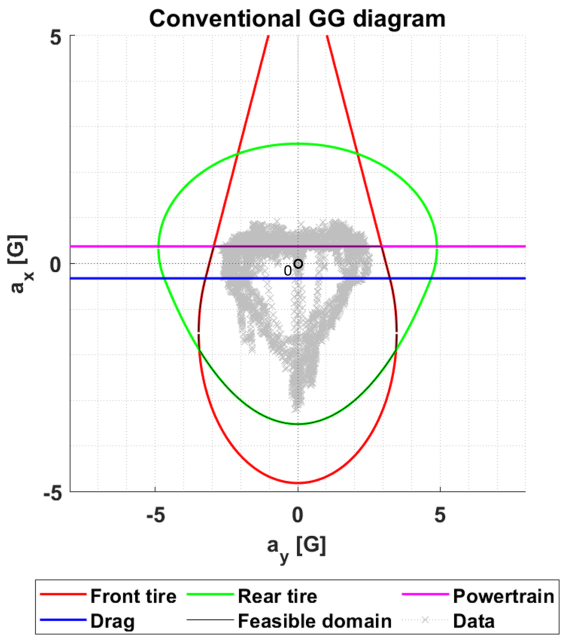


Figure 5.4: Visualization of a conventional GG diagram. The telemetry data is given in crosses, and the theoretical constraints are given in solid lines.

In essence, the higher up/lower down in the GG diagram, the more acceleration/braking is possible. The effect of aerodynamics on the heavy braking performance is clearly visible. At very large velocities, the aerodynamic downforce is the most significant contribution to the vertical force on the tires. Judging from the friction circle in (2.4), this allows for a very large longitudinal force in the absence of a lateral component in a straight line.

The lateral acceleration is defined as the product between the velocity squared and the path curvature as given in (2.18). This means that one tries to visualize a change in three parameters on a two dimensional grid, which is impossible. The problem is solved by evaluating the performance constraints with a constant velocity. In the conventional GG diagram, this can be interpreted as the further to the side one goes, the larger the curvature is of the current trajectory. As a result, the maximum achievable lateral acceleration indicates the tightest corner one can navigate at a fixed velocity. However, this method does make the constraints incompatible with an entire data set as seen in Figure 5.4, where the data points exceed the feasible domain of the performance constraints for the chosen velocity.

One can define the grip and power limits as given by the inequality constraints of the optimization problems. At any point of the track, the vehicle must satisfy all of these constraints. Therefore, the feasible domain will stretch from the origin to the closest constraint line in every direction.

5.3.2 Adapted GG diagram

Although the GG diagram is a very powerful tool, the optimization problems under consideration have lateral accelerations that are dependent on velocity, and at each data point the track has a constant curvature. Therefore, taking inspiration from the conventional GG diagram, an alteration is made, which is shown in Figure 5.5.

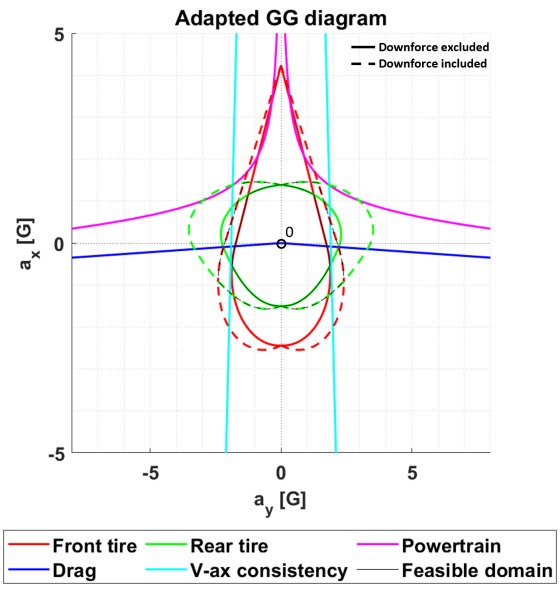


Figure 5.5: Visualization of the adapted GG diagram. The solid lines represent a vehicle without downforce, the dashed lines represent a vehicle with downforce.

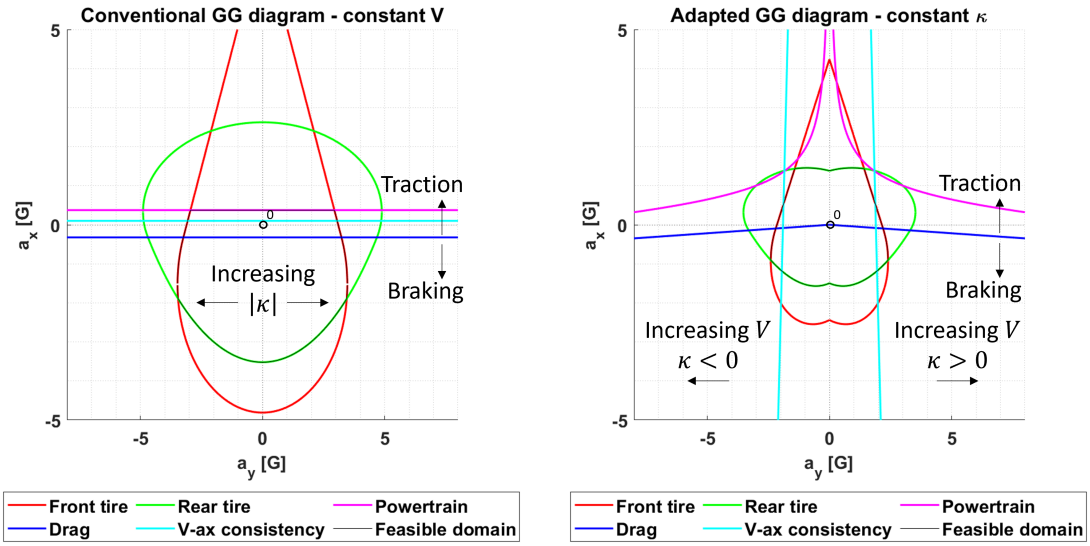
The adapted GG diagram is interpreted for corners with a constant curvature. As a result, the value on the horizontal axis corresponds to the velocity in a corner of that radius. However, due to the quadratic dependency between the lateral acceleration and the velocity in (2.18), the velocity increases quadratically when increasing the lateral acceleration.

In Figure 5.5, the theoretical vehicle performance limits are given for both a vehicle without and with aerodynamic downforce. Without downforce, the velocity only affects the amount of lateral force the tires can generate. As a result, the maximum longitudinal acceleration can be achieved when there is no lateral component, i.e. the vehicle is at a standstill. Increasing the lateral acceleration decreases the longitudinal acceleration until the maximum velocity in the corner is reached. However, for the individual tires this maximum is not reached when $a_x = 0$. This is due to longitudinal load transfer, where the front tire has a larger vertical force when the vehicle is braked, and thus friction force. This effect is limited as increasing the longitudinal acceleration also increases the force the tire has to generate in the longitudinal direction, which occupies the potential of the lateral force that increases the velocity in the corner. The same occurs for the rear tire, but this happens for $a_x > 0$.

Adding downforce to the vehicle model only increases the vertical force on the tires. Therefore, the friction force is increased with increasing velocity. This is observed by a temporary increase of both the longitudinal and lateral acceleration starting from standstill. As the velocity increases

quadratically, the lateral force will become increasingly more important, which closes the shape.

The adapted GG diagram is compared to the conventional GG diagram in Figure 5.6.



(a) Conventional GG diagram.

(b) Adapted GG diagram.

Figure 5.6: Comparison between the conventional GG diagram and the adapted GG diagram for the optimization problem.

Both GG diagrams are divided into an upper part where the vehicle is propelled, and a lower part where the vehicle is braking. The boundary between the two regions is defined by constraint g_4, which is the difference between the generated propulsion/braking force and the aerodynamic drag. In the conventional GG diagram, the powertrain (g_3), drag (g_4), and consistency (h_1) constraints appear as horizontal lines as they are curvature invariant. This means that irrespective of the curvature, the shapes of these constraints will not change in the adapted GG diagram. Finally, when moving away from the origin in the direction of the lateral acceleration, the curvature will increase in the conventional GG diagram, and the velocity will increase in the adapted GG diagram. In both GG diagrams, the curvature will be negative on the left side of the vertical axis, and positive on the right side. It is important to note that when $a_y = 0$ in the adapted GG diagram, the velocity is zero and not the curvature.

5.3.3 Shape of the constraints

The constraints have a certain shape, which can be modified by changing the vehicle parameters or the curvature. For the single-track model with an elevated CoG described in Chapter 2, the constraints will take the following form:

1. Front tire grip limit:

$$g_1 = F_{x_{1,i}}^2 + F_{y_{1,i}}^2 - \left(\mu_1 F_{z_{1,i}}\right)^2 \tag{5.10}$$

A derivation of the constraint is given in Appendix A.1. The constraint is a second-order polynomial in both $a_{x,i}$ and V_i^2. At the point $\left(V_i, -\frac{k_D}{m} V_i^2\right)$, the vehicle switches between

being actively braked and propelled by the powertrain. This means that for $a_{x,i} < -\frac{k_D}{m} V_i^2$, the brake distribution p_B is used in the longitudinal force and the torque distribution p_D for larger accelerations. Therefore, for both the traction and the braking region, the function describes the trajectory of an ellipse, but the derivative will be discontinuous at the intersection point. Due to longitudinal load transfer, the maximum achievable velocity, and therefore lateral acceleration, will always be located at a longitudinal acceleration lower than $a_{x,i} < -\frac{k_D}{m} V_i^2$ as this causes an increase of vertical force on the front tire. The width of the ellipse is mainly determined by the friction coefficient μ_1. However, by tuning the downforce distribution D_d more towards the front, one can effectively raise the friction coefficient of the tire increasing the maximum available grip [36].

As the curvature κ_i decreases, the maximum possible velocity increases. This is illustrated in Figure 5.7.

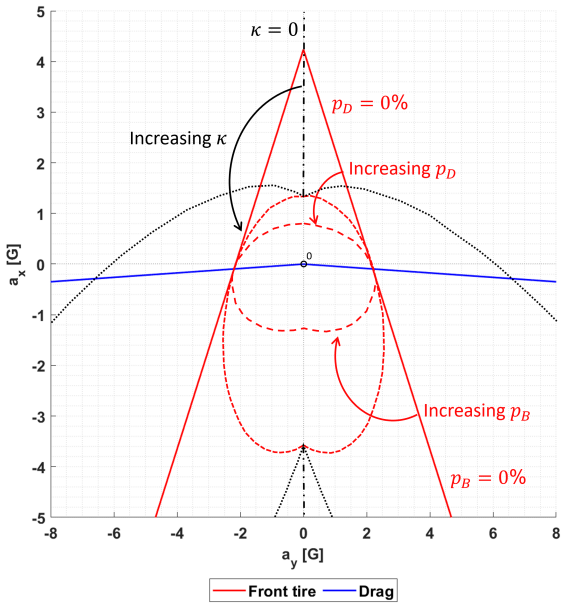


Figure 5.7: Relation between the shape of the front tire grip constraint g_1 and curvature κ (black) and torque p_D and brake p_B distributions (red).

At some point, the tire allows the vehicle to take the corner at infinite velocity, because the contribution of downforce in the vertical force is greater than the maximum possible lateral force on the tire. As a result, the curve of the traction and braking solution diverges. A similar phenomenon occurs when tuning the torque distribution p_D and brake distribution p_B, as can be seen in Figure 5.7. Due to longitudinal load transfer, the vertical force on the front tire will be directly proportional to the longitudinal acceleration with a negative slope. If the rear wheel is primarily driven or braked, the longitudinal force term in (5.10) becomes negligibly small, and the shape of the constraint is dominated by the linear dependency on the longitudinal acceleration. As a result, under traction the solution will converge to the a_x axis, and under braking the solution will diverge.

2. Rear tire grip limit:

$$g_2 = F_{x_{2,i}}^2 + F_{y_{2,i}}^2 - \left(\mu_2 F_{z_{2,i}}\right)^2 \tag{5.11}$$

A derivation of the constraint is given in Appendix A.2. The rear tire constraint follows exactly the same basic shape as the front tire constraint. However, longitudinal load transfer causes an increase of vertical force on the rear tire during traction. As a result, the maximum achievable lateral acceleration is always located at $a_{x,i} > -\frac{k_D}{m}V_i^2$. Depending on the magnitude of the aerodynamic drag force, this peak lateral performance can either be achieved with a positive or negative longitudinal acceleration. The difference in load transfer also causes the solution to converge to the a_x axis under braking, and to diverge under traction if only a small fraction of drive/brake torque is supplied to the rear wheel. This is illustrated in Figure 5.8.

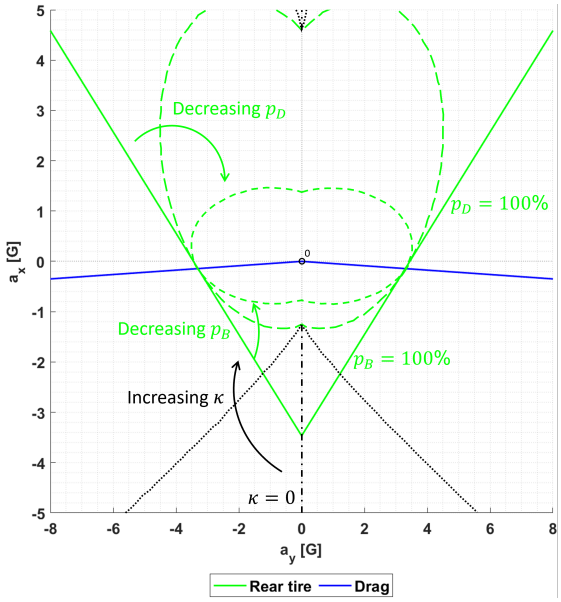


Figure 5.8: Relation between the shape of the rear tire grip constraint g_2 and curvature κ (black) and torque p_D and brake p_B distributions (green).

3. Power limit:

$$g_3 = ma_{x,i} - \frac{P}{V_i} + k_D V_i^2 + f_{rr}\left(mg + k_L V_i^2\right) \tag{5.12}$$

When active, the powertrain constraint will make the maximum longitudinal acceleration a function of the velocity. It consists of a positive term proportional to ${V_i}^{-1}$ and a negative term proportional to V_i^2. In the adapted GG diagram, this means that for low lateral accelerations, the maximum longitudinal acceleration tends to infinity. This then gradually decreases proportional to the inverse of the velocity and, because the lateral acceleration ($V_i^2 = a_{y,i}\kappa_i^{-1}$) is displayed on the horizontal axis, the constraint decreases with a linear decay due to drag and rolling resistance. The shape of the power limit is fairly consistent, and is independent of the curvature. The curve is scaled upwards or downwards by respectively increasing or decreasing the motor power P.

4. Drag limit:

$$g_4 = -ma_{x,i} - k_D V_i^2 \tag{5.13}$$

The drag limit constraint has a quadratic relation between the longitudinal acceleration and velocity. In the adapted GG diagram, it will always be a linearly decreasing line that

starts in the origin. Above this line, the vehicle is in traction, and below this line, the vehicle is braking.

5. Consistency between velocity and acceleration:

$$h_1 = V_{i-1}^2 - V_i^2 + 2\Delta s a_{x,i-1} \tag{5.14}$$

As the velocity at the next iteration V_i is taken to be constant, the longitudinal acceleration is proportional to the square of the velocity with a negative slope. In the adapted GG diagram, this means that the constraint (5.14) will be represented by a linearly decreasing line. When the longitudinal acceleration is zero, the velocity at the subsequent iterations is the same. Likewise, when the acceleration is positive, the velocity at iteration $i-1$ will be lower than the velocity at iteration i, and when the acceleration is negative, the velocity at iteration $i-1$ will be higher than the velocity at iteration i. This constraint is also independent of the track curvature. The track discretization Δs is very important for this constraint. For larger Δs, the discrepancy in velocity between subsequent iterations will be larger for the same longitudinal acceleration, meaning that the angle with the horizontal axis will become smaller.

5.3.4 Finding potential minimizers

All optimization problems have an objective function that is defined as the square of one of the design variables times the constant -1. This objective function is not well-bounded, which means that if it were to be optimized unconstrained, the solution to the optimization problem will tend to infinity. However, due to the shape of the constraints and the limit to the accepted longitudinal acceleration, the optimization problem becomes convex, which means that there is one unique solution to the optimization problem.

However, throughout the LTS process different constraints might become active. For this reason, all possible minimizers are analyzed by investigating the shape of the constraints, and finding design variable combinations where the KKT conditions (3.2)-(3.5) and sufficient condition (3.6) are satisfied.

The following sections cover how a (semi-)analytical solution of the minimizers to the optimization problems can be found. For this, the MATLAB Symbolic Math Toolbox [37] was used. A MATLAB script that calculates the minimizers to each optimization problem is included in Appendix B. The solutions are not given in this work as they provide little extra understanding of the problem.

It is important to factor out as many common terms in the solution as possible. This process was overlooked in this work. Utilizing this will notably boost solution readability while substantially reducing the source file size the compiler needs to process, ultimately leading to quicker computational processes.

Forward loop - optimization problem (5.6)

Optimization problem (5.6) is different from the other three optimization problems as only the longitudinal acceleration is taken as a design variable, and the longitudinal acceleration is maximized. The constraints can be rewritten as a solution to $a_{x,i}$. When equaling all constraints to zero, the optimization problem can be visualized as in Figure 5.9.

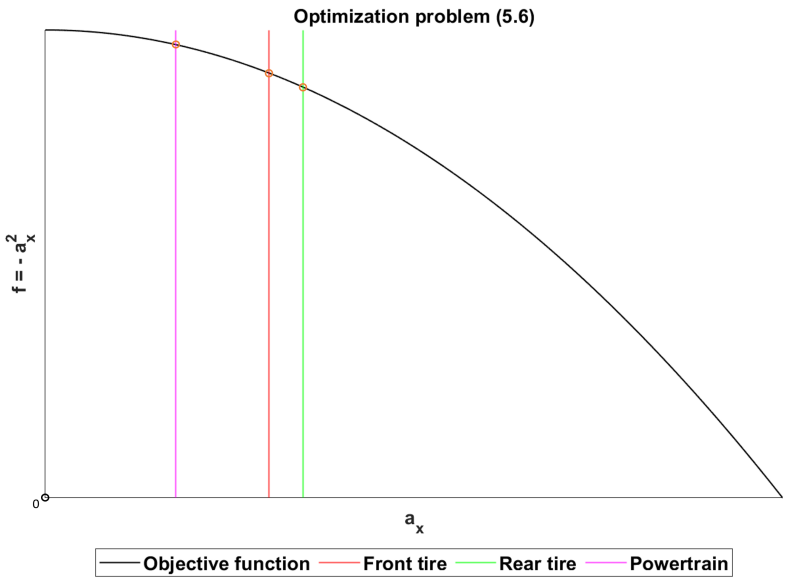


Figure 5.9: Visualization of optimization problem (5.6). The potential minimizers are indicated with an orange circle.

As the curvature κ_i and velocity V_i are constant, the maximum achievable longitudinal acceleration $a_{x,i}$ for each constraint is a constant value. Since the objective function is not well-bounded, the minimizer to this optimization problem can be found when one of the constraints becomes active. Depending on the vehicle parameters, the velocity, and the curvature of the track, this means that there are three possible minimizers to this optimization problem: either constraint g_1, g_2, or g_3 is active as indicated by the orange circles in Figure 5.9. The minimizer can then be found by setting the active constraint to zero, and solving the equation for $a_{x,i}$.

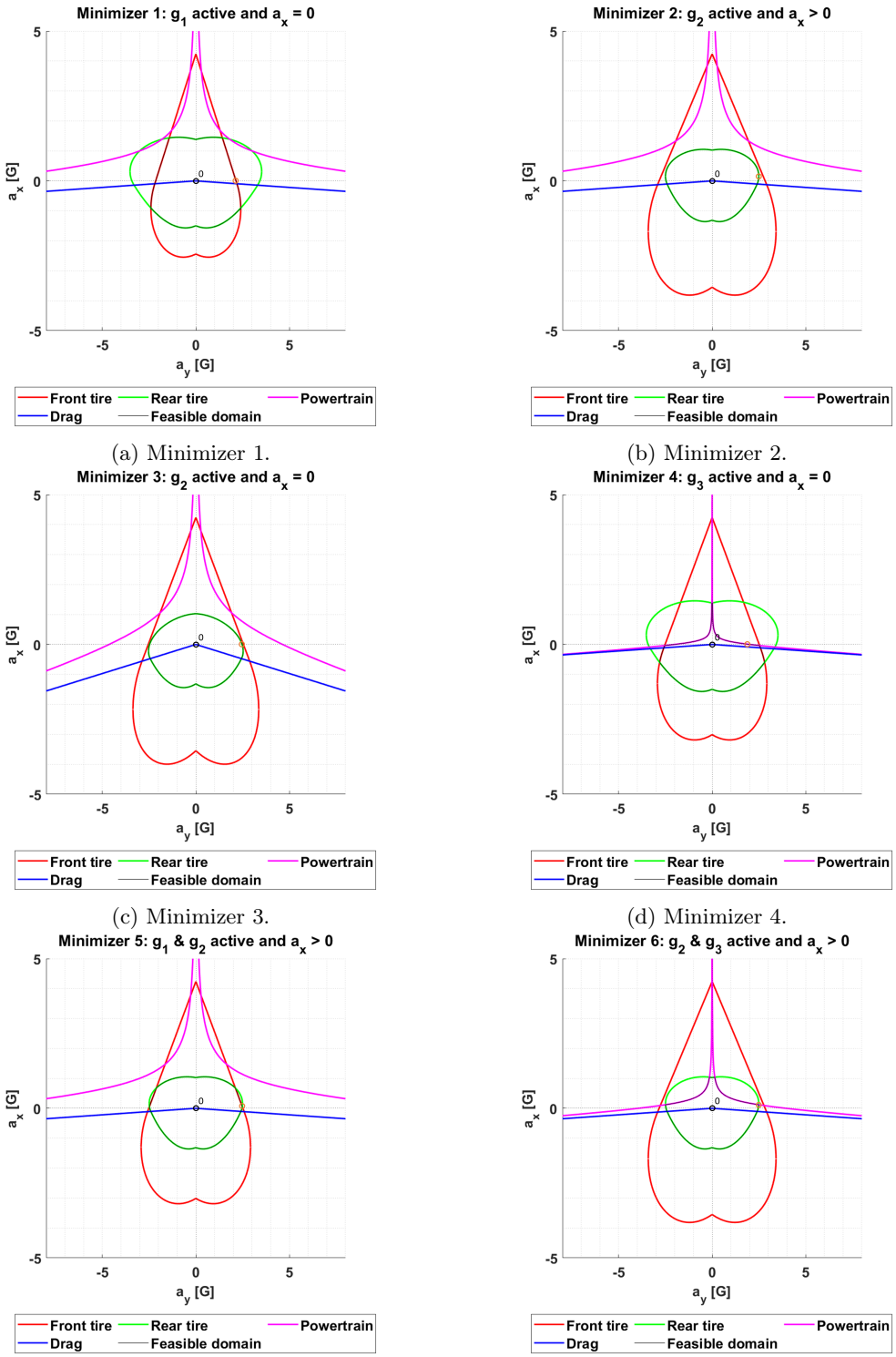
Forward loop - optimization problem (5.7)

In this optimization problem, the longitudinal acceleration must be non-negative as one only considers an increase of the forward velocity. From the shape of the constraints, it can be concluded that there are six possible minimizers. Example cases of the minimizers are shown in Figure 5.10.

The minimizers can be described as follows:

1. Only g_1 is active and $a_{x,i} = 0$. Due to load transfer, the front tire will lose grip under acceleration. Therefore, the front tire will be utilized as much as possible if the load transfer is as small as possible, i.e. $a_{x,i} = 0$. This minimizer is the most prevalent if the friction coefficient of the rear tire is larger than the friction coefficient of the front tire. An analytical solution to the problem can be found by solving $g_1 = 0$ for V_i, and then substituting $a_{x,i} = 0$.

2. Only g_2 is active and $a_{x,i} > 0$. In contrary with the front tire, the rear tire will gain grip under acceleration. If the rear wheel handles a significant amount of motor torque, i.e. p_D is not tuned towards the front wheel, the curve of the active constraint will have a maximum velocity at a positive longitudinal acceleration. In the case that p_D is tuned towards the front wheel, the curve will diverge, as seen in Figure 5.8, and the constraint will become



(a) Minimizer 1.

(b) Minimizer 2.

(c) Minimizer 3.

(d) Minimizer 4.

(e) Minimizer 5.

(f) Minimizer 6.

Figure 5.10: Example solutions for each minimizer of optimization problem 5.7. The minimizers are indicated with an orange circle.

not well-bounded. However, this proves no practical problem as the other constraints will be violated for insanely high velocities. This occurs when the friction coefficient of the front tire is larger than the friction coefficient of the rear tire. Constraints g_1 and g_3 will, in this case, not intersect with g_2 for an acceleration that is lower than the acceleration that maximizes the velocity. The minimizer can be found analytically by solving $g_2 = 0$ for V_i. One takes the partial derivative to the acceleration $\frac{\partial V_i}{\partial a_{x,i}}$, and sets it equal to zero to find the maximum of the curve. Then the equation can be solved for $a_{x,i}$. By substituting the minimizer value of the acceleration $a^*_{x,i}$ into the expression for the velocity, also the minimizing velocity can be found.

3. Only g_2 is active and $a_{x,i} = 0$. When the drag coefficient of the vehicle is very high, the acceleration for which the velocity is maximized may become negative. If the rear wheel is sufficiently driven, the shape converges and the maximum velocity will be found when $a_{x,i} < 0$, as seen in Figure 5.10c. Since this maximum occurs when the vehicle is decelerating, $a_{x,i} = 0$ will become the maximum possible state under traction. The solution to the minimizer can be found in a similar way with g_2 as minimizer 1.

4. Only g_3 is active and $a_{x,i} = 0$. The power limit monotonically decreases with velocity. At $a_{x,i} = 0$, the motor power is equal to the aerodynamic drag and rolling resistance, which is the velocity limit of the powertrain. Since the power limit is independent of the curvature, this velocity limit will be a constant. The solution to the minimizer can be found in a similar way with g_3 as minimizer 1.

5. g_1 and g_2 are active and $a_{x,i} > 0$. In this case, the same conditions hold as for minimizer 2. However, g_1 will now intersect with g_2 at a longitudinal acceleration that is below the maximum achievable lateral grip of the rear tire and above $a_{x,i} = 0$. As g_1 decreases due to load transfer, the two constraints will both become active. The minimizer can be found by setting both constraints to active and solving them for $a_{x,i}$. Since the values of V_i and $a_{x,i}$ must be the same for both constraints, one can equate the derived accelerations and find a solution for V_i. This results in a fourth-order polynomial in V_i^2, which cannot be solved analytically. However, a numerical approximation of the velocity can be found through, for example, polynomial root finding. This approximation can then be substituted back into one of the derived accelerations to find the associated minimizing velocity.

6. g_2 and g_3 are active and $a_{x,i} > 0$. This case is similar to the previous case, except that the intersection is between g_2 and g_3. The value of the minimizer can be found in a similar way as minimizer 5. This results in a sixth-order polynomial in V_i, and the value of the minimizer will always be the largest real solution to the polynomial.

Reverse loop - optimization problem (5.8)

In the reverse loop, equality constraint h_1 is introduced to ensure consistency between the velocity and longitudinal acceleration. Equality constraints are always active. However, h_1 is, along with the objective function, not well-bounded as decreasing the longitudinal acceleration increases the velocity at the previous iteration $i - 1$. As a result, the minimizer of the optimization problem will always reside where both the equality constraint and one of the inequality constraints are active. This leads to two potential minimizers under braking as can be seen in Figure 5.11.

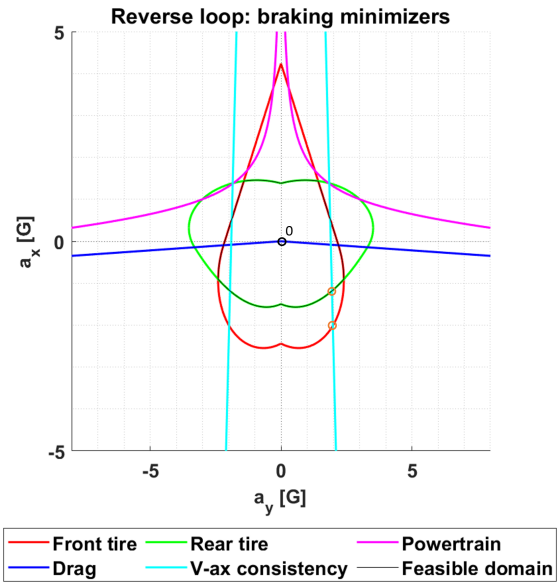


Figure 5.11: Visualization of optimization problem (5.8). The potential minimizers are indicated with an orange circle.

The minimizers can be described as follows:

1. h_1 and g_1 are active and $a_{x,i-1} < -\frac{k_D}{m} V_{i-1}^2$.
2. h_1 and g_2 are active and $a_{x,i-1} < -\frac{k_D}{m} V_{i-1}^2$.

The potential minimizers can be found by starting on the equality constraint at $a_{x,i-1} = -\frac{k_D}{m} V_{i-1}^2$, and decreasing the acceleration while staying on the constraint until it intersects the inequality constraints. The value of the minimizer can be found by solving both active constraints for $a_{x,i-1}$, equating the solutions, and calculating the solution for V_{i-1}. In all cases this is a second-order polynomial in V_{i-1}^2, which can be solved analytically. The corresponding acceleration is then found by substituting the minimizing velocity into the derived expression for the acceleration.

Reverse loop - optimization problem (5.9)

Similar to optimization problem (5.8), the traction variant optimization problem in the reverse loop also depends on constraint h_1. However, since h_1 monotonically decreases with V_{i-1}, the most preferable outcome would be when g_4 is also active, or $a_{x,i-1} = -\frac{k_D}{m} V_{i-1}^2$, as this always defines the lower bound of the longitudinal acceleration. This means that the vehicle is constantly braking across the lap, and the velocity in previous iteration $i-1$ quadratically increases. This is not a realistic result, as after braking for a corner, the vehicle accelerates until a performance limit is reached or the vehicle has to brake for the next corner.

This problem is solved with the addition of equality constraint h_2. This constraint ensures that the performance constraint that results in a maximum velocity is active. With this change, it is ensured that the vehicle will always remain on the boundary of the feasible domain. As a result, there are four potential minimizers under traction as can be seen in Figure 5.12.

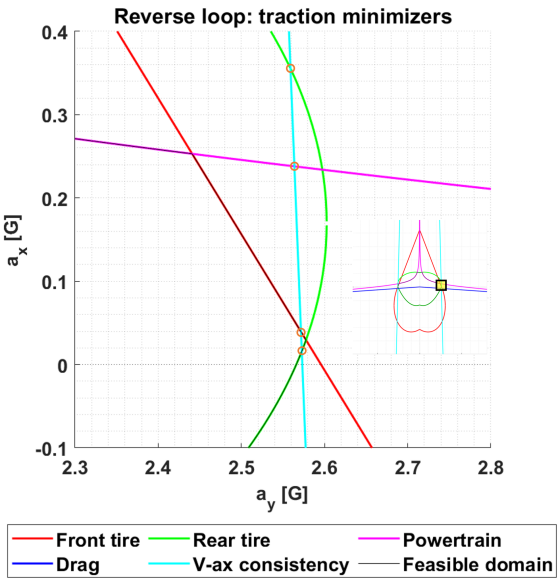


Figure 5.12: Visualization of optimization problem (5.9). The region around the minimizers of the adapted GG diagram is enhanced. The potential minimizers are indicated with an orange circle.

The minimizers can be described as follows:

1. h_1 and g_1 are active and $a_{x,i-1} \geq -\frac{k_D}{m} V_{i-1}^2$.
2. h_1 and g_2 are active and $a_{x,i-1} \geq a_{x,i-1}|_{\max(g_2)}$.
3. h_1 and g_2 are active and $-\frac{k_D}{m} V_{i-1}^2 \leq a_{x,i-1} < a_{x,i-1}|_{\max(g_2)}$.
4. h_1 and g_3 are active and $a_{x,i-1} \geq -\frac{k_D}{m} V_{i-1}^2$.

As the shape of the rear tire grip constraint has its maximum when $a_{x,i-1} \geq -\frac{k_D}{m} V_{i-1}^2$, there are two intersections between g_2 and h_1 when the vehicle is close to the lateral grip limit of the rear tire, as can be seen in Figure 5.12. This means that both minimizer solutions 2 and 3 are valid. However, due to the linear decay of h_1 with increasing lateral acceleration, minimizer 3 will result in a slightly higher velocity, and thus will be chosen as the correct minimizer.

Finding the solutions of potential minimizers is done in the same manner as for optimization problem (5.8). For additional minimizer 4, equating the accelerations from g_3 and h_1 results in a third-order polynomial in V_{i-1}, which can be solved analytically and is independent of the curvature.

5.4 Increasing vehicle model complexity

The algorithm is set up in such a way that vehicle models are easily interchangeable. The only change that needs to be made is the formulation of the optimization problems, where the inequality constraints are altered depending on the vehicle model. As long as the constraints remain continuous and their shape changes in a predictable way, the minimizer determination can be

done through visual proof as done in this work. The practical drawback of increasing the vehicle model complexity is an increase in analytical solution size. The coding files grow significantly in size, resulting in a substantial decline in computation speed.

When including Magic Formula parameters into the algorithm, for example creating a load-dependent friction coefficient, as described in Figure 2.4, the vertical force limits of the tire test data have to be included as inequality constraints. In a model with downforce, the analytical expression will violate the maximum vertical force on a tire, and extrapolate the friction coefficient as a result.

Increasing the amount of wheels in the vehicle model can be done by adding new inequality constraints for the friction ellipse of each wheel that is added. For a four-wheel model, this means that the optimization problems will have four inequality constraints based on the friction ellipse of each wheel. The number of potential minimizers will change because of this, depending on the inclusion of certain physical phenomena such as load transfer. When including load transfer in the lateral direction, one needs to include the direction of curvature in the optimization problem. This is because the direction of the turn influences the friction force the tires can develop on each side of the track. In all optimization problems this can be done by making the objective function linear and multiplying it with $\text{sgn}(\kappa)$, the sign function of the curvature. Since the curvature is considered to be a constant in the optimization problem, this multiplication is just a scaling of the objective and the properties of the function are retained.

An important change is to upgrade the powertrain model. For example, Hendrikx, et al. [12] reformulate the angular velocity to engine torque curve of a combustion engine to a relation between forward velocity and longitudinal force using the gear ratios of the drivetrain and the wheel radius. The same method can be used for a brushed DC electric motor that has a linearly declining relation between its rotational speed and provided motor torque [38], as seen in Figure 5.13.

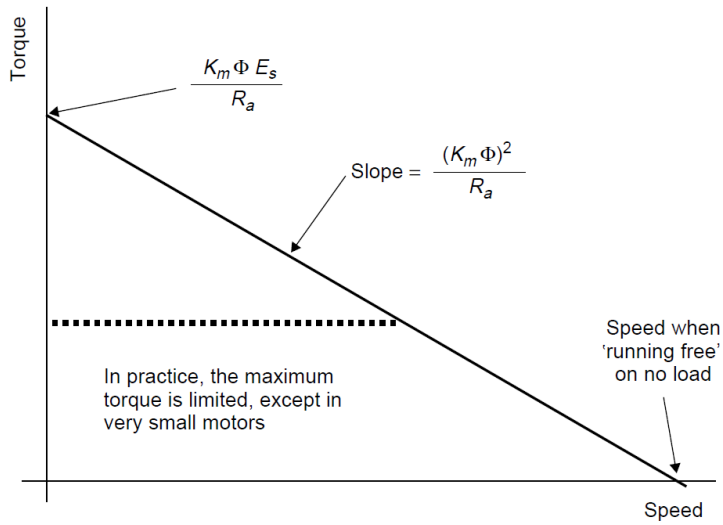


Figure 5.13: Relation between motor speed and motor torque for a brushed DC electric motor. Taken from [38].

The relation between motor torque T_m and rotational velocity ω_m is

$$T_m = \frac{K_m \Phi E_s}{R_a} - \frac{(K_m \Phi)^2}{R_a} \omega_m \tag{5.15}$$

where K_m is the motor constant, Φ is the total flux passing through the coil of the motor, E_s is the supply voltage, and R_a is the armature resistance. The torque and rotational velocity of the electric motor can be converted to the torque and rotational velocity of the wheel, which in return can be turned into a tractive force and translational velocity using the gear ratio G, drivetrain efficiency η_m, and wheel radius R_t.

$$T_w = T_m G \eta_m = F_M R_t \tag{5.16}$$

$$\omega_w = \frac{\omega_m}{G} \eta_m = \frac{V}{R_t} \tag{5.17}$$

The tractive force and velocity can then be substituted into (5.15) to find the relation between force and velocity.

$$F_M = \frac{G \eta_m}{R_t} \frac{K_m \Phi E_s}{R_a} - \frac{G^2}{R_t^2} \frac{(K_m \Phi)^2}{R_a} V \tag{5.18}$$

This expression still exhibits the same linearly declining behavior, which may then be used as F_M in (2.11).

5.5 Summary

In this chapter, the lap time simulation (LTS) algorithm is explained. The conventional method for a point mass vehicle model using the forward and reverse loop is illustrated, and it is deduced that more realistic vehicle models cannot be implemented in this framework. With the use of optimization theory, the forward and reverse loop are revamped and a single-track vehicle model with an elevated center of gravity is implemented. Both loops include two optimization problems that maximize the vehicle behavior at each point of the track based on the tires and powertrain. With an alteration to the GG diagram, the shapes of the constraints are explored, and are used to find (semi-)analytical solutions to the potential minimizers of the optimization problems. Finally, recommendations are given to upgrade the vehicle to a four-wheel model, include load-dependent friction coefficients, and improve the powertrain model.

6 Applications of lap time simulation

Before implementing the algorithm in an optimized code environment for commercial use, it is important that the workings of the model are well understood. In this chapter, the model is compared to the simplified version of the lap time simulation model to assess the benefits of the more complex vehicle model and computational effort. Moreover, it is tested against telemetry taken directly from race cars to identify areas of improvement. However, first some limitations that the method faces are described such that they can be accounted for in the later discussion.

6.1 Model deficiencies

Before discussing applications of the proposed methodology, some of the expected shortcomings are noted. These shortcomings are the solution falling outside the feasible domain and a rapid loss of velocity in a straight line. Both cases are discussed in the following sections.

6.1.1 Solution outside of the feasible domain

When a sudden increase in curvature occurs, the calculated velocity V_i may result in constraint h_1 being located outside of the feasible domain defined by the inequality constraints in the subsequent iteration $i-1$. As a result, no solution can be found in the reverse loop optimization problem. The intersection between h_1 and the inequality constraints will always at least violate one other inequality constraint. This is a fundamental problem for algorithms with a single-step lookahead distance. The future states of the track are not considered, and can thus not be anticipated upon.

There are two situations where this can occur:

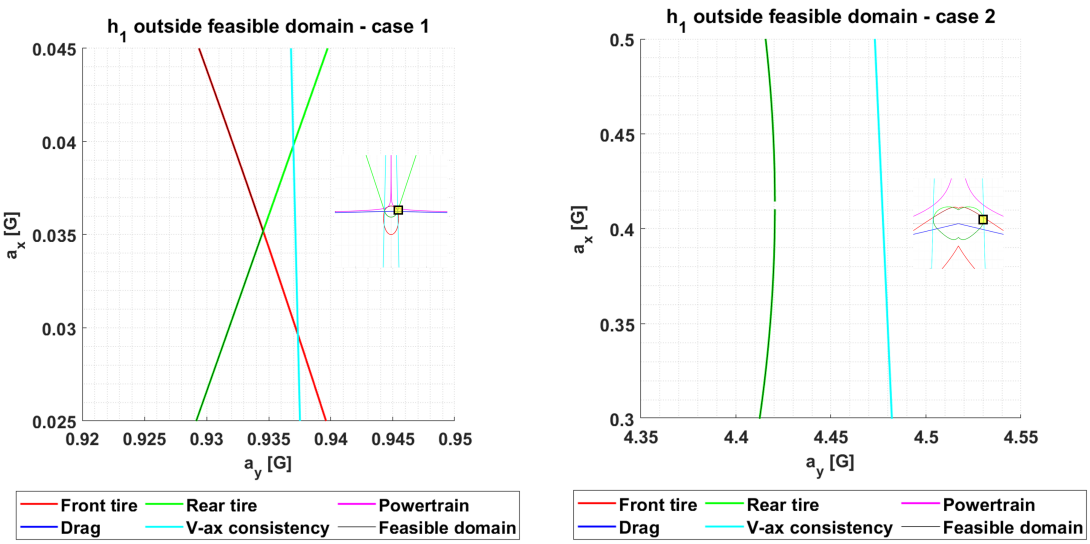
1. h_1 has their intersections with g_1 and g_2 very close to each other, but either intersection violates the other constraint. This is depicted in Figure 6.1a.
2. h_1 does not intersect with g_2 as it requires a higher velocity than the maximum possible velocity allowed by the rear tire. This is depicted in Figure 6.1b.

Either situation can occur due to a sudden increase in curvature, which pushes h_1 outside the feasible domain. In both cases, the violation is initially only marginal, which, from a practical point of view, still approximates the optimal driving behavior. However, both situations require a different solution to their problem.

In the first situation, it is chosen to use minimizer 3 since the constraint violations of this and minimizer 1 are about the same size. However, this strategy will not work in the second situation as g_2 and h_1 do not intersect. In this case, it is chosen to take the acceleration for which g_2 is maximum and calculate V_{i-1} by substituting this acceleration into (5.2). Both solutions will overestimate the velocity with, at maximum, a few extra km/h.

Usually, the curvature does not significantly increase for extended duration as this would mean that the vehicle is spinning out. For that reason, the violation remains relatively small, and is solved at the apex of the turn, after which point the curvature decreases.

A more robust approach to this problem would be to find the velocity and acceleration that minimize the violation of the performance constraints subject to the consistency and drag constraints.



(a) Intersection g_1 & g_2.

(b) Maximum g_2.

Figure 6.1: Examples of solutions outside the feasible domain.

This optimization problem can be formulated as follows:

$$\begin{aligned} & \underline{x} = \begin{bmatrix} V_{i-1} & a_{x,i-1} \end{bmatrix}^T \\ \min_{\underline{x}} \quad & f(\underline{x}) = g_1^2 + g_2^2 + g_3^2 \\ \text{s.t.} \quad & g(\underline{x}) = -ma_{x,i-1} - k_D V_{i-1}^2 \leq 0 \\ & h(\underline{x}) = V_{i-1}^2 - V_i^2 + 2\Delta s a_{x,i-1} = 0 \end{aligned} \tag{6.1}$$

The performance constraints are evaluated in the non-negative domain, meaning that negative values of g_1, g_2 and g_3 are set to zero. This method ensures that the consistency constraint is satisfied and the vehicle is evaluated under traction conditions. However, generally it is not possible to find an analytical solution for the minimizer through visual inspection. This means that the optimization problem has to be solved through numerical optimization. It was chosen not to use this method as the numerical optimization is significantly slower than analytical solutions, and the discontinuous behavior of the objective function is problematic for gradient-based optimization algorithms.

6.1.2 Rapid loss of velocity

This problem occurs in a straight line or small curvature when the velocity is constrained by the power limit. The grip constraints diverge for small curvatures as a result of aerodynamic downforce, as can be seen in Figure 5.7 and Figure 5.8. Moreover, the consistency constraint h_1 and powertrain constraint g_3 are invariant with curvature, meaning that these two constraints will define the minimizer at the end of a straight for the traction optimization problem in the reverse loop as can be seen in Figure 6.2.

When in traction, velocity V_{i-1} decreases due to a positive longitudinal acceleration $a_{x,i-1}$ as required by constraint h_1. This means that in the subsequent iteration V_i has decreased. Since

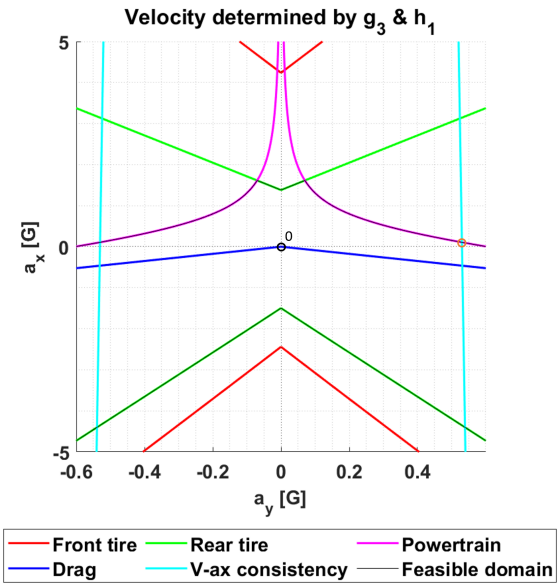


Figure 6.2: Example of velocity minimizer in small curvature. The minimizer is shown as an orange circle.

the curvature barely changes in a straight line, this iteration will also be subject to a decrease in V_{i-1}. For consecutive iterations in a straight line this then leads to a rapid decrease in velocity, as constraint g_3 is inversely proportional to V_{i-1}.

This behavior is a chain reaction and continues until the velocity of one of the minimizers of the braking optimization problem in the reverse loop is lower than the velocity at that point in the forward loop. As a result, the vehicle brakes for a brief moment on the straight, which increases the velocity again.

Obviously, this is not the fastest way to travel in a straight line. This momentary instance of braking can also be used to accelerate the vehicle up to its top speed. If the condition for braking is not triggered, the velocity would exponentially decrease due to the inverse dependency between the velocity and the longitudinal acceleration. This would be an even less desired outcome. Luckily, the time loss is only momentary as the correction resets the offset between the forward and reverse loop in a single iteration.

This implementation error is likely due to the formulation of the traction optimization problem in the reverse loop. In conventional LTS algorithms, the behavior under traction is only optimized in the forward loop. The maximum possible velocity at each point of the track is not achieved in the forward loop as the (estimated) velocity is calculated based on the consistency with the velocity and acceleration in the previous iteration. In the reverse loop, only the longitudinal acceleration is corrected in case of traction.

However, in (5.9) of the revised algorithm, the vehicle is again optimized to be on its performance limit due to constraint h_2. This extra optimization overestimates the longitudinal acceleration, leading to a more rapid loss of velocity due to its inverse relation with the longitudinal acceler-

ation.

A potential solution is to revert to the same method of calculating the longitudinal acceleration from the velocity at subsequent iterations like in the conventional algorithm. In a straight line this will not result in any problems. However, in a corner when the vehicle is grip limited, a small change in longitudinal acceleration will sometimes violate the grip limit due to a change in load transfer. Therefore, either a new solution should be found or a combination of the solutions should be implemented.

6.2 Comparison with conventional LTS

In order to highlight the extended vehicle model capabilities, the revised algorithm is compared to a conventional LTS algorithm that includes aerodynamics. The point mass model includes downforce, drag, rolling resistance, is driven by two wheels and braked by all four wheels. The three equations governing the motion are:

$$F_x = ma_x + F_D \tag{6.2}$$

$$F_y = mV^2\kappa \tag{6.3}$$

$$F_z = mg + F_L \tag{6.4}$$

The tire grip is described by

$$\left(\frac{F_x}{n \cdot \mu \frac{F_z}{4}}\right)^2 + \left(\frac{F_y}{4 \cdot \mu \frac{F_z}{4}}\right)^2 = 1 \tag{6.5}$$

where n is the number of driven/braked wheels. The models are evaluated for an LMP2 car with the vehicle parameters listed in Table 6.1.

Table 6.1: List of vehicle parameters. PM stands for point mass model, ST stands for single-track model.

Parameter	Description	Value	Model
Mass			
m	Vehicle mass	1030 [kg]	PM & ST
g	Gravitational acceleration	9.81 [m/s^2]	PM & ST
W_d	Weight distribution	55.0 [%]	ST
L	Wheelbase	3.005 [m]	ST
h	CoG height	0.390 [m]	ST
Aerodynamics			
k_D	Drag coefficient	0.8983 [N/(m^2/s^2)]	PM & ST
k_L	Lift coefficient	2.695 [N/(m^2/s^2)]	PM & ST
D_d	Downforce distribution	48.0 [%]	ST
d	Vertical distance CoG-CoP	0.05 [m]	ST
Drivetrain			
P	Engine power	450 [kW]	PM & ST
f_{rr}	Rolling resistance coefficient	0.02 [-]	PM & ST
n	Number of driven wheels	2 (traction), 4 (braking) [-]	PM
p_D	Torque distribution	0 [%]	ST
p_B	Brake distribution	62.5 [%]	ST

Tire data is not given due to confidential reasons. The single-track model is equipped with a rear tire that has a higher friction coefficient than the front tire. The point mass model has a single friction coefficient with the same value as the friction coefficient of the front tire. The validity of the determined minimizers is checked with the constraints. The maximum constraint value is set to be 10^{-1}. This ensures that the error of the performance constraints is a maximum of 0.1 N, which is unnoticeable, and allows for some computational inaccuracies of the analytical solutions. If any constraint exceeds this threshold, the potential minimizer is rejected.

6.2.1 Constant radius turn

At first, both models are evaluated on the theoretical track that is shown in Figure 6.3.

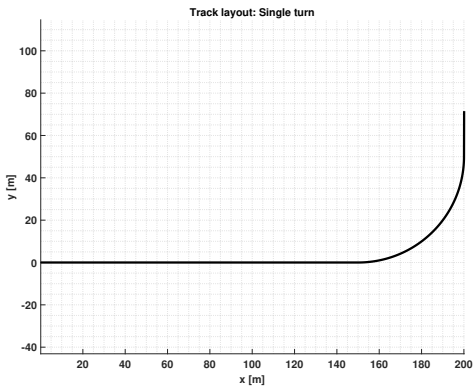


Figure 6.3: Track layout of the constant radius turn.

This track is discretized with a step size of $\Delta s = 0.1$ m and consists of three parts: a 150 meter long straight, a 90 degree curve with a constant curvature of $\kappa = 0.02$ 1/m, a 20 m straight at the corner exit. On this track, one can, for example, evaluate the following vehicle behavior:

- Acceleration out of a turn
- Braking before a turn
- Steady-state cornering
- Transition between the vehicle being grip limited and power limited

The curvature of the track layout given in Figure 6.3 is used as an input to the point mass and single-track LTS algorithms. Both simulations are started with an initial velocity of $V_1 = 150$ km/h. The resulting time gain $\Delta^2 t$, velocity, and acceleration profiles are shown in Figure 6.4.

The time gain is calculated as follows:

$$\Delta^2 t = \frac{\partial t_{\mathrm{PM}}}{\partial s} - \frac{\partial t_{\mathrm{ST}}}{\partial s} \tag{6.6}$$

where t is the cumulative time along the track. The units are displayed in milliseconds per meter. From this metric one can judge which model performs faster at each point of the track.

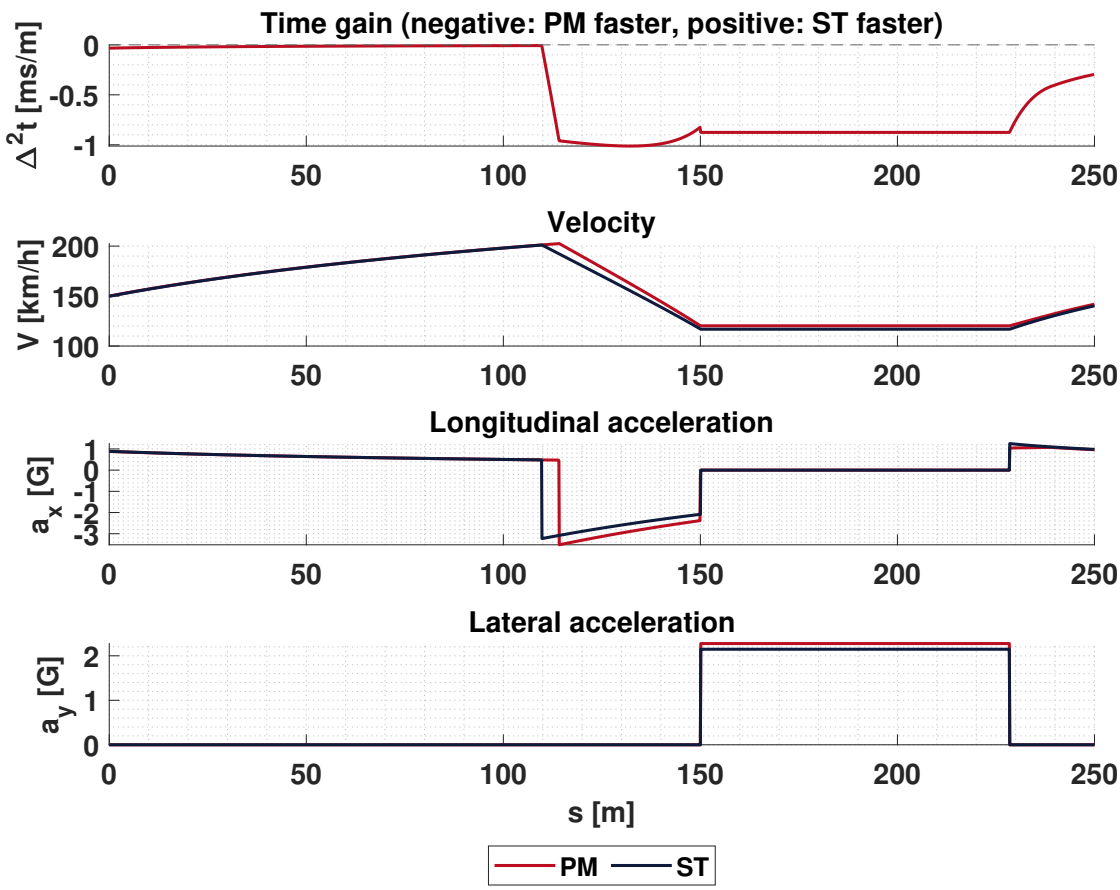


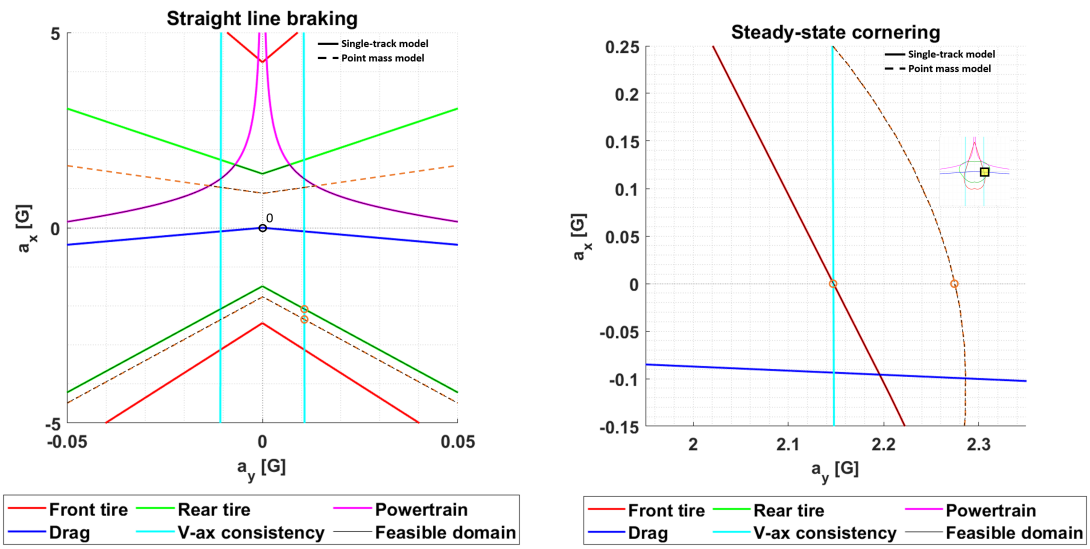
Figure 6.4: Comparison between the point mass and the single-track model LTS on the theoretical track. PM stands for point mass model, ST stands for single-track model.

A negative time gain indicates that the point mass model is faster, whereas a positive time gain illustrates an improvement in time by the single-track model.

For this circuit, it can be seen that the single-track model loses time compared to the point mass model at any given moment. Most notably is the performance loss under braking. The single-track model brakes earlier and achieves a lower peak deceleration than the point mass model. This is due to longitudinal load transfer and is shown in Figure 6.5a, which corresponds to the last data point before the corner. The figure is slightly altered by taking a curvature of $\kappa = 10^{-4}$ 1/m, as no curvature corresponds to the tire constraints being indistinguishable vertical lines. In both algorithms, one tries to maximize the braking performance at this point. As the algorithm with the point mass model calculates the velocity based on the maximum possible deceleration, its solution in the adapted GG diagram is around the intersection between the consistency constraint h_1 and the grip constraint of the point mass model given by the dashed orange line. The feasible domains of both models are different due to the performance of the tires. Under braking, the performance constraint of the point mass model is the equivalent tire, and for the single-track model this is the rear tire, as indicated by the black lines in Figure 6.5a.

The single-track model has a few elements in its favor for better performance. The brake distribution is shifted slightly to the front wheel, which means that the rear wheel has to generate less force than in the point mass model that has an equal brake distribution. Moreover, the load transfer due to the drag force increases the vertical force on the rear tire, the downforce distribution is tuned more towards the rear tire, and the friction coefficient of the rear tire is larger.

However, the weight distribution is tuned more towards the front tire and, more significantly, the load transfer due to inertia takes away a significant amount of vertical force under braking. Overall, this leads to a loss in total friction force, which means that in the single-track model, the rear tire is found to be a more constraining factor than the equivalent tire in the point mass model. As a result, the vehicle has less grip in the braking zone and can achieve a lower deceleration compared to the point mass model. This means that the vehicle has to brake earlier in the revised LTS algorithm.



(a) Straight line braking. (b) Steady-state cornering.

Figure 6.5: Explanation of phenomena through adapted GG diagrams. The minimizers to both models are shown as orange circles.

Another big difference between the two models is the speed carried through the corner. The velocity of the point mass model is larger than that of the single-track model. This can be explained with the adapted GG diagram in Figure 6.5b. Both models take this turn in steady-state as their longitudinal acceleration is zero and velocity constant. As can be seen in the adapted GG diagram, for the single-track model, the front tire constrains the motion for $a_x = 0$, whereas for the point mass model this is the equivalent tire. In both cases, the friction coefficient is the same, but the vertical force distribution of the single-track model is different. In comparison to the equivalent tire of the point mass model, the front tire gains load due to the forward weight distribution, but loses load due to the rearward downforce distribution and the load transfer due to drag. Overall, the decrease in vertical load is greater, meaning that the total achievable friction force is smaller. As a result, the single-track model achieves a lower steady-state velocity

through the turn than the point mass model.

The final noticeable difference is the higher acceleration out of the turn for the single-track model. This is mainly due to the problem described in Section 6.1.2. This is not due to longitudinal load transfer, as since one is driving in a straight line, the only potentially active performance constraint is the powertrain. As the velocity decreases, the possible longitudinal acceleration rapidly increases, which continues until, in the reverse loop, the vehicle is grip limited due to the increased track curvature. This problem is evident over both traction zones as can be seen in the marginally bigger time loss at iteration $i = 1$ than the time loss in the iteration before the braking zone. If the simulation was started at a lower initial velocity, the problem would be much more prevalent.

These results illustrate that the desired increase of vehicle model complexity has been implemented correctly. Vehicle setups with unequal force distributions between the front and rear axle can be investigated with the new algorithm and longitudinal load transfer is incorporated in the vehicle behavior.

Another important aspect of the new algorithm is the computational time compared to the conventional point mass model. The computational time is a device specific metric, but if the simulations are run under similar conditions on the same device, they may be compared. Both algorithms were programmed and run in MATLAB. The point mass model was evaluated in a time of 2.16 ms, whereas the single-track model had a computational time of 4436.8 ms, so the computational time has increased by over a factor of 2000. However, the simulation takes only a handful of seconds, which still makes it a practically feasible method. Furthermore, by using a programming language that allows for better optimization of code performance like C++, the computational time can be improved.

6.2.2 Circuit de Barcelona-Catalunya

To evaluate a more realistic use case, the LTS algorithms are evaluated for the permanent race track Circuit de Barcelona-Catalunya, which is located in Montmeló, Spain. The layout of this race track is given in Figure 6.6.

The curvature data was calculated from time-filtered telemetry and is discretized with a step size of $\Delta s = 1$ m. In many forms of motorsports, the Circuit de Barcelona-Catalunya is used for vehicle testing as it provides a good combination of high-speed, medium-speed and slow-speed corners [39]. Furthermore, the circuit was part of the European Le Mans Series calendar in 2023 [40], where LMP2 cars are used. A comparison of the models, starting with an initial velocity of $V_1 = 150$ km/h, can be found in Figure 6.7.

At a first glance, one sees a few single-point negative spikes in the longitudinal acceleration trace. This is due to the problem described in Section 6.1.2. In all instances, this occurs in a straight line, and the difference in performance between the models is smaller before the deceleration spike. Extrapolation of the data points on a straight line before the spike probably describes the real behavior of the vehicle more accurately.

Compared to the point mass model, the solution of the single-track model has a much lower velocity in the high-speed corners around 1300 m (turn 3), and 2900 m (turn 9). In these turns, the longitudinal acceleration is close to zero, so load transfer is insignificant. However, due to the

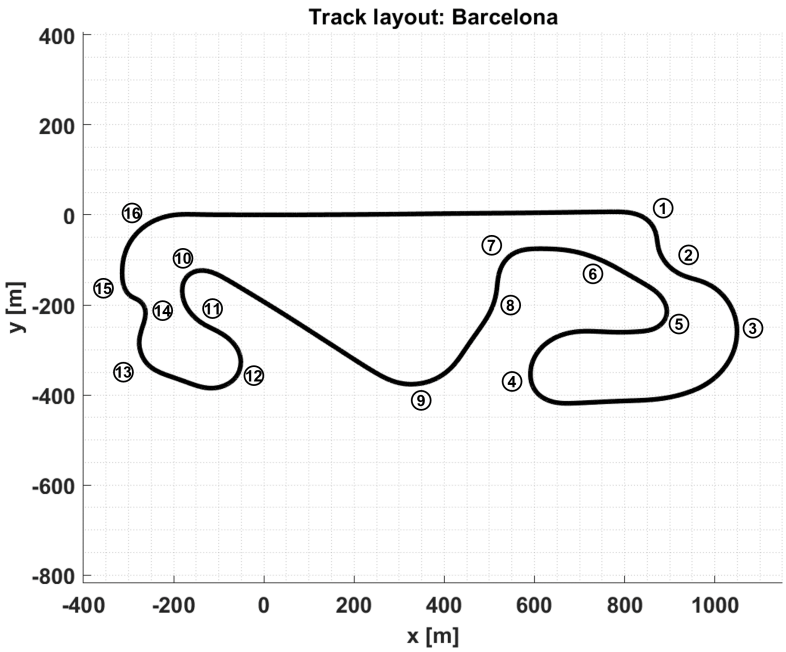


Figure 6.6: Track layout of Circuit de Barcelona-Catalunya.

high velocity, the aerodynamics of the vehicle are very important. As seen before, the downforce distribution is tuned more towards the rear of the vehicle and the front tire loses load due to the application of the aerodynamic drag force. Similarly to the steady-state cornering case discussed in the previous section, the front tire is the limiting factor of the vehicle here and compared to the point mass model, it can produce a significantly smaller friction force due to the imbalance of the vertical forces between the front and the rear wheel. As a result, the single-track model achieves lower velocities in high-speed corners.

Another interesting difference is found in the braking performance under heavy braking, which for example happens around 750 m (turn 1). A heavy braking zone consists of roughly four stages. First, the vehicle is at its top speed before the brakes are applied. In the LTS algorithm, the vehicle immediately brakes with its maximum possible deceleration. Throughout the braking zone, the velocity is lowered, which in return decreases the vertical force due to downforce. As a result, the maximum available grip decreases and the vehicle can only achieve lower longitudinal accelerations throughout the braking zone, until the final stage, the corner entry, is reached.

In comparison to the conventional algorithm, the peak deceleration of the single-track model is lower due to load transfer from the rear tire to the front tire as observed on the previous circuit as well. However, further into the braking zone, the single-track model is faster than the point mass model as indicated by the positive time gain. This is because of the implementation of the braking in the reverse loop of the conventional algorithm. The conventional algorithm calculates the value of $a_{x,i-1}$ based on V_i, which is incorrect. As $V_i < V_{i-1}$, the deceleration is underestimated compared to the physical limit. This small difference makes the single-track model faster during the braking zone and more accurate. This is done because both the acceleration and velocity have to be determined at iteration $i-1$, but only one equation, the friction ellipse (2.4),

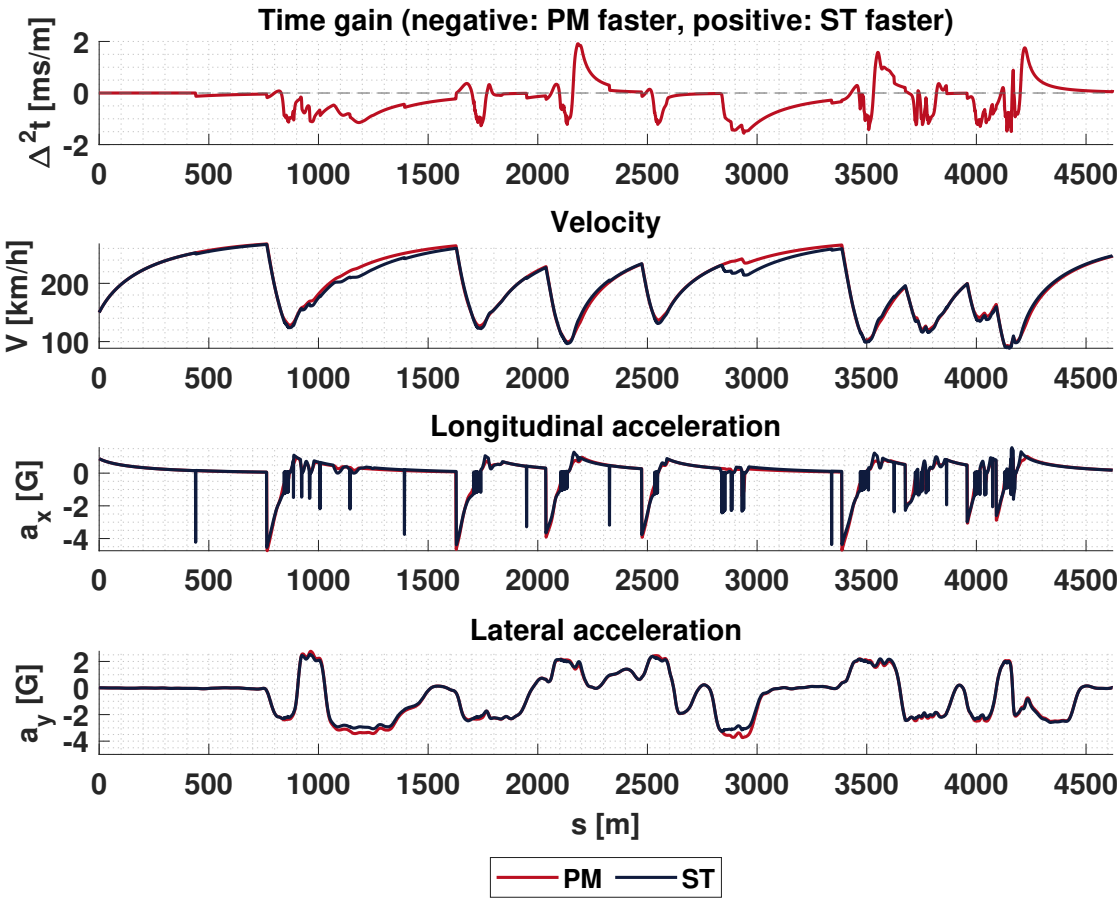


Figure 6.7: Comparison between the point mass and the single-track model LTS on Circuit de Barcelona-Catalunya. PM stands for point mass model, ST stands for single-track model.

is used. This can be avoided by using the consistency constraint (5.1) to substitute the velocity. However, this correction was overlooked in this work.

Finally, the data point before one starts braking will have a higher top speed in the conventional model. This is due to the implementation of the traction part of the reverse loop. In the conventional algorithm, the vehicle is assumed to be traveling at the velocity calculated in the forward loop V_{i-1}, and $a_{x,i-1}$ is calculated based on this V_{i-1} from the forward loop and V_i from the reverse loop to ensure consistency between the velocity and longitudinal acceleration. Since $V_i < V_{i-1}$, the longitudinal acceleration will still be negative. As a result, the tire is no longer fully utilized. This ensures that the velocity curve of the reverse loop returns to the velocity curve of the forward loop.

In the revised algorithm, this correction is not made resulting in a larger difference in velocity between the forward and reverse loop. This illustrates a drawback of the revised algorithm. However, the original traction implementation of the reverse loop cannot be applied for the single-track model, as a change in the longitudinal acceleration results in a change in the velocity.

The most notable positive time gain of the revised model is around 2200 m, which is the acceleration out of the slow turn 5. In the single-track model, the rear tire gains load from the front tire under traction. This phenomenon, along with the larger friction coefficient of the rear tire, increases the total friction force and thus increases the maximum possible velocity out of the turn compared to the conventional model.

Compared to the conventional model, the single-track model shows a lot of oscillations between a positive and negative longitudinal acceleration in the corners. These oscillations may be due to two factors: a rapid increase of curvature, or a rapid decrease of velocity. The first instance may be a result of a too large track step size Δs, whereas the second instance is mainly a result of the encountered problem in Section 6.1.2. Since the vehicle is very close to the limit of performance in a corner, a small perturbation may already lead to an invalidation of the solution.

It is important that the computational time of the revised algorithm is relatively low in a realistic use case like this one. The computational time has increased from 4.03 ms for the point mass model to 7726.2 ms for the single-track model. This increase is almost a factor 2000, but is still within ten seconds for an entire simulation. This is still an acceptable result, that would be feasible for commercial software.

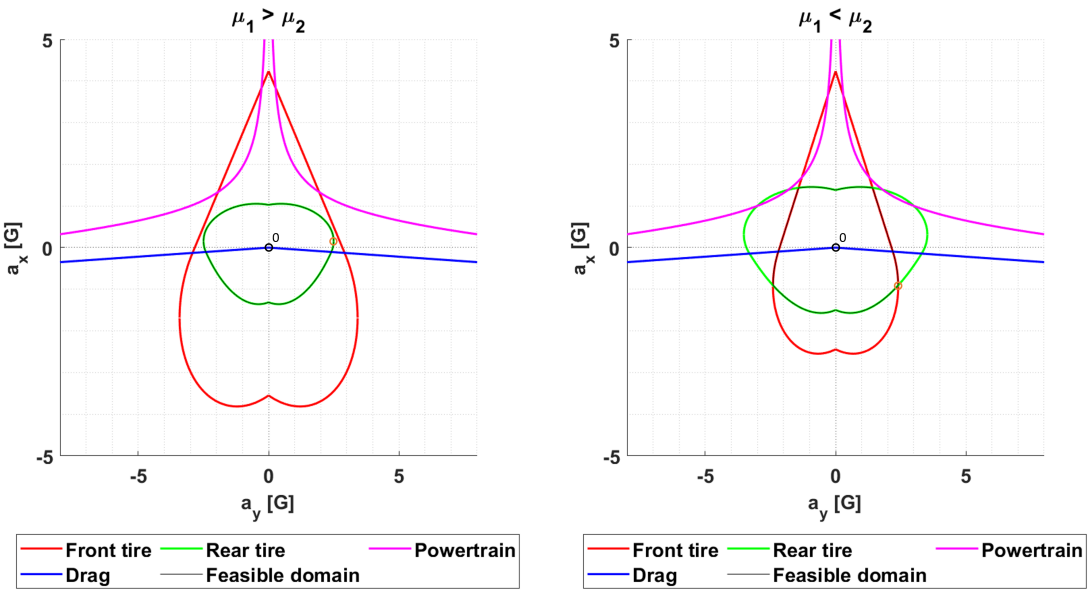
6.3 Effect of differences in tire friction coefficient

With the revised LTS algorithm, one can check how different front and rear tires affect the vehicle. The LMP2 car has different tires on the front and rear axles, where the rear tire has a higher friction coefficient. In this section, a normal LMP2 vehicle with the vehicle parameters listed in Table 6.1 is compared to the same LMP2 vehicle with its front and rear tires swapped. A visualization of the adapted GG diagram for both vehicles at $\kappa = 0.02$ 1/m is given in Figure 6.8.

Ignoring the consistency constraint h_1, the maximum possible velocity through this turn, which is equivalent to the maximum possible lateral acceleration, depends on the longitudinal acceleration in both cases. For the vehicle with a better front tire, a maximum lateral acceleration is achieved when the rear tire constraint g_2 is active and the longitudinal acceleration is positive. In the case of the vehicle with a better rear tire, the velocity is maximized when both the grip constraints g_1 and g_2 are active and the longitudinal acceleration is negative and below the intersection of the drag and grip constraints. However, in the case of $\mu_1 < \mu_2$, the algorithm is unable to utilize the available tire grip. In the forward loop, only non-negative longitudinal accelerations are considered, so the maximum assigned velocity in a corner from the algorithm is smaller than the vehicle can attain. Due to the restrictive nature of the consistency constraint h_1 in the reverse loop, the vehicle will brake before the turn and enter in steady-state. An example of this is given in Figure 6.9, which is driven on the constant radius turn track from Figure 6.3.

In the turn, the vehicle with the better front tire is able to use its full potential by slowly accelerating to its maximum achievable velocity, whereas the original LMP2 vehicle takes the turn in steady-state. This allows for a faster corner speed in the case where the tires are switched. For the original LMP2, it would be faster if the vehicle starts the turn at its maximum velocity indicated in Figure 6.8b and gradually brakes through the turn until the steady-state velocity is achieved.

This process called trail braking cannot be simulated with the revised version of the LTS algo-



(a) $\mu_1 > \mu_2$.

(b) $\mu_1 < \mu_2$.

Figure 6.8: Difference in adapted GG diagram due to friction coefficient difference. The minimizers are indicated by orange circles.

rithm, which highlights a point of improvement. A suggestion would be to also include deceleration in optimization problem (5.7) of the forward loop. By doing this, one is able to set the correct grip limited maximum velocity for vehicles with a better rear tire. In the subsequent iteration, the velocity will be lower due to the deceleration, thus within the maximum velocity limit. However, the velocity will be too high to allow for acceleration. Therefore, optimization problem (5.6) should be split into a part for $a_{x,i} \geq 0$ and $a_{x,i} < 0$. The new negative part would minimize the deceleration to slowly return the velocity to a steady-state cornering solution.

Another difference between the vehicles is the behavior under braking. In both occasions, the rear tire grip is the limiting factor under braking. In the case that the rear tire is the worse tire, the maximum available friction force for braking is smaller due to a lower friction coefficient. This is observed in a lower peak deceleration and an earlier application of the brakes compared to the original LMP2 vehicle.

6.3.1 Stability

The feasibility of these solutions can be checked through the understeer gradient η, which is defined as follows:

$$\eta = \frac{mg}{L}\left(\frac{b}{C_1} - \frac{a}{C_2}\right) \tag{6.7}$$

where C_1 and C_2 represent the cornering stiffness of the front and rear tire, respectively. The cornering stiffness is defined as the slope of the lateral force curve around SA = 0 deg. It can be approximated by adding an arbitrary small step in slip angle ϵ_α to the slip angle input in the Magic Formula (2.5) to find the lateral force F_y^ϵ at that step, and computing the derivative of

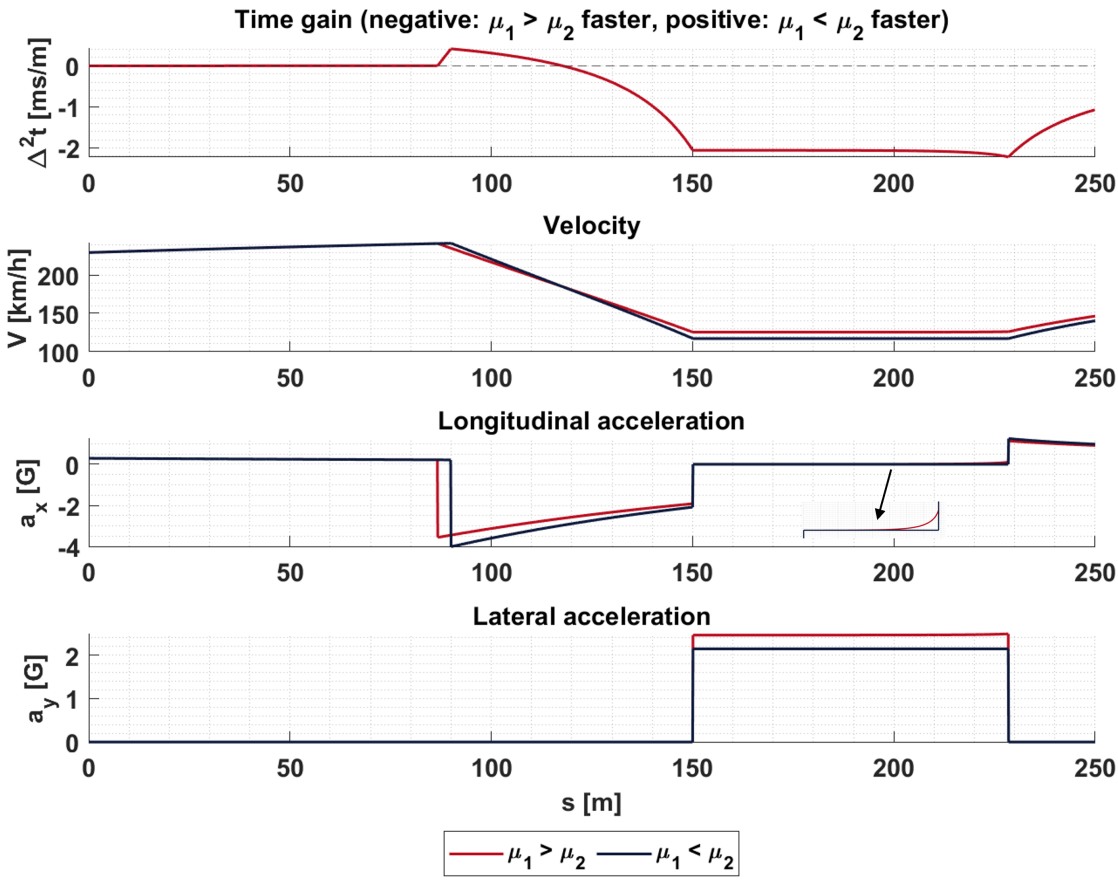


Figure 6.9: Comparison between vehicles with different tire combinations on the constant radius turn track.

the lateral force through a forward difference method similar to (3.11).

$$C = \frac{F_y^{\epsilon} - F_y}{\epsilon_\alpha} \tag{6.8}$$

The understeer gradient is a very important metric for steady-state cornering performance. It describes the required change in steering input for an increase in lateral acceleration, as defined by the following equation for small chassis slip angles β:

$$\delta = \frac{L}{R} + \frac{a_y}{g}\eta \tag{6.9}$$

For $\eta > 0$, the vehicle is understeered and the steering angle must be increased if the lateral acceleration is increased. For $\eta < 0$, the vehicle is oversteered and the steering angle must be decreased if the lateral acceleration is increased. Only oversteered vehicles can become unstable. This happens when the required steering angle to traverse the corner becomes negative.

Both vehicles have a positive understeer gradient for a free rolling wheel (SR = 0%) in the turn with $\kappa = 0.02$ 1/m while on the grip limit of the tires. The vehicle with a better front tire has

$\eta = 0.0028$, and the vehicle with the better rear tire has $\eta = 0.0045$. This means that for a free rolling tire, both vehicles follow stable trajectories. For the vehicle with the better front tire, a slight increase in traction (SR = 10%), like suggested by the revised LTS algorithm, leads to an increase of the understeer gradient, indicating that the solution is stable. In the case where the vehicle with the better rear tire is slightly braked (SR = −10%), the understeer gradient also increases. Therefore, both solutions are deemed to be stable.

In general, stability of the vehicle cannot always be guaranteed. However, this is mostly due to vehicle design and setup. A vehicle that has a front tire with a significantly higher cornering stiffness than the rear tire will be more likely to have oversteer as defined by (6.7). The higher the discrepancy in cornering stiffnesses, the lower the critical velocity becomes at which the vehicle becomes unstable.

To verify stability of the solution, one can calculate the understeer gradient with (6.7) at each point of the track, and derive the critical velocity in case of oversteer. The critical velocity can be calculated as follows:

$$V_{crit} = \sqrt{\frac{gL}{|\eta|}} \tag{6.10}$$

If the vehicle has oversteer in a corner ($\eta < 0$), and the velocity exceeds the critical velocity, the solution is unstable and therefore infeasible.

6.4 Comparison with real telemetry data

A comparison of the simulation results with data measured in a vehicle is a crucial step in the validation of the model and provides insights in its shortcomings. Due to the quasi steady-state approximation, one is already aware that the LTS algorithm will not include transient behavior like tire relaxation. Moreover, since a single-track vehicle model was used, lateral load transfer is not considered. The primary contrasts in the data obtained from simulation and telemetry can provide valuable pointers on where to begin enhancing the vehicle model and parameter tuning.

In both upcoming cases, the track curvature has been determined from the (time-filtered) telemetry using (2.18). As a result, an approximation of the track layout is created for one specific lap. Comparing the output of the LTS algorithm with this track layout for a different lap will lead to slight errors as the taken racing line may be marginally different.

6.4.1 LMP2 at Circuit de Barcelona-Catalunya

First, the LMP2 car with vehicle parameters listed in Table 6.1 will be compared to a qualifying run at the Circuit de Barcelona-Catalunya as shown in Figure 6.6. The engine power is reduced to 427.2 kW, to account for a 95% mechanical efficiency. The results can be found in Figure 6.10.

The curvature input for the LTS algorithm has been smoothed to minimize erratic behavior due to rapid changes in curvature. The simulation and telemetry are found to be in agreement on the apices in the low- and medium-speed corners. The only times that the apex speeds are dissimilar are at the high-speed turns 3 (1200 m) and 9 (2900 m). These turns are taken at a significantly high speed, which results in a large lateral acceleration. In the simulation, this has no side effect, but in reality the vehicle experiences lateral load transfer from the inside tires to the outside tires. In turn 9, the vertical force balance on each tire is as given in Figure 6.11b.

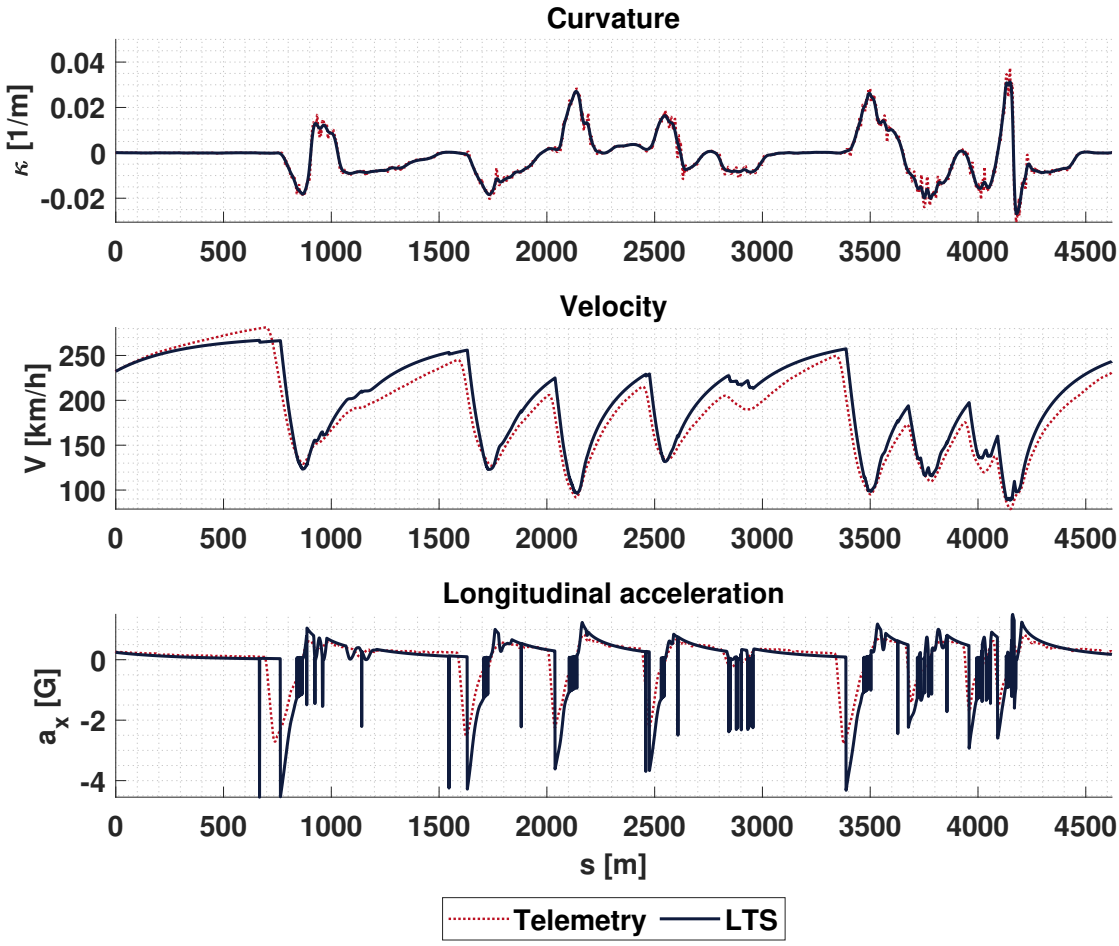
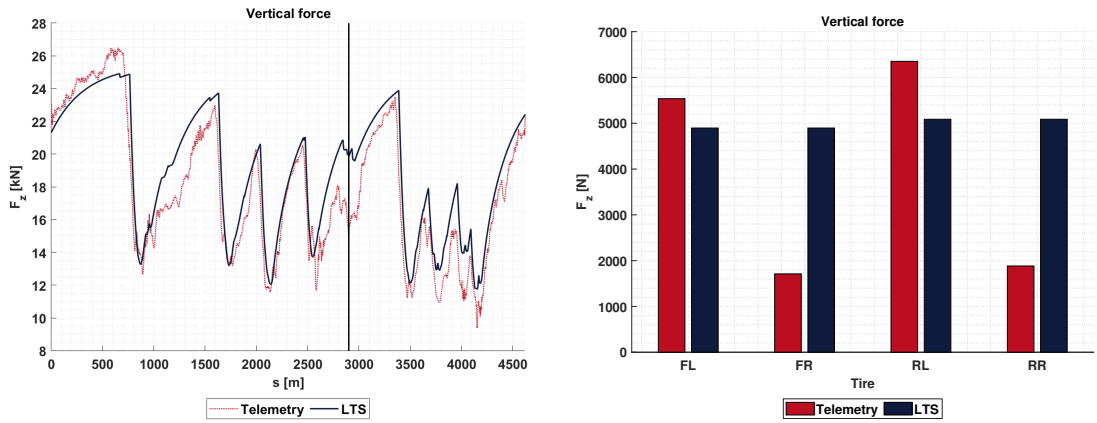


Figure 6.10: Comparison between telemetry and the single-track LTS on Circuit de Barcelona-Catalunya.

In this right-hand turn, the load is transferred from the right tires to the left tires. In the simulation, it was already determined that the front tire limits the performance in this turn. Due to the lateral load transfer, the front right tire loses a significant amount of vertical force, which in return limits the available friction force. As a result, the vehicle is already grip limited at a lower velocity.

However, the addition of all vertical forces is not consistent between the telemetry and the LTS in this turn. Due to the omission of lateral load transfer, the vehicle can take tighter corners at increased velocity. This will, in return, increase the effect of the aerodynamic downforce on the vertical force, creating a bigger difference between the simulation and reality. Additionally, the vehicle body rolls in a corner, which changes the angle of attack of the aerodynamic elements. This effect decreases k_L and shifts the downforce distribution. Moreover, the aerodynamics of the vehicle are highly dependent on the wind direction and intensity, which is very difficult to quantify.



(a) Total vertical force throughout the lap.

(b) Vertical force decomposition at turn 9.

Figure 6.11: Vertical force on the vehicle.

Overall, the total vertical force on the vehicle is consistent between the simulation and data as can be seen in Figure 6.11a.

After the initial grip limited phase of traction out of the corner, the simulation is displaying a higher velocity than in the real case once the power limit becomes active. The powertrain model used in the lap time simulation algorithm is very simplistic, whereas the real power output depends on the selected gear, differential, thermal and mechanical efficiency, chemical processes inside the engine, and many more factors. As a result, the actual deliverable power is always lower than the maximum available power. This means that the simulation will have a higher velocity in the power limited areas when accelerating out of the grip limit.

Another clear difference is found in the braking performance. In the data, the maximum deceleration is -2.72 G, whereas the LTS manages a maximum deceleration of -4.53 G. At this point, the vertical force on one of the front tires is around 9 kN. This is around the same load as the weight of the entire vehicle, but concentrated on a single tire. In the case of such an extreme force, it is likely that the maximum wheel travel of around 16 mm is exceeded and the bump stop is triggered on the heave spring. In this region, the spring rate drastically increases as a separate spring is engaged that prevents a collapse of the suspension. This causes a loss in performance. With a heave spring rate of 250 N/mm and a motion ratio of 1 at the front suspension, this would mean a suspension travel of 36.5 mm. This indeed exceeds the desired operating window of the suspension, meaning that heavy braking is unrealistic.

Moreover, in the LTS the deceleration is instantaneous, while in the real case the vehicle takes around 40 meters to reach the peak deceleration. This is due to the inertia of the vehicle body. An instant deceleration means that the jerk on the CoG is infinite, which is physically impossible. This illustrates an example of a loss of transient behavior in the simulation.

The only time the telemetry has a higher velocity than the simulation is at the end of the main straight around 700 m. This might be due to an overestimation of k_D. k_D was estimated using the rule of thumb that it should be about three times smaller than k_L. This is a very crude assumption and is based on a constant that varies under different conditions. The actual drag

coefficient is probably slightly lower than given in this work.

6.4.2 Formula Student at AutoX

To test the versatility of the LTS algorithm, also a Formula Student car is evaluated on an AutoX track. Formula Student vehicles are much smaller and lighter than the LMP2 car, but also have a lower maximum power output of 80 kW as mandated by the regulations [41]. A list of vehicle parameters is given in Table 6.2.

Table 6.2: List of vehicle parameters for the Formula Student car.

Parameter	Description	Value
	Mass	
m	Vehicle mass	287 [kg]
g	Gravitational acceleration	9.81 [m/s^2]
W_d	Weight distribution	52.0 [%]
L	Wheelbase	1.535 [m]
h	CoG height	0.350 [m]
	Tires	
μ_1 & μ_2	Friction coefficient	1.60 [-]
	Aerodynamics	
k_D	Drag coefficient	1.53 [N/(m^2/s^2)]
k_L	Lift coefficient	2.00 [N/(m^2/s^2)]
D_d	Downforce distribution	50.0 [%]
d	Vertical distance CoG-CoP	0.00 [m]
	Drivetrain	
P	Engine power	45 [kW]
f_{rr}	Rolling resistance coefficient	0.015 [-]
p_D	Torque distribution	25.0 [%]
p_B	Brake distribution	70.0 [%]

Many of these parameters have been estimated based on telemetry data. For example, in reality, the power output is controlled by a control algorithm that considers many different vehicle states such as wheel speed, throttle input, and battery state of charge. For this reason, it is crucial that the power output in a constant power model is modeled as an average achievable peak power. Here, a power output of 45 kW was chosen as most peaks in the battery output match or exceed this threshold.

The drag and lift coefficients have been estimated through Computational Fluid Dynamics (CFD) simulations. CFD simulations, while vital for predicting race car aerodynamics, face challenges in fully capturing the complex, unsteady flow phenomena involved. It is nearly impossible to recreate a computer design and usually simplifications are made in the geometry of a simulation as a trade off for computational efficiency. Tire-wake interactions, real-world surface effects, and dynamic forces on the vehicle structure pose difficulties. Moreover, on-track conditions are always variable, which may lead to discrepancies with the simulation environment.

Furthermore, the vehicle model assumes a fixed torque and brake distribution. However, in reality the Formula Student car is equipped with an electric motor in each wheel, that can be controlled

individually through a process called torque vectoring. Torque vectoring is the regulation of the torque distribution between each wheel, and optimizes the tire grip in tight corners. Moreover, the brake distribution can be changed by the driver, although this does not happen in practice due to the twisty layout of Formula Student tracks and the effort it takes to accommodate such a change on the steering wheel. As a compromise, it was chosen to tune the torque distribution more towards the rear tire and the brake distribution more towards the front tire, as this allows for better utilization during longitudinal load transfer.

The AutoX is one of the four dynamic events of a Formula Student competition. The purpose of the AutoX is to test the speed of the car over a single lap at maximum power. AutoX tracks have a small track width and are characterized by numerous consecutive sharp corners with short straight sections, which are better suited to the lower power cars and effectively highlights the cornering capabilities of the vehicles. In this work, the manual AutoX layout at Formula Student Germany 2023 is used [42]. The track layout was determined through telemetry and discretized with a step size of $\Delta s = 0.5$ m. The track layout can be found in Figure 6.12.

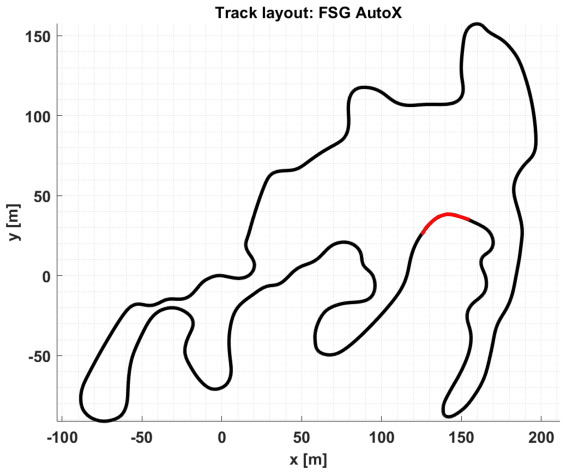


Figure 6.12: Track layout of the 2023 FSG Manual AutoX.

A comparison between the telemetry data and the LTS simulation is shown in Figure 6.13. One must note that no professional racing driver was used for this run, meaning that they unlikely will have extracted the maximum performance out of the vehicle at each point of the lap.

A clear observation is how the straight-line speed differs. The simulated velocity can reach up to 100 km/h, whereas the vehicle was actually driven no faster than 80 km/h. This is because of the powertrain model. The average battery power output is lower than the constant power assumed by the LTS algorithm. Moreover, not all the power from the battery is converted to propulsion as the electric motors and drivetrain suffer from mechanical power losses due to friction. This further limits the achievable velocity.

One can also notice that the acceleration out of tight corners is larger than the acceleration found by the simulation. This is a result of the torque vectoring on the electric motors in each wheel. The outer rear wheel receives more torque as it has more grip due to longitudinal and lateral load transfer. A static torque distribution between the front and rear wheel as used in the LTS

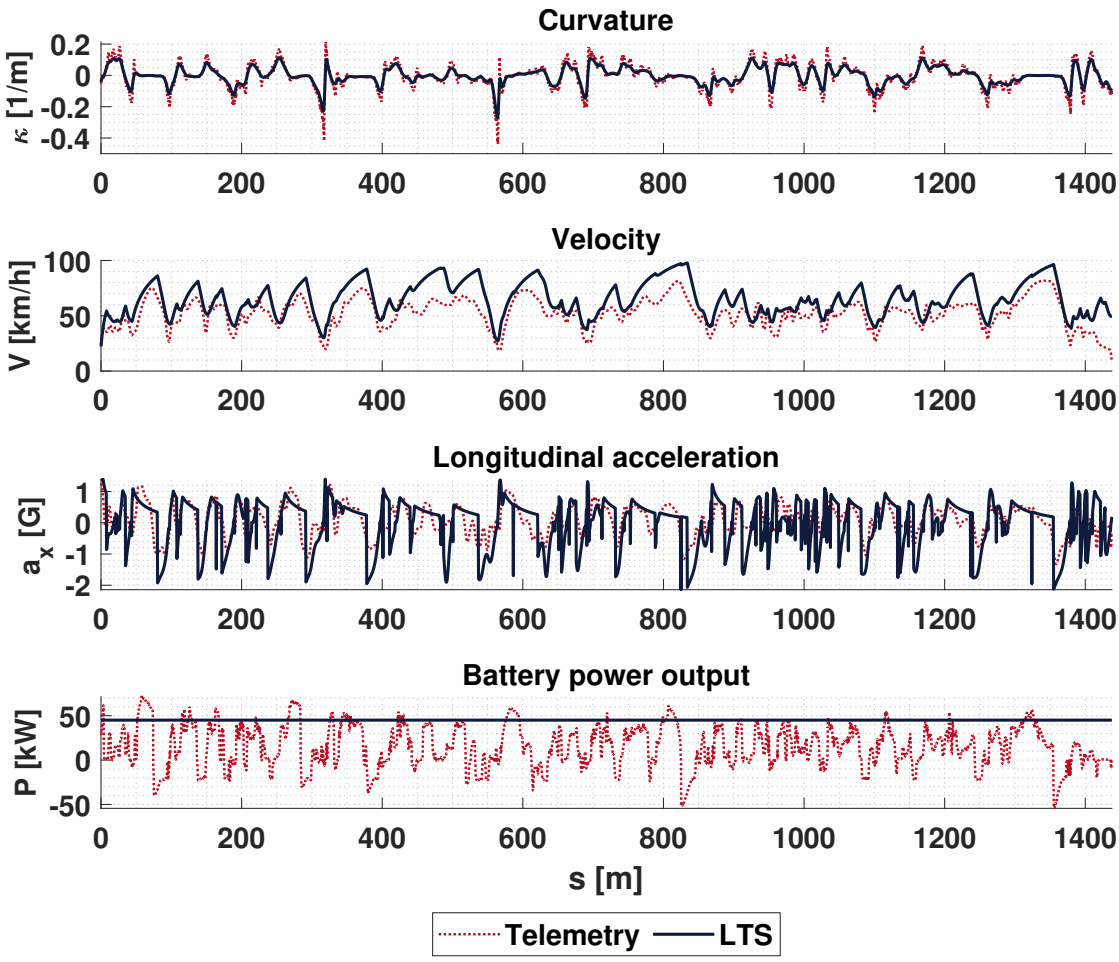


Figure 6.13: Comparison between telemetry and the single-track LTS on a Formula Student AutoX.

cannot be optimized for each corner individually.

In this example, the maximum deceleration of the LTS algorithm is larger than the actual maximum deceleration. Due to longitudinal load transfer, the front wheel will have an increased load. This Formula Student car was driven with a front spring rate of 65 kg/cm to account for non-smooth road surfaces. At peak deceleration, the vertical force on a single tire is almost equal to 1.6 kN in the LTS algorithm. This would result in a suspension travel of around 25 cm, which is far larger than the allowed suspension travel of 5 cm before triggering the bump stop. Moreover, the deceleration is instantaneous, which is impossible because of the inertia of the vehicle body.

Overall, the shape of the data appears to be similar, and the velocity is within a 10 km/h difference in the low speed corners where the vehicle is grip limited. From the LTS, one can also give some feedback to the driver. For example, around 790 m, the left-hand turn on the bottom right after the slalom indicated in red in Figure 6.12, the driver lifts off the throttle and loses

speed as a result. According to the LTS, this corner can be taken by a shorter application of the brakes. However, many important details that may limit the velocity like disruptive wind conditions or bumps in the track have not been taken into account, and should be considered when reviewing the driver performance.

6.5 Summary

The revised lap time simulation algorithm is found to have two deficiencies: the solution to the optimization problem may fall outside the feasible domain, and a rapid loss of velocity on stretches with small curvature. Neither shortcoming proves to be a major problem to performing a track simulation. The revised single-track LTS algorithm is first compared to a conventional point mass LTS algorithm for an LMP2 car on two different tracks. It is found that the enhanced physical models perform as expected, as the velocity is lower under braking and in (near) steady-state cornering, and the velocity is increased under traction. The computational effort has increased by a factor 1000 to 2000, but is still within ten seconds for an entire simulation. The maximum tire grip available for vehicles with a larger rear friction coefficient is found not to be exploited by the revised algorithm, whereas vehicles with better front tires fully utilize the tire grip while cornering. Both solutions are found to be stable through an analysis of the understeer gradient. A comparison with telemetry reveals that the implementation of an improved powertrain model and lateral load transfer is crucial for the improvement of the validity of the LTS algorithm.

7 Conclusions and recommendations

In this work, a lap time simulation algorithm was developed. It was decided to take a quasi steady-state approach with a predefined vehicle trajectory for the benefit of computational efficiency. A single-track model with an elevated center of gravity is used as the vehicle model, which includes longitudinal load transfer, split torque between the front and rear axle, and unequal weight, downforce and grip distributions. The powertrain model is a constant power model.

First, the direct optimization method proposed by Beckers [21] was changed to a nested optimization method. In this method, the state variables are removed from the design vector and replaced by only the vehicle inputs steering angle and motor torque. As a result, the vehicle state is computed inside the optimization problem by means of a Newton-Raphson algorithm. For a kinematic movement initial condition, the nested optimization was able to find the same gripping condition as the direct optimization in a corner radius range from 10 to 200 m with a computational time that is five times longer per initial guess, but with a higher average success rate of 57.1%. The drifting condition was not found, unless the solution of the direct optimization was provided as an initial condition to the Newton-Raphson solver. However, this was at the cost of a six times increased average computational time per initial guess and an average success rate of only 25.3%. This method is slower than the original direct optimization, but, when the initial guesses are chosen cleverly, it may be faster than the direct optimization due to requiring less samples to converge. However, the large computational times of the model deemed it to be not desirable for investigating further implementation into a lap time simulation algorithm.

Another approach is the implementation of optimization theory to find the maximum performance of the vehicle at each point of the track. A restructuring of the forward and reverse loop of conventional LTS algorithms took place, and two optimization problems were added to each loop. Due to the structure of the optimization problems, a set of potential minimizers could be determined and (semi-)analytical expressions for each potential minimizer can be obtained.

This method appears to have two inherent deficiencies: the solution to the optimization problem sometimes falls outside the feasible domain, and a rapid loss of velocity may take place on straight lines or paths with very small curvature. This is as a result of the optimization problem for traction in the reverse loop. The additional optimization step is unnecessary and overwrites the solution found in the optimization problem that maximizes the longitudinal acceleration in the forward loop.

Compared to a conventional lap time simulation algorithm with a point mass vehicle model, the revised lap time simulation algorithm clearly exhibits the effects of longitudinal load transfer and an unequal distribution of quantities between the front and rear axle. Moreover, a simulation of an LMP2 car at the Circuit de Barcelona-Catalunya takes 7726.2 ms, which makes it still attractive as an implementation in commercial software.

Furthermore, it has been found that vehicles with a higher friction coefficient on the front tire gradually accelerate through a constant radius corner. Vehicles with a higher friction coefficient on the rear tire are expected to decelerate to steady-state through a constant radius turn (trail braking). However, it was found that the current algorithm does not allow for trail braking in a turn. Both solutions were found to have a positive understeer gradient, indicating that the solutions are stable.

The LTS algorithm is also evaluated using telemetry of two very different race cars. It was found that the simulation is very similar to the telemetry in the grip limited region of the slow- and medium-speed corners. However, some inconsistencies are also observed under braking, in high-speed corners, and while the vehicle is power limited due to various factors that are unaccounted for.

Recommendations are given below in order of importance:

- One should not maximize the velocity again when the vehicle is in traction in the reverse loop. Currently, the behavior under traction is optimized twice. The first optimization of the longitudinal acceleration in the forward loop is similar to the conventional algorithm. However, in a straight line or small curvature, this is already a satisfactory result. In the cases that the vehicle is not braking and cornering, a simple correction of the longitudinal acceleration using the velocities from the subsequent iterations like in the conventional algorithm is not viable.
- The powertrain model should be improved. In both comparisons with telemetry, the vehicle receives too much power when it is power limited in the simulation. This is a result of the simplified powertrain model. An important improvement is an increase of complexity of the powertrain model. A method similar to Hendrikx, et al. [12] can be applied for a wide range of powertrain configurations, where the relation between the motor angular velocity and torque is converted into a relation between the vehicle velocity and longitudinal force.
- The forward loop optimization problems should be changed to also include deceleration. In the current forward loop, a vehicle with a larger friction coefficient on the rear tire is unable to fully utilize the tire grip while cornering. Allowing a maximum velocity at which the vehicle has to decelerate in the forward loop will create the desired trail braking effect.
- The size of the analytical solutions should be minimized. Since the obtained analytical solutions of the potential minimizers are very large, one should try to gather similar terms and make substitutions of a common combination of constants. This will significantly decrease the size of the source files that the compiler has to read and translate, and will therefore improve the computational time.
- The vehicle model should be upgraded to a four-wheel model with lateral load transfer. In comparison with telemetry, simulation vehicles gain a significant amount of performance in high-speed corners. This is mainly caused by a loss of vertical force on the inner tires due to lateral load transfer.
- The algorithm should be programmed in a programming environment that allows for better code optimization. This will improve the computational time of the model.

References

[1] C. Rouelle, *Advanced Vehicle Dynamics Applied to Race Car Design & Development*, Apr. 2022.

[2] W. F. Milliken and D. L. Milliken, *Race Car Vehicle Dynamics*, 1st Edition. Warrendale, PA: SAE International, 1995.

[3] M. Gadola, D. Vetturi, D. Cambiaghi, and L. Manzo, "A Tool for Lap Time Simulation," in *Motorsports Engineering Conference Exposition*, SAE International, Dec. 1996. DOI: https://doi.org/10.4271/962529.

[4] B. Siegler, A. Deakin, and D. Crolla, "Lap Time Simulation: Comparison of Steady State, Quasi- Static and Transient Racing Car Cornering Strategies," in *Motorsports Engineering Conference Exposition*, SAE International, Nov. 2000. DOI: https://doi.org/10.4271/2000-01-3563.

[5] D. L. Brayshaw and M. F. Harrison, "A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 219, no. 6, pp. 725–739, 2005. DOI: 10.1243/095440705X11211.

[6] T. Fujioka and T. Kimura, "Numerical simulation of minimum-time cornering behavior," *Jsae Review*, vol. 13, 1992.

[7] D. Casanova, "On minimum time vehicle manoeuvring: The theoretical optimal lap," 2000.

[8] D. P. Kelly, "Lap time simulation with transient vehicle and tyre dynamics," 2008.

[9] E. Velenis and P. Tsiotras, "Optimal velocity profile generation for given acceleration limits: Theoretical analysis," in *Proceedings of the 2005, American Control Conference, 2005.*, 2005, 1478–1483 vol. 2. DOI: 10.1109/ACC.2005.1470174.

[10] J. P. Timings and D. J. Cole, "Minimum Maneuver Time Calculation Using Convex Optimization," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 3, p. 031 015, Mar. 2013, ISSN: 0022-0434. DOI: 10.1115/1.4023400.

[11] M. Massaro and D. J. N. Limebeer, "Minimum-lap-time optimisation and simulation," *Vehicle System Dynamics*, vol. 59, no. 7, pp. 1069–1113, 2021. DOI: 10.1080/00423114.2021.1910718.

[12] J. Hendrikx, T. Meijlink, and R. Kriens, "Application of optimal control theory to inverse simulation of car handling," English, *Vehicle System Dynamics*, vol. 26, no. 6, pp. 449–461, 1996, ISSN: 0042-3114.

[13] H. Winner, T. Völkl, and M. Muehlmeier, "Extended Steady State Lap Time Simulation for Analyzing Transient Vehicle Behavior," *SAE International Journal of Passenger Cars - Mechanical Systems*, vol. 6, no. 1, pp. 283–292, Apr. 2013, ISSN: 1946-3995. DOI: https://doi.org/10.4271/2013-01-0806.

[14] R. Lot and S. Evangelou, "Lap Time Optimization of a Sports Series Hybrid Electric Vehicle," Jul. 2013. DOI: 10.13140/2.1.2166.3522.

[15] M. Masouleh and D. Limebeer, "Optimizing the Aero-Suspension Interactions in a Formula One Car," *IEEE Transactions on Control Systems Technology*, vol. 24, pp. 1–16, Oct. 2015. DOI: 10.1109/TCST.2015.2475396.

[16] A. J. Tremlett and D. J. N. Limebeer, "Optimal tyre usage for a Formula One car," *Vehicle System Dynamics, International Journal of Vehicle Mechanics and Mobility*, vol. 54, no. 10, pp. 1448–1473, Oct. 2016. DOI: 10.1080/00423114.2016.1213861.

[17] G. Perantoni and D. J. N. Limebeer, "Optimal Control of a Formula One Car on a Three-Dimensional Track—Part 1: Track Modeling and Identification," *Journal of Dynamic Systems, Measurement, and Control*, vol. 137, no. 5, p. 051 018, May 2015, ISSN: 0022-0434. DOI: 10.1115/1.4028253.

[18] F. Christ, A. Wischnewski, A. Heilmeier, and B. Lohmann, "Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients," *Vehicle System Dynamics*, vol. 59, pp. 1–25, Dec. 2019. DOI: 10.1080/00423114.2019.1704804.

[19] V. Cossalter, M. Lio, R. Lot, and L. Fabbri, "A General Method for the Evaluation of Vehicle Manoeuvrability with Special Emphasis on Motorcycles," *Vehicle System Dynamics - VEH SYST DYN*, vol. 31, pp. 113–135, Feb. 1999. DOI: 10.1076/vesd.31.2.113.2094.

[20] G. Radhakrishnan, C. Rouelle, I. J. M. Besselink, and P. Calorio, "A structured approach to analyzing vehicle dynamics," in *Advances in Dynamics of Vehicles on Roads and Tracks II*, A. Orlova and D. Cole, Eds., Cham: Springer International Publishing, 2022, pp. 617–641, ISBN: 978-3-031-07305-2.

[21] A. Beckers, "Numerical Optimization of Steady State Cornering in Racing Applications," DC 2023.010, M.S. thesis, Eindhoven University of Technology, Eindhoven, the Netherlands, 2022.

[22] S. C. Themen, "Velocity Control for a Dynamic Vehicle Model for Extreme Driving Conditions," Eindhoven University of Technology, Tech. Rep., 2021, DC 2021.095.

[23] P. A. Calorio, *Methodologies for optimizing the performance of race vehicles*, Escola de Engenharia de São Carlos da Universidade de São Paulo, 2022.

[24] H. B. Pacejka and I. J. M. Besselink, *Tire and Vehicle Dynamics*. Butterworth-Heinemann, 2012.

[25] Foster, P., *23 United Autosports USA Oreca 07 - Gibson LMP2 of Alexander Lynn, Oliver Jarvis, Joshua Pierson, Paul di Resta*, Photograph, 24 Hours of Le Mans test day, 2022. [Online]. Available: https://www.autosport.com/wec/news/hypercar-pace-will-decide-future-of-lmp2-class-in-wec-jarvis/10318912/ (visited on 05/12/2023).

[26] P. Y. Papalambros and D. J. Wilde, *Principles of Optimal Design*. USA: Cambridge University Press, 2003, ISBN: 0521627273.

[27] L. F. P. Etman, *Optimization problem formulation*, Lecture slides, 4DM20 Engineering Optimization, Eindhoven University of Technology, 2021.

[28] G. R. Radhakrishnan, "A Structured Approach To Analysing Vehicle Dynamics," DC 2020.106, M.S. thesis, Eindhoven University of Technology, Eindhoven, the Netherlands, 2020.

[29] C. J. J. Beckers, I. J. M. Besselink, and H. Nijmeijer, "Assessing the impact of cornering losses on the energy consumption of electric city buses," *Transportation Research Part D: Transport and Environment*, vol. 86, p. 102 360, 2020, ISSN: 1361-9209.

[30] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*. McGraw-Hill, 2010.

[31] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd. New York, NY: Springer Science & Business Media, 2006, ISBN: 9780387303031.

[32] MathWorks, *fmincon*. [Online]. Available: https://www.mathworks.com/help/optim/ug/fmincon.html.

[33] MathWorks, *lhsdesign*. [Online]. Available: https://www.mathworks.com/help/stats/lhsdesign.html.

[34] I. J. M. Besselink, *Spectrum of Automotive - 5ATA0*, Lecture notes Vehicle Dynamics, 2021.

[35] R. Rice, "MEASURING CAR-DRIVER INTERACTION WITH THE g-g DIAGRAM," in *SAE Technical Paper*, 1973.

[36] W. Toet, "Aerodynamics and aerodynamic research in Formula 1," *The Aeronautical Journal*, vol. 117, no. 1187, pp. 1–26, 2013.

[37] MathWorks, *Symbolic Math Toolbox*. [Online]. Available: `https://www.mathworks.com/products/symbolic.html`.

[38] J. Larminie and J. Lowry, *Electric Vehicle Technology Explained*. John Wiley & Sons, 2003.

[39] L.-J. Buckle. "Why Testing happens at Barcelona – About the Circuit." (2021), [Online]. Available: `https://www.insidef2.com/why-testing-happens-at-barcelona-about-the-circuit/`.

[40] European Le Mans Series. "4 Hours of Barcelona." (2023), [Online]. Available: `https://www.europeanlemansseries.com/race/4746`.

[41] Formula Student Germany, *Formula Student Rules 2023*, 2023. [Online]. Available: `https://www.formulastudent.de/fsg/rules/`.

[42] Formula Student Germany, *Team Briefing Saturday*, 2023. [Online]. Available: `https://www.formulastudent.de/fileadmin/user_upload/all/2023/important_docs/FSG2023_Teambriefing_Saturday.pdf`.

A Derivation of constraints

A.1 Front tire grip

The grip limit of a tire is given by the friction circle in (2.4):

$$F_{x_1}^2 + F_{y_1}^2 = (\mu_1 F_{z_1})^2 \tag{A.1}$$

The values of the longitudinal, lateral, and vertical force can be taken from (2.23), (2.21), and (2.19) respectively:

$$F_{x_1} = p_V \left(ma_x + F_D\right) \tag{A.2}$$

$$F_{y_1} = \frac{b}{L} m a_y \tag{A.3}$$

$$F_{z_1} = \frac{b}{L} mg - \frac{h}{L} m a_x - \frac{h+d}{L} F_D + \frac{b-c}{L} F_L \tag{A.4}$$

Filling this into the friction circle gives:

$$\begin{gathered} g_1 = \left(p_V \left(ma_x + F_D\right)\right)^2 + \left(\frac{b}{L} m a_y\right)^2 - \\ \left(\mu_1 \left[\frac{b}{L} mg - \frac{h}{L} m a_x - \frac{h+d}{L} F_D + \frac{b-c}{L} F_L\right]\right)^2 \end{gathered} \tag{A.5}$$

Substituting the external forces $F_L = k_L V^2$, $F_D = k_D V^2$, and $a_y = V^2 \kappa$, and considering LTS iteration i results in:

$$\begin{gathered} g_1 = \left(p_V \left(ma_{x,i} + k_D V_i^2\right)\right)^2 + \left(\frac{b}{L} m V_i^2 \kappa_i\right)^2 - \\ \left(\mu_1 \left[\frac{b}{L} mg - \frac{h}{L} m a_{x,i} - \frac{h+d}{L} k_D V_i^2 + \frac{b-c}{L} k_L V_i^2\right]\right)^2 \end{gathered} \tag{A.6}$$

A.2 Rear tire grip

The grip limit of a tire is given by the friction circle in (2.4):

$$F_{x_2}^2 + F_{y_2}^2 = (\mu_2 F_{z_2})^2 \tag{A.7}$$

The values of the longitudinal, lateral, and vertical force can be taken from (2.24), (2.22), and (2.20) respectively:

$$F_{x_2} = \left(1 - p_V\right)\left(ma_x + F_D\right) \tag{A.8}$$

$$F_{y_2} = \frac{a}{L} m a_y \tag{A.9}$$

$$F_{z_2} = \frac{a}{L} mg + \frac{h}{L} m a_x + \frac{h+d}{L} F_D + \frac{a+c}{L} F_L \tag{A.10}$$

Filling this into the friction circle gives:

$$\begin{gathered} g_2 = \left(\left(1 - p_V\right)\left(ma_x + F_D\right)\right)^2 + \left(\frac{a}{L} m a_y\right)^2 - \\ \left(\mu_2 \left[\frac{a}{L} mg + \frac{h}{L} m a_x + \frac{h+d}{L} F_D + \frac{a+c}{L} F_L\right]\right)^2 \end{gathered} \tag{A.11}$$

Substituting the external forces $F_L = k_L V^2$, $F_D = k_D V^2$, and $a_y = V^2 \kappa$, and considering LTS iteration i results in:

$$g_2 = \left((1 - p_V)\left(m a_{x,i} + k_D V_i^2\right)\right)^2 + \left(\frac{a}{L} m V_i^2 \kappa_i\right)^2 - \left(\mu_2 \left[\frac{a}{L} m g + \frac{h}{L} m a_{x,i} + \frac{h+d}{L} k_D V_i^2 + \frac{a+c}{L} k_L V_i^2\right]\right)^2 \tag{A.12}$$

B Derivation of minimizers

For the derivation of the minimizers, a script was made using the Symbolic Math Toolbox in MATLAB [37]. In case of multiple solutions, the correct one was determined through visualization of the results.

```
clear
close all

%%%
```

```
%%% Minimizer 3: g3 active
axG3 = solve(g3 == 0, ax); % Three solutions, only one is correct
ax13 = axG3(1);

%% Forward loop - optimization problem (5.7)
%%% Minimizer 1: g1 active and ax = 0
vG1     = sqrt(solve(g1 == 0, vsq));    % Two solutions, only one is correct
v21     = subs(vG1(2), ax, 0);
ax21    = 0;

%%% Minimizer 2: g2 active and ax > 0
vG2         = sqrt(solve(g2 == 0, vsq));    % Two solutions, only one is correct

% Differentiate V to ax
dvG2        = diff(vG2(1), ax);

% Find maximum by setting the derivative equal to 0. Only the numerator may
% be considered
[num, ~]    = numden(dvG2);
num         = collect(num, (l^2*a^2*ax^2*kl^2*mu2^2 - l^2*a^2*ax^2*kappa^2*m^2 + l
    ^2*ax^2*c^2*kl^2*mu2^2 + l^2*ax^2*d^2*kd^2*mu2^2 + l^2*a^2*g^2*kd^2*mu2^2 + a
    ^4*g^2*kappa^2*m^2*mu2^2 + a^2*ax^2*h^2*kappa^2*m^2*mu2^2 + 2*l^2*a*ax^2*c*kl
    ^2*mu2^2 + 2*l^2*a^2*ax^2*kappa^2*m^2*pv - 2*l^2*a^2*ax^2*kl^2*mu2^2*pv - 2*l
    ^2*ax^2*c^2*kl^2*mu2^2*pv - 2*l^2*ax^2*d^2*kd^2*mu2^2*pv - 2*l^2*a^2*g^2*kd^2*
    mu2^2*pv - l^2*a^2*ax^2*kappa^2*m^2*pv^2 + l^2*a^2*ax^2*kl^2*mu2^2*pv^2 + l^2*
    ax^2*c^2*kl^2*mu2^2*pv^2 + l^2*ax^2*d^2*kd^2*mu2^2*pv^2 + l^2*a^2*g^2*kd^2*mu2
    ^2*pv^2 - 2*l^2*a*ax*d*g*kd^2*mu2^2 + 2*l^2*a*ax^2*d*kd*kl*mu2^2 + 2*l^2*ax^2*
    c*d*kd*kl*mu2^2 - 2*l^2*a^2*ax*g*kd*kl*mu2^2 - 4*l^2*a*ax^2*c*kl^2*mu2^2*pv +
    2*a^3*ax*g*h*kappa^2*m^2*mu2^2 + 2*l^2*a*ax^2*c*kl^2*mu2^2*pv^2 + 2*l^2*a*ax
    ^2*d*kd*kl*mu2^2*pv^2 + 2*l^2*ax^2*c*d*kd*kl*mu2^2*pv^2 - 2*l^2*a^2*ax*g*kd*kl
    *mu2^2*pv^2 - 2*l^2*a*ax*c*g*kd*kl*mu2^2 + 4*l^2*a*ax*d*g*kd^2*mu2^2*pv - 4*l
    ^2*a*ax^2*d*kd*kl*mu2^2*pv - 4*l^2*ax^2*c*d*kd*kl*mu2^2*pv + 4*l^2*a^2*ax*g*kd
    *kl*mu2^2*pv - 2*l^2*a*ax*d*g*kd^2*mu2^2*pv^2 - 2*l^2*a*ax*c*g*kd*kl*mu2^2*pv
    ^2 + 4*l^2*a*ax*c*g*kd*kl*mu2^2*pv)^(1/2));

% Isolate square root and square both sides
eq          = (num == 0) - (+ (g*l^2*a^2*kd*kl*m*mu2^2*pv^2 - 2*g*l^2*a^2*kd*kl*m*
    mu2^2*pv + g*l^2*a^2*kd*kl*m*mu2^2 - ax*l^2*a^2*kl^2*m*mu2^2*pv^2 + 2*ax*l^2*a
    ^2*kl^2*m*mu2^2*pv - ax*l^2*a^2*kl^2*m*mu2^2 + ax*l^2*a^2*kappa^2*m^3*pv^2 -
    2*ax*l^2*a^2*kappa^2*m^3*pv + ax*l^2*a^2*kappa^2*m^3 + g*l^2*a*c*kd*kl*m*mu2
    ^2*pv^2 - 2*g*l^2*a*c*kd*kl*m*mu2^2*pv + g*l^2*a*c*kd*kl*m*mu2^2 - 2*ax*l^2*a*
    c*kl^2*m*mu2^2*pv^2 + 4*ax*l^2*a*c*kl^2*m*mu2^2*pv - 2*ax*l^2*a*c*kl^2*m*mu2^2
     + g*l^2*a*d*kd^2*m*mu2^2*pv^2 - 2*g*l^2*a*d*kd^2*m*mu2^2*pv + g*l^2*a*d*kd^2*
    m*mu2^2 - 2*ax*l^2*a*d*kd*kl*m*mu2^2*pv^2 + 4*ax*l^2*a*d*kd*kl*m*mu2^2*pv - 2*
    ax*l^2*a*d*kd*kl*m*mu2^2 - ax*l^2*c^2*kl^2*m*mu2^2*pv^2 + 2*ax*l^2*c^2*kl^2*m*
    mu2^2*pv - ax*l^2*c^2*kl^2*m*mu2^2 - 2*ax*l^2*c*d*kd*kl*m*mu2^2*pv^2 + 4*ax*l
    ^2*c*d*kd*kl*m*mu2^2*pv - 2*ax*l^2*c*d*kd*kl*m*mu2^2 - ax*l^2*d^2*kd^2*m*mu2
    ^2*pv^2 + 2*ax*l^2*d^2*kd^2*m*mu2^2*pv - ax*l^2*d^2*kd^2*m*mu2^2 - g*a^3*h*
    kappa^2*m^3*mu2^2 - ax*a^2*h^2*kappa^2*m^3*mu2^2));
eq          = expand(eq^2);

% Move all terms to the same side and create a polynomial in ax
eq          = eq - rhs(eq);
eq          = collect(eq, ax);

% Minimizers
axS         = solve(eq, ax);                    % Two solutions, only one is correct
ax22        = axS(2);
v22         = subs(vG2(1), ax, ax22);

%%% Minimizer 3: g2 active and ax = 0
v23     = subs(vG2(1), ax, 0);
```

```
ax23     = 0;

%%% Minimizer 4: g3 active and ax = 0
vG3      = solve(subs(g3 == 0, vsq, v^2), v, "MaxDegree", 3);    % Three solutions, only one is correct
v24      = subs(vG3(1), ax, 0);
ax24     = 0;

%%% Minimizer 5: g1 & g2 active and ax > 0
% Equate solutions and multiply out denominators
eq = axG1(2) == axG2(1);
[~, denG1] = numden(axG1(2));
[~, denG2] = numden(axG2(1));
eq = expand(eq * denG1 * denG2);

% Isolate square roots and square both sides
left = collect(lhs(eq), (b^2*g^2*l^2*m^2*mu1^2*pv^2 + b^2*h^2*kappa^2*m^2*mu1^2*vsq^2 - b^2*kappa^2*l^2*m^2*pv^2*vsq^2 + b^2*kl^2*l^2*mu1^2*pv^2*vsq^2 + c^2*kl^2*l^2*mu1^2*pv^2*vsq^2 + d^2*kd^2*l^2*mu1^2*pv^2*vsq^2 - 2*b*c*kl^2*l^2*mu1^2*pv^2*vsq^2 - 2*b*d*kd*kl*l^2*mu1^2*pv^2*vsq^2 + 2*c*d*kd*kl*l^2*mu1^2*pv^2*vsq^2 + 2*b^2*g*kl*l^2*m*mu1^2*pv^2*vsq - 2*b*c*g*kl*l^2*m*mu1^2*pv^2*vsq - 2*b*d*g*kd*l^2*m*mu1^2*pv^2*vsq)^(1/2));
right = collect(rhs(eq), (a^2*g^2*l^2*m^2*mu2^2 - a^2*kappa^2*l^2*m^2*vsq^2 + a^2*kl^2*l^2*mu2^2*vsq^2 + c^2*kl^2*l^2*mu2^2*vsq^2 + d^2*kd^2*l^2*mu2^2*vsq^2 + a^2*g^2*l^2*m^2*mu2^2*pv^2 + a^2*h^2*kappa^2*m^2*mu2^2*vsq^2 - a^2*kappa^2*l^2*m^2*pv^2*vsq^2 + a^2*kl^2*l^2*mu2^2*pv^2*vsq^2 + c^2*kl^2*l^2*mu2^2*pv^2*vsq^2 + d^2*kd^2*l^2*mu2^2*pv^2*vsq^2 + 2*a*c*kl^2*l^2*mu2^2*vsq^2 - 2*a^2*g^2*l^2*m^2*mu2^2*pv + 2*a^2*kappa^2*l^2*m^2*pv*vsq^2 - 2*a^2*kl^2*l^2*mu2^2*pv*vsq^2 - 2*c^2*kl^2*l^2*mu2^2*pv*vsq^2 - 2*d^2*kd^2*l^2*mu2^2*pv*vsq^2 + 2*a*c*kl^2*l^2*mu2^2*pv^2*vsq^2 + 2*a*d*kd*kl*l^2*mu2^2*vsq^2 + 2*c*d*kd*kl*l^2*mu2^2*vsq^2 + 2*a^2*g*kl*l^2*m*mu2^2*vsq - 4*a*c*kl^2*l^2*mu2^2*pv*vsq^2 + 2*a*d*kd*kl*l^2*mu2^2*pv^2*vsq^2 + 2*c*d*kd*kl*l^2*mu2^2*pv^2*vsq^2 + 2*a^2*g*kl*l^2*m*mu2^2*pv^2*vsq + 2*a*c*g*kl*l^2*m*mu2^2*vsq + 2*a*d*g*kd*l^2*m*mu2^2*vsq - 4*a*d*kd*kl*l^2*mu2^2*pv*vsq^2 - 4*c*d*kd*kl*l^2*mu2^2*pv*vsq^2 - 4*a^2*g*kl*l^2*m*mu2^2*pv*vsq + 2*a*c*g*kl*l^2*m*mu2^2*pv^2*vsq + 2*a*d*g*kd*l^2*m*mu2^2*pv^2*vsq - 4*a*c*g*kl*l^2*m*mu2^2*pv*vsq - 4*a*d*g*kd*l^2*m*mu2^2*pv*vsq)^(1/2));
eq = left == right;
eq = eq - (+ (2*kd*l^4*m*pv^3*vsq - kd*l^4*m*pv^2*vsq - kd*l^4*m*pv^4*vsq + h^2*kd*l^2*m*mu1^2*vsq - h^4*kd*m*mu1^2*mu2^2*vsq + b*g*h^3*m^2*mu1^2*mu2^2 - b*g*h*l^2*m^2*mu1^2 - b*h*kl*l^2*m*mu1^2*vsq + c*h*kl*l^2*m*mu1^2*vsq + d*h*kd*l^2*m*mu1^2*vsq + 2*b*g*h*l^2*m^2*mu1^2*pv + b*h^3*kl*m*mu1^2*mu2^2*vsq - c*h^3*kl*m*mu1^2*mu2^2*vsq - d*h^3*kd*m*mu1^2*mu2^2*vsq - 2*h^2*kd*l^2*m*mu1^2*pv*vsq - b*g*h*l^2*m^2*mu1^2*pv^2 + h^2*kd*l^2*m*mu1^2*pv^2*vsq + h^2*kd*l^2*m*mu2^2*pv^2*vsq + 2*b*h*kl*l^2*m*mu1^2*pv*vsq - 2*c*h*kl*l^2*m*mu1^2*pv*vsq - 2*d*h*kd*l^2*m*mu1^2*pv*vsq - b*h*kl*l^2*m*mu1^2*pv^2*vsq + c*h*kl*l^2*m*mu1^2*pv^2*vsq + d*h*kd*l^2*m*mu1^2*pv^2*vsq));
eq = eq - ((h^2*m*mu1^2 - l^2*m*pv^2)*(a^2*g^2*l^2*m^2*mu2^2 - a^2*kappa^2*l^2*m^2*vsq^2 + a^2*kl^2*l^2*mu2^2*vsq^2 + c^2*kl^2*l^2*mu2^2*vsq^2 + d^2*kd^2*l^2*mu2^2*vsq^2 + a^2*g^2*l^2*m^2*mu2^2*pv^2 + a^2*h^2*kappa^2*m^2*mu2^2*vsq^2 - a^2*kappa^2*l^2*m^2*pv^2*vsq^2 + a^2*kl^2*l^2*mu2^2*pv^2*vsq^2 + c^2*kl^2*l^2*mu2^2*pv^2*vsq^2 + d^2*kd^2*l^2*mu2^2*pv^2*vsq^2 + 2*a*c*kl^2*l^2*mu2^2*vsq^2 - 2*a^2*g^2*l^2*m^2*mu2^2*pv + 2*a^2*kappa^2*l^2*m^2*pv*vsq^2 - 2*a^2*kl^2*l^2*mu2^2*pv*vsq^2 - 2*c^2*kl^2*l^2*mu2^2*pv*vsq^2 - 2*d^2*kd^2*l^2*mu2^2*pv*vsq^2 + 2*a*c*kl^2*l^2*mu2^2*pv^2*vsq^2 + 2*a*d*kd*kl*l^2*mu2^2*vsq^2 + 2*c*d*kd*kl*l^2*mu2^2*vsq^2 + 2*a^2*g*kl*l^2*m*mu2^2*vsq - 4*a*c*kl^2*l^2*mu2^2*pv*vsq^2 + 2*a*d*kd*kl*l^2*mu2^2*pv^2*vsq^2 + 2*c*d*kd*kl*l^2*mu2^2*pv^2*vsq^2 + 2*a^2*g*kl*l^2*m*mu2^2*pv^2*vsq + 2*a*c*g*kl*l^2*m*mu2^2*vsq + 2*a*d*g*kd*l^2*m*mu2^2*vsq - 4*a*d*kd*kl*l^2*mu2^2*pv*vsq^2 - 4*c*d*kd*kl*l^2*mu2^2*pv*vsq^2 - 4*a^2*g*kl*l^2*m*mu2^2*pv*vsq + 2*a*c*g*kl*l^2*m*mu2^2*pv^2*vsq + 2*a*d*g*kd*l^2*m*mu2^2*pv^2*vsq - 4*a*c*g*kl*l^2*m*mu2^2*pv*vsq - 4*a*d*g*kd*l^2*m*mu2^2*pv*vsq)^(1/2));
eq = expand(eq^2);
```

```
% Isolate remainder square root and square both sides
eq = combine(eq, "IgnoreAnalyticConstraints", true);
eq = collect(eq, ((b^2*g^2*l^2*m^2*mu1^2*pv^2 + b^2*h^2*kappa^2*m^2*mu1^2*vsq^2 -
    b^2*kappa^2*l^2*m^2*pv^2*vsq^2 + b^2*kl^2*l^2*mu1^2*pv^2*vsq^2 + c^2*kl^2*l^2*
    mu1^2*pv^2*vsq^2 + d^2*kd^2*l^2*mu1^2*pv^2*vsq^2 - 2*b*c*kl^2*l^2*mu1^2*pv^2*
    vsq^2 - 2*b*d*kd*kl*l^2*mu1^2*pv^2*vsq^2 + 2*c*d*kd*kl*l^2*mu1^2*pv^2*vsq^2 +
    2*b^2*g*kl*l^2*m*mu1^2*pv^2*vsq - 2*b*c*g*kl*l^2*m*mu1^2*pv^2*vsq - 2*b*d*g*kd
    *l^2*m*mu1^2*pv^2*vsq)*(a^2*g^2*l^2*m^2*mu2^2 - a^2*kappa^2*l^2*m^2*vsq^2 + a
    ^2*kl^2*l^2*mu2^2*vsq^2 + c^2*kl^2*l^2*mu2^2*vsq^2 + d^2*kd^2*l^2*mu2^2*vsq^2
    + a^2*g^2*l^2*m^2*mu2^2*pv^2 + a^2*h^2*kappa^2*m^2*mu2^2*vsq^2 - a^2*kappa^2*l
    ^2*m^2*pv^2*vsq^2 + a^2*kl^2*l^2*mu2^2*pv^2*vsq^2 + c^2*kl^2*l^2*mu2^2*pv^2*
    vsq^2 + d^2*kd^2*l^2*mu2^2*pv^2*vsq^2 + 2*a*c*kl^2*l^2*mu2^2*vsq^2 - 2*a^2*g
    ^2*l^2*m^2*mu2^2*pv + 2*a^2*kappa^2*l^2*m^2*pv*vsq^2 - 2*a^2*kl^2*l^2*mu2^2*pv
    *vsq^2 - 2*c^2*kl^2*l^2*mu2^2*pv*vsq^2 - 2*d^2*kd^2*l^2*mu2^2*pv*vsq^2 + 2*a*c
    *kl^2*l^2*mu2^2*pv^2*vsq^2 + 2*a*d*kd*kl*l^2*mu2^2*vsq^2 + 2*c*d*kd*kl*l^2*mu2
    ^2*vsq^2 + 2*a^2*g*kl*l^2*m*mu2^2*vsq - 4*a*c*kl^2*l^2*mu2^2*pv*vsq^2 + 2*a*d*
    kd*kl*l^2*mu2^2*pv^2*vsq^2 + 2*c*d*kd*kl*l^2*mu2^2*pv^2*vsq^2 + 2*a^2*g*kl*l
    ^2*m*mu2^2*pv^2*vsq + 2*a*c*g*kl*l^2*m*mu2^2*vsq + 2*a*d*g*kd*l^2*m*mu2^2*vsq
    - 4*a*d*kd*kl*l^2*mu2^2*pv*vsq^2 - 4*c*d*kd*kl*l^2*mu2^2*pv*vsq^2 - 4*a^2*g*kl
    *l^2*m*mu2^2*pv*vsq + 2*a*c*g*kl*l^2*m*mu2^2*pv^2*vsq + 2*a*d*g*kd*l^2*m*mu2
    ^2*pv^2*vsq - 4*a*c*g*kl*l^2*m*mu2^2*pv*vsq - 4*a*d*g*kd*l^2*m*mu2^2*pv*vsq))
    ^(1/2));
eq = eq - ( + (a^2*g^2*h^4*l^2*m^4*mu1^4*mu2^2*pv^2 - 2*a^2*g^2*h^4*l^2*m^4*mu1^4*
    mu2^2*pv + a^2*g^2*h^4*l^2*m^4*mu1^4*mu2^2 - 2*a^2*g^2*h^2*l^4*m^4*mu1^2*mu2
    ^2*pv^4 + 4*a^2*g^2*h^2*l^4*m^4*mu1^2*mu2^2*pv^3 - 2*a^2*g^2*h^2*l^4*m^4*mu1
    ^2*mu2^2*pv^2 + a^2*g^2*l^6*m^4*mu2^2*pv^6 - 2*a^2*g^2*l^6*m^4*mu2^2*pv^5 + a
    ^2*g^2*l^6*m^4*mu2^2*pv^4 + 2*a^2*g*h^4*kl*l^2*m^3*mu1^4*mu2^2*pv^2*vsq - 4*a
    ^2*g*h^4*kl*l^2*m^3*mu1^4*mu2^2*pv*vsq + 2*a^2*g*h^4*kl*l^2*m^3*mu1^4*mu2^2*
    vsq - 4*a^2*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^4*vsq + 8*a^2*g*h^2*kl*l^4*m^3*mu1
    ^2*mu2^2*pv^3*vsq - 4*a^2*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^2*vsq + 2*a^2*g*kl*l
    ^6*m^3*mu2^2*pv^6*vsq - 4*a^2*g*kl*l^6*m^3*mu2^2*pv^5*vsq + 2*a^2*g*kl*l^6*m
    ^3*mu2^2*pv^4*vsq + a^2*h^6*kappa^2*m^4*mu1^4*mu2^2*vsq^2 - a^2*h^4*kappa^2*l
    ^2*m^4*mu1^4*pv^2*vsq^2 + 2*a^2*h^4*kappa^2*l^2*m^4*mu1^4*pv*vsq^2 - a^2*h^4*
    kappa^2*l^2*m^4*mu1^4*vsq^2 - 2*a^2*h^4*kappa^2*l^2*m^4*mu1^2*mu2^2*pv^2*vsq^2
     + a^2*h^4*kl^2*l^2*m^2*mu1^4*mu2^2*pv^2*vsq^2 - 2*a^2*h^4*kl^2*l^2*m^2*mu1^4*
    mu2^2*pv*vsq^2 + a^2*h^4*kl^2*l^2*m^2*mu1^4*mu2^2*vsq^2 + 2*a^2*h^2*kappa^2*l
    ^4*m^4*mu1^2*pv^4*vsq^2 - 4*a^2*h^2*kappa^2*l^4*m^4*mu1^2*pv^3*vsq^2 + 2*a^2*h
    ^2*kappa^2*l^4*m^4*mu1^2*pv^2*vsq^2 + a^2*h^2*kappa^2*l^4*m^4*mu2^2*pv^4*vsq^2
     - 2*a^2*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 + 4*a^2*h^2*kl^2*l^4*m^2*mu1
    ^2*mu2^2*pv^3*vsq^2 - 2*a^2*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^2*vsq^2 - a^2*
    kappa^2*l^6*m^4*pv^6*vsq^2 + 2*a^2*kappa^2*l^6*m^4*pv^5*vsq^2 - a^2*kappa^2*l
    ^6*m^4*pv^4*vsq^2 + a^2*kl^2*l^6*m^2*mu2^2*pv^6*vsq^2 - 2*a^2*kl^2*l^6*m^2*mu2
    ^2*pv^5*vsq^2 + a^2*kl^2*l^6*m^2*mu2^2*pv^4*vsq^2 + 2*a*c*g*h^4*kl*l^2*m^3*mu1
    ^4*mu2^2*pv^2*vsq - 4*a*c*g*h^4*kl*l^2*m^3*mu1^4*mu2^2*pv*vsq + 2*a*c*g*h^4*kl
    *l^2*m^3*mu1^4*mu2^2*vsq - 4*a*c*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^4*vsq + 8*a*c
    *g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^3*vsq - 4*a*c*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv
    ^2*vsq + 2*a*c*g*kl*l^6*m^3*mu2^2*pv^6*vsq - 4*a*c*g*kl*l^6*m^3*mu2^2*pv^5*vsq
     + 2*a*c*g*kl*l^6*m^3*mu2^2*pv^4*vsq + 2*a*c*h^4*kl^2*l^2*m^2*mu1^4*mu2^2*pv
    ^2*vsq^2 - 4*a*c*h^4*kl^2*l^2*m^2*mu1^4*mu2^2*pv*vsq^2 + 2*a*c*h^4*kl^2*l^2*m
    ^2*mu1^4*mu2^2*vsq^2 - 4*a*c*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 + 8*a*c*h
    ^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 - 4*a*c*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv
    ^2*vsq^2 + 2*a*c*kl^2*l^6*m^2*mu2^2*pv^6*vsq^2 - 4*a*c*kl^2*l^6*m^2*mu2^2*pv
    ^5*vsq^2 + 2*a*c*kl^2*l^6*m^2*mu2^2*pv^4*vsq^2 + 2*a*d*g*h^4*kd*l^2*m^3*mu1^4*
    mu2^2*pv^2*vsq - 4*a*d*g*h^4*kd*l^2*m^3*mu1^4*mu2^2*pv*vsq + 2*a*d*g*h^4*kd*l
    ^2*m^3*mu1^4*mu2^2*vsq - 4*a*d*g*h^2*kd*l^4*m^3*mu1^2*mu2^2*pv^4*vsq + 8*a*d*g
    *h^2*kd*l^4*m^3*mu1^2*mu2^2*pv^3*vsq - 4*a*d*g*h^2*kd*l^4*m^3*mu1^2*mu2^2*pv
    ^2*vsq + 2*a*d*g*kd*l^6*m^3*mu2^2*pv^6*vsq - 4*a*d*g*kd*l^6*m^3*mu2^2*pv^5*vsq
     + 2*a*d*g*kd*l^6*m^3*mu2^2*pv^4*vsq + 2*a*d*h^4*kd*kl*l^2*m^2*mu1^4*mu2^2*pv
    ^2*vsq^2 - 4*a*d*h^4*kd*kl*l^2*m^2*mu1^4*mu2^2*pv*vsq^2 + 2*a*d*h^4*kd*kl*l^2*
    m^2*mu1^4*mu2^2*vsq^2 - 4*a*d*h^2*kd*kl*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 + 8*a*d
    *h^2*kd*kl*l^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 - 4*a*d*h^2*kd*kl*l^4*m^2*mu1^2*mu2
```

```
^2*pv^2*vsq^2 + 2*a*d*kd*kl*l^6*m^2*mu2^2*pv^6*vsq^2 - 4*a*d*kd*kl*l^6*m^2*mu2
^2*pv^5*vsq^2 + 2*a*d*kd*kl*l^6*m^2*mu2^2*pv^4*vsq^2 + b^2*g^2*h^4*l^2*m^4*mu1
^2*mu2^4*pv^2 - 2*b^2*g^2*h^2*l^4*m^4*mu1^2*mu2^2*pv^4 + 4*b^2*g^2*h^2*l^4*m
^4*mu1^2*mu2^2*pv^3 - 2*b^2*g^2*h^2*l^4*m^4*mu1^2*mu2^2*pv^2 + b^2*g^2*l^6*m
^4*mu1^2*pv^6 - 4*b^2*g^2*l^6*m^4*mu1^2*pv^5 + 6*b^2*g^2*l^6*m^4*mu1^2*pv^4 -
4*b^2*g^2*l^6*m^4*mu1^2*pv^3 + b^2*g^2*l^6*m^4*mu1^2*pv^2 + 2*b^2*g*h^4*kl*l
^2*m^3*mu1^2*mu2^4*pv^2*vsq - 4*b^2*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^4*vsq + 8*
b^2*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^3*vsq - 4*b^2*g*h^2*kl*l^4*m^3*mu1^2*mu2
^2*pv^2*vsq + 2*b^2*g*kl*l^6*m^3*mu1^2*pv^6*vsq - 8*b^2*g*kl*l^6*m^3*mu1^2*pv
^5*vsq + 12*b^2*g*kl*l^6*m^3*mu1^2*pv^4*vsq - 8*b^2*g*kl*l^6*m^3*mu1^2*pv^3*
vsq + 2*b^2*g*kl*l^6*m^3*mu1^2*pv^2*vsq + b^2*h^6*kappa^2*m^4*mu1^2*mu2^4*vsq
^2 - 2*b^2*h^4*kappa^2*l^2*m^4*mu1^2*mu2^2*pv^2*vsq^2 + 4*b^2*h^4*kappa^2*l^2*
m^4*mu1^2*mu2^2*pv*vsq^2 - 2*b^2*h^4*kappa^2*l^2*m^4*mu1^2*mu2^2*vsq^2 - b^2*h
^4*kappa^2*l^2*m^4*mu2^4*pv^2*vsq^2 + b^2*h^4*kl^2*l^2*m^2*mu1^2*mu2^4*pv^2*
vsq^2 + b^2*h^2*kappa^2*l^4*m^4*mu1^2*pv^4*vsq^2 - 4*b^2*h^2*kappa^2*l^4*m^4*
mu1^2*pv^3*vsq^2 + 6*b^2*h^2*kappa^2*l^4*m^4*mu1^2*pv^2*vsq^2 - 4*b^2*h^2*
kappa^2*l^4*m^4*mu1^2*pv*vsq^2 + b^2*h^2*kappa^2*l^4*m^4*mu1^2*vsq^2 + 2*b^2*h
^2*kappa^2*l^4*m^4*mu2^2*pv^4*vsq^2 - 4*b^2*h^2*kappa^2*l^4*m^4*mu2^2*pv^3*vsq
^2 + 2*b^2*h^2*kappa^2*l^4*m^4*mu2^2*pv^2*vsq^2 - 2*b^2*h^2*kl^2*l^4*m^2*mu1
^2*mu2^2*pv^4*vsq^2 + 4*b^2*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 - 2*b^2*h
^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^2*vsq^2 - b^2*kappa^2*l^6*m^4*pv^6*vsq^2 + 4*b
^2*kappa^2*l^6*m^4*pv^5*vsq^2 - 6*b^2*kappa^2*l^6*m^4*pv^4*vsq^2 + 4*b^2*kappa
^2*l^6*m^4*pv^3*vsq^2 - b^2*kappa^2*l^6*m^4*pv^2*vsq^2 + b^2*kl^2*l^6*m^2*mu1
^2*pv^6*vsq^2 - 4*b^2*kl^2*l^6*m^2*mu1^2*pv^5*vsq^2 + 6*b^2*kl^2*l^6*m^2*mu1
^2*pv^4*vsq^2 - 4*b^2*kl^2*l^6*m^2*mu1^2*pv^3*vsq^2 + b^2*kl^2*l^6*m^2*mu1^2*
pv^2*vsq^2 - 2*b*c*g*h^4*kl*l^2*m^3*mu1^2*mu2^4*pv^2*vsq + 4*b*c*g*h^2*kl*l^4*
m^3*mu1^2*mu2^2*pv^4*vsq - 8*b*c*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^3*vsq + 4*b*c
*g*h^2*kl*l^4*m^3*mu1^2*mu2^2*pv^2*vsq - 2*b*c*g*kl*l^6*m^3*mu1^2*pv^6*vsq +
8*b*c*g*kl*l^6*m^3*mu1^2*pv^5*vsq - 12*b*c*g*kl*l^6*m^3*mu1^2*pv^4*vsq + 8*b*c
*g*kl*l^6*m^3*mu1^2*pv^3*vsq - 2*b*c*g*kl*l^6*m^3*mu1^2*pv^2*vsq - 2*b*c*h^4*
kl^2*l^2*m^2*mu1^2*mu2^4*pv^2*vsq^2 + 4*b*c*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^4*
vsq^2 - 8*b*c*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 + 4*b*c*h^2*kl^2*l^4*m
^2*mu1^2*mu2^2*pv^2*vsq^2 - 2*b*c*kl^2*l^6*m^2*mu1^2*pv^6*vsq^2 + 8*b*c*kl^2*l
^6*m^2*mu1^2*pv^5*vsq^2 - 12*b*c*kl^2*l^6*m^2*mu1^2*pv^4*vsq^2 + 8*b*c*kl^2*l
^6*m^2*mu1^2*pv^3*vsq^2 - 2*b*c*kl^2*l^6*m^2*mu1^2*pv^2*vsq^2 - 2*b*d*g*h^4*kd
*l^2*m^3*mu1^2*mu2^4*pv^2*vsq + 4*b*d*g*h^2*kd*l^4*m^3*mu1^2*mu2^2*pv^4*vsq -
8*b*d*g*h^2*kd*l^4*m^3*mu1^2*mu2^2*pv^3*vsq + 4*b*d*g*h^2*kd*l^4*m^3*mu1^2*mu2
^2*pv^2*vsq - 2*b*d*g*kd*l^6*m^3*mu1^2*pv^6*vsq + 8*b*d*g*kd*l^6*m^3*mu1^2*pv
^5*vsq - 12*b*d*g*kd*l^6*m^3*mu1^2*pv^4*vsq + 8*b*d*g*kd*l^6*m^3*mu1^2*pv^3*
vsq - 2*b*d*g*kd*l^6*m^3*mu1^2*pv^2*vsq - 2*b*d*h^4*kd*kl*l^2*m^2*mu1^2*mu2^4*
pv^2*vsq^2 + 4*b*d*h^2*kd*kl*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 - 8*b*d*h^2*kd*kl*
l^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 + 4*b*d*h^2*kd*kl*l^4*m^2*mu1^2*mu2^2*pv^2*vsq
^2 - 2*b*d*kd*kl*l^6*m^2*mu1^2*pv^6*vsq^2 + 8*b*d*kd*kl*l^6*m^2*mu1^2*pv^5*vsq
^2 - 12*b*d*kd*kl*l^6*m^2*mu1^2*pv^4*vsq^2 + 8*b*d*kd*kl*l^6*m^2*mu1^2*pv^3*
vsq^2 - 2*b*d*kd*kl*l^6*m^2*mu1^2*pv^2*vsq^2 + c^2*h^4*kl^2*l^2*m^2*mu1^4*mu2
^2*pv^2*vsq^2 - 2*c^2*h^4*kl^2*l^2*m^2*mu1^4*mu2^2*pv*vsq^2 + c^2*h^4*kl^2*l
^2*m^2*mu1^4*mu2^2*vsq^2 + c^2*h^4*kl^2*l^2*m^2*mu1^2*mu2^4*pv^2*vsq^2 - 4*c
^2*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 + 8*c^2*h^2*kl^2*l^4*m^2*mu1^2*mu2
^2*pv^3*vsq^2 - 4*c^2*h^2*kl^2*l^4*m^2*mu1^2*mu2^2*pv^2*vsq^2 + c^2*kl^2*l^6*m
^2*mu1^2*pv^6*vsq^2 - 4*c^2*kl^2*l^6*m^2*mu1^2*pv^5*vsq^2 + 6*c^2*kl^2*l^6*m
^2*mu1^2*pv^4*vsq^2 - 4*c^2*kl^2*l^6*m^2*mu1^2*pv^3*vsq^2 + c^2*kl^2*l^6*m^2*
mu1^2*pv^2*vsq^2 + c^2*kl^2*l^6*m^2*mu2^2*pv^6*vsq^2 - 2*c^2*kl^2*l^6*m^2*mu2
^2*pv^5*vsq^2 + c^2*kl^2*l^6*m^2*mu2^2*pv^4*vsq^2 + 2*c*d*h^4*kd*kl*l^2*m^2*
mu1^4*mu2^2*pv^2*vsq^2 - 4*c*d*h^4*kd*kl*l^2*m^2*mu1^4*mu2^2*pv*vsq^2 + 2*c*d*
h^4*kd*kl*l^2*m^2*mu1^4*mu2^2*vsq^2 + 2*c*d*h^4*kd*kl*l^2*m^2*mu1^2*mu2^4*pv
^2*vsq^2 - 8*c*d*h^2*kd*kl*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 + 16*c*d*h^2*kd*kl*l
^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 - 8*c*d*h^2*kd*kl*l^4*m^2*mu1^2*mu2^2*pv^2*vsq^2
+ 2*c*d*kd*kl*l^6*m^2*mu1^2*pv^6*vsq^2 - 8*c*d*kd*kl*l^6*m^2*mu1^2*pv^5*vsq^2
+ 12*c*d*kd*kl*l^6*m^2*mu1^2*pv^4*vsq^2 - 8*c*d*kd*kl*l^6*m^2*mu1^2*pv^3*vsq
^2 + 2*c*d*kd*kl*l^6*m^2*mu1^2*pv^2*vsq^2 + 2*c*d*kd*kl*l^6*m^2*mu2^2*pv^6*vsq
^2 - 4*c*d*kd*kl*l^6*m^2*mu2^2*pv^5*vsq^2 + 2*c*d*kd*kl*l^6*m^2*mu2^2*pv^4*vsq
^2 + d^2*h^4*kd^2*l^2*m^2*mu1^4*mu2^2*pv^2*vsq^2 - 2*d^2*h^4*kd^2*l^2*m^2*mu1
```

```
    ^4*mu2^2*pv*vsq^2 + d^2*h^4*kd^2*l^2*m^2*mu1^4*mu2^2*vsq^2 + d^2*h^4*kd^2*l^2*
    m^2*mu1^2*mu2^4*pv^2*vsq^2 - 4*d^2*h^2*kd^2*l^4*m^2*mu1^2*mu2^2*pv^4*vsq^2 +
    8*d^2*h^2*kd^2*l^4*m^2*mu1^2*mu2^2*pv^3*vsq^2 - 4*d^2*h^2*kd^2*l^4*m^2*mu1^2*
    mu2^2*pv^2*vsq^2 + d^2*kd^2*l^6*m^2*mu1^2*pv^6*vsq^2 - 4*d^2*kd^2*l^6*m^2*mu1
    ^2*pv^5*vsq^2 + 6*d^2*kd^2*l^6*m^2*mu1^2*pv^4*vsq^2 - 4*d^2*kd^2*l^6*m^2*mu1
    ^2*pv^3*vsq^2 + d^2*kd^2*l^6*m^2*mu1^2*pv^2*vsq^2 + d^2*kd^2*l^6*m^2*mu2^2*pv
    ^6*vsq^2 - 2*d^2*kd^2*l^6*m^2*mu2^2*pv^5*vsq^2 + d^2*kd^2*l^6*m^2*mu2^2*pv^4*
    vsq^2));
eq = expand(eq^2);

% Find polynomial coefficients for vsq
eq = eq - rhs(eq);
eq = collect(eq, vsq);

% The polynomial coefficients can be found by exporting eq to a text file.
% The coefficients can be found manually or by writing a text interpreter.
% The equation size is too large for the use of MATLAB function coeffs.
fileName     = "FL2DMinimizer5.txt";     % File name
fileID       = fopen(fileName, "w");     % Open document
fprintf(fileID, "%s", string(eq));       % Rewrite document
fclose(fileID);                          % Close document

% The obtained coefficients will determine the value of V^2, so after
% evaluation with a root solver one still needs to take the square root.
% This will result in four solutions. The correct solution will always be
% the largest value with no imaginary part. The value of ax can be
% found by substituting the value of V in axG1(2) or axG2(1).

%%% Minimizer 6: g2 & g3 active and ax > 0
% Equate solutions and multiply out denominators
eq       = subs(axG2(1) == axG3(1), vsq, v^2);
denG3    = m*v;
eq       = expand(eq * denG2 * denG3);

% Isolate square root and square both sides
eq = eq - (+ l^2*v^3*kd*m - 2*l^2*v^3*kd*m*pv + l^2*v^3*kd*m*pv^2 - v^3*h^2*kd*m*
    mu2^2 - v*a*g*h*m^2*mu2^2 - v^3*a*h*kl*m*mu2^2 - v^3*c*h*kl*m*mu2^2 - v^3*d*h*
    kd*m*mu2^2);
eq = expand(eq^2);

% Find polynomial coefficients for V
eq = eq - rhs(eq);
eq = collect(eq, v);

% The polynomial coefficients can be found by exporting eq to a text file.
% The coefficients can be found manually or by writing a text interpreter.
% The equation size is too large for the use of MATLAB function coeffs.
fileName     = "FL2DMinimizer6.txt";     % File name
fileID       = fopen(fileName, "w");     % Open document
fprintf(fileID, "%s", string(eq));       % Rewrite document
fclose(fileID);                          % Close document

% The obtained coefficients will determine the value of V after
% evaluation with a root solver. This will result in six solutions. The
% correct solution will always be the largest value with no imaginary part.
% The value of ax can be found by substituting the value of V in
% axG2(1) or axG3(1).

%% Reverse loop - optimization problem (5.8)
%%% Minimizer 1: h1 & g1 active and ax < -kD/m*V^2
axH1 = solve(h1 == 0, ax);                    % Two solutions, only one is correct
eq = subs(axH1(1), v^2, vsq) == axG1(2);
```

```
% Multiply out denominators
[~, denH1] = numden(axH1(1));
eq = expand(eq * denG1 * denH1);

% Isolate square root and square both sides
eq = eq - lhs(eq);
eq = eq - (- 2*ds*(l^2*b^2*kl^2*mu1^2*pv^2*vsq^2 - l^2*b^2*kappa^2*m^2*pv^2*vsq^2
    + l^2*c^2*kl^2*mu1^2*pv^2*vsq^2 + l^2*d^2*kd^2*mu1^2*pv^2*vsq^2 + b^2*h^2*
    kappa^2*m^2*mu1^2*vsq^2 + l^2*b^2*g^2*m^2*mu1^2*pv^2 - 2*l^2*b*c*kl^2*mu1^2*pv
    ^2*vsq^2 - 2*l^2*b*d*kd*kl*mu1^2*pv^2*vsq^2 + 2*l^2*c*d*kd*kl*mu1^2*pv^2*vsq^2
     + 2*l^2*b^2*g*kl*m*mu1^2*pv^2*vsq - 2*l^2*b*c*g*kl*m*mu1^2*pv^2*vsq - 2*l^2*b
    *d*g*kd*m*mu1^2*pv^2*vsq)^(1/2));
eq = expand(eq^2);

% Bring all terms to one side and collect polynomial coefficients
eq = eq - rhs(eq);
eq = collect(eq, vsq);

% Solve equation
vS      = sqrt(solve(eq, vsq));     % Two solutions, only one is correct
v31     = vS(2);
ax31    = subs(axH1(1), v, v31);

%%% Minimizer 2: h1 & g2 active and ax < -kD/m*V^2
eq = subs(axH1(1), v^2, vsq) == axG2(1);

% Multiply out denominators
eq = expand(eq * denG2 * denH1);

% Isolate square root and square both sides
eq = eq - lhs(eq);
eq = eq - (2*ds*(l^2*a^2*g^2*m^2*mu2^2 - l^2*a^2*kappa^2*m^2*vsq^2 + l^2*a^2*kl^2*
    mu2^2*vsq^2 + l^2*c^2*kl^2*mu2^2*vsq^2 + l^2*d^2*kd^2*mu2^2*vsq^2 - l^2*a^2*
    kappa^2*m^2*pv^2*vsq^2 + l^2*a^2*kl^2*mu2^2*pv^2*vsq^2 + l^2*c^2*kl^2*mu2^2*pv
    ^2*vsq^2 + l^2*d^2*kd^2*mu2^2*pv^2*vsq^2 + a^2*h^2*kappa^2*m^2*mu2^2*vsq^2 +
    2*l^2*a*c*kl^2*mu2^2*vsq^2 - 2*l^2*a^2*g^2*m^2*mu2^2*pv + 2*l^2*a^2*kappa^2*m
    ^2*pv*vsq^2 - 2*l^2*a^2*kl^2*mu2^2*pv*vsq^2 - 2*l^2*c^2*kl^2*mu2^2*pv*vsq^2 -
    2*l^2*d^2*kd^2*mu2^2*pv*vsq^2 + l^2*a^2*g^2*m^2*mu2^2*pv^2 + 2*l^2*a*c*kl^2*
    mu2^2*pv^2*vsq^2 + 2*l^2*a*d*kd*kl*mu2^2*vsq^2 + 2*l^2*c*d*kd*kl*mu2^2*vsq^2 +
     2*l^2*a^2*g*kl*m*mu2^2*vsq - 4*l^2*a*c*kl^2*mu2^2*pv*vsq^2 + 2*l^2*a*d*kd*kl*
    mu2^2*pv^2*vsq^2 + 2*l^2*c*d*kd*kl*mu2^2*pv^2*vsq^2 + 2*l^2*a^2*g*kl*m*mu2^2*
    pv^2*vsq + 2*l^2*a*c*g*kl*m*mu2^2*vsq + 2*l^2*a*d*g*kd*m*mu2^2*vsq - 4*l^2*a*d
    *kd*kl*mu2^2*pv*vsq^2 - 4*l^2*c*d*kd*kl*mu2^2*pv*vsq^2 - 4*l^2*a^2*g*kl*m*mu2
    ^2*pv*vsq + 2*l^2*a*c*g*kl*m*mu2^2*pv^2*vsq + 2*l^2*a*d*g*kd*m*mu2^2*pv^2*vsq
    - 4*l^2*a*c*g*kl*m*mu2^2*pv*vsq - 4*l^2*a*d*g*kd*m*mu2^2*pv*vsq)^(1/2));
eq = expand(eq^2);

% Bring all terms to one side and collect polynomial coefficients
eq = eq - rhs(eq);
eq = collect(eq, vsq);

% Solve equation
vS      = sqrt(solve(eq, vsq));     % Two solutions, only one is correct
v32     = vS(2);
ax32    = subs(axH1(1), v, v32);

%% Reverse loop - optimization problem (5.9)
%%% Minimizer 1: h1 & g1 active and ax >= -kD/m*V^2
eq = subs(axH1(1), v^2, vsq) == axG1(1);

% Multiply out denominators
```

```
eq = expand(eq * denG1 * denH1);

% Isolate square root and square both sides
eq = eq - lhs(eq);
eq = eq - (2*ds*(l^2*b^2*kl^2*mu1^2*pv^2*vsq^2 - l^2*b^2*kappa^2*m^2*pv^2*vsq^2 +
    l^2*c^2*kl^2*mu1^2*pv^2*vsq^2 + l^2*d^2*kd^2*mu1^2*pv^2*vsq^2 + b^2*h^2*kappa
    ^2*m^2*mu1^2*vsq^2 + l^2*b^2*g^2*m^2*mu1^2*pv^2 - 2*l^2*b*c*kl^2*mu1^2*pv^2*
    vsq^2 - 2*l^2*b*d*kd*kl*mu1^2*pv^2*vsq^2 + 2*l^2*c*d*kd*kl*mu1^2*pv^2*vsq^2 +
    2*l^2*b^2*g*kl*m*mu1^2*pv^2*vsq - 2*l^2*b*c*g*kl*m*mu1^2*pv^2*vsq - 2*l^2*b*d*
    g*kd*m*mu1^2*pv^2*vsq)^(1/2));
eq = expand(eq^2);

% Bring all terms to one side and collect polynomial coefficients
eq = eq - rhs(eq);
eq = collect(eq, vsq);

% Solve equation
vS      = sqrt(solve(eq, vsq));     % Two solutions, only one is correct
v41     = vS(1);
ax41    = subs(axH1(1), v, v41);

%%% Minimizer 2: h1 & g2 active and ax >= axG2max
eq = subs(axH1(1), v^2, vsq) == axG2(2);

% Multiply out denominators
eq = expand(eq * denG2 * denH1);

% Isolate square root and square both sides
eq = eq - lhs(eq);
eq = eq - (- 2*ds*(l^2*a^2*g^2*m^2*mu2^2 - l^2*a^2*kappa^2*m^2*vsq^2 + l^2*a^2*kl
    ^2*mu2^2*vsq^2 + l^2*c^2*kl^2*mu2^2*vsq^2 + l^2*d^2*kd^2*mu2^2*vsq^2 - l^2*a
    ^2*kappa^2*m^2*pv^2*vsq^2 + l^2*a^2*kl^2*mu2^2*pv^2*vsq^2 + l^2*c^2*kl^2*mu2
    ^2*pv^2*vsq^2 + l^2*d^2*kd^2*mu2^2*pv^2*vsq^2 + a^2*h^2*kappa^2*m^2*mu2^2*vsq
    ^2 + 2*l^2*a*c*kl^2*mu2^2*vsq^2 - 2*l^2*a^2*g^2*m^2*mu2^2*pv + 2*l^2*a^2*kappa
    ^2*m^2*pv*vsq^2 - 2*l^2*a^2*kl^2*mu2^2*pv*vsq^2 - 2*l^2*c^2*kl^2*mu2^2*pv*vsq
    ^2 - 2*l^2*d^2*kd^2*mu2^2*pv*vsq^2 + l^2*a^2*g^2*m^2*mu2^2*pv^2 + 2*l^2*a*c*kl
    ^2*mu2^2*pv^2*vsq^2 + 2*l^2*a*d*kd*kl*mu2^2*vsq^2 + 2*l^2*c*d*kd*kl*mu2^2*vsq
    ^2 + 2*l^2*a^2*g*kl*m*mu2^2*vsq - 4*l^2*a*c*kl^2*mu2^2*pv*vsq^2 + 2*l^2*a*d*kd
    *kl*mu2^2*pv^2*vsq^2 + 2*l^2*c*d*kd*kl*mu2^2*pv^2*vsq^2 + 2*l^2*a^2*g*kl*m*mu2
    ^2*pv^2*vsq + 2*l^2*a*c*g*kl*m*mu2^2*vsq + 2*l^2*a*d*g*kd*m*mu2^2*vsq - 4*l^2*
    a*d*kd*kl*mu2^2*pv*vsq^2 - 4*l^2*c*d*kd*kl*mu2^2*pv*vsq^2 - 4*l^2*a^2*g*kl*m*
    mu2^2*pv*vsq + 2*l^2*a*c*g*kl*m*mu2^2*pv^2*vsq + 2*l^2*a*d*g*kd*m*mu2^2*pv^2*
    vsq - 4*l^2*a*c*g*kl*m*mu2^2*pv*vsq - 4*l^2*a*d*g*kd*m*mu2^2*pv*vsq)^(1/2));
eq = expand(eq^2);

% Bring all terms to one side and collect polynomial coefficients
eq = eq - rhs(eq);
eq = collect(eq, vsq);

% Solve equation
vS      = sqrt(solve(eq, vsq));     % Two solutions, only one is correct
v42     = vS(1);
ax42    = subs(axH1(1), v, v42);

%%% Minimizer 3: h1 & g2 active and -kD/m*V^2 <= ax < axG2max
eq = subs(axH1(1), v^2, vsq) == axG2(1);

% Multiply out denominators
eq = expand(eq * denG2 * denH1);

% Isolate square root and square both sides
eq = eq - lhs(eq);
```

```
eq = eq - (2*ds*(l^2*a^2*g^2*m^2*mu2^2 - l^2*a^2*kappa^2*m^2*vsq^2 + l^2*a^2*kl^2*
    mu2^2*vsq^2 + l^2*c^2*kl^2*mu2^2*vsq^2 + l^2*d^2*kd^2*mu2^2*vsq^2 - l^2*a^2*
    kappa^2*m^2*pv^2*vsq^2 + l^2*a^2*kl^2*mu2^2*pv^2*vsq^2 + l^2*c^2*kl^2*mu2^2*pv
    ^2*vsq^2 + l^2*d^2*kd^2*mu2^2*pv^2*vsq^2 + a^2*h^2*kappa^2*m^2*mu2^2*vsq^2 +
    2*l^2*a*c*kl^2*mu2^2*vsq^2 - 2*l^2*a^2*g^2*m^2*mu2^2*pv + 2*l^2*a^2*kappa^2*m
    ^2*pv*vsq^2 - 2*l^2*a^2*kl^2*mu2^2*pv*vsq^2 - 2*l^2*c^2*kl^2*mu2^2*pv*vsq^2 -
    2*l^2*d^2*kd^2*mu2^2*pv*vsq^2 + l^2*a^2*g^2*m^2*mu2^2*pv^2 + 2*l^2*a*c*kl^2*
    mu2^2*pv^2*vsq^2 + 2*l^2*a*d*kd*kl*mu2^2*vsq^2 + 2*l^2*c*d*kd*kl*mu2^2*vsq^2 +
     2*l^2*a^2*g*kl*m*mu2^2*vsq - 4*l^2*a*c*kl^2*mu2^2*pv*vsq^2 + 2*l^2*a*d*kd*kl*
    mu2^2*pv^2*vsq^2 + 2*l^2*c*d*kd*kl*mu2^2*pv^2*vsq^2 + 2*l^2*a^2*g*kl*m*mu2^2*
    pv^2*vsq + 2*l^2*a*c*g*kl*m*mu2^2*vsq + 2*l^2*a*d*g*kd*m*mu2^2*vsq - 4*l^2*a*d
    *kd*kl*mu2^2*pv*vsq^2 - 4*l^2*c*d*kd*kl*mu2^2*pv*vsq^2 - 4*l^2*a^2*g*kl*m*mu2
    ^2*pv*vsq + 2*l^2*a*c*g*kl*m*mu2^2*pv^2*vsq + 2*l^2*a*d*g*kd*m*mu2^2*pv^2*vsq
    - 4*l^2*a*c*g*kl*m*mu2^2*pv*vsq - 4*l^2*a*d*g*kd*m*mu2^2*pv*vsq)^(1/2));
eq = expand(eq^2);

% Bring all terms to one side and collect polynomial coefficients
eq = eq - rhs(eq);
eq = collect(eq, vsq);

% Solve equation
vS      = sqrt(solve(eq, vsq));     % Two solutions, only one is correct
v43     = vS(2);
ax43    = subs(axH1(1), v, v43);

%%% Minimizer 4: h1 & g3 active
eq = subs(axG3(1), vsq, v^2) == axH1(1);

% Multiply out denominators
eq = expand(eq * denG3 * denH1);

% Bring all terms to one side and collect polynomial coefficients
eq = eq - rhs(eq);
eq = collect(eq, v);

% Solve equation
vS      = solve(eq, v, "MaxDegree", 3); % Three solutions, only one is correct
v44     = vS(1);
ax44    = subs(axH1(1), v, v44);

%% Numerical analysis
%%% Define constant parameters
global M G L B A H KD KL C D P FRR PD PB MU1 MU2 VI DS
M   = 1030;          % Vehicle mass [kg]
G   = 9.81;          % Gravitational acceleration [m/s^2]
WD  = 0.55;          % Weight distribution [-]
L   = 3.005;         % Wheelbase [m]
B   = WD*L;          % Longitudinal distance CoG-rear axle [m]
A   = L - B;         % Longitudinal distance CoG-front axle [m]
H   = 0.390;         % Vertical height CoG [m]
KD  = 0.8983;        % Drag coefficient [N/(m^2/s^2)]
KL  = 2.695;         % Lift coefficient [N/(m^2/s^2)]
DD  = 0.48;          % Downforce distribution [-]
C   = (WD - DD)*L;   % Longitudinal distance CoG-CoP [m]
D   = 0.05;          % Vertical distance CoG-CoP [m]
P   = 60*1e3;        % Motor power [W]
FRR = 0.02;          % Rolling resistance coefficient [-]
PD  = 0.00;          % Torque distribution [-]
PB  = 0.625;         % Brake distribution [-]
MU1 = 1.68;          % Friction coefficient front tire [-]
MU2 = 1.5;           % Friction coefficient rear tire [-]
VI  = 31.1111;       % Velocity at iteration i [m/s]
```

```
VFL = 15;            % Velocity for (5.6) [m/s]
DS  = 0.1;           % Track step-size [m]

%%% Define sweep parameters
N       = 1e6;                          % Number of data points
KAPPA   = 0.02;                         % Curvature [1/m]
V       = linspace(0.1, 300, N)'/3.6 ;  % Velocity [m/s]

%%% Calculate constraints
[G1, G2, G3, G4, H1, AY] = calculateConstraints(V, KAPPA);

%%% Calculate minimizers
% Define replacement arrays
oldArray = [m, g, l, b, a, h, kd, kl, c, d, p, frr, mu1, mu2, vi, ds, kappa];
newArray = [M, G, L, B, A, H, KD, KL, C, D, P, FRR, MU1, MU2, VI, DS, KAPPA];

% Forward loop - optimization problem (5.6)
AX11 = double(subs(ax11, [oldArray, pv, vsq], [newArray, PD, VFL^2]));
AX12 = double(subs(ax12, [oldArray, pv, vsq], [newArray, PD, VFL^2]));
AX13 = double(subs(ax13, [oldArray, v, vsq], [newArray, VFL, VFL^2]));
[V1, AX1] = physicalData(VFL*ones(1, 3), [AX11, AX12, AX13], "positive");

% Forward loop - optimization problem (5.7)
AX21 = 0;
V21  = double(subs(v21, [oldArray, pv], [newArray, PD]));
AX22 = double(subs(ax22, [oldArray, pv], [newArray, PD]));
V22  = double(subs(v22, [oldArray, pv], [newArray, PD]));
AX23 = 0;
V23  = double(subs(v23, [oldArray, pv], [newArray, PD]));
AX24 = 0;
V24  = double(subs(v24, oldArray, newArray));
AX25 = 0.7121;  % Manually determined
V25  = 31.2816; % Manually determined
AX26 = 0.7611;  % Manually determined
V26  = 31.2835; % Manually determined
[V2, AX2] = physicalData([V21, V22, V23, V24, V25, V26],...
    [AX21, AX22, AX23, AX24, AX25, AX26], "positive");

% Reverse loop - optimization problem (5.8)
AX31 = double(subs(ax31, [oldArray, pv], [newArray, PB]));
V31  = double(subs(v31, [oldArray, pv], [newArray, PB]));
AX32 = double(subs(ax32, [oldArray, pv], [newArray, PB]));
V32  = double(subs(v32, [oldArray, pv], [newArray, PB]));
[V3, AX3] = physicalData([V31, V32], [AX31, AX32], "braking");

% Reverse loop - optimization problem (5.9)
AX41 = double(subs(ax41, [oldArray, pv], [newArray, PD]));
V41  = double(subs(v41, [oldArray, pv], [newArray, PD]));
AX42 = double(subs(ax42, [oldArray, pv], [newArray, PD]));
V42  = double(subs(v42, [oldArray, pv], [newArray, PD]));
AX43 = double(subs(ax43, [oldArray, pv], [newArray, PD]));
V43  = double(subs(v43, [oldArray, pv], [newArray, PD]));
AX44 = double(subs(ax44, [oldArray, pv], [newArray, PD]));
V44  = double(subs(v44, [oldArray, pv], [newArray, PD]));
[V4, AX4] = physicalData([V41, V42, V43, V44],...
    [AX41, AX42, AX43, AX44], "traction");

%% Visualization
% Color array
colors = [
    128, 0, 128     % Purple
    139, 69, 19     % Brown
```

```
    236, 145, 52     % Orange
    256, 256, 0      % Yellow
    ]/256;

% Create figure
fig = figure("Name", "Adapted GG diagram");
t = tiledlayout(1, 1, "TileSpacing", "compact", "Padding", "compact");

nexttile
hold on
xline(0, "LineStyle", ":", "LineWidth", 0.5)
yline(0, "LineStyle", ":", "LineWidth", 0.5)

% Constraints
pLine        = plot(AY/G, G1/G, "Color", "r", "LineWidth", 1.5);
pLine(2)     = plot(AY/G, G2/G, "Color", "g", "LineWidth", 1.5);
pLine(3)     = plot(AY(length(G3) + (1:length(G3)))/G, G3/G, ...
    "Color", "m", "LineWidth", 1.5);
pLine(4)     = plot(AY(length(G4) + (1:length(G4)))/G, G4/G, ...
    "Color", "b", "LineWidth", 1.5);
pLine(5)     = plot(AY(length(H1) + (1:length(H1)))/G, H1/G, ...
    "Color", "c", "LineWidth", 1.5);

%%% Minimizers
% (5.6)
xline(VFL^2*KAPPA/G, "LineStyle", ":", "LineWidth", 0.5,...
    "Color", colors(:, 1))
pLine(6)     = plot(V1.^2*KAPPA/G, AX1/G, "LineStyle", "none",...
    "Marker", "o", "Color", colors(1, :), "MarkerFaceColor", colors(1, :));

% (5.7)
pLine(7)     = plot(V2.^2*KAPPA/G, AX2/G, "LineStyle", "none",...
    "Marker", "o", "Color", colors(2, :), "MarkerFaceColor", colors(2, :));

% (5.8)
pLine(8)     = plot(V3.^2*KAPPA/G, AX3/G, "LineStyle", "none",...
    "Marker", "o", "Color", colors(3, :), "MarkerFaceColor", colors(3, :));

% (5.9)
pLine(9)     = plot(V4.^2*KAPPA/G, AX4/G, "LineStyle", "none",...
    "Marker", "o", "Color", colors(4, :), "MarkerFaceColor", colors(4, :));

title("Adapted GG diagram")
axis([-8, 8, -5, 5], "square")
xlabel("a_y [G]")
ylabel("a_x [G]")
grid minor
lgd = legend(pLine, ["Front tire", "Rear tire", "Powertrain", ...
    "Drag", "V-ax consistency", "Minimizers (5.6)",...
    "Minimizers (5.7)", "Minimizers (5.8)", "Minimizers (5.9)"], ...
    "Orientation", "horizontal", "NumColumns", 3, "AutoUpdate", "off");
lgd.Layout.Tile = "south";

set(findall(fig, "-property", "FontSize"), "FontSize", 14)
set(findall(fig, "-property", "FontWeight"), "FontWeight", "bold")
exportgraphics(t, "AdaptedGGDiagram.png")   % Save image

%% Functions
% Calculate constraints
function [G1, G2, G3, G4, H1, AY] = calculateConstraints(V, KAPPA)
global M G L B A H KD KL C D P FRR PD PB MU1 MU2 VI DS

```

```
%%% Rewrite Input
AY       = V.^2.*KAPPA;       % Lateral acceleration

%%% Front tire
axG1(:,1) = ((L^2*V.^4*B^2*KL^2*MU1^2*PD^2 - 2*L^2*V.^4*B*C*KL^2*MU1^2*PD^2 - 2*L
    ^2*V.^4*B*D*KD*KL*MU1^2*PD^2 + L^2*V.^4*C^2*KL^2*MU1^2*PD^2 + 2*L^2*V.^4*C*D*
    KD*KL*MU1^2*PD^2 + L^2*V.^4*D^2*KD^2*MU1^2*PD^2 + 2*L^2*V.^2*B^2*G*KL*M*MU1^2*
    PD^2 - 2*L^2*V.^2*B*C*G*KL*M*MU1^2*PD^2 - 2*L^2*V.^2*B*D*G*KD*M*MU1^2*PD^2 - L
    ^2*AY.^2*B^2*M^2*PD^2 + L^2*B^2*G^2*M^2*MU1^2*PD^2 + AY.^2*B^2*H^2*M^2*MU1^2)
    .^(1/2) + V.^2*H^2*KD*MU1^2 - L^2*V.^2*KD*PD^2 - V.^2*B*H*KL*MU1^2 + V.^2*C*H*
    KL*MU1^2 + V.^2*D*H*KD*MU1^2 - B*G*H*M*MU1^2)./(M*(L^2*PD^2 - H^2*MU1^2));
axG1(:,2) = ((L^2*V.^4*B^2*KL^2*MU1^2*PB^2 - 2*L^2*V.^4*B*C*KL^2*MU1^2*PB^2 - 2*L
    ^2*V.^4*B*D*KD*KL*MU1^2*PB^2 + L^2*V.^4*C^2*KL^2*MU1^2*PB^2 + 2*L^2*V.^4*C*D*
    KD*KL*MU1^2*PB^2 + L^2*V.^4*D^2*KD^2*MU1^2*PB^2 + 2*L^2*V.^2*B^2*G*KL*M*MU1^2*
    PB^2 - 2*L^2*V.^2*B*C*G*KL*M*MU1^2*PB^2 - 2*L^2*V.^2*B*D*G*KD*M*MU1^2*PB^2 - L
    ^2*AY.^2*B^2*M^2*PB^2 + L^2*B^2*G^2*M^2*MU1^2*PB^2 + AY.^2*B^2*H^2*M^2*MU1^2)
    .^(1/2) + V.^2*H^2*KD*MU1^2 - L^2*V.^2*KD*PB^2 - V.^2*B*H*KL*MU1^2 + V.^2*C*H*
    KL*MU1^2 + V.^2*D*H*KD*MU1^2 - B*G*H*M*MU1^2)./(M*(L^2*PB^2 - H^2*MU1^2));
axG1(:,3) = -((L^2*V.^4*B^2*KL^2*MU1^2*PB^2 - 2*L^2*V.^4*B*C*KL^2*MU1^2*PB^2 - 2*L
    ^2*V.^4*B*D*KD*KL*MU1^2*PB^2 + L^2*V.^4*C^2*KL^2*MU1^2*PB^2 + 2*L^2*V.^4*C*D*
    KD*KL*MU1^2*PB^2 + L^2*V.^4*D^2*KD^2*MU1^2*PB^2 + 2*L^2*V.^2*B^2*G*KL*M*MU1^2*
    PB^2 - 2*L^2*V.^2*B*C*G*KL*M*MU1^2*PB^2 - 2*L^2*V.^2*B*D*G*KD*M*MU1^2*PB^2 - L
    ^2*AY.^2*B^2*M^2*PB^2 + L^2*B^2*G^2*M^2*MU1^2*PB^2 + AY.^2*B^2*H^2*M^2*MU1^2)
    .^(1/2) - V.^2*H^2*KD*MU1^2 + L^2*V.^2*KD*PB^2 + V.^2*B*H*KL*MU1^2 - V.^2*C*H*
    KL*MU1^2 - V.^2*D*H*KD*MU1^2 + B*G*H*M*MU1^2)./(M*(L^2*PB^2 - H^2*MU1^2));
wrongAcc = find(axG1(:,1)<-KD/M*V.^2 | abs(imag(axG1(:,1))) > 1e-6);          %
    Braking under traction
wrongDec = find(axG1(:,2:3)>-KD/M*V.^2 | abs(imag(axG1(:,2:3))) > 1e-6) +... %
    Traction under braking
    numel(axG1(:,1));
axG1([wrongAcc ; wrongDec]) = NaN;
axG1 = real(axG1);
G1 = [axG1(:,1) ; NaN ; axG1(:,2) ; NaN ; axG1(:,3)];
G1 = [G1 ; NaN ; G1];

%%% Rear tire
axG2(:,1) = -(V.^2*H^2*KD*MU2^2 - L^2*V.^2*KD - (L^2*V.^4*A^2*KL^2*MU2^2*PD^2 - 2*
    L^2*V.^4*A^2*KL^2*MU2^2*PD + L^2*V.^4*A^2*KL^2*MU2^2 + 2*L^2*V.^4*A*C*KL^2*MU2
    ^2*PD^2 - 4*L^2*V.^4*A*C*KL^2*MU2^2*PD + 2*L^2*V.^4*A*C*KL^2*MU2^2 + 2*L^2*V
    .^4*A*D*KD*KL*MU2^2*PD^2 - 4*L^2*V.^4*A*D*KD*KL*MU2^2*PD + 2*L^2*V.^4*A*D*KD*
    KL*MU2^2 + L^2*V.^4*C^2*KL^2*MU2^2*PD^2 - 2*L^2*V.^4*C^2*KL^2*MU2^2*PD + L^2*V
    .^4*C^2*KL^2*MU2^2 + 2*L^2*V.^4*C*D*KD*KL*MU2^2*PD^2 - 4*L^2*V.^4*C*D*KD*KL*
    MU2^2*PD + 2*L^2*V.^4*C*D*KD*KL*MU2^2 + L^2*V.^4*D^2*KD^2*MU2^2*PD^2 - 2*L^2*V
    .^4*D^2*KD^2*MU2^2*PD + L^2*V.^4*D^2*KD^2*MU2^2 + 2*L^2*V.^2*A^2*G*KL*M*MU2^2*
    PD^2 - 4*L^2*V.^2*A^2*G*KL*M*MU2^2*PD + 2*L^2*V.^2*A^2*G*KL*M*MU2^2 + 2*L^2*V
    .^2*A*C*G*KL*M*MU2^2*PD^2 - 4*L^2*V.^2*A*C*G*KL*M*MU2^2*PD + 2*L^2*V.^2*A*C*G*
    KL*M*MU2^2 + 2*L^2*V.^2*A*D*G*KD*M*MU2^2*PD^2 - 4*L^2*V.^2*A*D*G*KD*M*MU2^2*PD
     + 2*L^2*V.^2*A*D*G*KD*M*MU2^2 - L^2*A^2*AY.^2*M^2*PD^2 + 2*L^2*A^2*AY.^2*M^2*
    PD - L^2*A^2*AY.^2*M^2 + L^2*A^2*G^2*M^2*MU2^2*PD^2 - 2*L^2*A^2*G^2*M^2*MU2^2*
    PD + L^2*A^2*G^2*M^2*MU2^2 + A^2*AY.^2*H^2*M^2*MU2^2).^(1/2) + 2*L^2*V.^2*KD*
    PD - L^2*V.^2*KD*PD^2 + V.^2*A*H*KL*MU2^2 + V.^2*C*H*KL*MU2^2 + V.^2*D*H*KD*
    MU2^2 + A*G*H*M*MU2^2)./(M*(- L^2*PD^2 + 2*L^2*PD - L^2 + H^2*MU2^2));
axG2(:,2) = -((L^2*V.^4*A^2*KL^2*MU2^2*PD^2 - 2*L^2*V.^4*A^2*KL^2*MU2^2*PD + L^2*V
    .^4*A^2*KL^2*MU2^2 + 2*L^2*V.^4*A*C*KL^2*MU2^2*PD^2 - 4*L^2*V.^4*A*C*KL^2*MU2
    ^2*PD + 2*L^2*V.^4*A*C*KL^2*MU2^2 + 2*L^2*V.^4*A*D*KD*KL*MU2^2*PD^2 - 4*L^2*V
    .^4*A*D*KD*KL*MU2^2*PD + 2*L^2*V.^4*A*D*KD*KL*MU2^2 + L^2*V.^4*C^2*KL^2*MU2^2*
    PD^2 - 2*L^2*V.^4*C^2*KL^2*MU2^2*PD + L^2*V.^4*C^2*KL^2*MU2^2 + 2*L^2*V.^4*C*D
    *KD*KL*MU2^2*PD^2 - 4*L^2*V.^4*C*D*KD*KL*MU2^2*PD + 2*L^2*V.^4*C*D*KD*KL*MU2^2
     + L^2*V.^4*D^2*KD^2*MU2^2*PD^2 - 2*L^2*V.^4*D^2*KD^2*MU2^2*PD + L^2*V.^4*D^2*
    KD^2*MU2^2 + 2*L^2*V.^2*A^2*G*KL*M*MU2^2*PD^2 - 4*L^2*V.^2*A^2*G*KL*M*MU2^2*PD
     + 2*L^2*V.^2*A^2*G*KL*M*MU2^2 + 2*L^2*V.^2*A*C*G*KL*M*MU2^2*PD^2 - 4*L^2*V
    .^2*A*C*G*KL*M*MU2^2*PD + 2*L^2*V.^2*A*C*G*KL*M*MU2^2 + 2*L^2*V.^2*A*D*G*KD*M*
```

```
    MU2^2*PD^2 - 4*L^2*V.^2*A*D*G*KD*M*MU2^2*PD + 2*L^2*V.^2*A*D*G*KD*M*MU2^2 - L
    ^2*A^2*AY.^2*M^2*PD^2 + 2*L^2*A^2*AY.^2*M^2*PD - L^2*A^2*AY.^2*M^2 + L^2*A^2*G
    ^2*M^2*MU2^2*PD^2 - 2*L^2*A^2*G^2*M^2*MU2^2*PD + L^2*A^2*G^2*M^2*MU2^2 + A^2*
    AY.^2*H^2*M^2*MU2^2).^(1/2) - L^2*V.^2*KD + V.^2*H^2*KD*MU2^2 + 2*L^2*V.^2*KD*
    PD - L^2*V.^2*KD*PD^2 + V.^2*A*H*KL*MU2^2 + V.^2*C*H*KL*MU2^2 + V.^2*D*H*KD*
    MU2^2 + A*G*H*M*MU2^2)./(M*(- L^2*PD^2 + 2*L^2*PD - L^2 + H^2*MU2^2));
axG2(:,3) = -(V.^2*H^2*KD*MU2^2 - L^2*V.^2*KD - (L^2*V.^4*A^2*KL^2*MU2^2*PB^2 - 2*
    L^2*V.^4*A^2*KL^2*MU2^2*PB + L^2*V.^4*A^2*KL^2*MU2^2 + 2*L^2*V.^4*A*C*KL^2*MU2
    ^2*PB^2 - 4*L^2*V.^4*A*C*KL^2*MU2^2*PB + 2*L^2*V.^4*A*C*KL^2*MU2^2 + 2*L^2*V
    .^4*A*D*KD*KL*MU2^2*PB^2 - 4*L^2*V.^4*A*D*KD*KL*MU2^2*PB + 2*L^2*V.^4*A*D*KD*
    KL*MU2^2 + L^2*V.^4*C^2*KL^2*MU2^2*PB^2 - 2*L^2*V.^4*C^2*KL^2*MU2^2*PB + L^2*V
    .^4*C^2*KL^2*MU2^2 + 2*L^2*V.^4*C*D*KD*KL*MU2^2*PB^2 - 4*L^2*V.^4*C*D*KD*KL*
    MU2^2*PB + 2*L^2*V.^4*C*D*KD*KL*MU2^2 + L^2*V.^4*D^2*KD^2*MU2^2*PB^2 - 2*L^2*V
    .^4*D^2*KD^2*MU2^2*PB + L^2*V.^4*D^2*KD^2*MU2^2 + 2*L^2*V.^2*A^2*G*KL*M*MU2^2*
    PB^2 - 4*L^2*V.^2*A^2*G*KL*M*MU2^2*PB + 2*L^2*V.^2*A^2*G*KL*M*MU2^2 + 2*L^2*V
    .^2*A*C*G*KL*M*MU2^2*PB^2 - 4*L^2*V.^2*A*C*G*KL*M*MU2^2*PB + 2*L^2*V.^2*A*C*G*
    KL*M*MU2^2 + 2*L^2*V.^2*A*D*G*KD*M*MU2^2*PB^2 - 4*L^2*V.^2*A*D*G*KD*M*MU2^2*PB
     + 2*L^2*V.^2*A*D*G*KD*M*MU2^2 - L^2*A^2*AY.^2*M^2*PB^2 + 2*L^2*A^2*AY.^2*M^2*
    PB - L^2*A^2*AY.^2*M^2 + L^2*A^2*G^2*M^2*MU2^2*PB^2 - 2*L^2*A^2*G^2*M^2*MU2^2*
    PB + L^2*A^2*G^2*M^2*MU2^2 + A^2*AY.^2*H^2*M^2*MU2^2).^(1/2) + 2*L^2*V.^2*KD*
    PB - L^2*V.^2*KD*PB^2 + V.^2*A*H*KL*MU2^2 + V.^2*C*H*KL*MU2^2 + V.^2*D*H*KD*
    MU2^2 + A*G*H*M*MU2^2)./(M*(- L^2*PB^2 + 2*L^2*PB - L^2 + H^2*MU2^2));
wrongAcc = find(axG2(:,1:2)<-KD/M*V.^2 | abs(imag(axG2(:,1:2))) > 1e-6);  %
    Braking under traction
wrongDec = find(axG2(:,3)>-KD/M*V.^2 | abs(imag(axG2(:,3))) > 1e-6) +... %
    Traction under braking
    numel(axG2(:,1:2));
axG2([wrongAcc ; wrongDec]) = NaN;
axG2 = real(axG2);
G2 = [axG2(:,1) ; NaN ; axG2(:,2) ; NaN ; axG2(:,3)];
G2 = [G2 ; NaN ; G2];

%%% Powertrain
G3 = (-(V.^2*KD - P./V + FRR*(KL*V.^2 + G*M))./M).*ones(size(AY));
G3 = [G3 ; NaN ; G3];

%%% Drag
G4 = (-KD/M*V.^2).*ones(size(AY));
G4 = [G4 ; NaN ; G4];

%%% Velocity-acceleration consistency
H1 = (1/(2*DS)*(VI^2 - V.^2)).*ones(size(AY));
H1 = [H1 ; NaN ; H1];

%%% Store data
AY = [AY ; NaN ; AY ; NaN; AY];
AY = [-AY ; NaN ; AY];
end

% Replace non-physical data points
function [V, AX] = physicalData(V, AX, dir)
global KD M

% Include acceleration direction
switch dir
    case "positive"
        direction = AX >= 0;
    case "negative"
        direction = AX < 0;
    case "traction"
        direction = AX >= -KD/M*V.^2;
    case "braking"
```

```
        direction = AX < -KD/M*V.^2;
    otherwise
        direction = AX >= 0;
end

% Find non-physical data points
nonPhysical = V < 0 | abs(imag(V)) > 1e-6 | abs(imag(AX)) > 1e-6 | ~direction;

% Copy data for real array and set fake points as NaN
V               = real(V);
V(nonPhysical)  = NaN;
AX              = real(AX);
AX(nonPhysical) = NaN;
end
```

Matlab 1: Symbolic derivations to the minimizers of each optimization problem.

The code creates Figure B.1, which verifies the calculation of the minimizers. A zoomed in version of the minimizers near the velocity limit is given in Figure B.2.

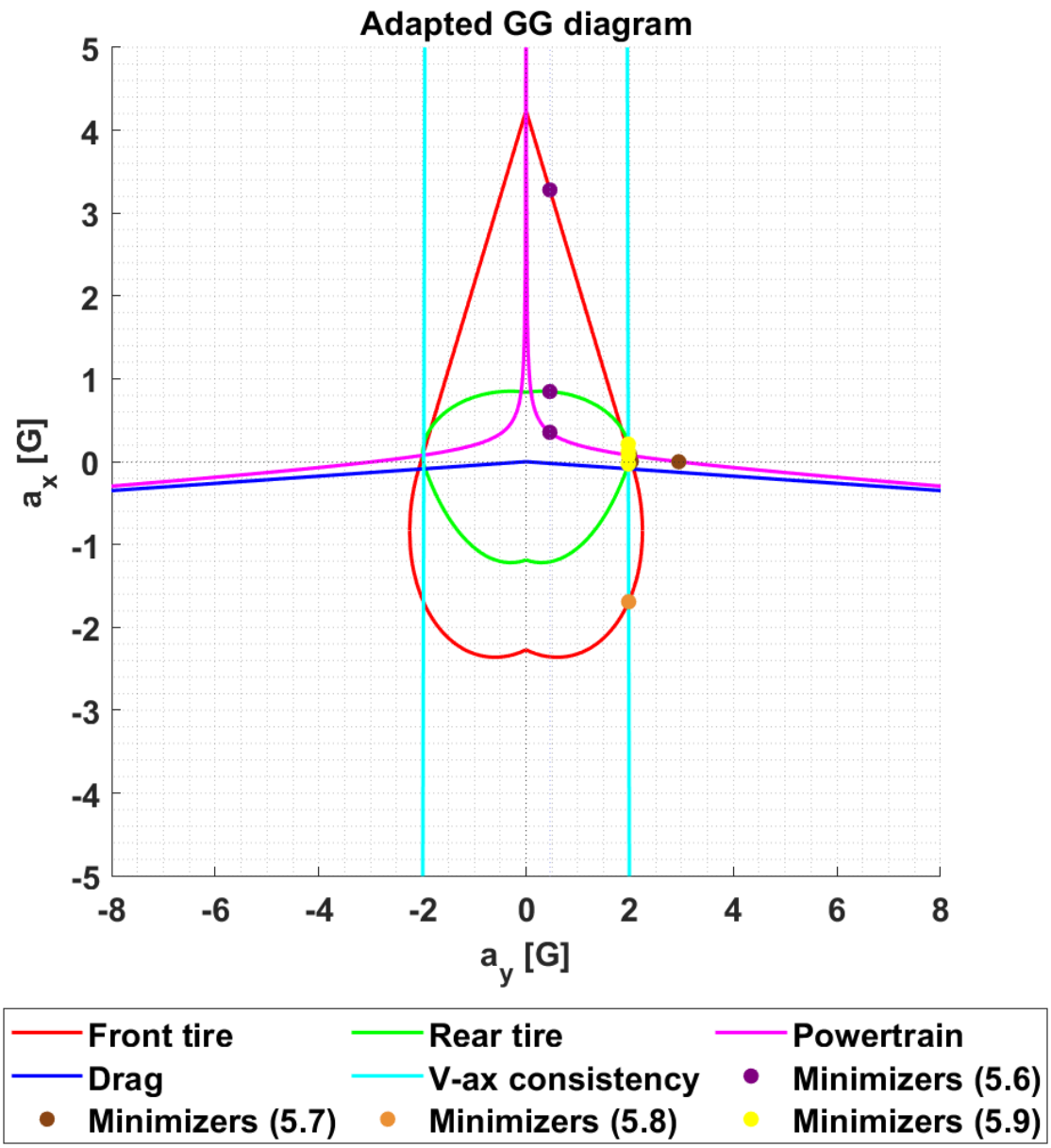


Figure B.1: Adapted GG diagram with the minimizers of each optimization problem given as dots in different colors.

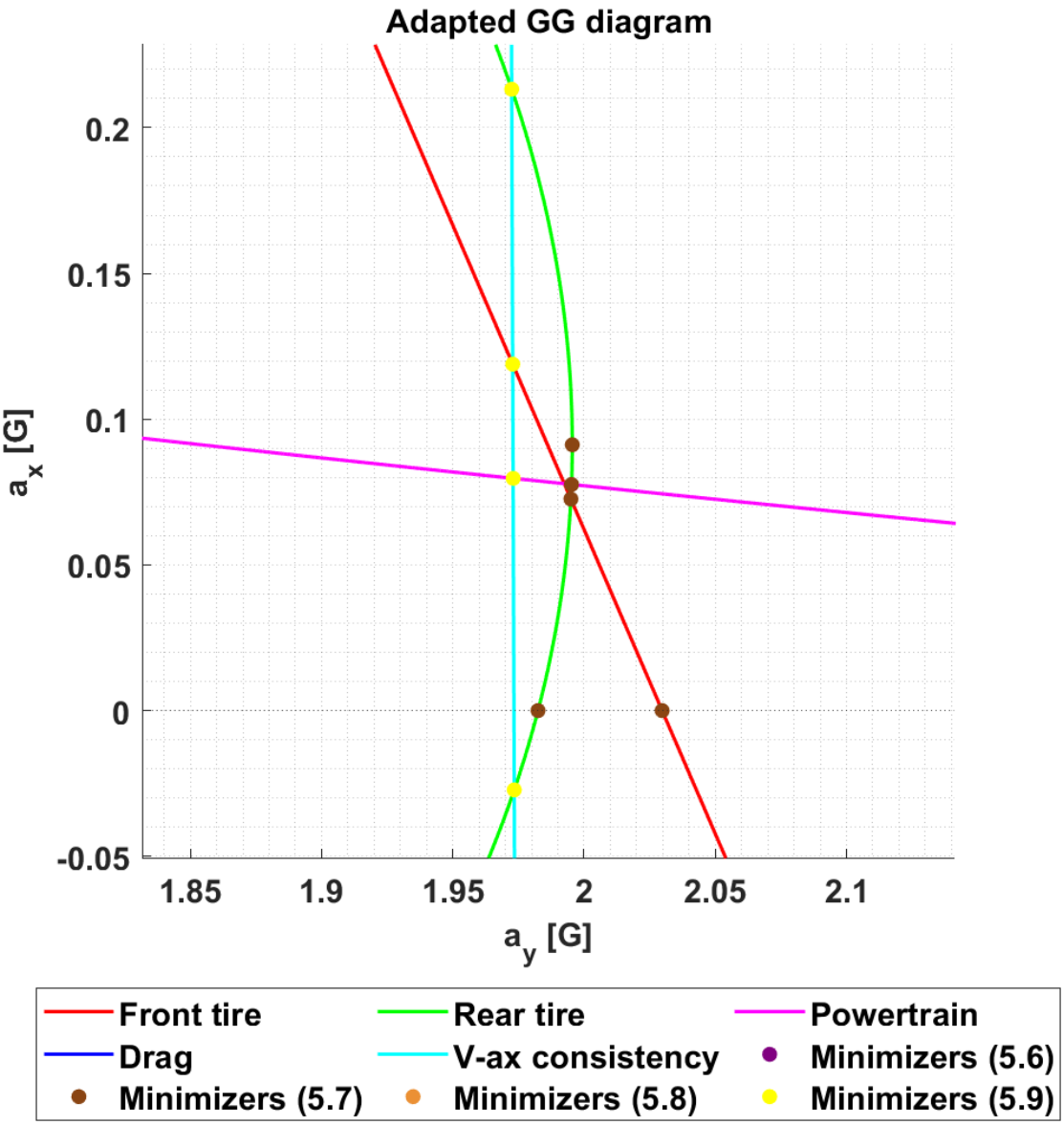


Figure B.2: Adapted GG diagram with the minimizers of each optimization problem given as dots in different colors. Zoomed in on the minimizers near the velocity limit.