Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Vehicle traction control algorithms for low speeds

Jan Kohout

Supervisor: doc. Ing. Tomáš Haniš, Ph.D.
Field of study: Cybernetics and Robotics
April 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Kohout Jan**	Personal ID number:	**492192**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Control Engineering**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Vehicle traction control algorithms for low speeds

Bachelor’s thesis title in Czech:

Trakční algoritmy vozu pro nízké rychlosti

Guidelines:

The goal of this thesis is to develop traction algorithms for conventional vehicle operating in low speed regions, where conventional mathematical models and control strategies are failing. The motivation is to improve vehicle traction forces utilization, while guarantee vehicle maneuver stability.
1. Get familiar with vehicle dynamic and tire mathematical models, implement suitable model.
2. Get familiar with traction algorithms, with focus on low-speed regime.
3. Develop and implement traction control strategy.
4. Test implementation using the sub-scale experimental platform.

Bibliography / sources:

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini – Vehicle Dynamics – Duisburg 2014
[2] Hans B. Pacejka - Tire and Vehicle Dynamics – The Netherlands 2012
[3] Robert Bosch GmbH - Bosch automotive handbook - Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. : Bentley Publishers

Name and workplace of bachelor’s thesis supervisor:

doc. Ing. Tomáš Haniš, Ph.D. Department of Control Engineering FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until:
by the end of summer semester 2023/2024

doc. Ing. Tomáš Haniš, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.
Date of assignment receipt

Student’s signature

Acknowledgements

I would like to thank my supervisor doc. Ing. Tomáš Haniš, Ph.D for his advice and supportive supervision of my work.

Many great thanks belongs to my family for their support during my studies. My appreciation also extends to my girlfriend, without whom I could not have finished this thesis. Last but not least, I want to express a special thanks to Matěj Golda for proofreading this thesis.

Declaration

I hereby declare that this bachelor's thesis was finished on my own and that I have cited all the used information sources in compliance with the Methodical guideline on the observance of ethical principles in the preparation of university graduate thesis.

In Prague, 20. April 2023

Abstract

In this thesis, I deal with the development of a vehicle traction control system. The goal was to develop a traction control system that, works at speeds close to zero. Firstly, I described available mathematical models for tires and vehicles, then I used Pacejka's magic formula and identified a single-track model to develop and test control systems. I started with a simple rule-based slip ratio control algorithm, then I set up a PID controller. I verified my tests in the Car-maker simulation software. Last, but not least, I tested both systems on a scaled-down platform with two independent rear motors.

Keywords: PID, Slip ratio, Single-track model, Launch control, Vehicle dynamic, Longitudinal dynamic

Supervisor: doc. Ing. Tomáš Haniš, Ph.D.

Abstrakt

V této práci se zabývám vývojem systému řízení trakce vozidla. Cílem bylo vyvinout systém řízení trakce, který funguje při rychlostech blízkých nule. Nejprve jsem popsal dostupné matematické modely pro pneumatiky a vozidlo, poté jsem použil magickou formuli Pacejka a identifikoval jsem jednoduchý model jednokolejového vozidla pro vývoj a testování řídicích systémů. Začal jsem s jednoduchým algoritmem řízení podle poměru skluzu, poté jsem nastavil PID regulátor. Své testy jsem ověřil v softwaru Carmaker. A nakonec jsem otestoval oba systémy na zmenšené platformě s dvěma nezávislými zadními motory.

Klíčová slova: PID, Poměr prokluzu, Jednostopý model, Kontrola rozjezdu, Dynamika vozu, Podélná dynamika

Překlad názvu: Trakční algoritmy vozu pro nízké rychlosti

Contents

Figures

Tables

Chapter 1

Introduction

Electric cars are becoming more popular as their prices are getting lower, and the market provides a variety of options, from family sedans to hypercars. One of the key differences is that an electric motor provides high torque output from low RPMs. This is different from the commonly used combustion engines that progressively build up torque with RPM, with peak torque available only in a smaller range of RPMs. The development of control algorithms provides additional safety and comfort to end-users while improving performance. Additionally, it should also reduce non-exhaust particles emitted by tires and brakes, enabling cars to adhere more easily to the newly discussed EURO 7 norm. However, these developed algorithms will not only benefit electric cars but also gas-powered ones.

Since most commercially used traction control algorithms such as ABS, DTC, ESP, and others are either less effective or non-functional at low speeds or while the vehicle is stationary, the implementation of a low speeds control has been proposed and tested.

The first part of thesis focuses on modeling and simulation. Tire models are discussed as well as vehicle modeling. Slip-ratio estimation for low speed is proposed. Logical control algorithm and PID controllers are described, created and tested in simulations and on mathematical models in MATLAB/Simulink.

Second part focuses on testing on a scaled platform. The platform is described. Traction control algorithms are implemented and tested on various surfaces and under different driver input scenarios.

1.1 Motivation

There are many advantages in controlling tire slip at low speeds. To begin with, we can improve the safety of passengers. Depending on car manufacturer, safety systems a have minimal speed when they are activated. This is often due to lack of accurate speed measuring. For example BMW owner's manual states that ABS becomes operational when speed exceeds 8 km/h [2]. Another source suggests that the minimal speed can be as high as 25km/h [8]. As electric cars have more torque available during start up, there is more potential for a wheel slip than on traditional fossil fuel car. Implementing a control algorithm can avoid collisions when car cannot safely start. For example on stop lights or assist during start-up on roads covered with ice or snow.

Another aspect of controlling wheel slip is to maximize performance of the car. Car manufacturers are using as a benchmark value the time it takes to accelerate from stand still to cartain speed. Implementing algorithm, that can maximize available tire grip will increase acceleration thus allowing the car to reach higher speeds in a shorter time. There are many new racing series just for electric cars. Being able to reliably start during different weather conditions would benefit every racing team. Maximizing acceleration and limiting wheel slip during start is important not only to be quicker but also to protect and not overheat tires.

Last but not least there is the environmental impact. Euro 7 norm is being discussed [19]. This norm wants to regulate non-exhaust particles. That means, among other things, cars will need to reduce the amount of particles released from breaks and tires. Once again control system that minimizes tire slip would prolong tire life thus emit less particles during a car life cycle. Many low speeds traction control algorithms use breaks to limit amount of torque applied on tire. This causes break pads to wear out. Using a different style of torque control will lower the amount of particles emitted by breaks. This would also positively impact servicing costs as prolonging service intervals is to be expected.

1.2 State of the art

Car manufacturers have implemented different types of low speeds traction control systems. There are systems to help with maintaining speed and control of vehicle while descending from hills. For example, BMW has implemented Hill Descent Control [10]. HDC is a downhill driving assistant that automatically controls vehicle speed on steep downhill gradients in speeds under 15 km/h. The system holds set speed without needing drivers break input. This system is featured on their 4x4 cars since 2000. Volvo, Renault, Range Rover and others also have similar systems. As stated on Volvo website [28], alongside of brakes control, HDC also lowers the sesitivity of the gas pedal and sets on engine rev limitter. On their website they also say that the system does not have to work in all situations.

As many cars nowadays come with electronic handbreaks they often, include feature called the auto hold. When engaged, the car automatically applies brake pressure to maintain a stationary position and eliminates undesired movement. When driver steps on the throttle pedal, the system automatically releases breaks and allows the car to roll. This assistent eliminates creeping on cars with automatic transmision. This is often used in a combination with hill start assist [24]. HSA has two primary usages. Firstly, it helps drivers with manual transmission to an easier uphill start. Secondly, it helps to start the vehicle on less adhesive surfaces, where an excessive throttle would cause a tire slip.

Sports cars manufacturers have often implemented the so-called launch control. In cars with combustion engines, it often consists of a rev-limiter and a harder shifting pattern. For example, the ZF8 transmission used in BMWs does only 50-100 launches with hard shiftting pattern then the functionality is limited [16]. In Electric cars the launch systems are different. Currently the production car with the fastest acceleration from 0-97 km/h 100 (mph) is Tesla Model S Plaid [7]. The measured time is 1,98s. This however was achieved on a specially prepared surface and the time of the first foot rolled was subtracted. After adding this time the, overall acceleration time is 2,07s. The car uses height adjustable suspension to lower front end. It also preaplies power to the rear motors of the car to maximize the grip [4]. Another recently released electrical sports car the Porche Tycan also features launch control. The car is only a rear-wheel drive, however it features overboost technology. This means that an output of 560 kW is available during the start up (instead of the peak output of 460 kW). According to the datasheet [5]. the 0-100 time is 2.8 seconds In September 2022 a new record for the fastest 0-100 km/h has been set by a team of students from the University of Stuttgart. They have used electric Formula Student car with four wheel drive. The newly set record is $1.461s$ [9].

To develope a new control system I can first start with antilock brake systems. ABS system is a control device that prevents wheel lock during breaking and, as a result, retains vehicle steering ability and stability [23]. As I want to prevent an excessive slip on powered wheels and also retain stability, I can use principal functions of ABS but with the intention to increase the speed. Systems that limit wheel spin during acceleration often use parts of the ABS system. ABS is mandatory on new vehicles. As stated by Automotive Handbook [23] there are two main interventions when limiting wheel slip. At low speeds or when just one wheel spins the break intervention is used. If both of the driven wheels spin, the engine intervention is used to reduce the overall torque.

In this paper [13] researchers used open loop control as well as PID controller to control the slip ratio by reducing torque during start up. In [15] researchers clutch position controller and subsequently torque controls are used to control slip on a motorcycle. Using a fuzzy PID controller to control independent rear wheels was described in [17]. In this paper [21] a sliding mode controller is designed and tested on a real car with two independent motors in rear wheels. To replace Hall-effect sensors this paper [26] describes the design and development of a non-intrusive inertial speed sensor. This article [27]describes a model-based tire slip control that utilizes an online estimation of the road friction coefficient. To ensure lateral stability torque a vectoring is often used as staed in Automotive Handbook [23]. The description [20] of torque vectoring solution by feedforward and feed back control is described in thesis. However it is disabled at low speeds.

Chapter 2

Tire-to-road interface

There is a wide range of tire models developed. Each type has different complexity and accuracy. Pacejka divides them into four categories ranging from empirical models based on experimental data to complex physical models. Each model can be used to describe forces or torques in different scenarios. These models can be also divided into a steady-state and a not-steady-state. Here, only four models are described. More can be found in Tire and vehicle dynamics [22].

2.1 Tire coordinates system

Multiple physical quantities need to be accounted for and described while modeling tires. To describe them accurately I need to define a coordinate system. As there are multiple, I choose the one used by Pacejka as it is also used in the Single-track model described in the chapter 3. In figure 2.1 there is cartesian system of coordinates in the center of a wheel. $\dot{\rho}$ is rotational speed of the wheel. Vector v is the velocity vector projection of the i-th wheel's center on its x-axis. In figure 2.1 we can also see a top view of a tire. There are x and y axies, forces F_x, F_y acting upon the tire, and the tire slip-angle α.

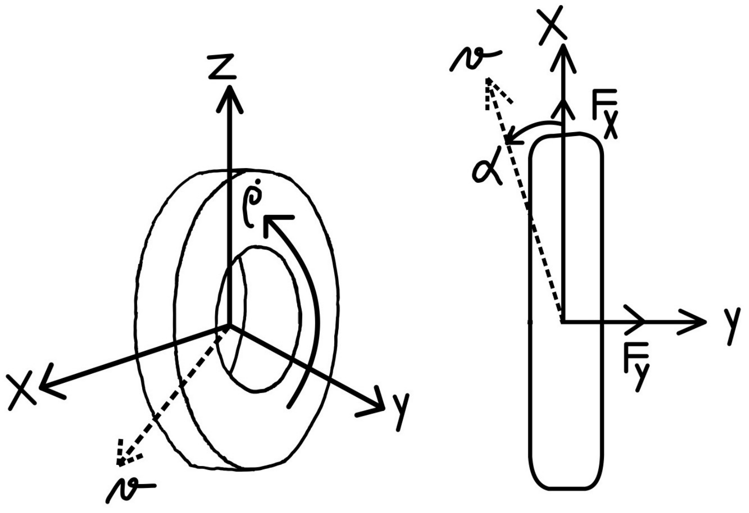


Figure 2.1: Wheel coordinates system

2.2 Brush tire model

It is a steady-state model that can be sorted as a simple physical model. The brush model is a row of bristles that touchs road surface. When the wheel is free rolling in a straight line without any slip the bristles touching the road surface are perpendicular to it. When break or drive torque is applied the corresponding slip is created, as can be seen in figure 2.2. We can also see the change in the amount of slip as the thread travels through the contact patch. This model is purely physical and does not need any measured data.

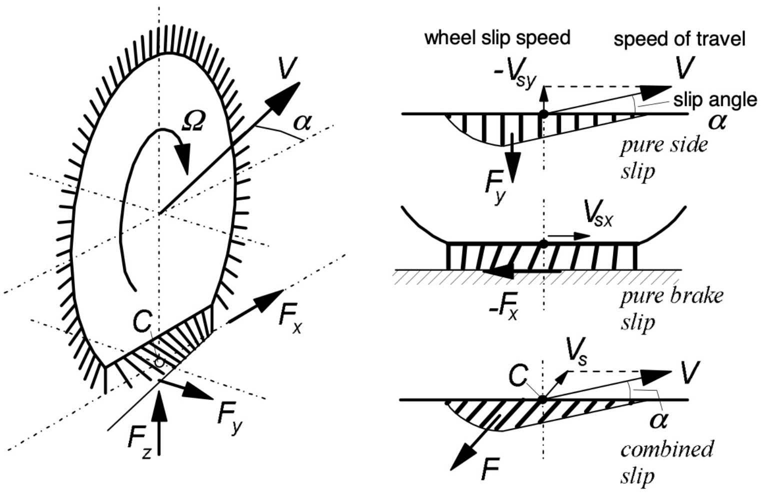


Figure 2.2: Brush model (image used from [22])

2.3 String tire model

The string tire model is sorted as a non-steady-state model. This model consists of an endless string that is kept under a certain pretension by a uniform radial force. The string has finite contact with the road. It can elastically move with respect to the wheel-center plane but is prevented from moving in the circumferential direction. This model can be extended by adding parallel strings.

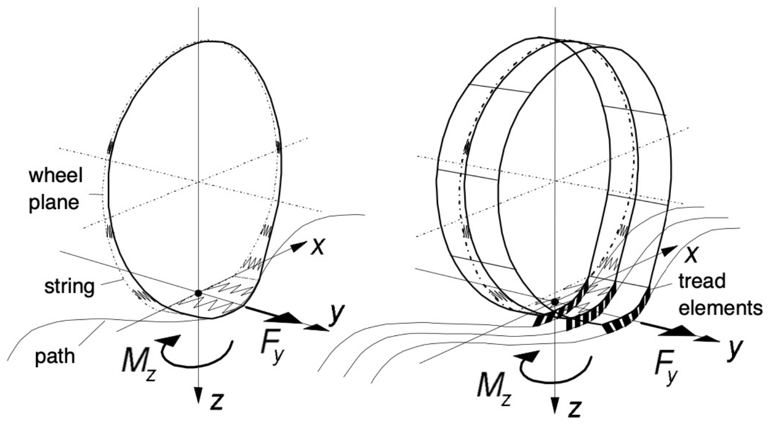


Figure 2.3: String tyre model
(image used from [22])

2.4 Pacejka Magic formula

Well known model was developed by Hans B. Pacejka and described in [22]. It is called magic formula and it uses more than 20 parameters. He also introduced simplified version called Simplified Pacejka Magic formula. This simplified version uses only four shaping coefficients. It uses only goniometric functions. It allows us to compute forces and also torques acting upon the tire. For the force F the formula is defined as:

$$F(\alpha) = DF_z sin(C \arctan(B\alpha - E(B\alpha - \arctan(B\alpha)))), \tag{2.1}$$

where

B is stiffness factor
C is shape factor

D is peak value
E is curvature factor
F_z is wheel-load.

To get the longitudinal force F_x, we use 2.4, as the function of a slip ratio λ. To get lateral force F_y the F is then function of slip angle α. This model can be sorted as semi-empirical steady-state model. The maximum of F_x is dependent on the wheel-load F_z as can be seen in figure 2.4.

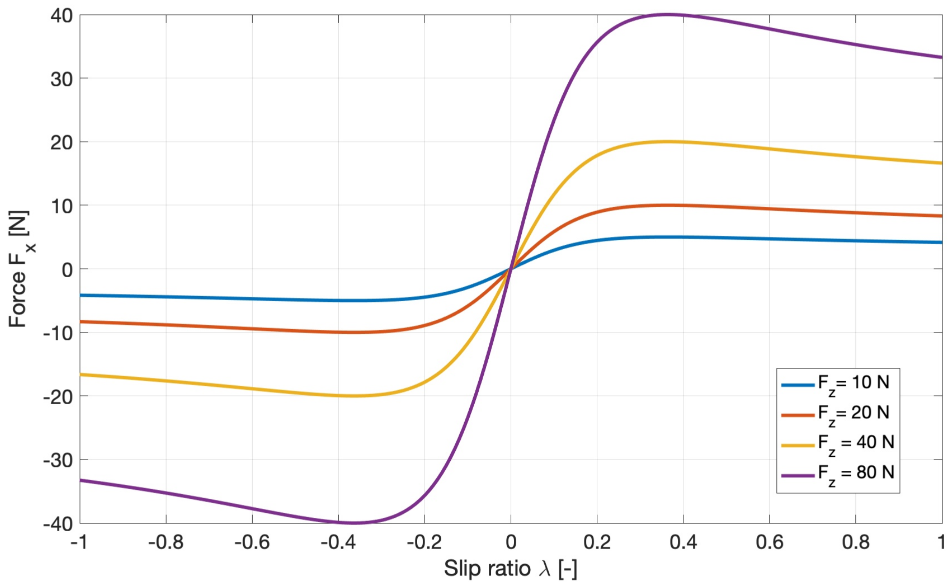


Figure 2.4: Dependence of force F_x on load force F_z using Magic formula

2.5 Two-Lines Tire Model

The two-lines model is composed of three parts. It has two saturations and one linear part. Force F is a function of either the tire slip angle for lateral force or a function of the slip ratio λ in the case of longitudinal force. Equation for F_x is:

$$F_x = \begin{cases} C\lambda & |\lambda| < \frac{\mu_{max}}{C} \\ \mu_{max} & |\lambda| \geq \frac{\mu_{max}}{C} \end{cases} \tag{2.2}$$

Chapter 3

Single-track implementation

In previous chapter, tire-to-road models were presented. They are an essential part of every vehicle model. For the development process, mathematical model of vehicle was used. As with the tire models, there are numerous models of vehicles. Since my testing platform is 1:10 scaled-down platform of a race buggy with two independent rear motors I decided to start with nonlinear Single-Track model as it is faster to identify than a twin-track. Testing the algorithms first in a simulation will speed up the process and eliminate dangers associated with deploying it on the hardware untested.

In my work I used an implemented non-linear single-track model from github repository[13]. It is implemented in MATLAB/Simulink environment.

3.1 Assumptions

Single-track model simplifies description of a vehicle. It uses just one wheel per axle. It neglects all lifting, pitching and rolling motion. Vehicle mass is assumed to be concentrated at the center of gravity. Mass distribution is considered to be constant. Pneumatic trail and aligning torque resulting from a side-slip angle of a tire are neglected.

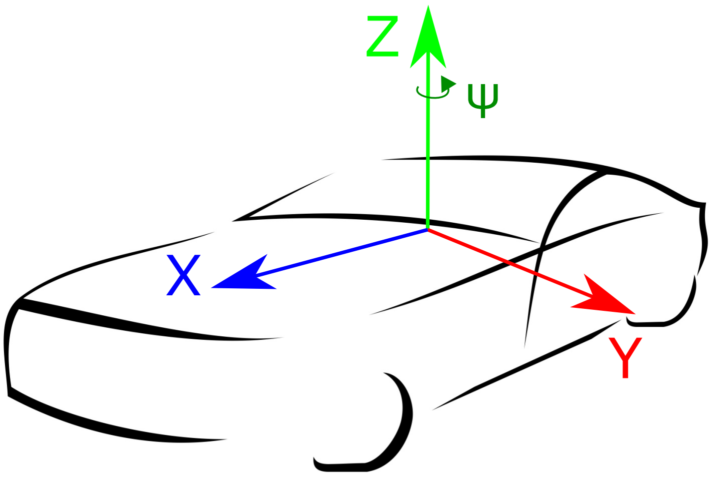


Figure 3.1: Car coordinates system (taken from [14])

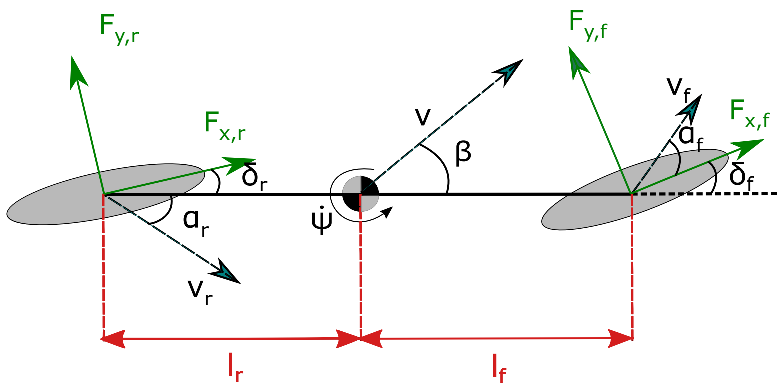


Figure 3.2: Single-track coords system (taken from [14])

3.2 Coordinate system and variables description

Cartesian coordinate system of a vehicle can be seen in figure 3.1. The x axis goes toward the front of the car from its center of gravity. The y axes go from center of gravity towards left side of the vehicle and z axis points from center of gravity upwards to the roof of vehicle. The vehicle's yaw angle ψ has a positive increment while turning left.

A detailed description of the single-track coordinate system is shown in figure 3.2. Here we can see previously defined forces acting on a wheel. $F_{x,i}$ and $F_{y,i}$ are forces acting upon the center of gravity of the i-th wheel along the x and y axes. β is side-slip angle. δ_i is steering angle of i-th wheel.

The model has three degrees of freedom.

1. Longitudinal motion:

$$F_x = -mv(\dot{\beta} + \dot{\psi}) \sin \beta + m\dot{v} \cos \beta \tag{3.1}$$

2. Lateral motion

$$F_y = mv(\dot{\beta} + \dot{\psi}) \cos \beta + m\dot{v} \sin \beta \tag{3.2}$$

3. Moment acting around z axes (Yaw moment):

$$M_z = I_z \ddot{\psi} \tag{3.3}$$

In equations 3.1 to 3.3 F_x and F_y are forces applied on the center of gravity in x resp. y axes, m is the vehicle mass, v is velocity of the COG, I_z is the moment of inertia of the vehicle around the z axis.

To get forces acting upon each wheel the steering angle projection has to be done.

$$\begin{pmatrix} F_x \\ F_y \\ M_z \end{pmatrix} = \begin{pmatrix} \cos(\delta_f) & -\sin(\delta_f) & \cos(\delta_r) & -\sin(\delta_r) \\ \sin(\delta_f) & \cos(\delta_f) & \sin(\delta_r) & \cos(\delta_r) \\ l_f \sin(\delta_f) & l_f \cos(\delta_f) & -l_r \sin(\delta_r) & -l_r \cos \delta_r \end{pmatrix} \begin{pmatrix} F_{x,f} \\ F_{y,f} \\ F_{x,r} \\ F_{y,r} \end{pmatrix} \tag{3.4}$$

3.3 Impllemented tire-to-surface model

Each wheel uses its coordinate system described in 2.1. Each wheel model has an internal state that describes its rotation acceleration using the following formulas adopted from [25]

$$\ddot{\rho}_f = \frac{1}{J}(\tau_f - R_f F_{x,f} - sign(\dot{\rho}_f)\tau_{Bf} - k_f v_{xf}), \tag{3.5}$$

$$\ddot{\rho}_r = \frac{1}{J}(\tau_r - R_r F_{x,r} - sign(\dot{\rho}_r)\tau_{Br} - k_r v_{xr}), \tag{3.6}$$

Pacejka Magic Formula was used a a tire-to-surface model. It is decribed in section 2.4.

3.4 Slip ratio definition

Translation speed for each wheel can be defined as:

$$v_{ci} = \dot{\rho}_i R_i, \tag{3.7}$$

where ρ_i is rotational velocity of i-th wheel with coresponding radius R_i.

The tavel velocity of each wheel can be calulated using the side-slip angle β the velocity of the vehicle's COG v, the yaw rate of the vehicle $\dot{\psi}$, and the steering angle projection δ_i applied on that wheel:

$$v_{xf} = v\cos(\beta)\cos(\delta_f) + (v\sin(\beta) + lf\dot{\psi})\sin(\delta_f). \tag{3.8}$$

$$v_{xr} = v\cos(\beta)\cos(\delta_r) + (v\sin(\beta) - lr\dot{\psi})\sin(\delta_r). \tag{3.9}$$

Slip ratio is usually defined as:

$$\lambda_i = \frac{v_{ci} - v_{xi}}{\max(v_{ci}, |v_{xi}|)}, \quad -1 \leq \lambda_i \leq 1. \tag{3.10}$$

This definition is widely used. The single-track model is also using it. If the slip ratio is -1 then the wheel is locked and sliding. If $\lambda = 1$ then the wheel is rotating and slipping but the travel velocity of the wheel and therefore the car is significantly smaller. As per the Automotive Handbook [23] the car is stable while the slip ratio is between $< -0.2\ ,\ 0.2 >$. This is going to be the maximum reference value for any further controllers I will design.

When the speeds approach zero the definition 3.10 becomes unstable as division by zero is undefined on interval $< -1\ ,\ 1 >$. To solve this I have defined switching speed $v_{sw} = 3ms^{-1}$ when the definition 3.10 can be used. In practice, this is the lowest speed when the speed read from wheel sensors can be relied on. In figure 2.4, we can see slip curve where the dependence between force F_x and slip ratio is shown. We know from 2.4 that the peek of forward force F_x is dependent on the wheel load but it is around the interval $< 0.05\ ,\ 0.4 >$.

To define value of slip ratio in speeds lowet than v_{sw} I have used equations 3.11, 3.12 and 3.13. The equations 3.11, 3.12 are not as precise as equation 3.13. Function arctan has similiar shape to 3.10 and is well definde for small numbers. I used this definiton inmy thesis.

$$\lambda = \begin{cases} 0.1, & v_{xi} < 3 \\ \dfrac{v_{ci} - v_{xi}}{\max(v_{ci}, |v_{xi}|)} & \text{otherwise} \end{cases} \tag{3.11}$$

$$\lambda = \begin{cases} \dfrac{v_{ci} - 0.1}{\max(v_{ci}, 0.1)}, & v_{xi} < 3, \\ \dfrac{v_{ci} - v_{xi}}{\max(v_{ci}, |v_{xi}|)} & \text{otherwise} \end{cases} \tag{3.12}$$

$$\lambda = \begin{cases} \dfrac{\arctan(v_{ci} - v_{xr}) * 1.95}{\pi}, & v_{xi} < 3, \\ \dfrac{v_{ci} - v_{xi}}{\max(v_{ci}, |v_{xi}|)} & \text{otherwise} \end{cases} \tag{3.13}$$

3.5 Identification

Implementation of a single-track model initially comes with several pre-defined vehicles and their weight distribution. To initially develop algorithms I have used preset named Fabia. However, to be able to create accurate design of a future traction control system the identification of model variables is necessary. The overall weight of the vehicle is divided into seven sections named: *Engine, AxleFront, AxleRare, Gearing, Transmission, Tank, Coachbuilder.* Those are assigned to their places in the modeled car. These weight segments are summed up in three sections: *Front, Center, Back.* I measured the weight distribution of the SDP by measuring the weight on each axle and the overall weight. The distance from a COG to each axle is also a model parameter and it was measured. Measured values are in table 3.1. From the measured values we can see that weight distribution is 53:47. The ideal weight distribution is considered to be a 50:50 ratio.

-	Mass [kg]	d center [m]
Rear axel	0.825	0.141
Front axel	0.935	0.159
Total	1.760	0.3

Table 3.1: SDP weights

To model the wheels of SDP their radius was calculated from the circumference. The moment of inertia was calculated using the assumption that wheel is a uniform solid cylinder. Parameters and results are in table 3.2.

Moment of inertia I_z around the z axis is computed from values set in MATLAB configuration file. Computed value is $I_z = 0.0427\ kg/m^2$. To confirm this value I have used the three string method described in [3].

-	**Front**	**Rear**
Circumference[m]	0.267	0.267
Radius[m]	0.0425	0.0425
Weight[g]	190	230
Moment of inertia[kg^{-2}]	00003	0.0004

Table 3.2: SDP wheels properties

Difference between results of measuring and computed value was smaller than 0.01 kg/m^2 to the one computed in MATLAB.

As a last parameter, I need to estimate the maximal torque that can be achieved by the SDP. According to the datasheet [1] the maximal current for one motor is 30A for 30s. Ing. Hostačný who created the SDP says that the maximal current is between 35-40A [18]. For my calculation I am going to consider $I_{max} = 35A$. The equation for constant velocity value can be written as:

$$Kv = 1035 \ \frac{RMP}{V} = 108.38 \ \frac{Rad \cdot s^{-1}}{V}. \tag{3.14}$$

Car battery has voltage range from 6.4 to 8.4V. I am gonna assume a state of chage of battery to be 80% thus $V = 8.0 \ V$.We can compute the value of Kt:

$$Kt = \frac{1}{Kv} = 9.22 \ \frac{mNm}{A}. \tag{3.15}$$

We can calculate the motor torque:

$$Tq_m = Kt * I_{max} = 322.7 \ mNm. \tag{3.16}$$

To compute the torque on a wheel we need to consider a gear ratio $gr = 2.125$ of the belt driven wheel. Aa there are two motors on the rear axel the final estimation of the maximal torque of SDP is:

$$2 \cdot Tq_w = 2 \cdot Tq_m \cdot gr = 2 \cdot 685,7 = 1371.4 \ mNm. \tag{3.17}$$

3.6 Road friction simulation

In section 3.7 different test scenarios were created for different road surfaces. I want to be able to start off controllably and effectively with the car on various surfaces, ranging from ice to specifically prepared surfaces for drag racing.This means surfaces with different friction coefficient.

A tire can not generate combined force in the longitudinal and lateral direction greater then vertical force F_z. this restriction is given by friction ellipse [14]. To simulate the friction I have modified parameter D in Pacejka's formula 2.4. This allows me to test different road friction conditions.

3.7 Test scenarios

For testing purposes I have created a number of test scenarios. I started with three different friction coefficient (0.3, 0.5, 0.8) these should represent an ice, wet and dry track. As said at the beginning of this chapter all pitching movements are neglected so testing different slopes of road was not possible. However, I added two different initial speeds $(0.001, 1)$ $[ms^{-1}]$, and also tested with different steering angles $(-1, 0, 10)$. To simulate real-time platform I added noise in form of uniformly distributed random number. The number is generated from interval $< -0.02,\ 0.02 >$ and added directly to the slip ratio signal.

Chapter 4

Slip control algorithms

4.1 Logical control

To start with development and set benchmark values I have proposed a simple algorithm that controls torque output based on predefined slip-ratio values. In some ways it behaves similarly to ABS. Algorithm outputs value y that is then integrated and sent as a torque request for the engines. Output decisions are based on set slip ratio thresholds. Flow chart of the algorithm can be seen in figure 4.1. When the torque request is greater than zero the algorithm starts. I have pre-defined value **s**$= 0.009$. Initial values of output **y** and of variables **angle_up**, **angle_down** are set to zero. Variables **angle_up**, **angle_down** are used to store the latest output for the integrator. Each cycle, the algorithm evaluates current value of the slip ratio. If λ is lower or equal to 0.09, the algorithm adds the values s to the previously stored value of **angle_up**. This value is then used as input for an integrator. While λ is in this region the torque grows exponentially. When λ is in interval $(0.09, 0.13 >$ the value of **angle_up** is no longer modified thus the torque grows linearly. If the slip ratio is in $(0.13, 0.16 >$ the output for the integrator is set to zero thus no longer increasing the torque demand.When value of λ is in $(0.26, 0.20 >$ the output for integrator is set to be negative, ten times the number **s** $(-10s)$. Torque will decrease linearly. When the value of λ is greater than 0.20, the value **s** is added to the previously stored value of **angle_down**, and the torque is decreasing exponentially. The output of integretor is staurated be smaller or equal to the trorque request from the user.

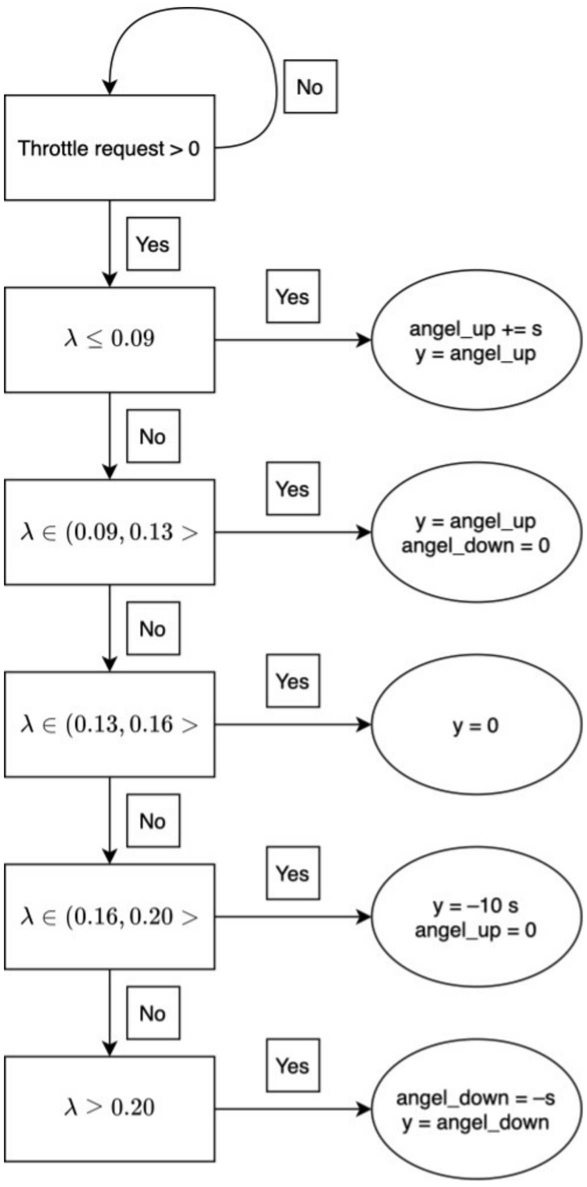


Figure 4.1: Flowchart of logical algorithm

4.2 Slip-ratio PID controller

I have decided to use close-loop PID contoller. I can describe transfer function of PID controler as:

$$H(s) = K_p + \frac{K_i}{s} + K_d s \tag{4.1}$$

To determine values of K_p,K_i, and K_d there are many methods. Basic trial and error method is possible but not always effective. The Zinger-Nichols method is heuristic tuning method that starts with just P controller. We increase the K_p until the output starts oscillating. When in this state we measure oscilation period T and used K_p as K_u. Then I can compute values of coefficients:

$$K_p = 0.6K_u \quad K_i = 1.2\frac{K_u}{T} \quad K_d = \frac{3K_uT}{40}. \tag{4.2}$$

As this method is not very precise and I am tuning PID for simulation, Ihave used a more systematic approach. I can describe the system as:

$$\ddot{\rho} = \frac{1}{I_z}(\tau_r - R_rFx_r), \tag{4.3}$$

$$\dot{\lambda} = \frac{\ddot{\rho}v_x}{\dot{\rho}^2R_r} - \frac{v_x}{\dot{\rho}R_r}, \tag{4.4}$$

$$\dot{v} = \frac{1}{m}Fx_f. \tag{4.5}$$

Equation 4.8 is simpliofied equation 3.6. I set $\tau_{Bf} = 0$ this means no break torque is applied and I neglect wind and road resistance. To get stae-space decription I substitue:

$$\ddot{\rho} = \dot{\omega}, \quad \tau_r = u, \quad \lambda = y \tag{4.6}$$

and also use tire model 2.4:

$$Fx_r = I_z - R_{(}DF_z\sin(C\arctan(B\lambda - E - (B\lambda - \arctan(B\lambda))))) \tag{4.7}$$

I get this state space desciprion:

$$\dot{\omega} = \frac{1}{I_z}(u - R_rFx_r), \tag{4.8}$$

$$\dot{\lambda} = \frac{\dot{\omega}v_x}{\omega^2R_r} - \frac{v_x}{\omega R_r}, \tag{4.9}$$

$$\dot{v} = \frac{1}{m}Fx_f, \tag{4.10}$$

$$y = \lambda. \tag{4.11}$$

I linearize this model i operating point p:

$$p_0 = \begin{pmatrix}\omega_0 & \lambda_0 & v_0\end{pmatrix} = \begin{pmatrix}2.35 & 0.1 & 0.1\end{pmatrix} \tag{4.12}$$

Matrix decription is:

$$A = \begin{pmatrix} \frac{\partial\dot{\omega}}{\partial\omega} & \frac{\partial\dot{\omega}}{\partial\lambda} & \frac{\partial\dot{\omega}}{\partial v} \\ \frac{\partial\dot{\lambda}}{\partial\omega} & \frac{\partial\dot{\lambda}}{\partial\lambda} & \frac{\partial\dot{\lambda}}{\partial v} \\ \frac{\partial\dot{v}}{\partial\omega} & \frac{\partial\dot{v_x}}{\partial\lambda} & \frac{\partial\dot{v}}{\partial v} \end{pmatrix} = \begin{pmatrix} 0 & \frac{CR_R}{I_z} & 0 \\ \frac{v}{\omega_0^2R_r} & 0 & \frac{-1}{\omega_0R_r} \\ 0 & \frac{C}{m} & 0 \end{pmatrix}, \tag{4.13}$$

$$B = \begin{pmatrix} \frac{\partial \dot{\omega}}{\partial u} \\ \frac{\partial \dot{\lambda}}{\partial u} \\ \frac{\partial \dot{v}}{\partial u} \end{pmatrix} = \begin{pmatrix} \frac{1}{I_z} \\ 0 \\ 0 \end{pmatrix}, \tag{4.14}$$

$$C = \begin{pmatrix} \frac{\partial y}{\partial \omega} & \frac{\partial y}{\partial \lambda} & \frac{\partial y}{\partial v} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}, \tag{4.15}$$

$$D = \begin{pmatrix} \frac{\partial y}{\partial u} \end{pmatrix} = \begin{pmatrix} 0 \end{pmatrix}, \tag{4.16}$$

I get transferfunction of the system:

$$H(s) = C(sI - A)^{-1} \cdot B + D = \frac{55 \cdot 10^3}{22s^2 - 4425}. \tag{4.17}$$

Using MATLAB and PID tuner I tuned the PID to agressive enough to react quickly but also to have minimal overshoot possible. The foloving constants were used:

$$K_p = 1.04 \quad K_i = 6.89 \quad K_p = 0.01. \tag{4.18}$$

4.3 Acceleration controller

To get reference value for Slip-ratio regulator I can either use constant value or use acceleration regulator. I used PI regulator. I set its output saturation to 0.2 as that is the value seen as a limit for uncontrollable car. On surface with $\mu = 1$ the maximal acceleration is around 4 ms^{-2}. I used this value as a reference for the acceleration controller. Values of PI regulator were set experimentally:

$$K_p = 0.06, \quad K_i = 0.16. \tag{4.19}$$

4.4 Yaw rate controller

To ensure lateral control i used a simple yaw controller that changes reference value for slip-ratio controller. As I will test the control systems on scaled-down platform with two independent motors I need to ensure their coordination. Yaw controller can achieve just that. As used on single-track it only limits acceleration to retain stability during a turn. When used on SDP I will change reference slip ratio for each wheel differently. I found two possible solutions to generate reference signal for yaw controller. First form thesis [12]

$$\dot{\psi} = \frac{v_x}{L + K_{us}\frac{v_x^2}{g}}\delta_f \tag{4.20}$$

and second found in Automotive Handbook [23]

$$\dot{\psi} = \frac{v_x}{L}\delta_f\frac{1}{1+(\frac{v_x}{v_{char}})^2} \tag{4.21}$$

By implementing torque vectoring I am trying to control vehicle yaw moment as it is described in 3.3. This is done by changing forces F_x acting on rear wheels. I can change them by changing torque value. I designed PI regulator which uses Eq. 4.21 as a reference signal. I have defined direction of yaw rate and yaw moment in Chapter 3. This definition is same as in SDP I can use output from PI regulator to alter reference value for slip-ratio regulators. To test my regulator I used twin-track model. I used previously implemented twintrack model [11]. I used predefined car values and added slip-ratio, acceleration, and yaw rate regulator. The model diagram is in figure 4.2. We can see if $\delta_f > 0$ then the driver is trying to turn left. This means that our yaw rate will be also greater than zero. When the yaw rate will be lower than reference value the error will be also greater than zero. As we want to turn left the right wheel needs to turn faster thus from Eq. 3.6 we need to increase torque on this wheel.

As said before single-track torque vectoring is impossible to implement as there is onyl one wheel per axel. Output of yaw rate controller lowers slip-ratio refference thus lowers acceleration to maintain speed in wich the car is able to make the turn. Schema of single-track launch control is in figure 5.1

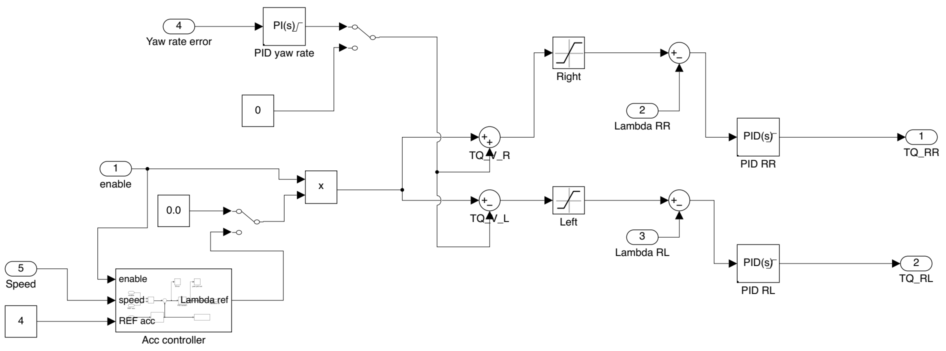


Figure 4.2: Simulink model of launch control in twin-track model

Chapter 5

Single-track simulation results

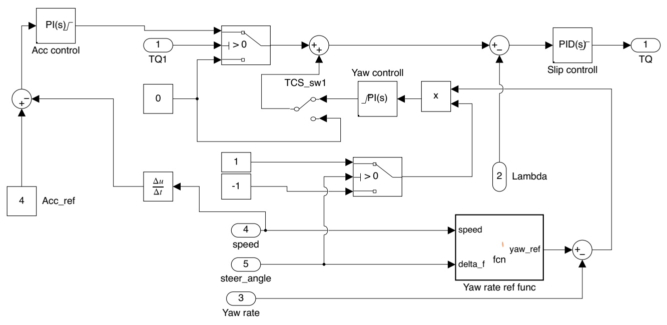


Figure 5.1: Simulink model of launch control in single-track model

In this section only graphs for the initial speed 0.001 m/s with $\delta_f = 0°$ and $\delta_f = 10°$ are presented. Graphs for the initial speed 0.001 m/s with $\delta_f = -1°$ are in the apendix of this thesis. Each simulation has its time limit set to 10 s as it is enough time for algorithms to achieve speeds grater than 10 m/s.

5.1 No control

In figure 5.2 the torque output is firmly set to 0.65 Nm as this is the value I computed in chapter 3.5 to be the maximal torque output of one motor. When no torque control is applied the slip ratio immediately reaches its maximal value of 1. The value stays at its maximal level throughout the entire simulation. This indicates the presence of a large amount of wheel spin. Because the steering angle is set to be zero the car drives in a straight line. We can clearly see when the road friction coefficient is higher the acting force F_X is greater thus the speed is increasing faster.

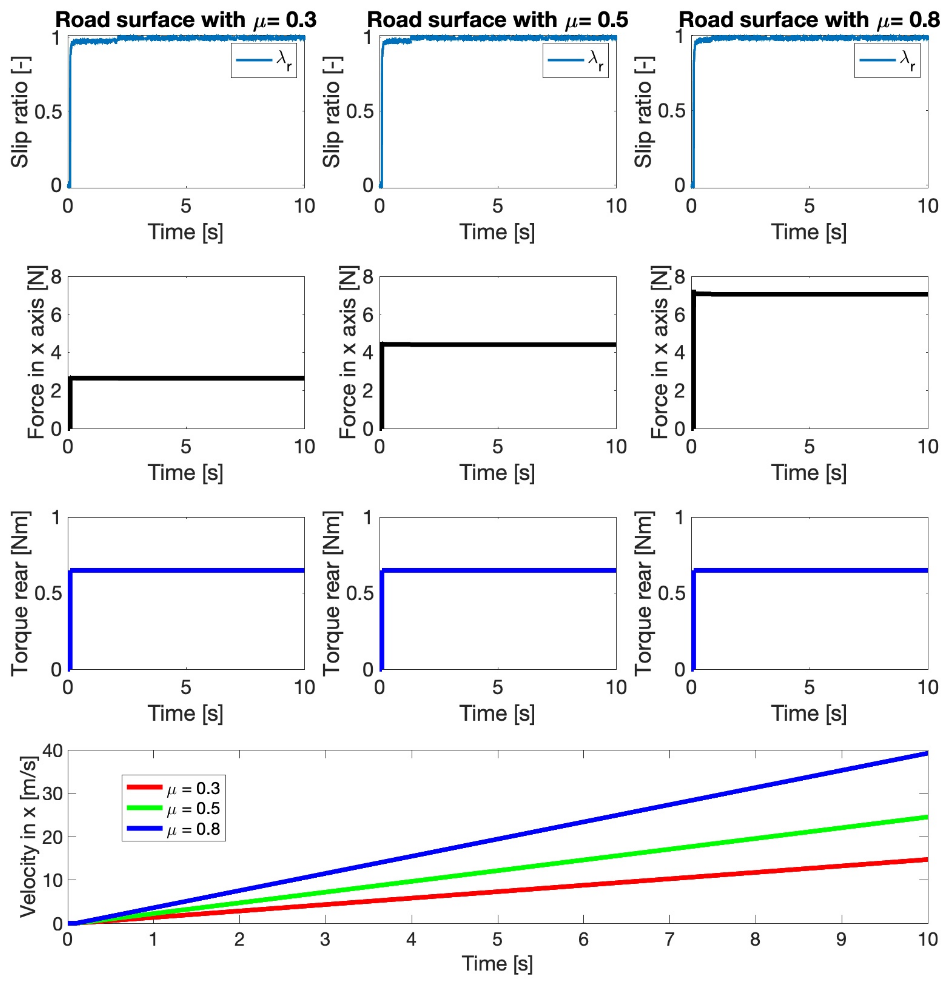


Figure 5.2: Simulation results with no controll and initial speed 0.001 m/s with $\delta_f = 0°$

Figure 5.3 shows the results of the simulation when the steering angle is set to 10° throughout the simulation. The output torque is also set to value 0.65 Nm throughout the simulation. The resulting slip ratios are similar as in the previous figure 5.2, the slip ratio quickly raises to its maximal value and stays that way the whole simulation. However, the force F_x is oscillating, and with it so is the longitudinal speed.

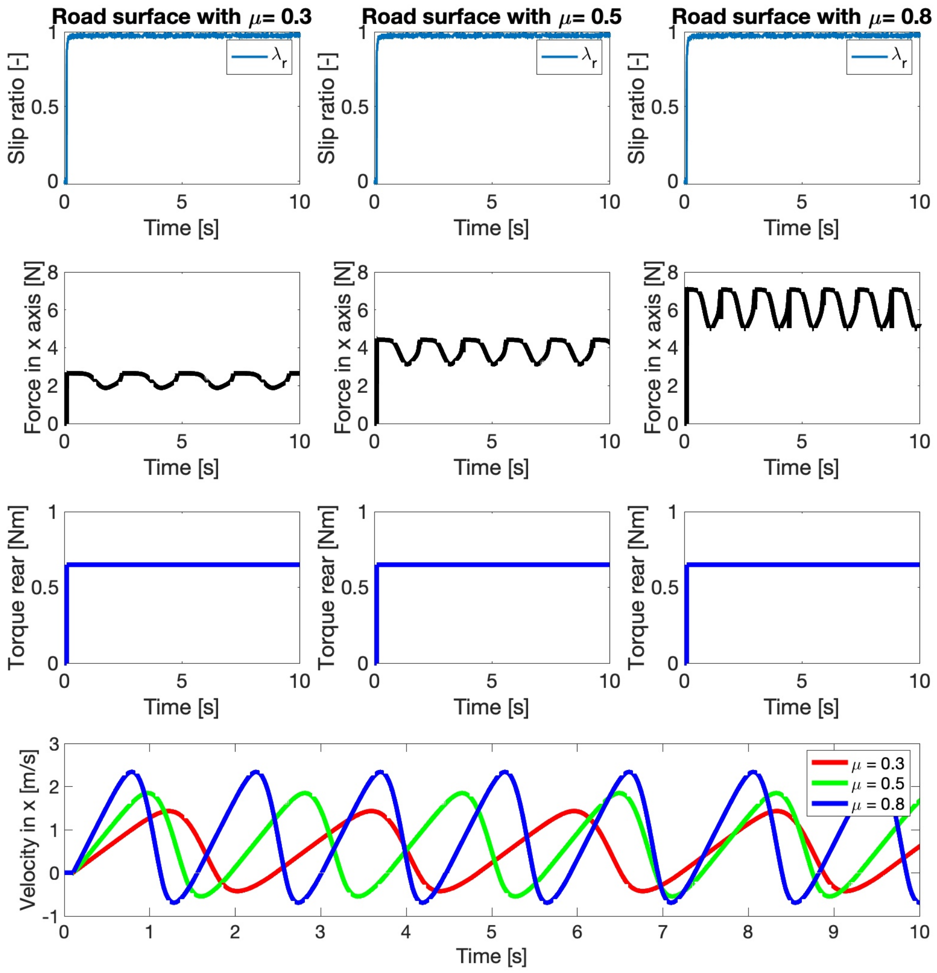


Figure 5.3: Simulation results with no controll and initial speed 0.001 m/s with $\delta_f = 10°$

Figure 5.4 shows that when the steering angle is applied the car becomes unstable and spins. As present from the position graph when the car starts to turn the yaw rate spikes and the SDP spins. Each yaw rate spike corresponds to the car spinning over. As the car spins it also loses its speed. This can be seen in the previous figure 5.3, where the speed oscillates.

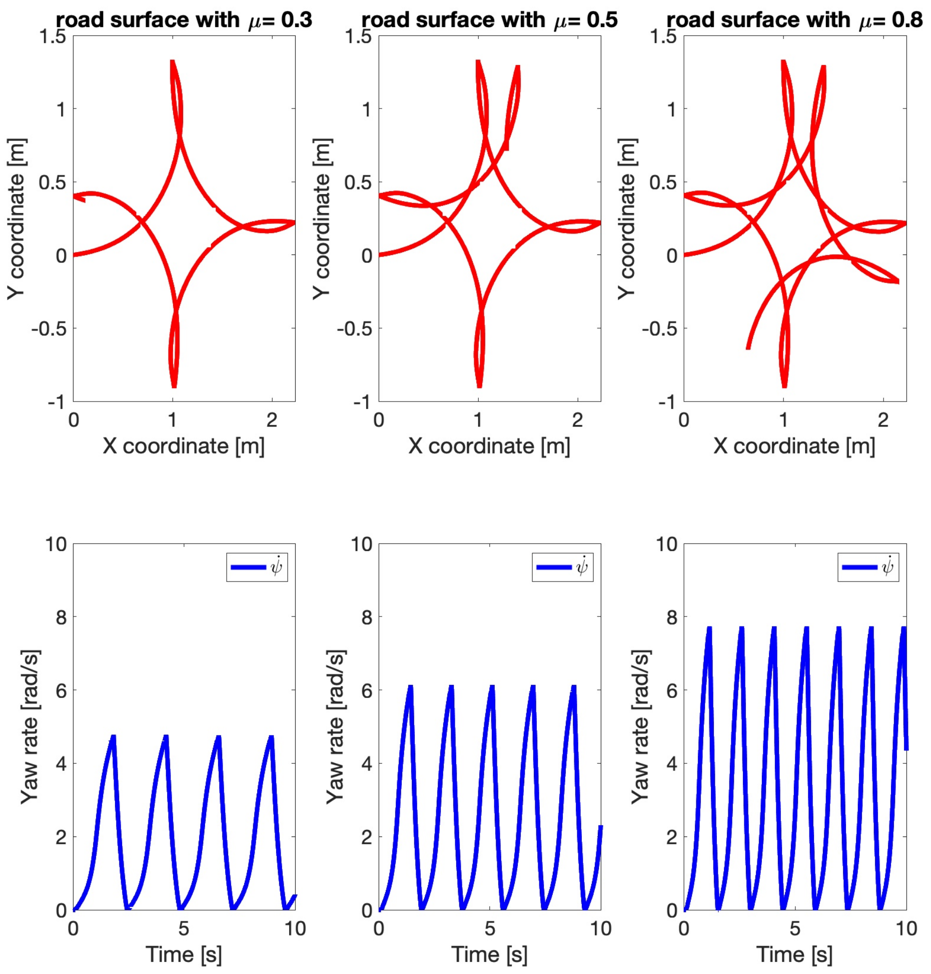


Figure 5.4: Position and yaw rate with no controll and initial speed 0.001 m/s with $\delta_f = 10°$

5.2 Logical algorithm

In figure 5.5 the results of the simulation with the logical algorithm are shown. The algorithm receives the same amount of torque requests as in the previous section. The algorithm reaches the region of slip ratio $(0.13, 0.16 >$ where the output is stable and the car accelerates without excessive tire slip. We can also see, that due to added noise, the slip-ratio in the case of $\mu = 0.3$ and $\mu = 0.5$ goes over the value 0.16 and the algorithm lowers torque output.

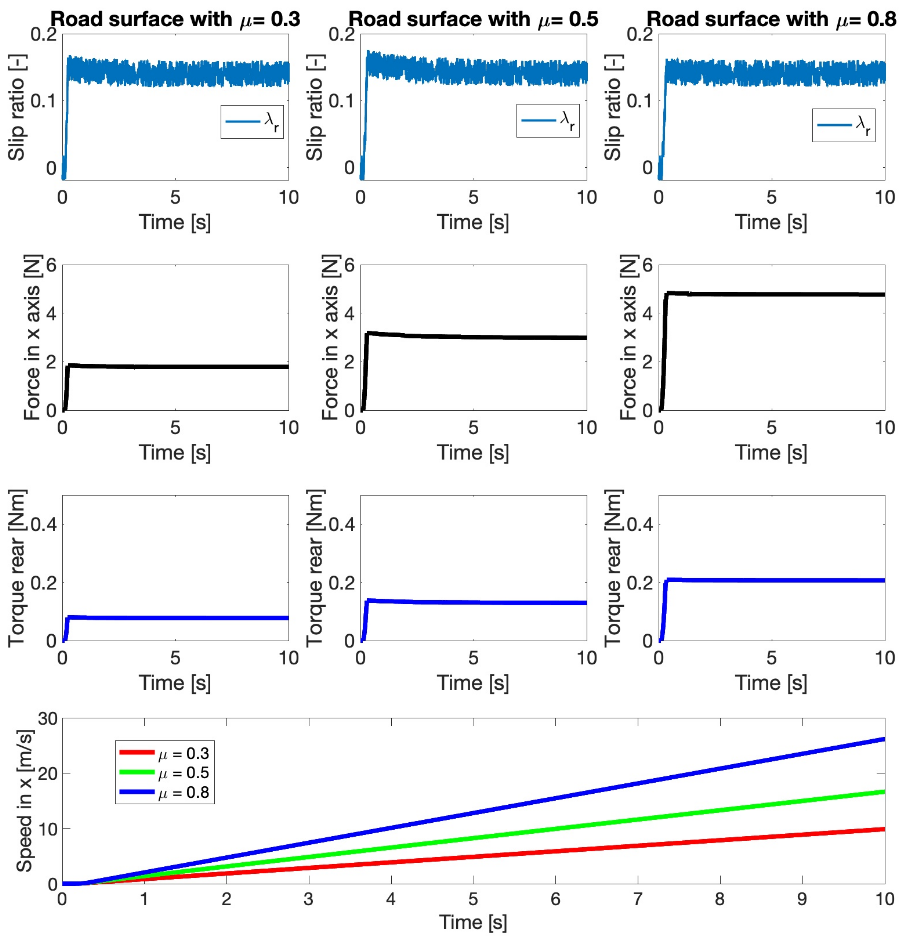


Figure 5.5: Simulation results for logical algorithm and inital speed 0.001 m/s with $\delta_f = 0°$

Figure 5.6 shows the results of a simulation where the steering angle was set to 10°. In the case of friction coefficients 0.3 and 0.5, we can see that initially the slip ratio grows and settles around a value of 0.16. However, after 2 seconds the slip ratio spikes. The algorithm reacts and lowers the output torque. In the case of $\mu = 0.8$, the algorithm settles the slip ratio in an interval $(0.13, 0.16 >$ but the speed achieved is close to zero and the car did not accelerate as low torque is applied. Spiking is also visible in F_x.

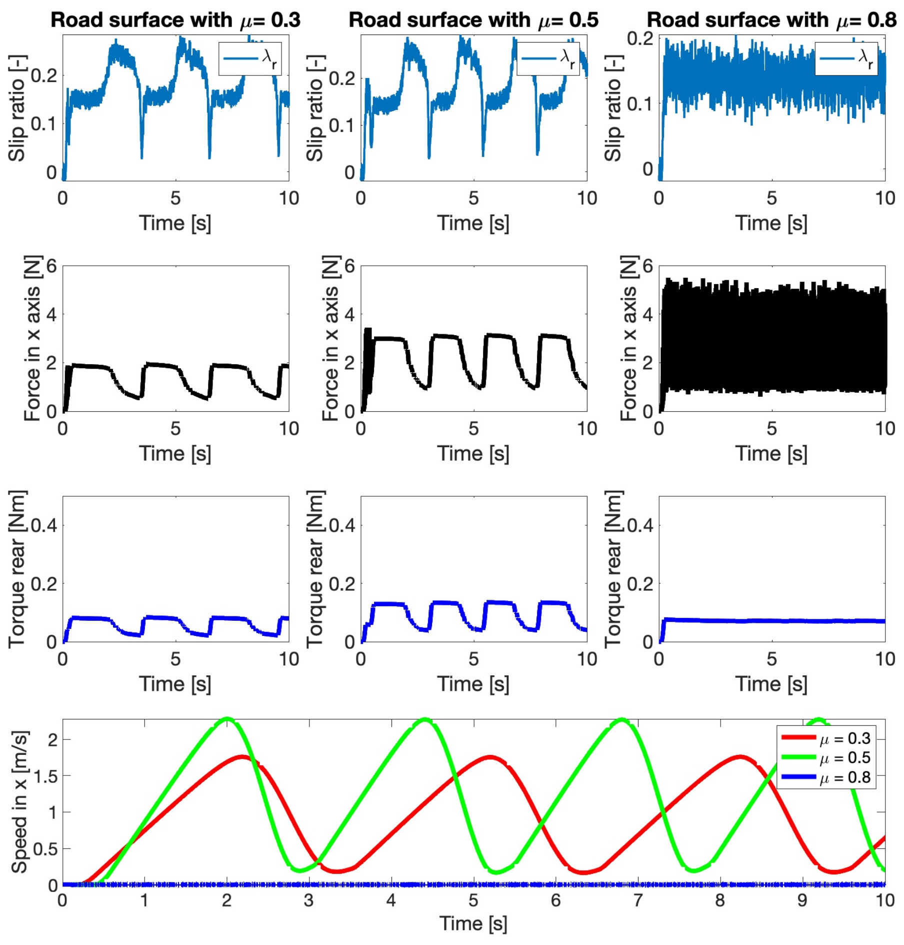


Figure 5.6: Simulation results for logical algorithm and initial speed 0.001 m/s with $\delta_f = 10°$

Figure 5.7 displays the position of the car. The car still spins, but as opposed to no control scenario, the radius of spin is grater and the number of spins is lower. The spikes in the yaw rate are lower than in the case of no control. In the case of $\mu = 0.8$, the car barely moved by a millimeter.

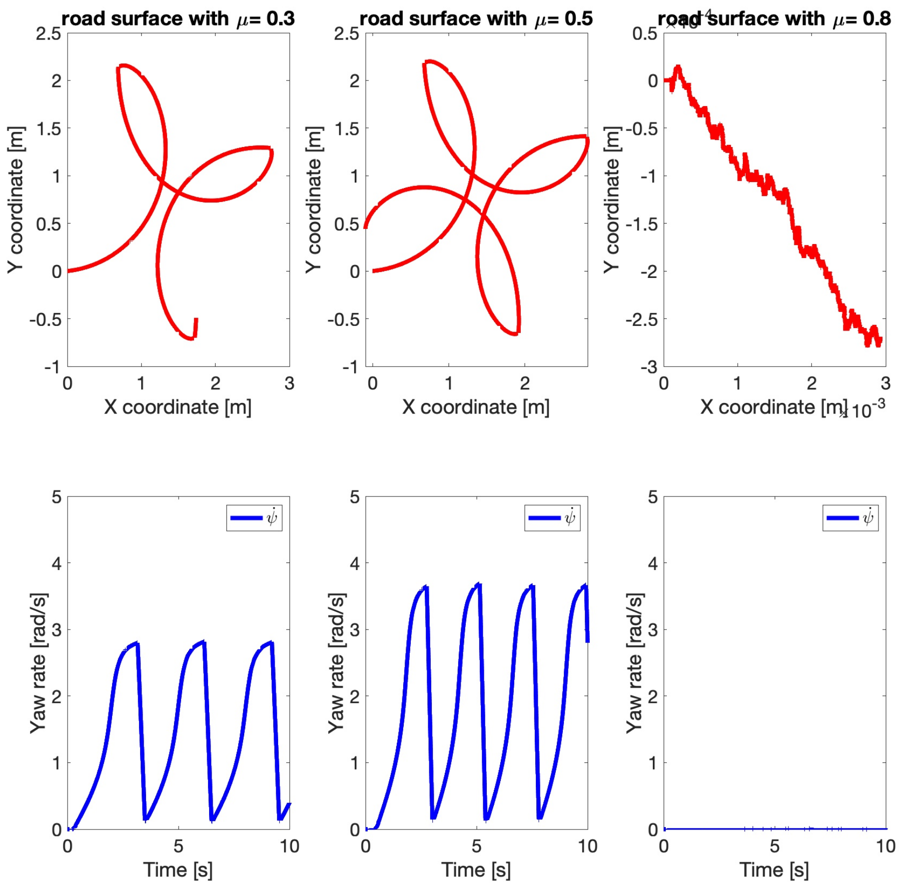


Figure 5.7: Position and yaw rate for logical algorithm and initial speed 0.001 m/s with $\delta_f = 10°$

5.3 PID control

In figure 5.8 results of the simulation with zero steering angle are shown. The acceleration controller sets slip-ratio reference to 0.2 this is the saturation level set on the controller. The slip-ratio PID controller manages to achieve to track the reference value of the slip ratio and accelerate the car. The torque output is stable and not oscillating. Once again we can see that with more grip the car can generate more longitudinal force F_x. As expected, achieved speeds are growing as the friction coefficient grows.

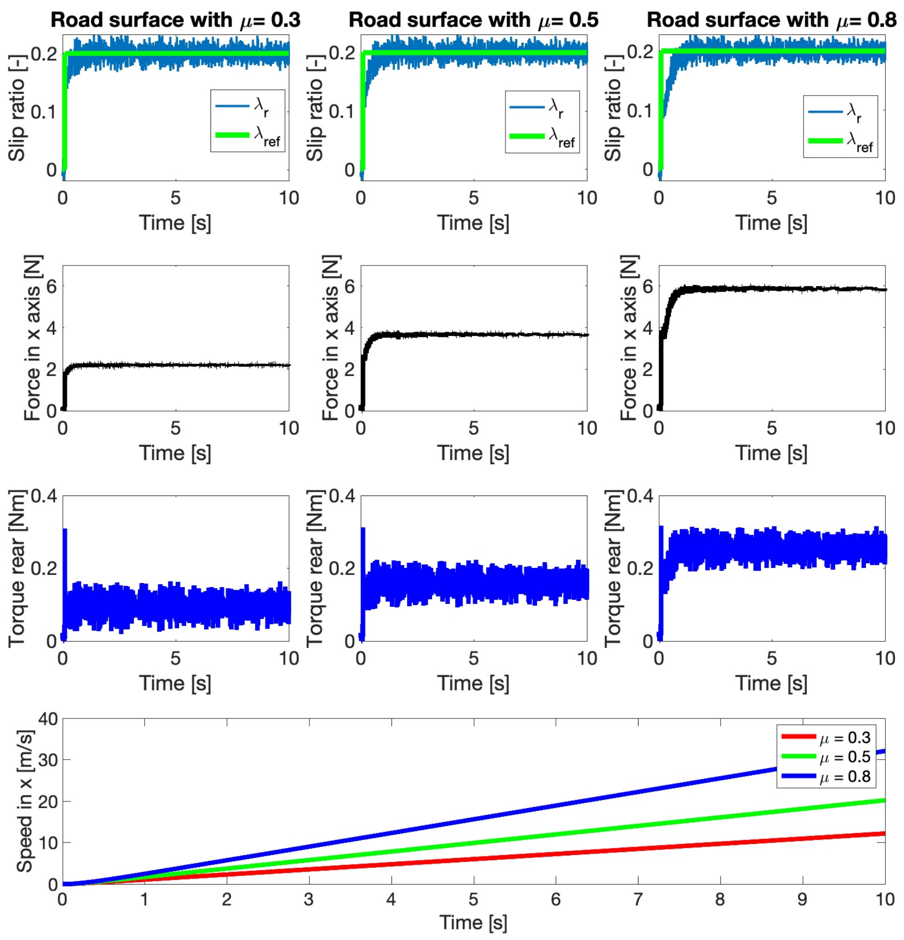


Figure 5.8: Simulation results for PID control and initial speed 0.001 m/s with $\delta_f = 0°$

Figure 5.9 shows results for the steering angle 10° is applied. We can see the intervention of the yaw rate controller. It lowers the slip-ratio reference thus lowering acceleration. The speed settles and does not grow. The settling speed is different for each μ. Any spiking of slip ratio as seen in previous results is eliminated.

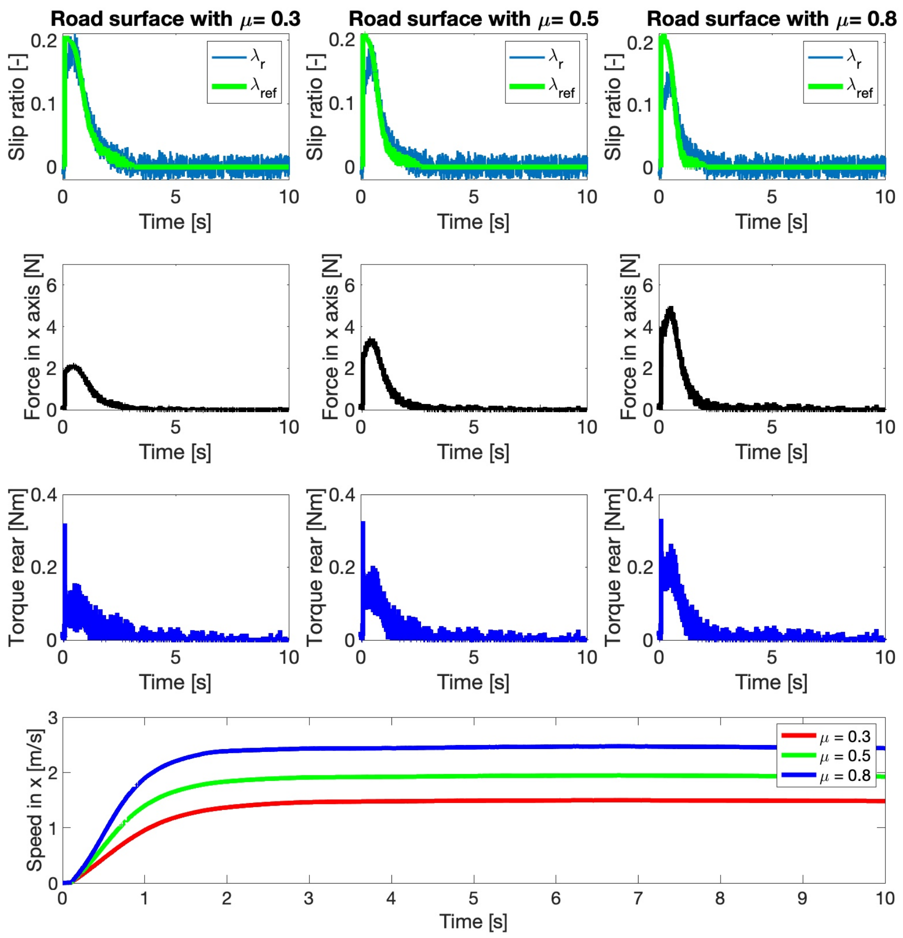


Figure 5.9: Simulation results for PID control and initial speed 0.001 m/s with $\delta_f = 10°$

The position of the car and yaw rate with its reference values are shown in figure 5.10 . We can see that the yaw rate controller can track the reference signal. By lowering acceleration the car is able to turn in a circle without spinning.

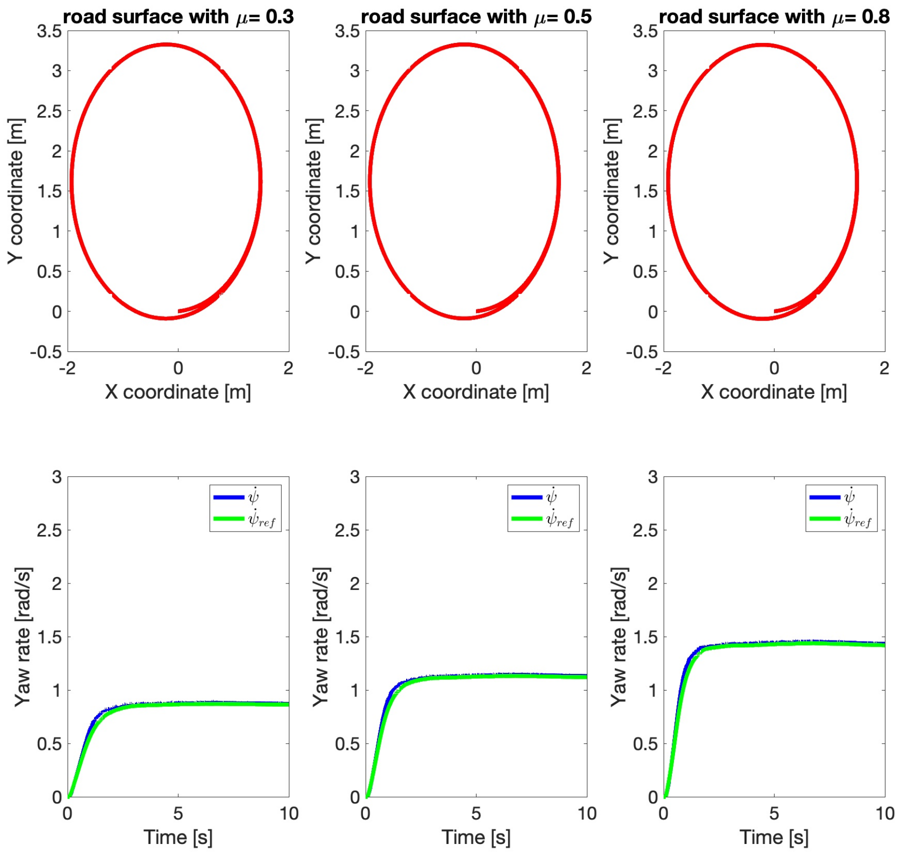
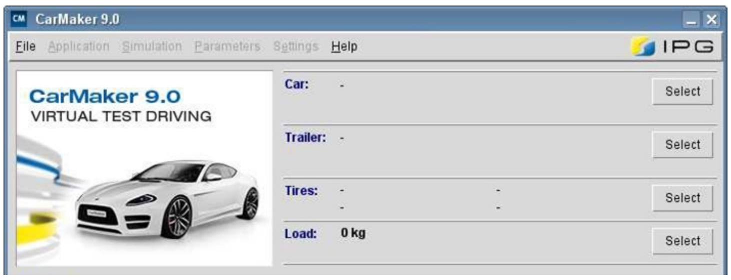


Figure 5.10: Position and yaw rate for PID control and initial speed 0.001 m/s with $\delta_f = 10°$

Chapter 6

Verification in Carmaker

As a the next step of algorithms development, I used IPG's Carmaker 11 software `https://ipg-automotive.com/en/products-solutions/software/carmaker/`. This program allows the simulation of multiple pre-defined cars with different types of tires. It offers integration with Simulink which means I do not have to recreate my algorithms and I can just copy them over. To begin testing I have used a car model of Tesla Model S with a dual motor setup. One motor for each of the two rear wheels. As a tire model, I have used predefined R16 tires with 195mm track width. I have tested both control systems proposed in Chapter 4. The control system controls the position of the gas pedal and subsequently torque on the wheels. This is different from the Single-track model where the torque was controlled directly. This difference means I cannot use the PID values I have derived from the Single-track model. However, I can still test the logical algorithm with only the scaling factor. Since the Carmaker software has a more complex model of a car and tires I can get more realistic results and thus verify created controller designs. In figure 6.1 we can see the initial interface of the Carmaker where all parameters can be set.



Car: -

Trailer: -

Tires: - -
- -

Load: 0 kg

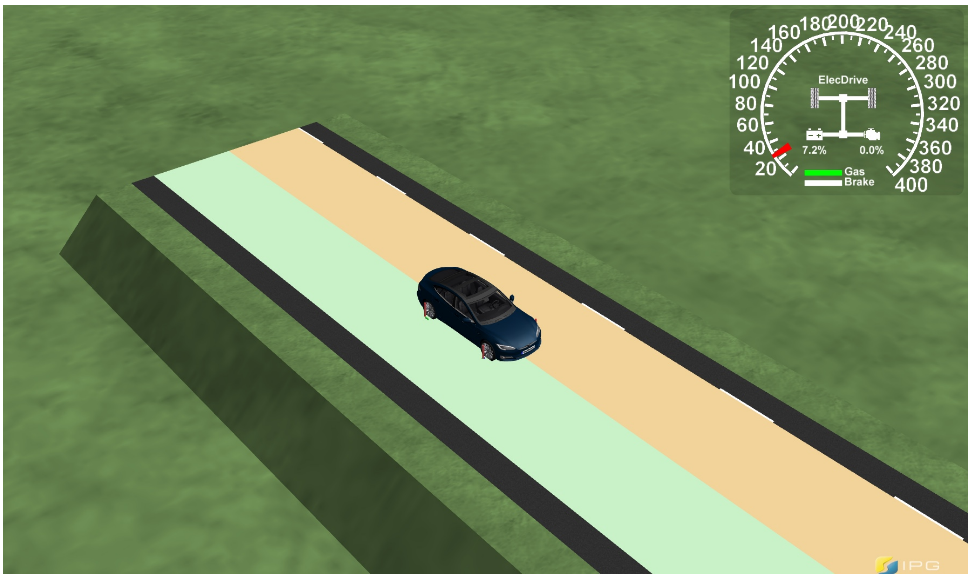
Figure 6.1: Carmaker GUI (imgae taken from `https://blog.csdn.net/xiaoming0907/article/details/125249472`)

6.1 Test scenarios

The Carmaker offers a number of driving scenarios and tracks. However, for my purposes, I decided to create my own testing tracks. Carmaker's model of a car is more complex than the single-track used in chapter 3. I can create a road with a slope allowing me to test more realistic scenarios. I can also test the split μ situation where one side of the car has wheels on a surface with a lower friction coefficient then other side. All test scenarios start with the car at a standstill. In the first second one of the simulation, the driver wants to accelerate at maximal speed. Steering angle δ_f is zero thus the car is going straight. Main scenario parameters are described in table 6.1. The advantage of the Carmaker is also its visualization interface IPGMovie. We can see split μ track in figure 6.2.

Road File	**Left** μ [-]	**Right** μ [-]	**Slope** [%]
straight_03	0.3	0.3	0.0
straight_05	0.5	0.5	0.0
straight_08	0.8	0.8	0.0
split_L07_R03	0.7	0.3	0.0
descent_03	0.3	0.3	-10.0

Table 6.1: Table of crated scenarios



.

Figure 6.2: Visualisation of a car on the split μ test track

6.2 Algortihm updates

As opposed to the Single-track model now I have two slip ratios for two wheels on the rear axel. If I was to control each wheel separately there would be problems on split μ surfaces. The difference in generated forces would cause the car to steer and deviate from the desired path. As a simple solution, I am selecting the higher value from those two slip ratios. In this way, I am limiting the force generated on the wheel with more possible grip, to the value, that the wheel on a surface with a lower friction coefficient is able to generate.

6.2.1 Logical control

In the single-track model, the maximal torque output is saturated at 0.685 Nm. But in the Carmaker I am controlling the throttle position. It is a value from interval $< 0,\ 1 >$This means, that I need to adjust the maximal output. This can be done by multiplying the output by a coefficient $k = 1.45$. in this way, the maximal output from the algorithm will be 1. Other changes were not necessary.

6.2.2 PID controller

Since I have changed car parameters. I have set new values of the PID controller experimentally. I am using only one PID controller to control the position of the throttle pedal. PID constants were obtained experimentally.

6.3 Simulation Results

6.3.1 No control

In figure 6.3 we can see the results of the simulation where no control system was used. The throttle pedal was set to 100% during the simulation. Results are similar to those from the previous chapter only for friction coefficients 0.3 and 0.5. The slip ratio spikes immediately to its maximal value of 1. This means the rear wheels are slipping. This is confirmed by the graph of rotational speeds. The angular speeds of the rear wheels are greater than the angular speeds of the front wheels. In the case of $\mu = 0.8$ the road friction is sufficient so even when full throttle is applied the rear wheel slip ratios are below 0.2 thus the car is deemed stable. We can also see that the angular speeds of wheels are similar on every wheel.

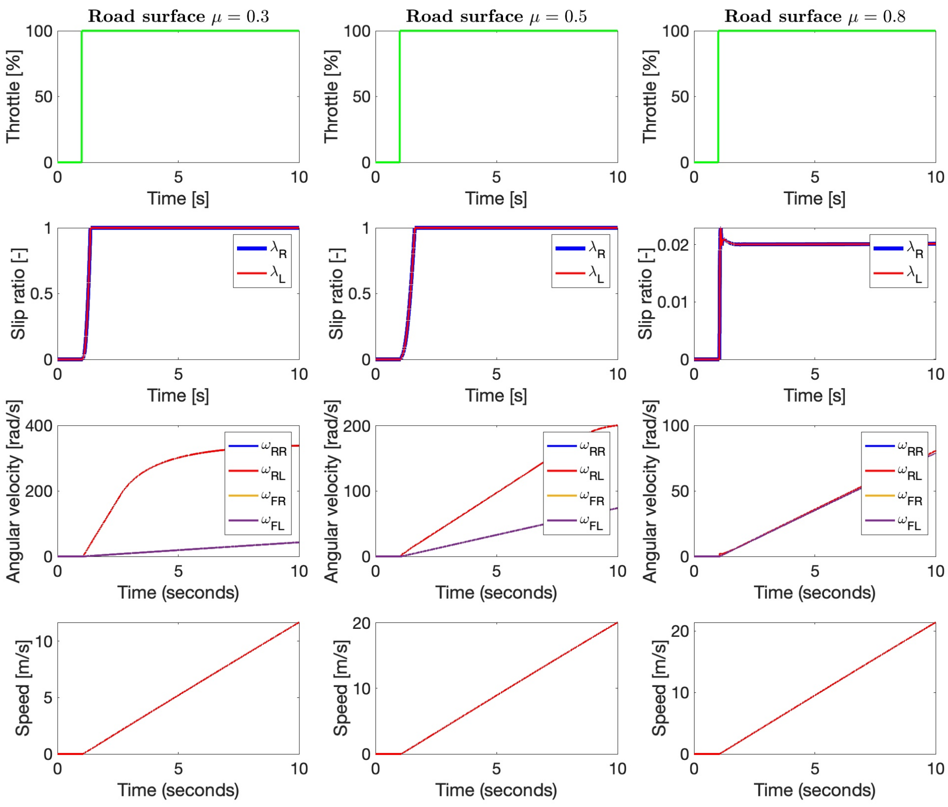


Figure 6.3: Simulation results no control and initial speed 0.0 m/s

In figure 6.4 we have results of simulations on split μ surface and on 10% descent. In the case of the split μ surface, the slip ratio of only one wheel spiked up. On descent the car still reaches maximal slip ratio.

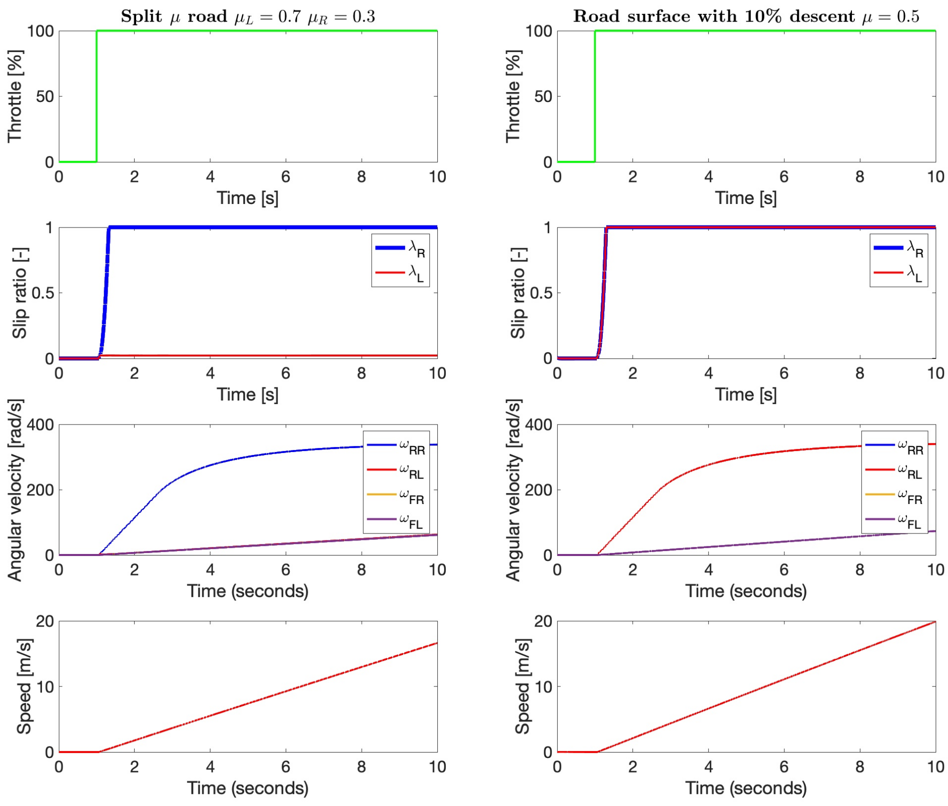


Figure 6.4: Simulation results no control and initial speed 0.0 m/s on split μ surface and on descent

6.3.2 Logical algorithm

Figure 6.5 shows that the designed logical algorithm is preventing tire slip. However, spikes in slip-ratio are visible. Those can be eliminated, or the amplitude can be lowered, by lowering the value s thus lowering the amount of throttle added to each iteration of algorithm. As downside of this approach the ability to efficiently use available grip will be lowered. We can see this also in figure 6.5 in the scenario where road surface has $\mu = 0.8$. The slip-ratios stay well below 0.09 therefore maximal acceleration is requested. There is a noticeable time difference before the maximal throttle request is reached. If we compare that to no control scenario shown in figure 6.3 we can see that the speed achieved at the same time is higher in case of no control.

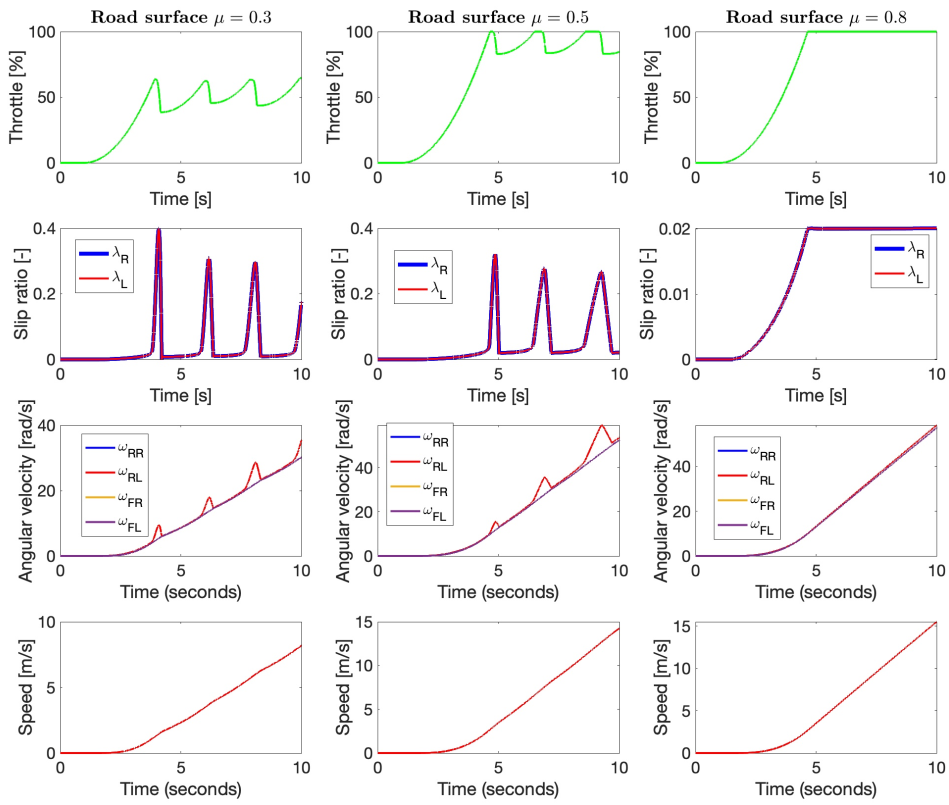


Figure 6.5: Simulation results for logical algorithm and initial speed 0.0 m/s

In figure 6.6 we can see that in the case of split μ test the logical algorithm is reacting to spikes in slip ratio. We can also see that the aplitude of spikes is getting smaller. This is also true for case of descent test. We can see oscialtion on the gaspedal that would be very unplesant to any of the car passengers.

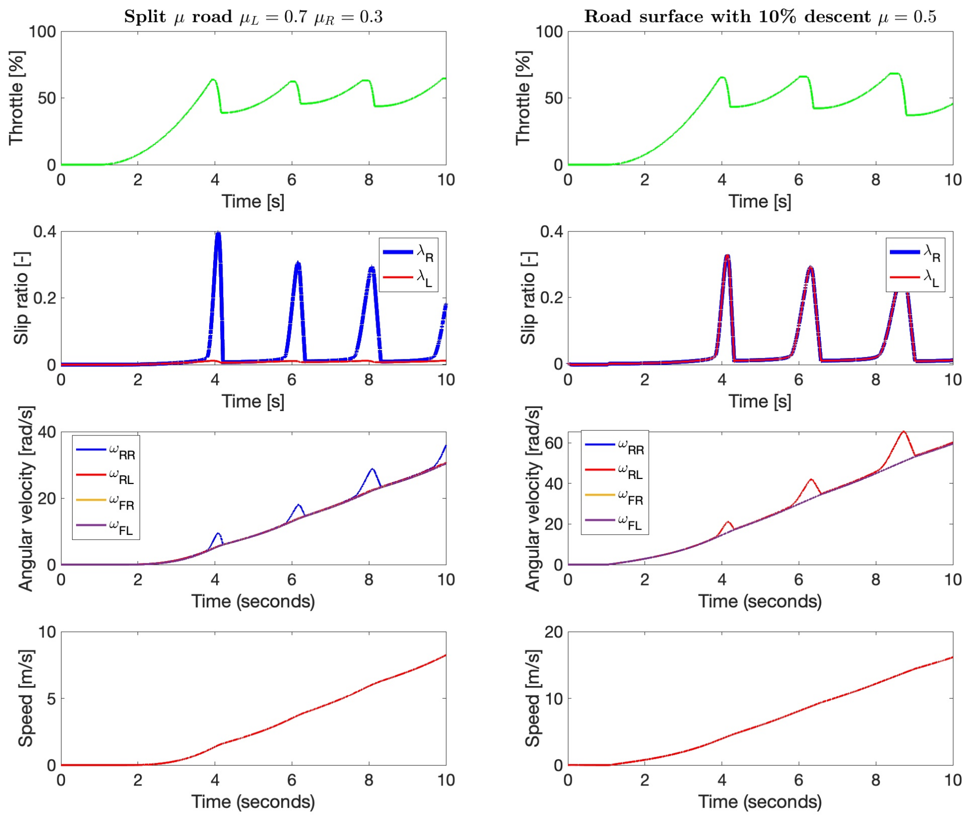


Figure 6.6: Position and yaw rate for logical algorithm and initial speed 0.0 m/s on split μ surface and on descent

6.3.3 PID control

PID controller has its reference set to 0.2. It manages to start the vehicle. Figure 6.7 shows that in contrast to logical algorithm the PID demands maximal available throttle instantly in case of $\mu = 0.8$. This means faster acceleration and better agility during the start up. In figure 6.8 we can see more slip on the right wheel as its the wheel with lower friction coefficient $\mu = 0.3$. The method of choosing higher slip-ratio works as its limiting the maximal force on the wheel with higher friction. But also maximizes the available grip for the slipper wheel.

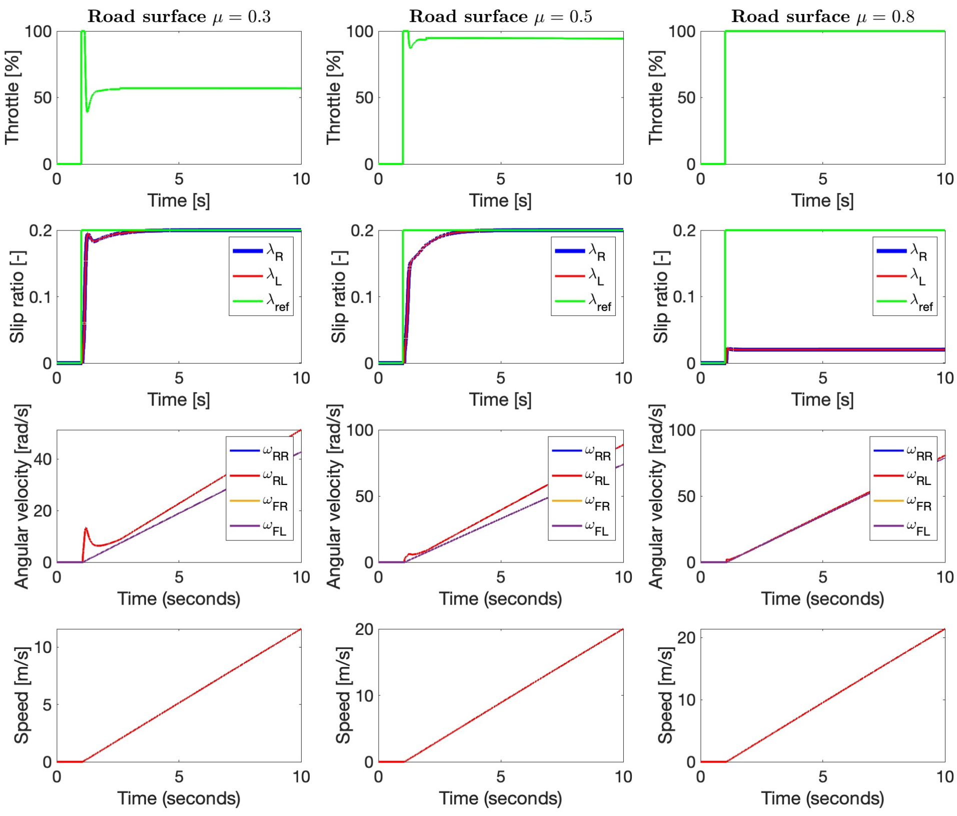


Figure 6.7: Simulation results for PID control with initial speed 0.0 m/s

Here in figure 6.8 are results of Carmaker simulation. In case of split μ there is intial wheel slip but then the PID controller reacts and lowers throttle apllied. For rest of the simulation the slip reatio tracks refence value. In case of descent we once again see that PID controller accelerates faster then logical control and manges to track the reference value. The PID overshoots in both cases.

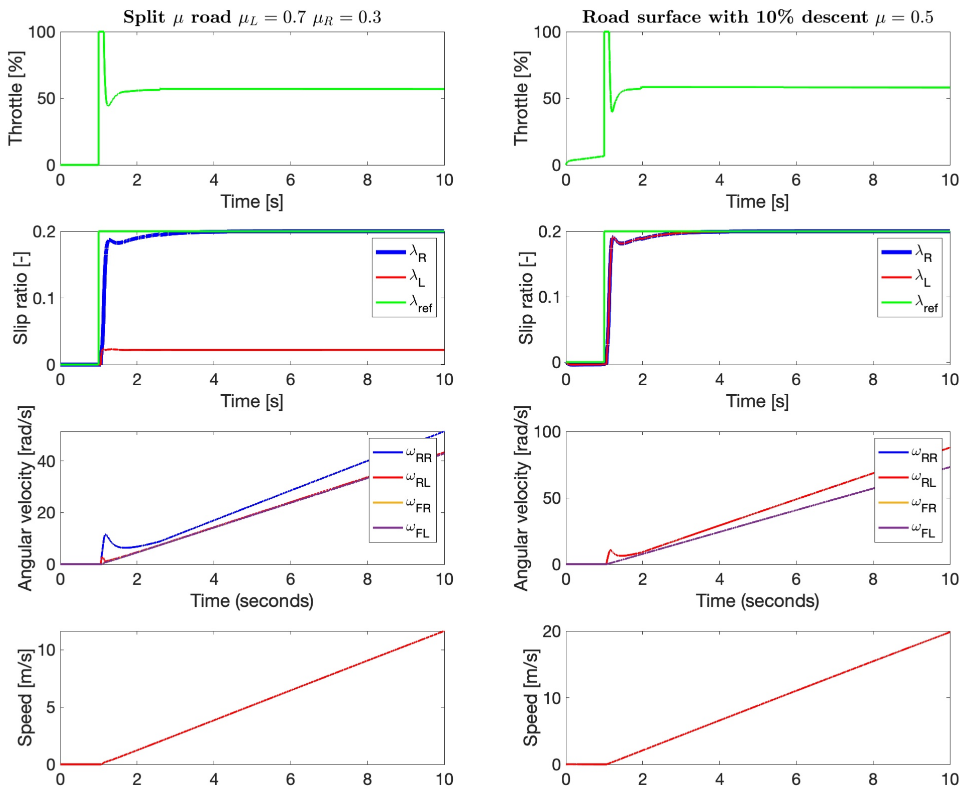


Figure 6.8: Simulation results for PID control with initial speed 0.0 m/s on split μ surface and on descent

Chapter 7

Scaled-down platform

Algorithms were tested on a scaled-down platform outside of simulations The SDP used was built as a master thesis by Ing. Hostačný. It is 1:10 RC car model with all-wheel drive capability, however the front differential is used with a combination of Hall effect sensor for contact-based measuring of speed. Each rear wheel of SDP is powered independently by an electric motor connected by belt and pulley to half-shafts. Two VESC controllers are used to drive the motors. 96Boards Mezzanine board with STM32F446 MCU

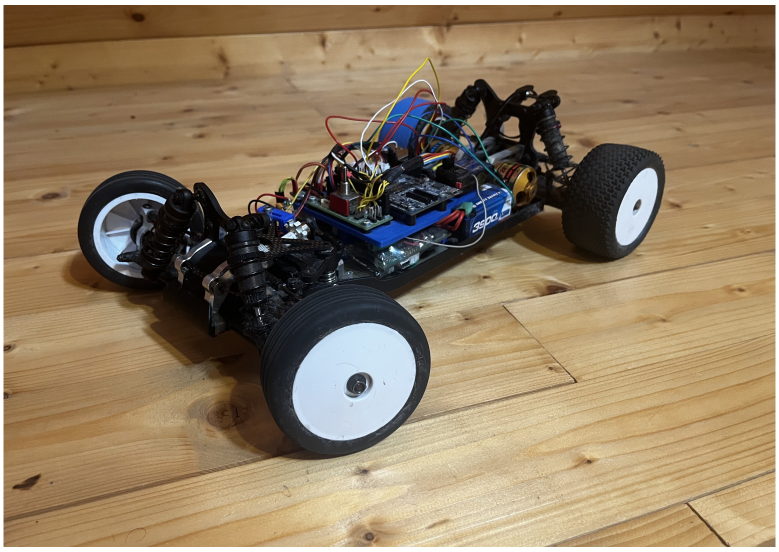


Figure 7.1: Scaled-down platform

is used as central control board. It is higher performance board with 180Hz ARM-based processor. The board proceses signals from VESC controllers received via CAN bus. Interprets signals received from radio control and generates PWM signal for steering servo motor. It also receives IMU data. Block diagram of individual peripherals and type of their connections is shown in figure 7.2. Implementation of control algorithms was to be done in Simulink template, where code for the F446 controller was also generated.

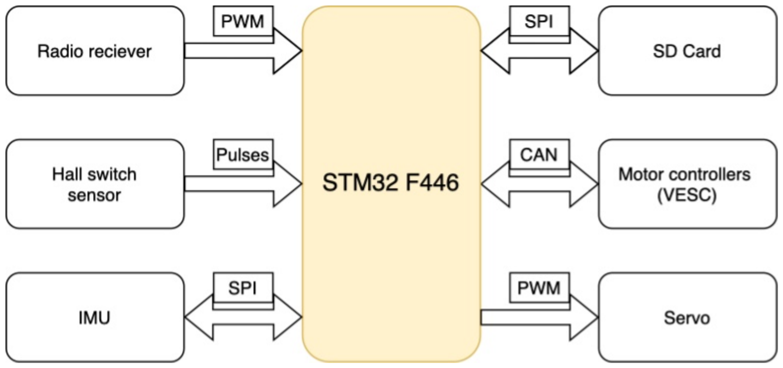


Figure 7.2: Block diagram of connections

7.1 Hardware modification of SDP

When I received the SDP, it was without its GPS receiver. This means, that accurate position tracking is not possible. However, my focus is on low speeds thus it is not necessary. Also, as the GPS antennas were mounted on the top of the chassis and were quite bulky by their removal has lowered the overall weight. COG is also lower. This in general helps maneuverability and acceleration.

7.1.1 Data logging

Data logging was handled by Bluetooth module HC-06, which received data via UART. I had complications with connecting to the module. I replaced it with a SD card module connected via SPI. We can see it mounted on the SDP in figure 7.3. This means that there is no on-line data logging. However, using SD card for data logging is reliable and numeric formats larger than 8 bits(limit of data bits sent over UART) can be stored without additional parsing. Each test that is running on the platform is stored in a plain text file. Afterwards I have created MATLAB scripts for displaying various measured variables. Most of them display essential variables such as driver input, front and rear wheel speeds, current supplied to motors, and calculated slip ratios. Additionally, there is a dedicated script to visualize the IMU output, followed by a separate debugging script specifically designed for testing and fine-tuning my control algorithms.

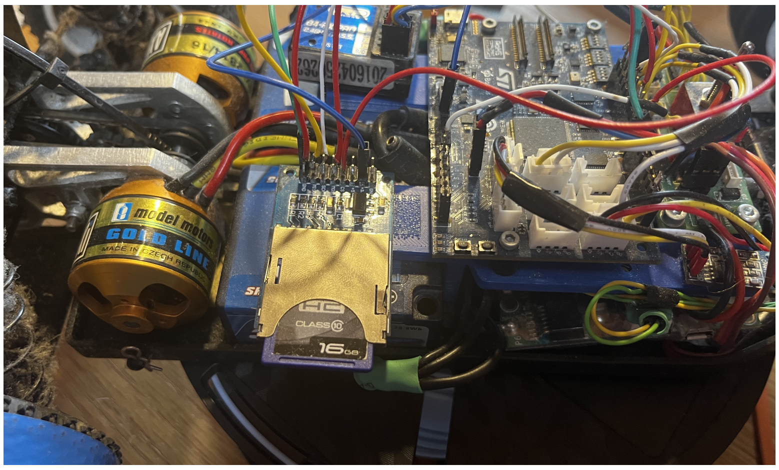


Figure 7.3: SD card used for data logging mounted on the SDP

7.2 Software modification of SDP

The intended way of implementing and testing new control algorithms on SDP was to firstly use created template file for Simulink to implement the algorithms. Then use toolbox for C code generation and STM32-MAT/TARGET to generate code. Generated C code was then supposed to be compiled for target board F446 and uploaded using one of the supported IDEs. I was also having issues with Simulink template not recognizing appended C functions and subsequent code generation. I was not able to compile the generated code using neither KEIL IDE nor TrueSTUDIO. Another issue was that STM32-MAT/TARGET is supported only on Windows operating systems. Since the functions communicating with peripherals were written in C I decided to implement my control algorithms in C as well and avoid using code generation via Simulink. Since I do not use GPS and UART data logging I do not need to include those functions. Implementation in C also gives me more control over CPU resources and the ability to add additional features ie. SD card data log. It also enables different sampling rates other than 1kHz. Table with all signals is in the apendix.

7.2.1 Code generation without Simulink

New software process is shown in figure 7.4. STM32CUBEMX allows to generate code needed to setup the ARM processor assigning GPIO pins, setting up timers and interrupts. STM32IDE is then used to implement control algorithm and for upload to the central board. It also allows for step-by-step debugging via USB and resource monitoring.

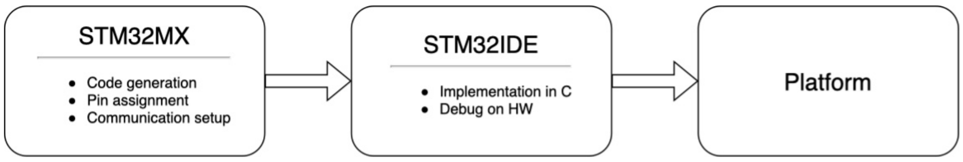


Figure 7.4: Block diagram of software creation process

7.3 CAN-VESC controllers

VESC controllers are open-source electric motor controllers. They communicate with the control board via CAN bus. They use extended frame format, which allows up to 256 controllers with specific IDs. This can be configured via provided VESC Tool software.

7.3.1 Sending comands

I have used previously defined IDs of motors. Left has ID=0x00 and right has ID=0x01. Several parameters can be set. As seen in table 7.1 there are three main ways to control the motor.

Message type	**Extended ID**
Set duty cycle	0x0+ID
Set current	0x1+ID
Set RPM	0x3+ID

Table 7.1: CAN bus extended ID

Controlling duty cycle, controls voltage and consequently the speed of the motor. It does not provide torque control nor does it sustains speed under heavy loads.

Controlling RPM uses a built-in PID controller to sustain set speed even under heavy loads but again it does not allow for torque control.

Controlling current provides torque control but does not provide speed control. Values of current are sent in mA. As I am controlling torque I am using this message type to send commands to VESC controllers. For example if I want to set left motor to $2.5A = 2500mA$ the extended ID would be ID = 0x11 and the first four bytes of message would be MSG = 0x09C4.

7.3.2 Recieveing information

Each VESc controller is configured to send informations about ERPM, current and duty cycle. The functions to phrase CAN messages were used from Ing. Hostačný. After receiving, the current signal is filtered by a bi-quad filter with a cut-off frequency of 20Hz.

7.4 Radio-PWM-Timer

Driver sends commands of throttle and steering wirelessly to the receiver. The receiver interprets the signal and generates two PWMs. It uses 50Hz frequency with an active state ranging from $1ms$ to $2ms$. This is the same for throttle and steering. For throttlem no input is at $1.5ms$. Maximum breaking is at $1ms$ and maximum throttle is at $2ms$.

As mentioned at Ing. Hostačný thesis, current radio set has a quantization problem. To prevent unwanted interpretation of throttle input, dead zones were implemented. The interval for zero input has been set to $1.47 - 1.53$ ms. As another safety feature, central board periodically checks, if it is receiving signal from the radio. If PWM Input capture-compare register overflows I set the recieved data to be invalid. Then command is sent to VESC controllers to break and stop.

7.4.1 Mapping throttle

One of the possible solutions to limit the amount of torque applied is to change the shape of the throttle curve. As I implemented the acceleration controller in chapter 4.2 I can map throttle position to its reference value. A possible solution is to linearly map acceleration to the thrrottle position. Another way is to use some function to have a faster or slower acceleration with a lower throttle position. In my opinion, this would not be comfortable to drive as when the driver wants to hold the current speed it is not possible. When the throttle is released the car will coast and when the throttle is applied minimal acceleration will be present. This can be eliminated by setting the first 10% of the throttle to set acceleration reference to 0 ms^{-1}. Possible throttle curves are shown in figure 7.5.

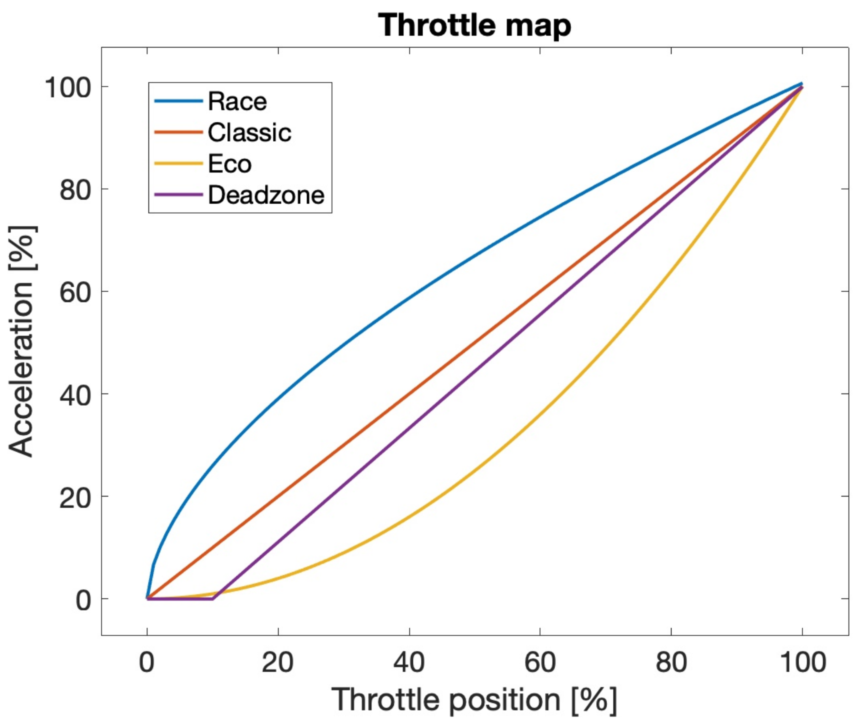


Figure 7.5: Graph of possible throttle curves

7.5 Contact based speed measurement

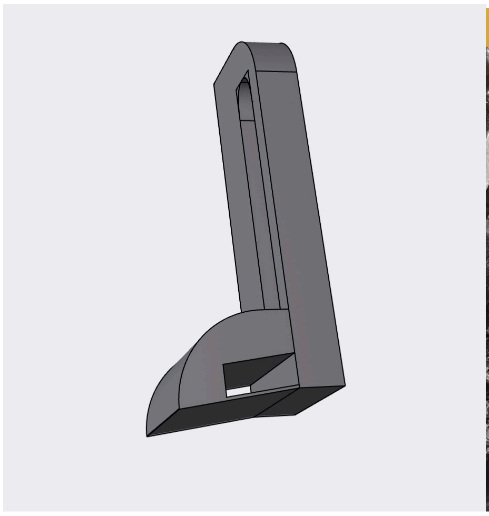
The SDP uses a contact-based solution to determine its speed [18]. It is a Hall switch (TLE4905L). According to the datasheet [6] it was developed for the automotive industry. The output of the switch is digital as it has an internal comparator with hysteresis that compares the analog output from Hall sensor in the IC. The measuring system consists of a gear mounted on the front differential. The gear has 8 teeth. In combination with the differential ratio of 2.6, the switch should provide 46 impulses per one front wheel revolution. To capture the states of the switch the timer with a resolution of $100\mu s$ is used. Timer-sensing the signal from the switch is noise sensitive. Manufacturing of the wheel is not ideal. To solve this inadequacy only every fourth impulse is captured. Mounting of the gear to the differential is also not ideal as it has an end play.

In my thesis, I am most interested in behaviors at low speeds coming to zero. In this region the sensor poses two major problems. It is not precise and fast enough to detect small changes of speed from standstill. It is also noisy with spikes as high as $0.7ms^{-1}$. Unfiltered signal can be seen 7.7.

7.5.1 Hall switch position adjustment

As said before, the signal from the Hall switch has spikes. Before I tried applying any software filtering I tried adjusting the sensor itself. It is placed very close to the magnetic gear. I have widened the gap between the sensor and the wheel. This removed most of the spikes from signal. However, after further tests on uneven terrain such as tarmac the vibrations from road caused slight sensor movement and additional signal spikes came back.

The sensor is soldered on a prototype board which is fastened by screws to the car chassis. Before designing new bracket to hold the sensor, it had to meet some requirements. It has to be adjustable because changes around just 0.5 mm are noticeable. As said in Ing. Hostačný's thesis the probe can be damaged at higher speeds. I want to design some level of protection for the sensor. In figure 7.6a we can see the final version of the bracket. There is a slot where the sensor can be slid in. In Figure 7.6b we can see the sensor in the bracket and mounted on SDP. It meets the two requirements well. It is adjustable so the position can be fine-tuned on the SDP. Since the sensor slides into the bracket it is protected from the spinning wheel. Figure 7.7 shows measured raw data from front axel. In figure 7.8 we can see that most of the spikes were eliminated just by adjusting the mounting position. Still some spiking was present and additional filtering was required.


(a) : Mounting bracket in modeling software

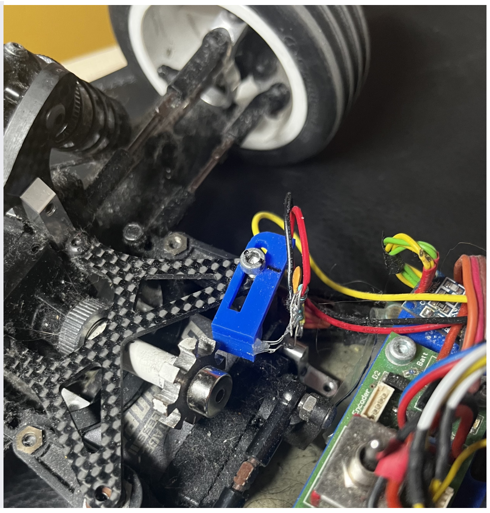

(b) : Hall switch in new bracket on SDP

Figure 7.6: Hall switch mounting

7.5.2 Spike filter

To filter out spikes from signal I have designed spike filter that uses data from accelerometer. I can compute the change of speed as:

$$\Delta v = acc_{ms} \cdot \Delta t, \tag{7.1}$$

where acc_{ms} is acceleration from accelerometer in $[ms^{-2}]$ and Δt is time since last sample. Ideally, the change of speed read by the Hall switch should be the same. In reality, I will leave some offset to account for tire slip and sensor error. I set the offset to 40% of the Δv. this creates envelope in which I expect the sample from front differential to be. If not, the sample is ignored and the last value is used. Additionally when the Δv is positive I do not expect the speed to decrease. If the newest sample has lower value than previous sample, the value is also ignored.

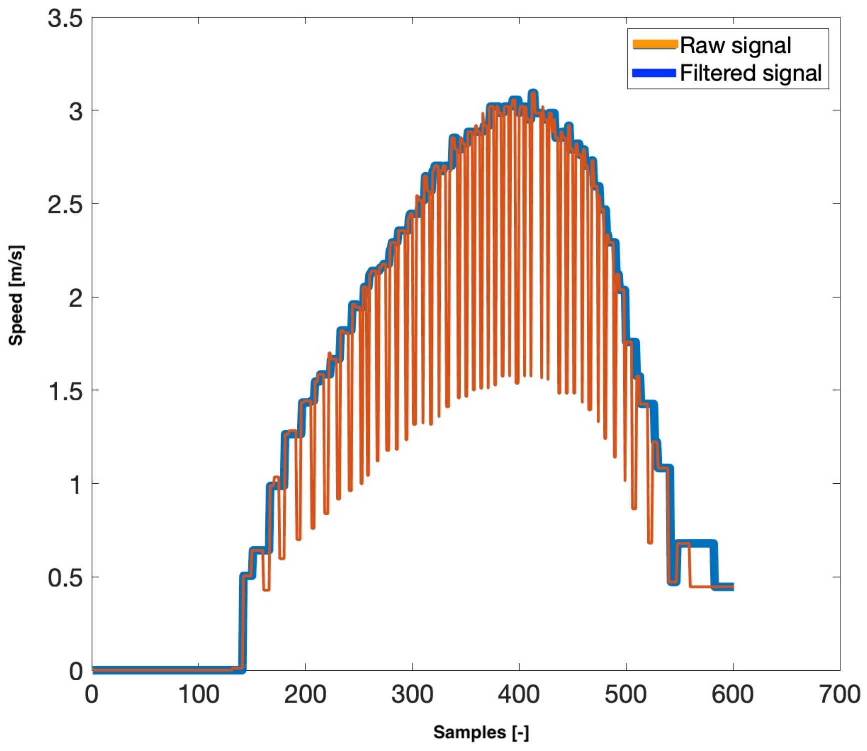


Figure 7.7: Raw and filtered signal

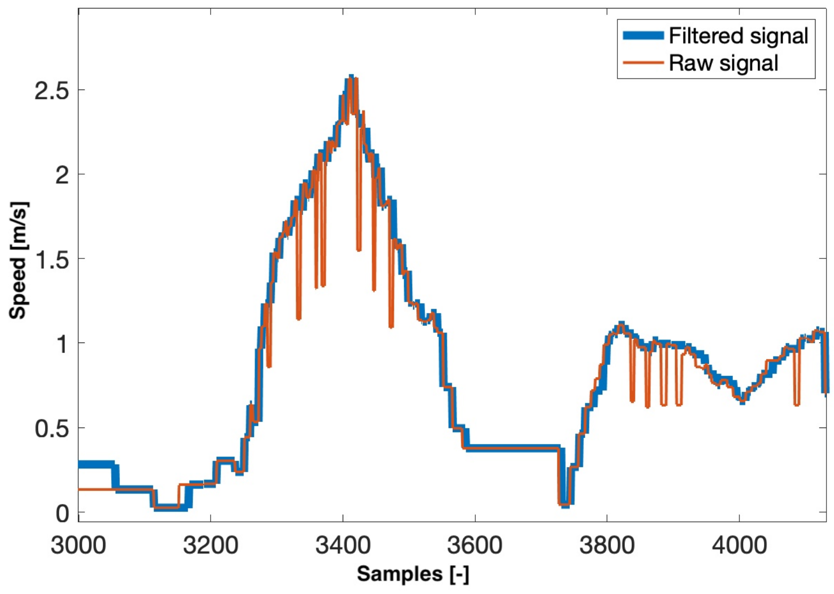


Figure 7.8: Raw and filtered signal after Hall switch position adjusment

7.6 Accelerometr based speed measurement

As said earlier, control board has MEMS 9DOF sensor, which can measure accelerations with ranges up to ±16g. The Linear acceleration measurement range of the LSM6DS3H sensor is set to $\pm 4g$. Update rate is set to maximal for accelerometer of $6.66kHz$. Sensor provides 16 bits of data in two's complement. To get value in ms^{-2} the output needs to be calculated using:

$$acc_{ms} = \frac{k_{acc} \cdot g}{1000} = \frac{0.122 \cdot 9.81}{1000}, \tag{7.2}$$

where the k_{acc} is coefficient provided by the datasheet and g is gravitational acceleration. I used the value of $g = 9.81\ ms^{-2}$ which corresponds to g meassured value in Prague. To get the value of speed I need to integrate the values of acceleration over time. Since the signal is discrete and I am using counter to measure number of cycles between each reading I can compute the speed v_{acc} as

$$v_{acc} = \sum acc_{ms}[n] * \Delta t[n], \tag{7.3}$$

where Δt is time between readings and can be computed as

$$\Delta t = \frac{n_{counter}}{f_{Timer}}, \tag{7.4}$$

where $n_{counter}$ is number of ticks counted by counter with frequency f_{Timer}. For my purpos, I choose $f_{Timer} = 2MHz$. I used 16bit counter that overflows every $0,0327\ s$. This is sufficient since I am sampling every $0.00085\ s$.

Accelerometer drift

To obtain valid accelerometr data it is necessary to calibrate it. The accelerometr data can be represented as

$$acc_{ms} = acc_{real} + acc_{error}. \tag{7.5}$$

If I put 7.5 to 7.3 the resulting value of speed will integrate the error over time and will be extremely inaccurate. To solve this problem I implemented calibration proccedure. The procedure run at start of each ride and assumes that SDP is not moving. It then takes 400 samples with sampling frequency 100Hz. The offset is computed as a mean value of taken samples.

$$acc_{error} = \frac{\sum_{n=1}^{400} acc_{ms}}{400}. \tag{7.6}$$

In figure 7.9 we can clearly see the presence of an offset in an unfiltered signal. During this experiment the car was stationary. We can also see when calibration was done, the offset was eliminated. This method sufficiently eliminates signal drift for it to be used in a complementary filter as can be seen in figure 7.11. The signal is more accurate, but when the vehicle comes to a stop, the integrated error is still present and the value is 0.4 m/s greater than it should be.

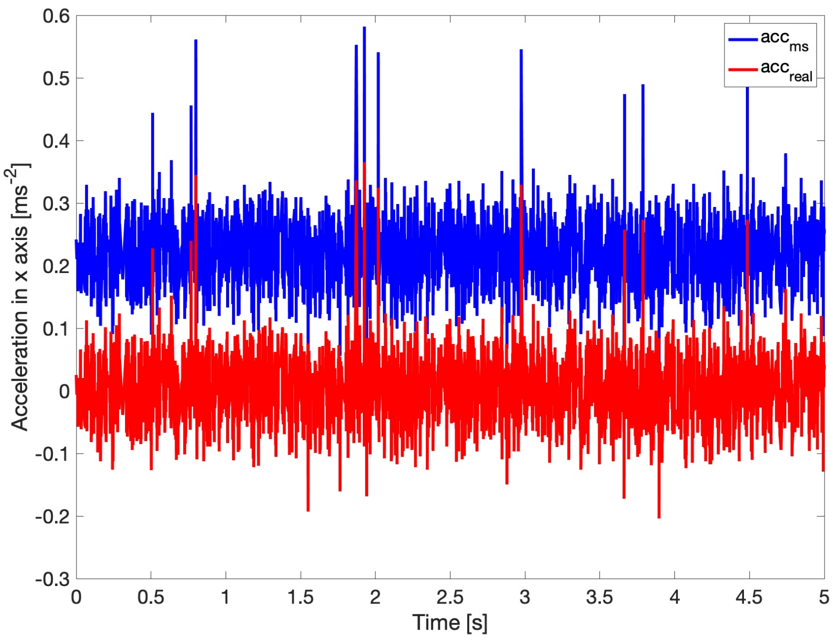


Figure 7.9: Speed values from accelerometer when the SDP is stationary

7.7 Complementary filter

To fuse signals from the accelerometer and Hall switch I have decided to implement the complementary filter. The idea of a complementary filter is shown at 7.10. I have signal from Hall switch that is stable over time but has spikes and the signal from accelerometer that is accurate over a short period of time but drifts over time. I can use a high pass filter with cut-off frequency f_c to filter out the drift and a low pass filter with the same frequency f_c to filter out the spikes from contact-based speed measuring as well as eliminate the jump in value upon start.

Integrated signal from accelerometer is shown in figure 7.9. From the graph, I have determined that in the first 0.5 s of the signal is the drift negligible.

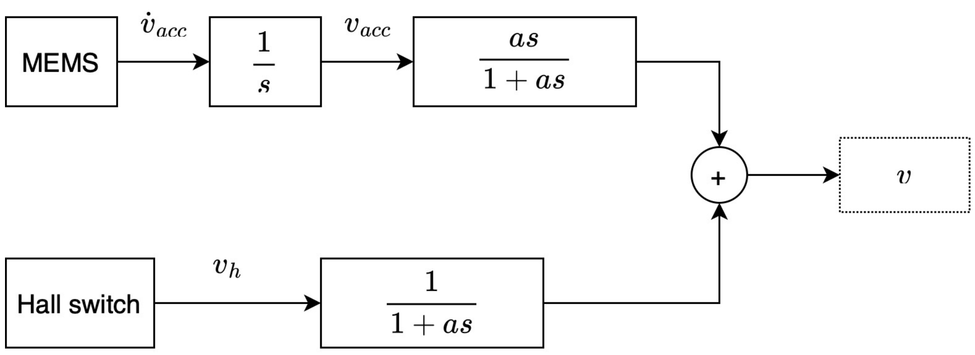


Figure 7.10: Block diagram of complementary filter

That gives me a cutoff frequency of $f_c = 2Hz$.

The transfer function of low pass filter can be written as:

$$L(s) = \frac{1}{1+as} \tag{7.7}$$

The transfer function of high pass filter can be written as:

$$H(s) = \frac{as}{1+as}, \tag{7.8}$$

where $a = \frac{1}{\omega} = \frac{1}{2\pi f_c}$. Final transfer can be:

$$v = \frac{1}{s}(\frac{as}{1+as}) \cdot \dot{v}_{acc} + (\frac{1}{1+as}) \cdot v_h \tag{7.9}$$

that is equivalent to:

$$(1+as)v = v_h + a \cdot \dot{v}_{acc} \tag{7.10}$$

Now I can use inverse Laplace transformation to get time domain function:

$$v(t) + \frac{dv(t)}{dt} = v_h(t) + a \cdot \dot{v}_{acc} \tag{7.11}$$

Next step is discretization using backward Euler method where derivative can be approximated:

$$\frac{dv(t)}{dt} \approxeq \frac{v[n] - v[n-1]}{T}, \tag{7.12}$$

where T is sampling time [s]. Using 7.12 I get descrete time version of 7.11

$$v[n] + a(\frac{v[n] - v[n-1]}{T} = v_h[n] + a \cdot \dot{v}_{acc}[n] \tag{7.13}$$

We can adjust the equation to its final form.

$$v[n] = \frac{a}{a+T}v[n-1] + \frac{T}{a+T}v_n[n] + \frac{aT}{a+T}\dot{v}_{acc}[n] \tag{7.14}$$

This equation can be easily implemented in software. Final comparison is in figure 7.11. We can see benefits of complementary filter. Filter removes the rest of spikes and smooths out the steps in signal from front wheels.

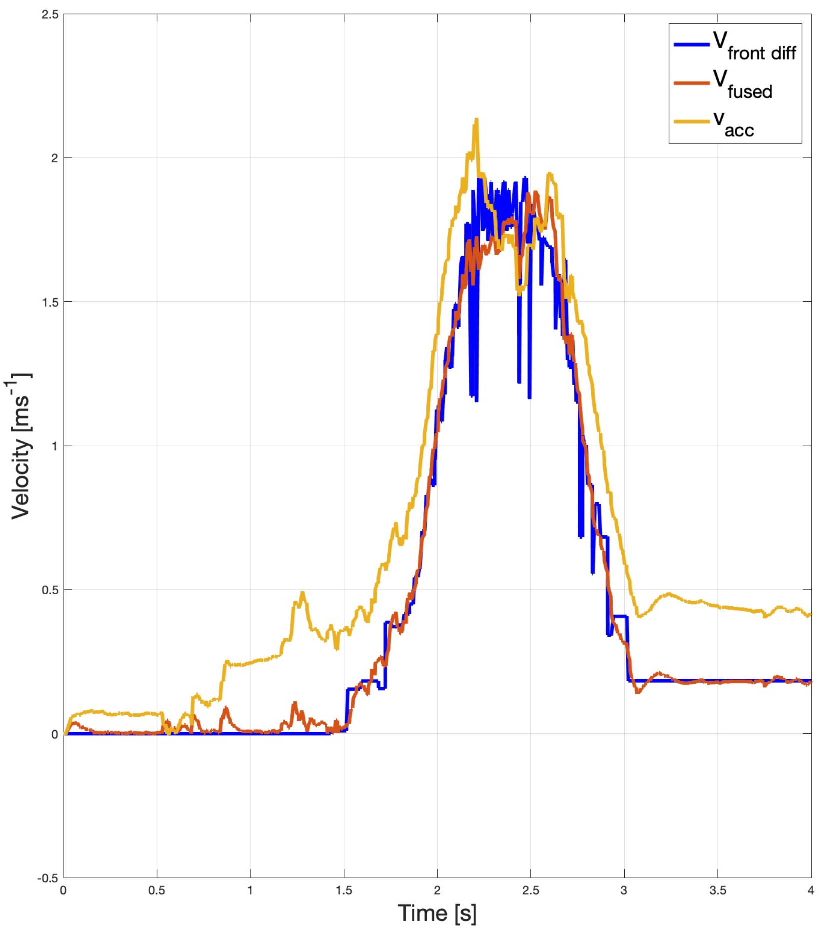


Figure 7.11: Filtered and raw speed data

7.8 Front and rear axel speed comparison

To compare rear and front speed calculation I drove the SDP to a certain speed then let go of the throttle and let it coast to stop. This method provides the least amount of slip thus speed on the front and rear axles should be the same. Speeds read from motors were within the margin of error to the speed measured by Hall switch fused with the accelerometer.

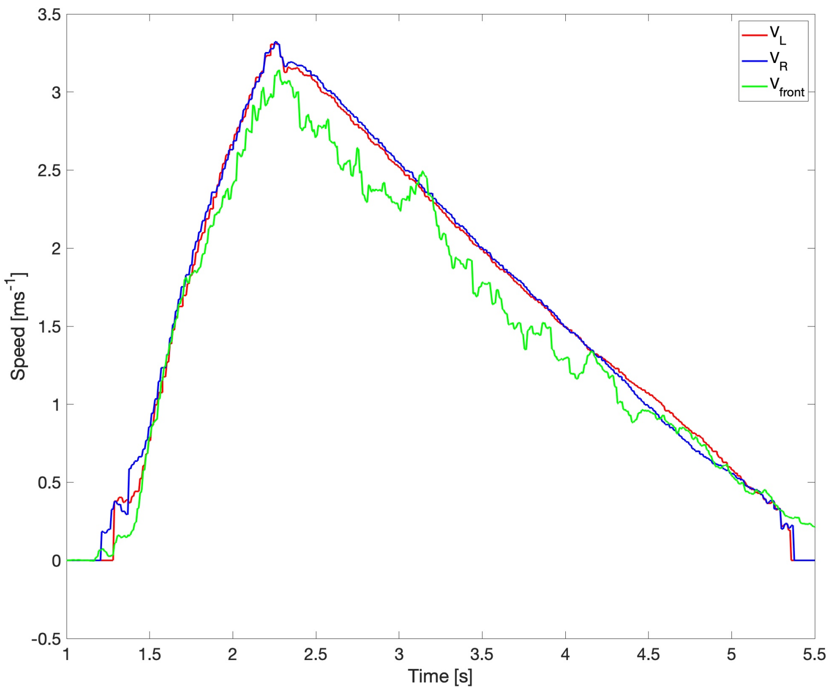


Figure 7.12: Front and rear axel speeds

7.9 Algorithm updates

I need to adapt developed algorithms to SDP. Since each reae wheel has its own motor I will you two PID controllers one for each wheel.

7.9.1 Logical algorithm

As said before the maximal current is between 35000 to 40000 mA this is a different maximal value than used in Single-track simulation. I could have used a similar technique as in Carmaker and just use a coefficient to multiply the output. However, I decided to change the value of s to make the algorithm more aggressive during acceleration. The new value is $s = 0.4$. I also change the last two states shown in Flowchart. I combined them together. This means when the slip ratio is in the interval $(0.13, 0.20)$ the output to the integrator is 0 thus no more torque is requested. When the value of λ is greater than 0.2 the output is decreasing exponentially.

7.9.2 PID

I used values derived for PID in the single-track model as base values. Since I am controlling current I have different interval of output values. I have set the saturation on slip ratio PIDs to be 35000. This means a maximum 35 A of current will be applied to the motor. As I do not want to break with PIDs I set the minimal output of the PIDs to be 0. As the maximal torque of one motor (computed in eq. 3.5) used in the simulation was 0.65. I added a gain of 51000 at the output of PID. After initial tests, the PID was too aggressive and output was oscillating. I lower all the values and after a few tests, I settled on values:

$$K_p = 0.15, \quad K_i = 0.6, \quad K_d = 0.007. \tag{7.15}$$

For the yaw controller I used PI controller. I have saturated its output to interval $< -0.15, 0.15 >$. Experimentally I determined its values to be:

$$K_p = 0.01, \quad K_i = 0.1. \tag{7.16}$$

Initial reference value for slip ratio controllers is 0.2. The Yaw rate controller can change it, but maximal reference value is set to be 0.25.

Chapter 8

SDP driving tests results

8.1 Driving scenarios

Testing was held on three different surfaces to represent different types of roads. I used carpet and a wooden floor. I also tested a split μ scenario where one of the wheels had a significantly lower coefficient μ. This was achieved by applying electrical tape on one of the wheels. In all tests, the SDP starts at a standstill. As starting steering angle I used zero, and full steering lock around 47°. As in the previous experiments, I will test without control then with the logical algorithm, and lastly with linear control.

8.2 No control

As shown in simulations, when full throttle is applied the car loses traction immediately. As can be seen in figure 8.1 I have sent a maximal throttle request. Both wheels spun and the slip ratio Instantaneously grew over the previously defined safe value 0.2. From a graph of speeds, we can see that the left wheel had more traction from the beginning however, near the end, it also spun. The test drive was shorter than expected due to the fact that the SDP spun. We can see the change of steering input as I have tried to prevent the spin and keep the car in a straight line. I am only showing results from testing on the carpet as the experiment on the wooden floor end up with the same result.

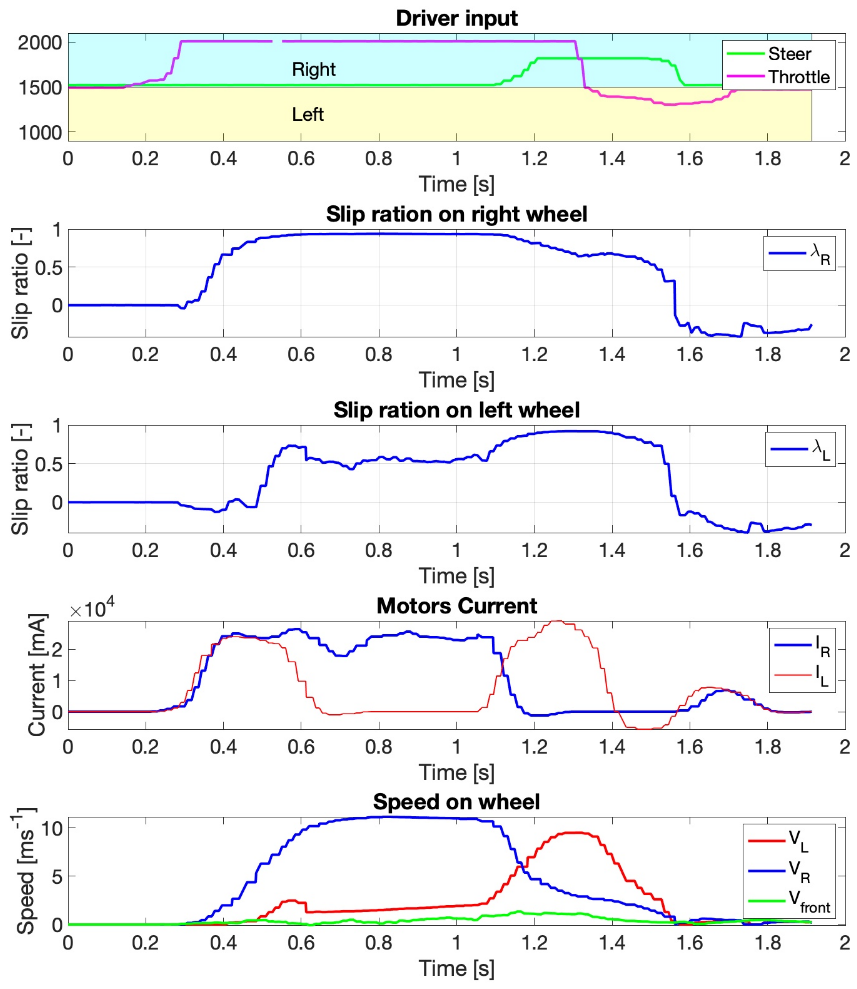


Figure 8.1: No control start on carpet

In figure 8.2 are shown results of split μ test. On the right wheel, the electrical tape was applied thus the friction coefficient was significantly lowered. When I have applied full throttle both wheels spun. However, we can see that the right wheel spins faster and requires less amount of current supplied. This time the SDP also spun despite my effort to steer correct the SDP.

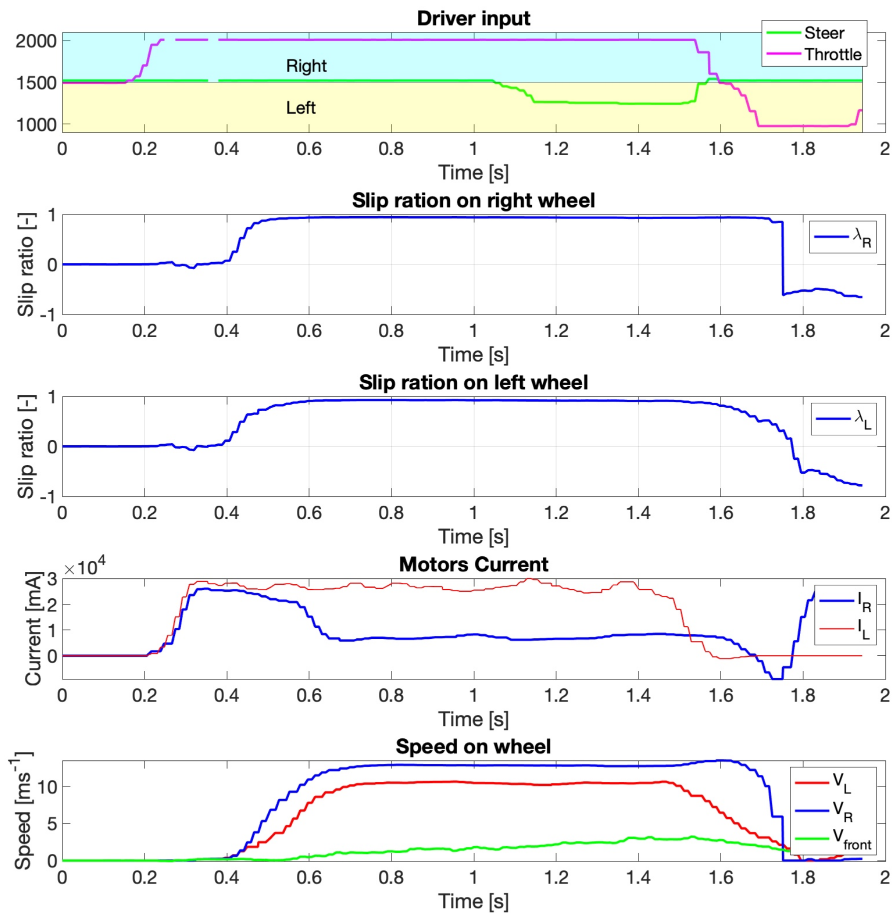


Figure 8.2: No control applied on split μ test drive

8.3 Logical control algorithm

Figure 8.3 we can see the results of the test drive conducted on a wooden floor. This surface was used to simulate a surface with a very low friction coefficient. The ramping up of current supplied to the motors can be seen.

We can see that around the two-second mark of the test, the wheels slipped. This is also indicated by an increase in the slip ratio on both wheels. In response to this increase, the logical algorithm has reduced the current to the motors. Some steering requests were made at the end of the test to avoid collision due to the shortness of the test track.

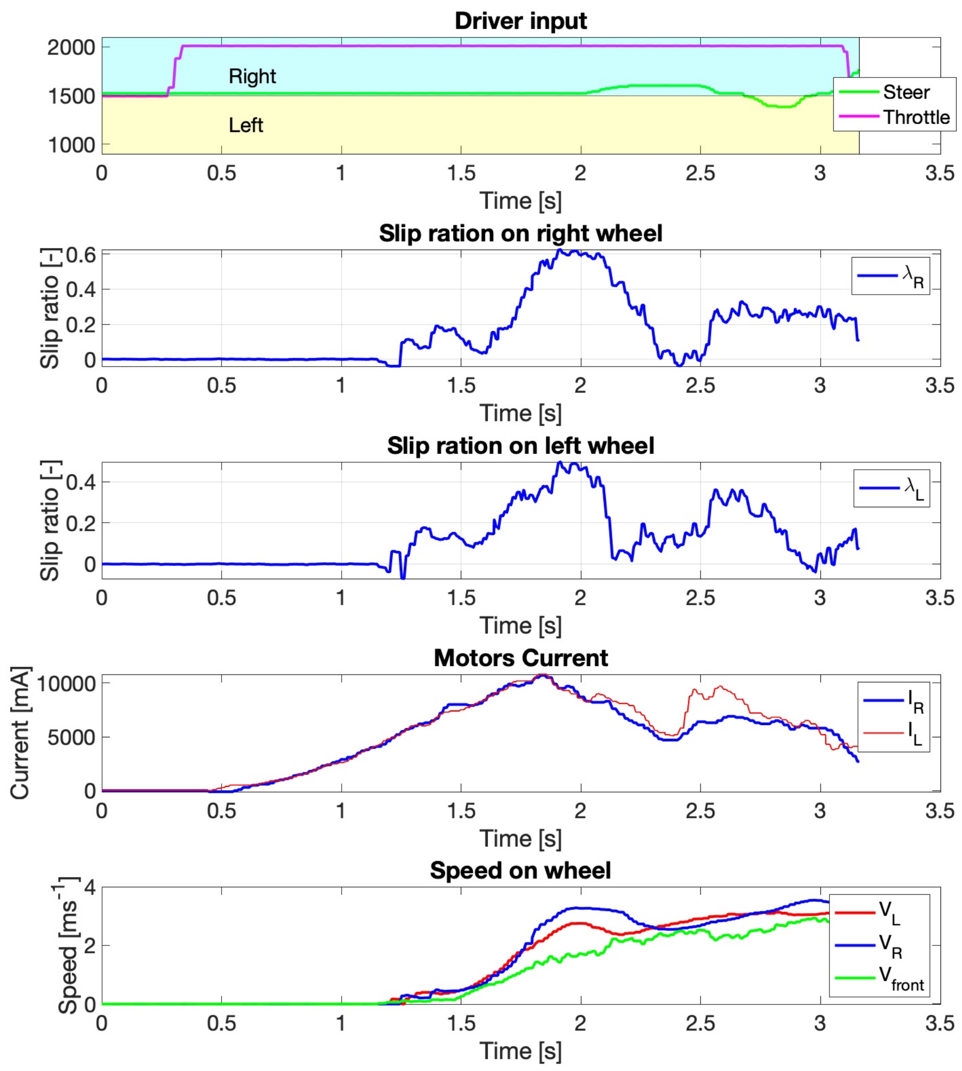


Figure 8.3: Logic based λ torque control on wood

Test results from the drive on the carpet are displayed in figure 8.4. This test again starts with full throttle applied. We can see the current ramping up. The SDP accelerates without an accessive wheel slip. We can see that after initial faster growth, the algorithm acts and slows it down. As the SDP reaches around 4 m/s, the slip ratio grows as the difference between speeds gets bigger. The bigger difference can be caused by inaccuracy in wheel speed computation. At higher speeds, the diameter of the tire grows. This is the design of the tire as it provides more grip during the start and less rolling resistance at higher speeds.

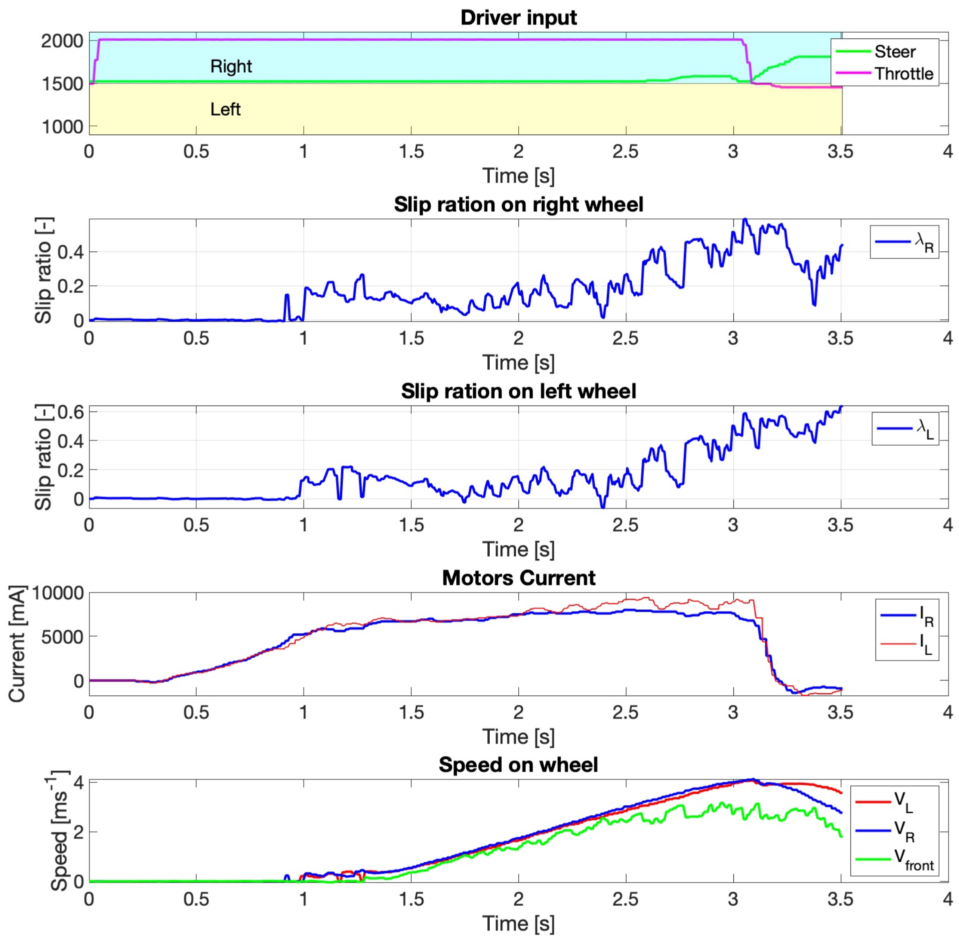


Figure 8.4: Logic based λ torque control on carpet

Figure 8.5 shows the results of the test drive on a wooden floor with full steering lock. This test was longer than previous ones, and the SDP completed multiple cycles during this test. As I was making left turns, the speed of the left wheel was expected to be lower than the speed of the right wheel. This was confirmed in the test drive. Using the same method of choosing between left and right slip ratio values for the logical algorithm as in Carmaker simulations, the algorithm mostly controlled by the right wheel throughout the test run. While there were spikes, the algorithm successfully maintained stability, and the SDP did not spin. This result differs from the simulations.

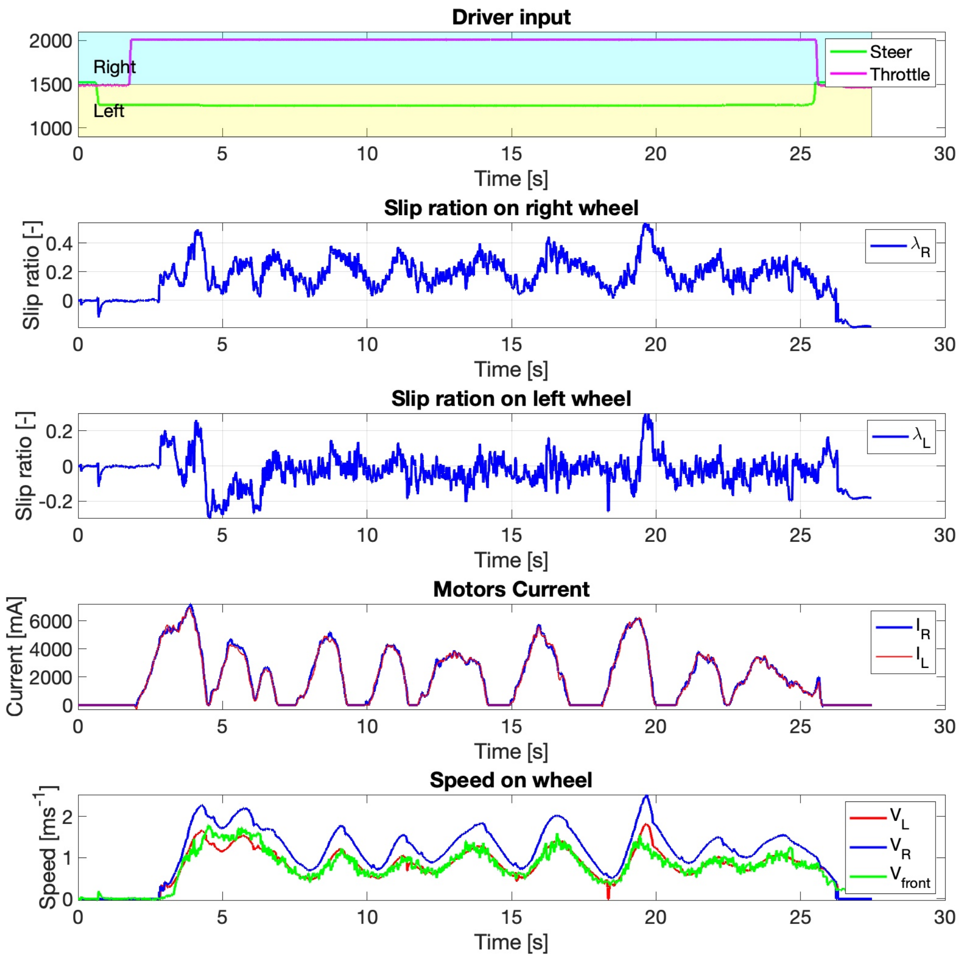


Figure 8.5: Logic based λ torque control on wood with full steer to the left

Results for the split μ test scenario are shown in Figure 8.6. The right wheel had the lower friction coefficient. We can observe multiple instances of wheel slip, but the algorithm consistently manages to reduce the current and prevent further slipping of the wheel. However, some steering input was necessary to keep the SDP moving in a straight line.

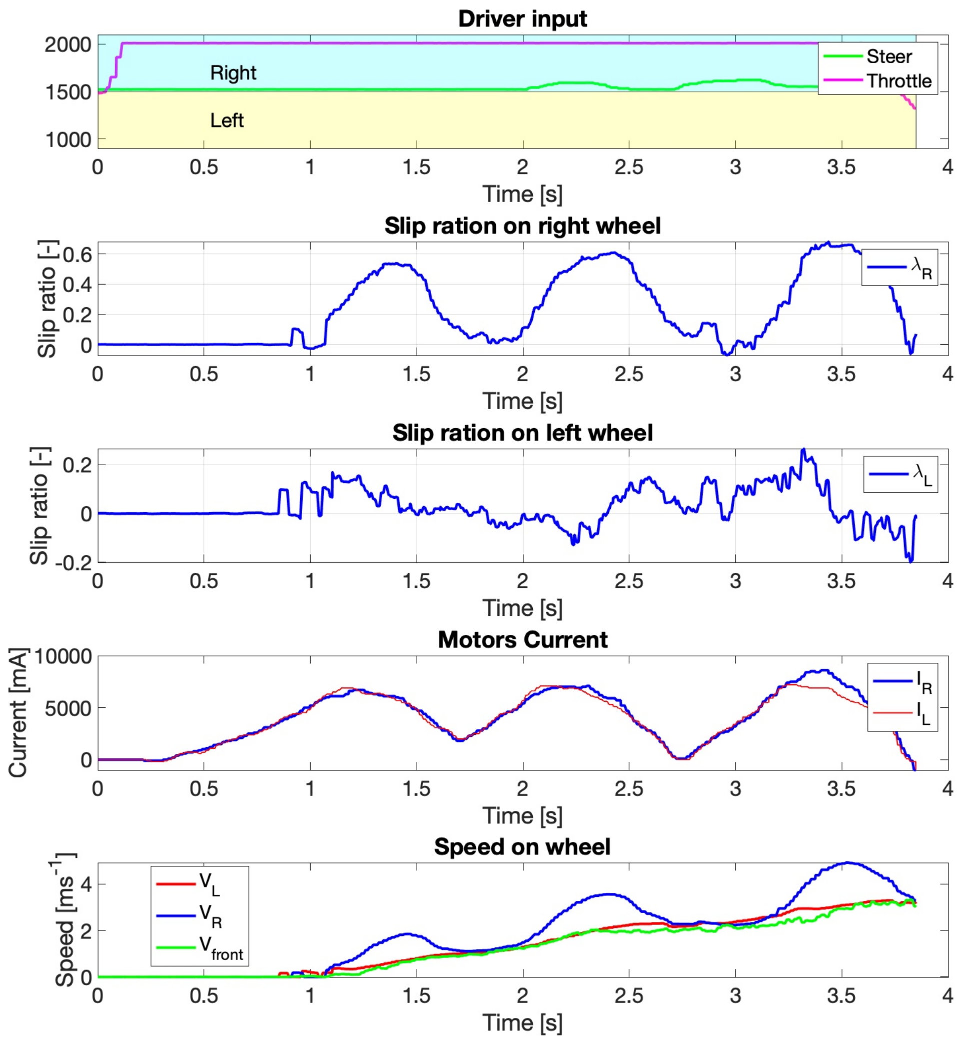


Figure 8.6: Logic based λ torque control on split μ surface

8.4 PID controll

Linear control was used in the following tests. In Figure 8.7, the results of the test drive on carpet are shown. When a throttle request is received, a reference value of 0.2 is set for both slip ratio PID controllers. We can see that torque requests are generated and sent to the motors. The Subscale Development Platform (SDP) accelerates without wheel slip. We can also compare the time from $0 - 3\ m/s$. As previously shown in simulations, the PID is quicker to set the optimal torque (current). This can be observed here as well. The time from the initial request to the car reaching a speed greater than $0.2\ m/s$ is around $0.6\ s$. It takes the logical algorithm on the same surface around $1\ s$ to surpass this value.

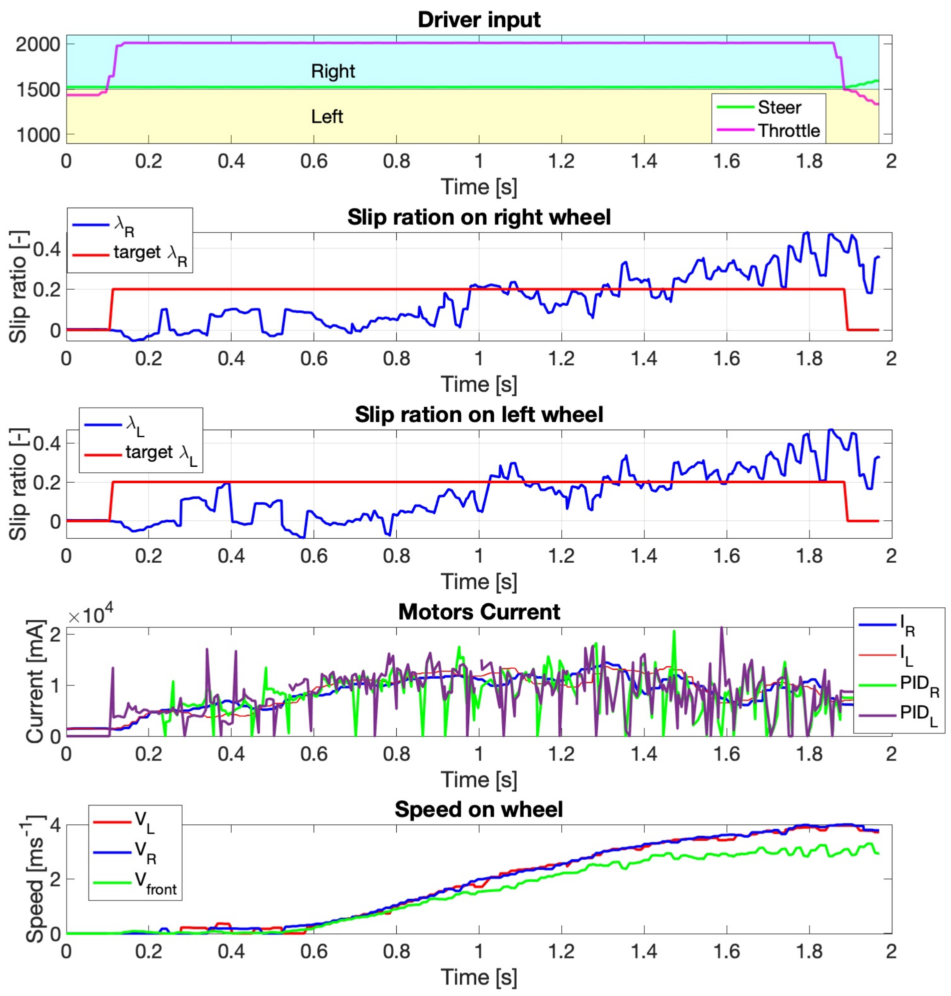


Figure 8.7: PID controll of slip ratio on carpet

In figure 8.8 we can see the progress of the test drive on a wooden floor. Wood has lower friction coefficient than carpet. We can see that the SDP starts up and during the acceleration two spikes in rear wheel speeds are visible. We can compare this test run to the test run with logical algorithm. The amplitude of spikes is much smaller in comparison to the start up with logical algorithm. After every spike the PID regulates the current and the slip ratio drops down on the reference value.

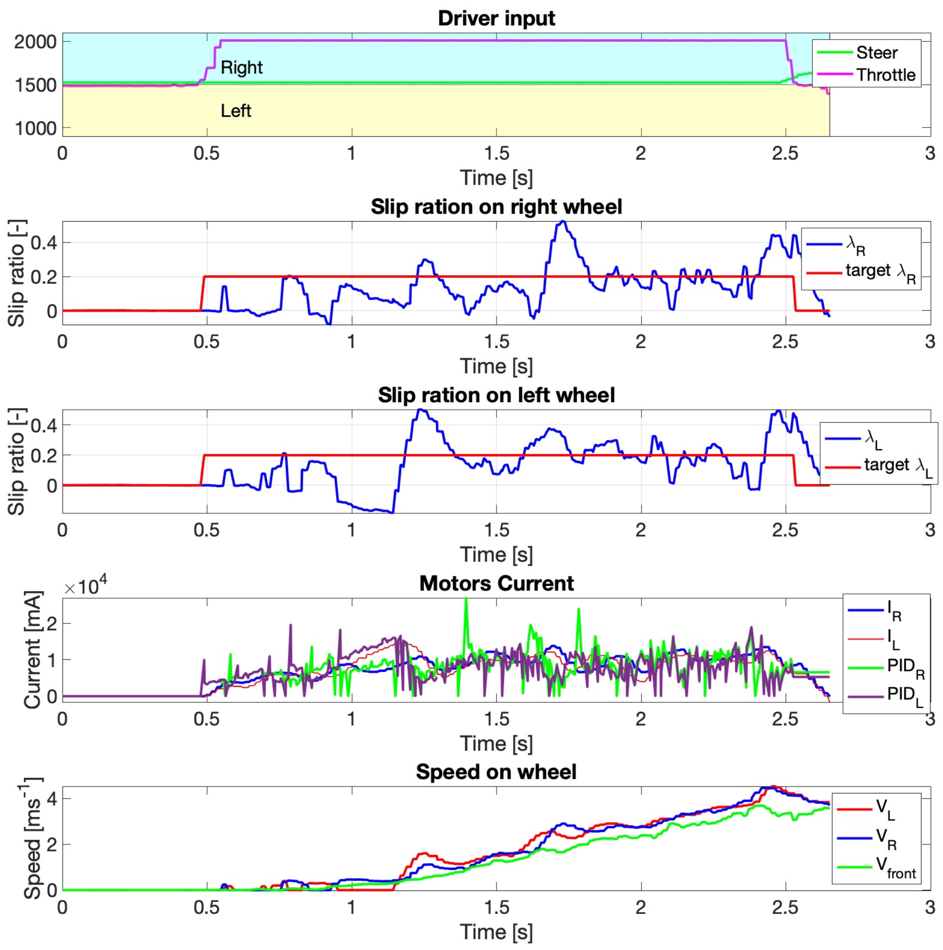


Figure 8.8: PID controll of slip ratio on wood

In Figure 8.9, the results of the split μ test drive are shown. Similar to the test drive with the logical algorithm, the right wheel has lower friction. We can observe that immediately after the start, the right wheel loses grip, but its controller reacts accordingly and reduces the current to the motor. After this initial slip, the right wheel remains in the controlled region. We can clearly see the lower torque demand on the right wheel compared to the left one.

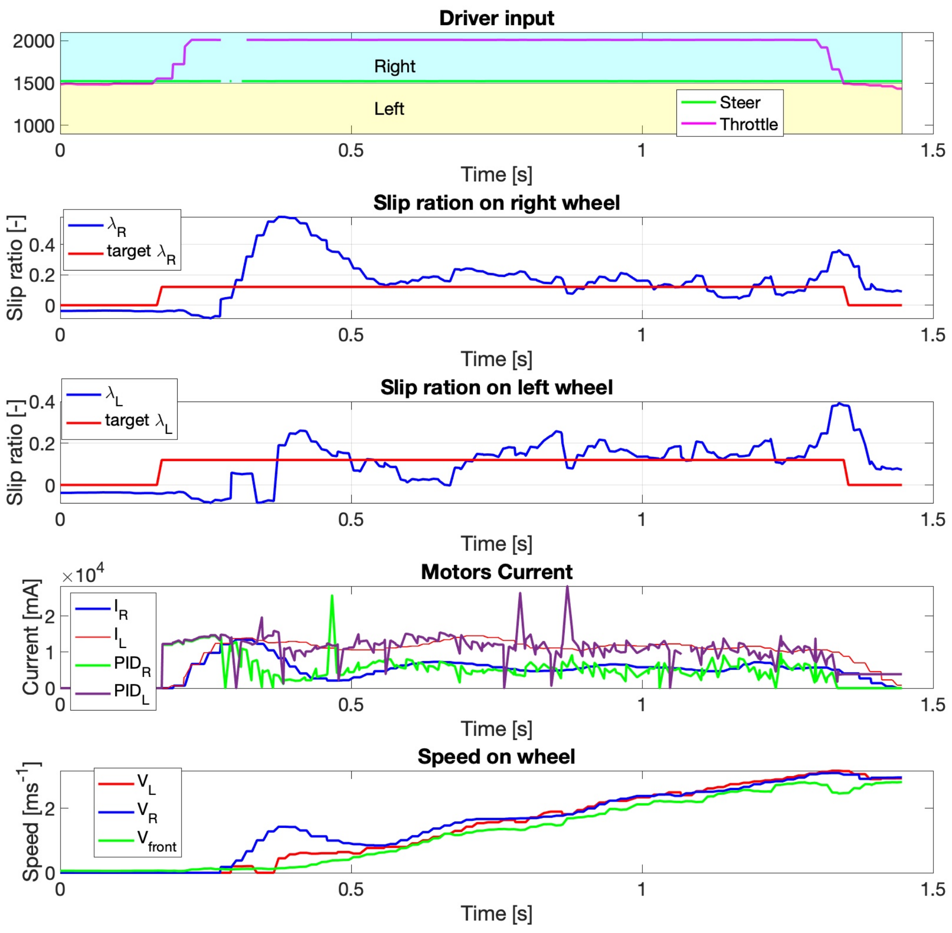


Figure 8.9: PID controll of slip ratio on split μ surface

In Figure 8.10, we can see the startup with full left steering lock. As I have implemented a simple PI torque vectoring controller, after startup, the controller lowers the reference value for inner wheel slip ratio, as well as slightly raises the outer wheel slip ratio reference. During startup, there is slight tire slip present, but the PID controller acts and manages to control the slip ratio. As opposed to the logical control algorithm, the speed is kept steady and without oscillation.

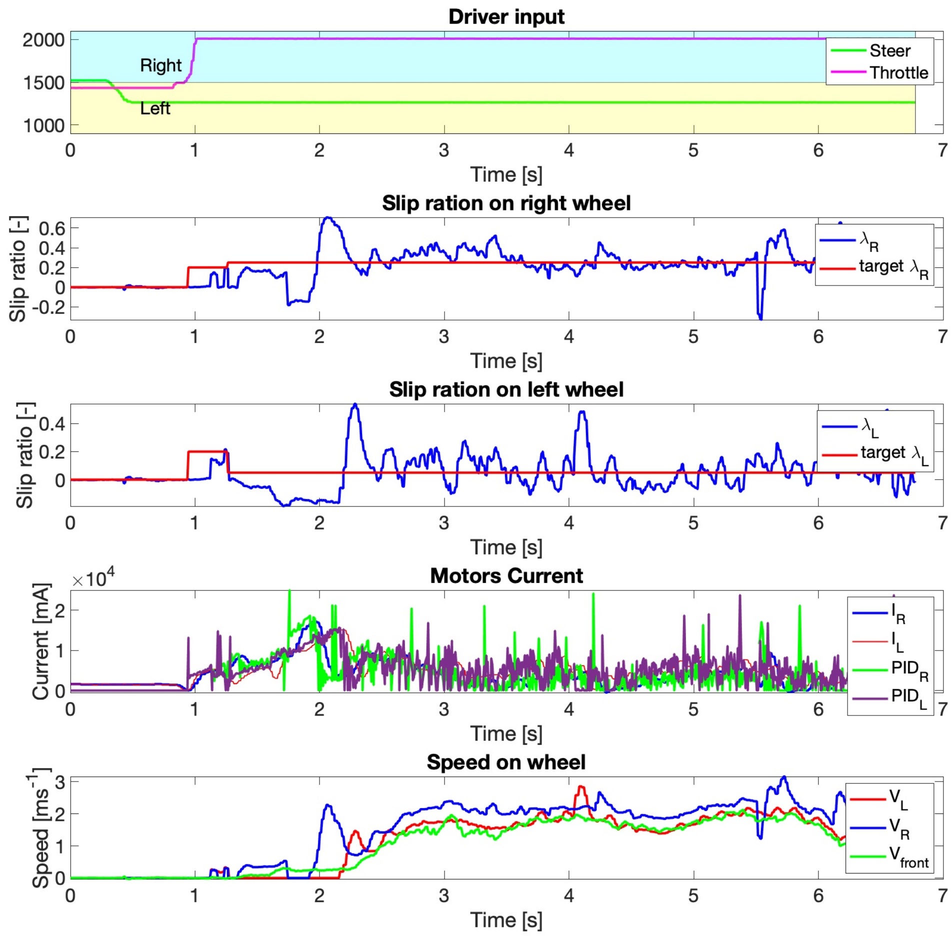


Figure 8.10: Start with full left turm using torque vectoring on tiles

Chapter 9

Conclusion

The goal of this thesis was to develop traction algorithms for conventional vehicle operating at low speed region.

In this thesis current traction control algorithms were summerised and descibed. I have used mathematical models to develop and subsequently test two traction control strategies. Both were based on controling slip ratio. Slip ratio definition was changed for use at low speeds. Logical traction control system was devoloped and tested. In simulation it managed to startup the car realiably when no steering input was applied. The PID control worked well and managed to startup the car effectively in every situation tested. Carmaker software was used to verify simulation results on single-track model.

Scaled-Down Platform was used for real life testing. SDP was updated for use at low speeds speed signal filtering was set up. Both control algorithms were implemented. Both algorithms were able to limit wheel spin and keep the SDP controllable while acceleration.

9.1 Future work

- Further develop torque vectoring
- Update SDP for testing either by creating new one or updating current one to be reliable
- Identify SDP and tire-road model in Carmaker
- Test control systems on full size road vehicle

Bibliography

[1] *Axi 2814/16 gold line.* https://www.modelmotors.cz/product/detail/213. Accessed:2023-02-27.

[2] *Bmw 318i owners manual.* https://www.manualslib.com/manual/728617/Bmw-318i.html?page=104#manual. Accessed:2023-02-27.

[3] *Rotational moment of inertia apparatus.* https://lambdasys.com/products/detail/5. Accessed:2023-04-13.

[4] *Tesla engineer explains what a model s plaid does when you activate 'drag strip mode'.* https://jalopnik.com/tesla-model-s-plaid-drag-strip-mode-cheetah-explained-1850256094. Accessed: 2023-04-28.

[5] *The powertrain pure performance.* https://media.porsche.com/mediakit/taycan/en/porsche-taycan/der-antrieb. Accessed: 2023-04-28.

[6] *Uni- and bipolar hall ic switches for magnetic field applications.* https://cz.mouser.com/datasheet/2/196/Infineon_TLE49X5L_DataSheet_v01_05_en-2322416.pdf. Accessed:2023-02-27.

[7] *List of fastest production cars by acceleration.* https://en.wikipedia.org/wiki/List_of_fastest_production_cars_by_acceleration#cite_note-AWD-10, 2001-.

[8] *The ultimate guide to abs: How it works.* https://driving.ca/column/how-it-works/how-it-works-abs#:~:text=You%20won%27t%20feel%20your,usually%20going%20slower%20than%20that, 2019. Accesed: 2023-4-29.

[9] *Fastest 0-100 km/h acceleration by an electric car*. `https://www.guinnessworldrecords.com/world-records/fastest-0-100-kmh-acceleration-electric-car`, 2022. Accessed: 22.2.2023.

[10] *Hill descent control hdc - the concept*. `https://www.bavarianmw.com/guide-686.html`, 2022. Accessed: 20.2.2023.

[11] *Twintrack*. `https://gitlab.fel.cvut.cz/hanistom/VehicleModel/-/tree/master/TwinTrack`, 2022.

[12] J. Andersson, *Vehicle dynamics - optimization of electronic stability program for sports cars*, 2008. Luleå University of Technology Department of Applied Physics and Mechanical Engineering Division of Computer Aided Design.

[13] A. Biswas and R. Yadav, *Modeling and simulation of launch control system for formula student electric vehicle*, in 2021 IEEE Transportation Electrification Conference (ITEC-India), 2021, pp. 1–7.

[14] D. Efremov, *Singletrackfullmodel*. `https://gitlab.fel.cvut.cz/hanistom/VehicleModel/-/tree/master/SingleTrackFullModel`, 2022.

[15] P. Giani, M. Tanelli, S. M. Savaresi, and M. Santucci, *Launch control for sport motorcycles: A clutch-based approach*, Control Engineering Practice, 21 (2013), pp. 1756–1766.

[16] gpdriver17, *Launch control is limited to 50 uses at maximum shift aggressiveness*. `https://g80.bimmerpost.com/forums/showthread.php?t=1806262`. Accessed: 2023-05-20.

[17] Y. Guodong, Z. Chengjie, and Z. Ning, *The torque distribution and anti-slip regulation control for two-wheel independent drive electric vehicle*, in 2016 Chinese Control and Decision Conference (CCDC), 2016, pp. 4444–4449.

[18] L. Hostačný, *Vehicle measurement system*, 2019.

[19] E. Kramer and A. Cantarelli, *Updating euro emission standards (euro 7)*. `https://www.europarl.europa.eu/RegData/etudes/BRIE/2023/740246/EPRS_BRI(2023)740246_EN.pdf`. Accessed: 2023-05-19.

[20] M. Mondek, *Active torque vectoring systems for electric drive vehicles*, 2018.

[21] K. Nam, Y. Hori, and C.-Y. Lee, *Wheel slip control for improving traction-ability and energy efficiency of a personal electric vehicle*, Energies, 8 (2015), pp. 1–21.

[22] H. B. Pacejka and I. J. M. Besselink, *Tire and vehicle dynamics*, Elsevier, Amsterdam, third edition ed., 2012.

[23] K. Reif and K.-H. Dietsche, *Automotive handbook*, Robert Bosch, Karlsruhe, 9th edition, revised and extended ed., [2014].

[24] J. Sajdl, *Hsa (hill start assist)*. `https://www.autolexicon.net/cs/articles/hsa-hill-start-assist/`. Accessed: 2023-05-19.

[25] D. Schramm, M. Hiller, and R. Bardini, *Vehicle Dynamics: Modeling and Simulation*, no. Book, Whole, Springer Berlin / Heidelberg, Berlin, Heidelberg, 2014 ed., 2014.

[26] H. Shah, S. Haldar, R. Ner, S. Jha, and D. Chakravarty, *Ground vehicle odometry using a non-intrusive inertial speed sensor*, in 2019 IEEE International Conference on Industrial Technology (ICIT), 2019, pp. 120–125.

[27] J. Stellet, M. Giessler, F. Gauterin, and F. León, *Model-based traction control for electric vehicles*, ATZelektronik worldwide, 9 (2014), pp. 44–50.

[28] Volvo, *Hill descent control (hdc)*. `https://www.volvocars.com/cz/support/car/v40-cross-country/18w17/article/1a8488fe3f1a836bc0a801e801c79357`. Accessed: 2023-03-12.

Appendix A

Acronyms

ABS	Anti-lock Braking System
COG	Center Of Gravity
HDC	Hill Descent Control
HSA	Hill Start Assist
ESC	Electric Speed Controller
VESC	Vedder Electric Speed Controller (the Benjamin's Vedder motor controller)
SDP	Scaled-Down Platform
GPS	Global Positioning System
PWM	Pulse Width Modulation
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver-Transmitter
CAN	Controller Area Network
IMU	Inertial Measurement Unit
SD	Secure Digital
MEMS	Micro-electromechanical Systems

Appendix B

Simulation results Single-track

B.1 Speed $= 0.001\ m/s\ \delta_f = -1$

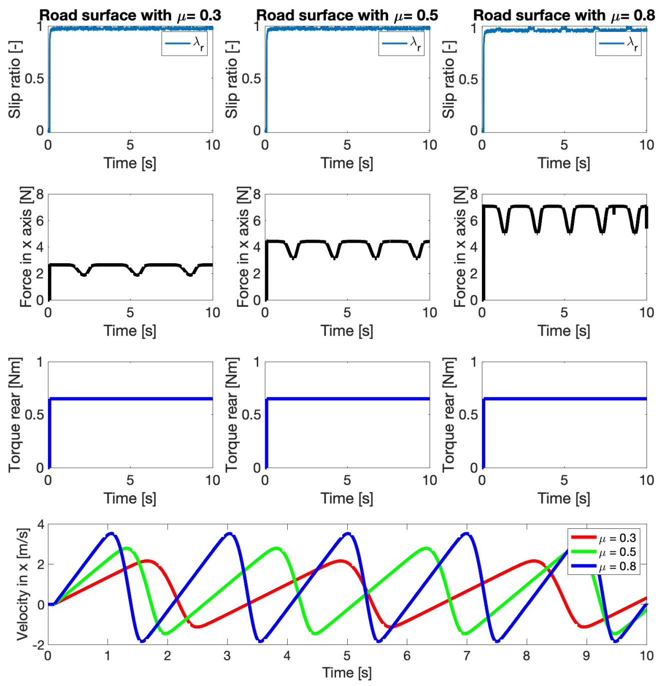


Figure B.1: Simulation results with no control and $\delta_f = -1$

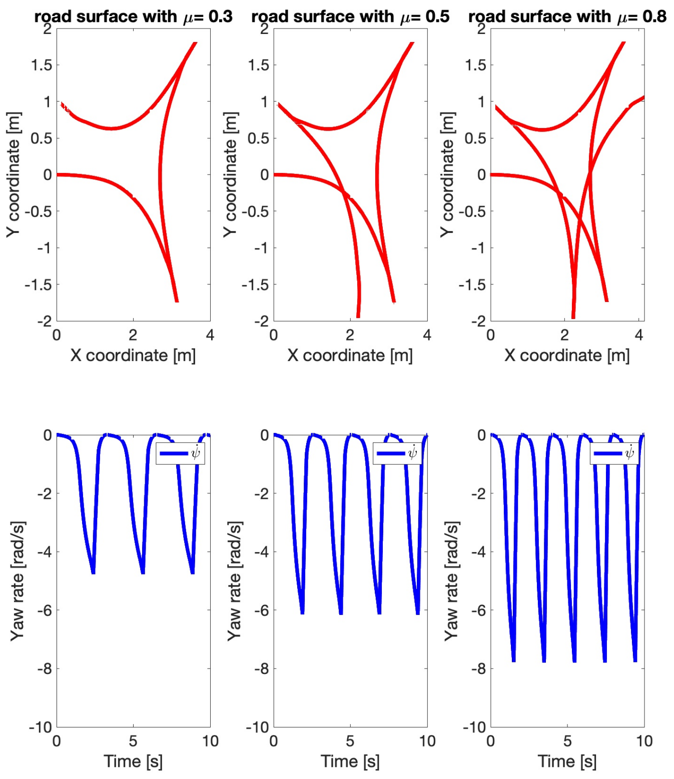


Figure B.2: Simulation results with no control and $\delta_f = -1$

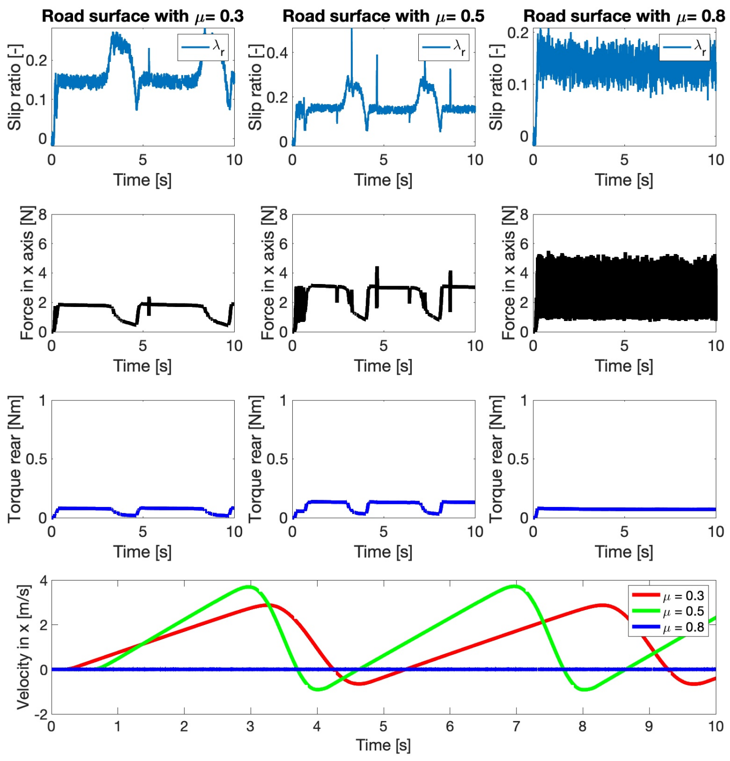


Figure B.3: Simulation results with logical control algrithm and $\delta_f = -1$

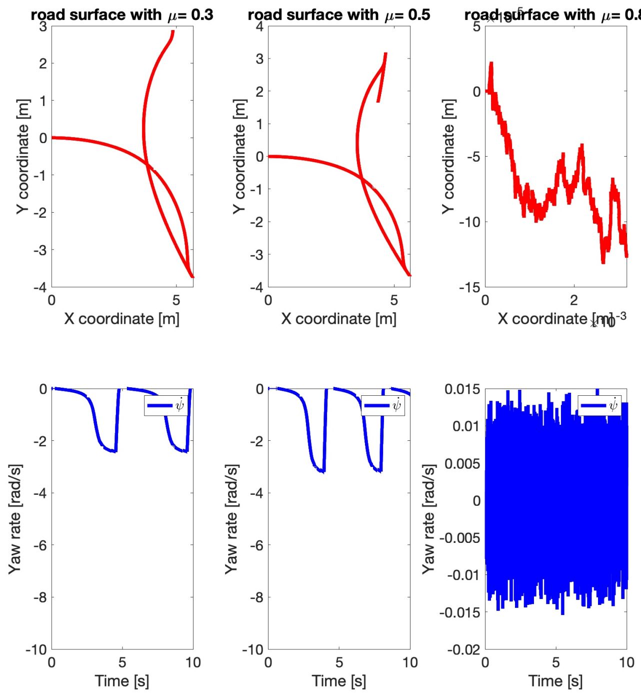


Figure B.4: Simulation results with logical control algrithm and $\delta_f = -1$

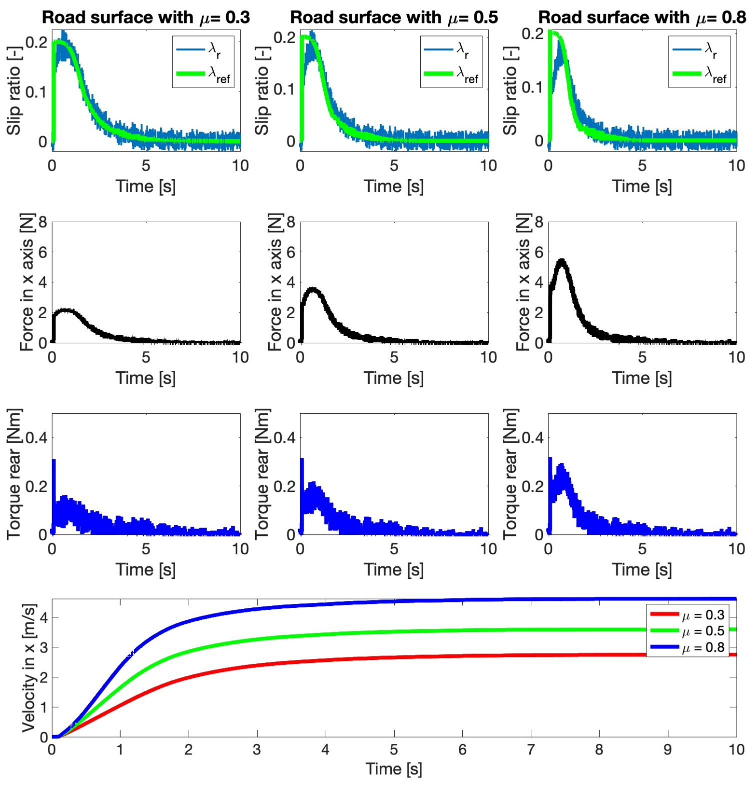


Figure B.5: Simulation results with linear control and $\delta_f = -1$

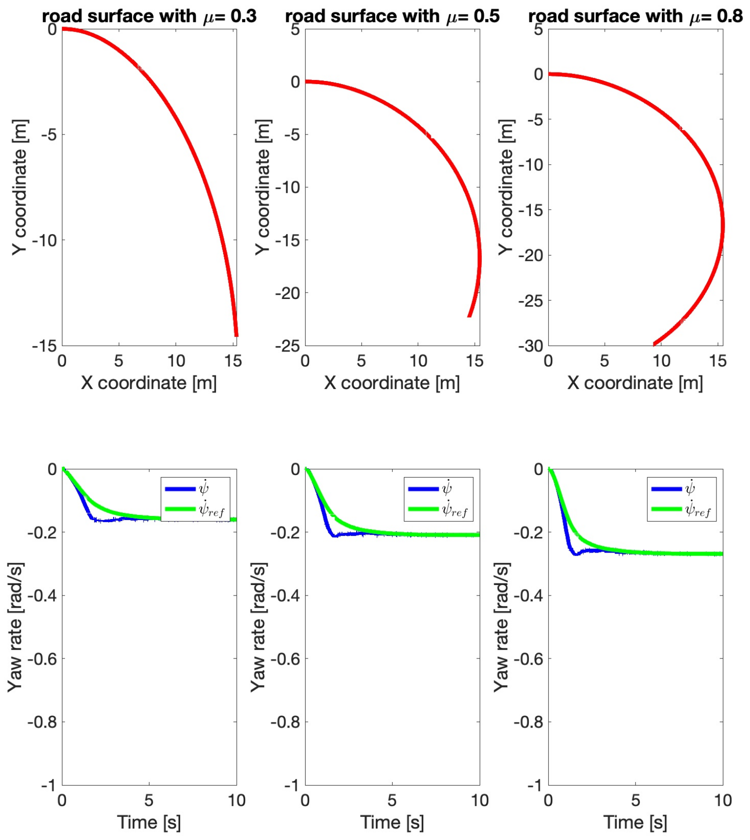


Figure B.6: Simulation results with linear control and $\delta_f = -1$

Appendix C

Table with SDP signals

Table C.1: SDP signals and connection protocols

#	Name	Variable name	Sensor	Interface	Unit
1	Current in right motor	curr_R	VESC	CAN	A*1000
2	Current in left motor	curr_L	;	CAN	A*1000
3	ERPM right motor	erpm_R	;	CAN	-
4	ERPM left motor	erpm_L	;	CAN	-
5	Duty cycle right motor	duty_R	;	CAN	[-] cdot 100000
6	Duty cycle left motor	duty_L	;	CAN	-*100000
7	Steering request	steer	-	PWM	-
8	Throttle request	throttle	-	PWM	-
9	Velocity from differential	diff_vel	Hall sensor	PWM	$m/s \cdot 10000$
10	Longitudnal acceleration	accData[0]	LSM6DS3	SPI	;
11	Lateral acceleration	accData[1]	;	SPI	;
12	Vertical acceleration	accData[2]	;	SPI	;
13	Roll	aglData[0]	;	SPI	;
14	Pitch	aglData[1]	;	SPI	;
15	Yaw	aglData[2]	;	SPI	;

Appendix D

CD contents

```
STM_MX
│
└── Platform_Kohout_Launch.ioc
MATLAB
├── init.m
├── LoadCarConfigs.m
├── plot_PID.m
└── test_file_combV.m
Simulink
├── MODEL.slx
└── SingleTrack5states.slx
C files
├── Include
│   ├── main.h
│   ├── Mezz_read_SPI.h
│   ├── Mezz_send_via_CAN.h
│   └── My_calculations.h
└── Source
    ├── main.c
    ├── Mezz_read_SPI.c
    ├── Mezz_send_via_CAN.c
    └── My_calculations.c
```