Ing. Manfred Großmann, BSc.

MOVES^2 - Modeling of ABS & TCS Algorithms

MASTER'S THESIS

to achieve the university degree of

Master of Science

Master's degree programme: Telematics

submitted to

Graz University of Technology

Supervisor:

Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Eugen Brenner

Institute of Technical Informatics

Ass.Prof. Dipl.-Ing. Dr.techn. Univ.-Doz. Arno Eichberger
Dipl.-Ing. Martin Ackerl
Dipl.-Ing. Harald Kraus

Institute of Automotive Engineering
Member of [**FSI**]

Graz, Dezember 2014

Acknowledgement

I would like to thank the chiefs, my superiors, my colleagues, and basically everybody here, at the Institute of Automotive Engineering.
To name just a few, I want to mention my supervisor at the FTG, Harald Kraus, who brought an end to this never-ending story, and Prof. Arno Eichberger, who started it a long time ago, during a lecture I attended[1].
At the Institute of Technical Informatics I want to say thanks to my mentor, Prof. Eugen Brenner.

A huge amount of gratitude goes to my parents Sonja and Alois (also known as Mama and Papa and soon to be known as Oma and Opa) for raising me, my sister Martina and the countless rest of our big family.
The biggest appreciation is kept for my beautiful, loving, and supporting wife, Dashka, and our unborn child! I couldn't do it without you!

Thanks to my fellow students and friends at the university, at Graz, and its surroundings. This one goes especially to my friend, Gerhard...
Finally, but not less important another "thank you" to my dear friend, Robert!

[1]331.191 - Fahrzeugtechnik Grundlagen für Elektrotechnik und Telematik

Affidavit

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

Datum Unterschrift

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

Date Signature

Abstract

The importance of simulation of driver assistance systems increases steadily. One advantage is reduced costs compared to prototype-tests. Furthermore, within a simulation there are a multitude of tests possible, which would maybe destroy the prototype in real-life. Therefore, it is reasonable to test the functionality of driver assistance systems within and beyond its limits using a save simulation-environment.

At the Institute of Automotive Engineering an inhouse developed simulation-environment is used. MOVES^2 is a MOdular VEhicle Simulation System which contains customizable vehicle-models. It is adjustable according to given demands. During the preceding seminar-project, the graphical user-interface of this simulation-environment has been redesigned. Backwards-compatibility, a simple usage and maintenance, as well as modular expandability were the main focus.

Building on this, this master-thesis concentrates on modeling an ABS- and TCS-algorithm within MOVES^2. Now the main focus is not to implement an ideal slip-control, but to stay near to reality. It relies on the designs of BOSCH, which were adapted and enhanced where necessary. The models were created using existing templates to stay backwards-compatible. Furthermore, future extensions are already considered accordingly. Therefore, an implementation of an ESP would be the next logical step.

A second simulation-environment, IPG Carmaker, was used for verification of the developed algorithms.

Kurzfassung

Die Simulation von Fahrassistenzsystemen gewinnt immer mehr an Bedeutung. Zum einen können Kosten gegenüber von Prototypen-Tests gespart werden. Zum anderen kann in der Simulation eine Vielzahl an denkbaren Tests durchgeführt werden, welche vielleicht bei Prototypen zur Zerstörung dieser führen würden. Es macht also durchaus Sinn, die Funktionalität von Fahrassistenz-Systemen in der Simulation an dessen Grenzen und auch darüber hinaus zu testen.

Das Institut für Fahrzeugtechnik Graz hat eine eigene Simulationsumgebung namens MOVES^2 entwickelt. MOVES^2 steht für MOdular VEhicle Simulation System und beinhaltet konfigurierbare Fahrzeugmodelle, welche je nach Bedarf entsprechend eingestellt werden können. Im Zuge der vorangegangen Seminararbeit wurde die graphische Oberfläche von MOVES^2 neu gestaltet. Abwärtskompatibilität, einfache Bedienung und Wartung, sowie Erweiterbarkeit standen dabei im Vordergrund.

Darauf aufbauend beschäftigt sich diese Diplomarbeit mit der Modellierung von ABS- und ASR-Algorithmen in MOVES^2. Hier liegt der Schwerpunkt darin, nicht eine optimale Schlupfregelung mit entsprechenden Controllern zu implementieren, sondern nahe an der Realität zu bleiben. Es beruht auf den Ausführungen von BOSCH, welche - wo nötig - entsprechend adaptiert wurden. Die Modelle sind anhand von Vorlagen erstellt worden um abwärtskompatibel zu bleiben. Weiters wurde auch schon Rücksicht auf zukünftige Erweiterungen gelegt, sodass die Implementierung eines ESP der nächste logische Schritt wäre.

Eine zweite Simulations-Umgebung, IPG Carmaker, wurde zur Verifizierung der entwickelten Algorithmen eingesetzt.

Contents

1. Introduction

Based upon the preceding seminar-project [14], this master-thesis implements ABS- (Anti-lock braking system, *"Antiblockiersystem"*) and TCS- (traction-control system, *"ASR, Antriebsschlupfregelung"*) algorithms within MOVES^2 (MOVES, MOdular VEhicle Simulation System).
MOVES is a homebrew simulation-environment, developed and maintained at the Institute of Automotive Engineering (FTG), Graz. ABS and TCS are common driver-assistance-systems which control the slip of wheels during braking or acceleration.

In contrast to real-word prototype-tests, simulations offer the possibility to test systems up to and beyond their limits without risking to damage or destroy the prototype. Furthermore, the simulation time is a fraction, which allows to increase the number of tests significantly. The importance of simulation-environments and their accuracy increases steadily.

As the author already improved the graphical user interface (GUI) during a seminar-project [14], he was eager to get a hands-on the newly redesigned tools and become a beta-tester. Thus, the author is able to see the intended simple usage and detect further bugs and unwanted behavior immediately.

This thesis introduces the program, MOVES and its preceding redesign. Furthermore, the ideas behind slip-controllers ABS and TCS are described.
Following chapters and sections explain in detail how those algorithms are built up using MATLAB Simulink and Stateflow. It also takes a closer look upon the two used simulation-environments, MOVES (Vehicle Model) and IPG Carmaker.

Discussions are directly appended to referring results. The summary will give an overview of achieved goals and an outlook of further improvements.

1.1. MOVES[2]

MOVES started out as a vehicle simulation system [18], but is able to do a lot more than just that. It is a complete development-environment with an according GUI based upon MATLAB[15], too. It uses its own (structured) parameter-files and customizable programs and functions to run simulations or optimizations. E.g. the main program is the initial vehicle-model, but there are already some other applications (like KOS or MuBOS [12]). Future implementations might use MOVES in completely unforeseen applications.

Motivation (Redesign)

The former GUI has been implemented using MATLAB GUIDE as a single-window program without the possibility of further, future extensions [19]. Back then, the involved programmers didn't think of any other usage besides the vehicle-model.

Today, there are a range of new applications and also the initial vehicle-program has become bigger and got more functions. Therefore, it is essential to have a flexible, modular environment which can be adjusted to the current demands. The new GUI relies on the functionality of the existing programs and intentions but splits the single-window dialog into multiple dialogs. Each dialog is only opened or presented when necessary to provide a better overview.

Functionality

The configuration of MOVES, its applications, and their referring functions and dialogs, are stored in newly introduced parameter-files. This allows also inexperienced users to set up new applications without deeper programming-skills. Furthermore, the parameter-files also allow again backwards-compatibility to include and handle GUIDE-created dialogs. A mix and combination of different dialogs (programmed, using GUIDE, or configured using parameter-files) is possible and concludes the requested modularity.

1.2. Antilock Braking System - ABS

ABS controls the slip of wheels during braking. Its goal is to avoid locking wheels when driver is braking, as this would mean the loss of control of the car. Controlled steering becomes impossible under locked-up wheels. Moreover, ABS should optimally use the current friction-potential between wheels and road-surface and, therefore, decrease the brake-distance.

This is done by influencing the brakes of each wheel. In case locking is imminent, the hydraulic brake-pressure is decreased by the ABS. During ABS-cycles the pressure is hold, decreased, or (again) increased.

The first electronic brake-system was introduced in 1978 [17], [20]. As guidance and basic layout for the implementation of an ABS, descriptions and documentation of BOSCH have been used [3], [11], [22], [17].

Requirements

The following requirements for ABS-controllers were taken from [17]:

- Its mandatory to provide driving-stability and steerability independent from current road-surface characteristics.
- Use the optimum current friction between tires and road.
- Adapt fast to surface-/friction- changes.
- Take care of μ-split-surface (different friction-conditions at left and right side of the road and vehicle). Common drivers should be able to compensate the resulting yaw-moment. It is unavoidable but ABS can decrease its slew-rate.
- The car needs to stay stable and steerable driving (braking) through curves.

All those criteria were considered during implementation and tested as shown in chapter 3 Results.

As (usually) all four wheels are braking and, therefore, all four wheels tend to lock, the main focus is to properly approximate the current slip or according helper-variables and indirect signals. This is addressed in the implementation (see section 2.3.1) and discussed in chapter 3 Results.

1.3. Traction Control System - TCS

TCS (*"Antriebsschlupfregelung"*, ASR) is a similar slip-controller compared to ABS. It avoids spinning of wheels during acceleration, independent from current road-surface conditions.

In contrast to ABS, the slip can be calculated very easily. This is true, if only one axle (front or rear) of the car is driven. In this thesis only front-driven vehicles are considered. Depending on the calculated slips the engine-torque is reduced to avoid spinning of the wheels. In case of μ-split-conditions, the low-friction sided wheel begins spinning, gaining the main load of the final-drive's distribution. Thus, TCS has to brake the spinning wheel to transfer torques onto to high-friction sided wheel.

The requirements are similar or equal to those of the ABS. Additionally, it is mandatory for TCS to avoid a shutdown of the engine, when reducing its power.

The first TCS was introduced in 1987 by BOSCH, [20].

2. Methodology

Contents

The following sections describe the implementation of ABS- and TCS-algorithms.

First, the vehicle, as part of a control-circuit, and basics of slip and tire-models are explained in the next chapters.
Thereafter, the two used simulation-environments, MOVES and Carmaker, are described.

2.1. Fundamentals

2.1.1. The controlled system - Vehicle

A control-circuit starts usually with a given setpoint-value that is compared to the current output. Depending on the difference (or error) the controller influences the referring system. The goal is to minimize this error between setpoint- and output-value.

Figure 2.1 shows the schematic layout of a vehicle-model as a control-system [5].

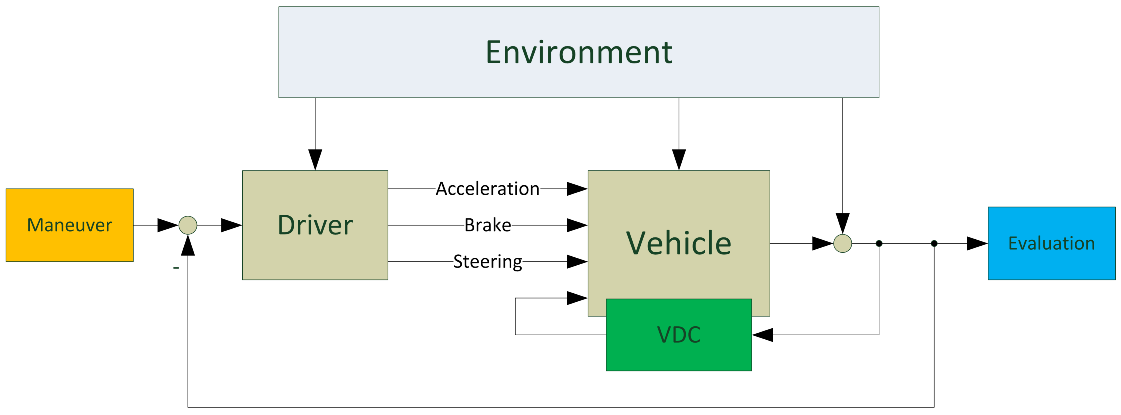


Figure 2.1.: Vehicle Model (schematic)

Maneuver provides the input-values and parameters to the system
(like road-layout, friction-potential, speed-profile,...).

Driver acts as controller within the outer circuit of the system.

Vehicle equals the controlled system, which reacts to driver-inputs.

VDC (vehicle dynamics controller, like ABS or TCS) control the inner circuit.

Evaluation: Results have to be evaluated
(e.g. reasonable braking distance, or uncontrollable behavior).

Environment influences the driver, vehicle, and acts as disturbance to the output.

Maneuvers are predefined (simulation-) scenarios, where all demands and requirements are defined to provide the setpoint-values of the driver.

The driver serves as controller to accelerate, decelerate, and steer the car through the given maneuver. In a closed-loop control-system the driver reacts to the difference between given setpoint- and current output-values. For instance, if the car starts to spin, the driver will counter-steer and try to keep the car on the road. An open-loop simulation has no feedback of the output-value. In this case the driver just passes through (or converts) the setpoint-values to regulate the system.

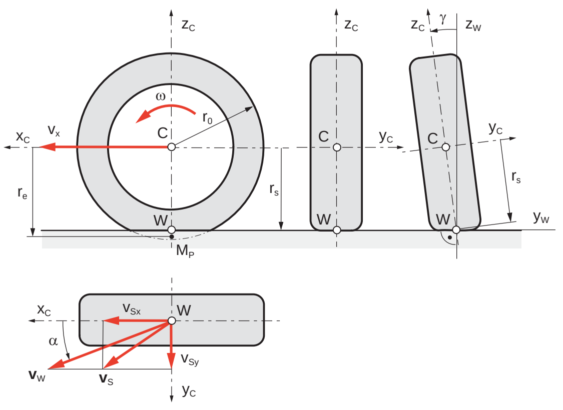
The vehicle is the controlled system. It moves according to the given inputs (acceleration-, deceleration-pedal positions, and steering-wheel angle) in-between its modeled (physical) limits.

Additionally, VDCs control the stability of the car. The reaction of those controllers is much faster than those of common drivers and there are more possibilities, like individual control of brakes per wheel. The main goal is to assist the driver to maintain control of the vehicle. These systems ensure stability and steerability during critical situations.
It is not a goal to replace the driver or take any responsibility off him. This is part of autonomous driving, which is not part of this thesis.

The surrounding environment influences all systems. For instance, a pedestrian, suddenly crossing the street, has to be detected by the driver, followed by a reasonable reaction to avoid collision. Another example is a change of the road-friction, e.g. after a tunnel with different climate conditions, like rain or snowfall. Thus basically, those changes result in errors or disturbances of output-values, which need to be handled by VDCs and the driver.

2.1.2. Tire characteristics

2.1.2.1. Slip



ω	...	wheel-speed in [rad/s]
v_x	...	longitudinal velocity in [m/s]
r_0	...	undeformed radius in [m]
r_s	...	static radius in [m]
r_e	...	effective radius in [m]
v_{Sx}	...	longitudinal slip-speed in [m/s]
v_{Sy}	...	lateral slip-velocity in [m/s]
α	...	slip-angle in [rad]
γ	...	camber angle in [rad]

Figure 2.2.: Tire kinematics, [5], [6]

no slip	brake slip		slip (traction)	
$v_x = \omega\, r_e$	$v_x > \omega\, r_e$		$v_x < \omega\, r_e$	
rolling wheel	braked wheel	blocking wheel	driven wheel	spinning wheel
$\lambda = 0$			$\lambda = \dfrac{\omega\, r_e\, -\, v_x}{\lvert\omega\rvert\, r_e}$	$\lambda = 1$
$\lambda_B = 0$	$\lambda_B = \dfrac{\omega\, r_e\, -\, v_x}{\lvert v_x\rvert}$	$\lambda_B = -1$		

Table 2.1.: Longitudinal slip definitions, [5], [6]

Figure 2.2 introduces the kinematic definitions of a vehicle's tire. In table 2.1 the slip (λ in [1] or [%]) is defined for acceleration and braking. The static radius (r_s) equals the distance between center (C) and wheelpoint (W). Basically, the slip is the ratio between the wheel-speed (ω) and the vehicle-velocity (v_x). Acceleration (or braking) causes slip depending on current torques and friction-conditions.

For braking the maximum slip (λ_B=-1 or -100%) is reached, when wheels start locking. In case of accelerating, spinning, wheels, above equation has its maximum slip at +1 (or 100%).
Driven wheels can turn n-times as fast as their reference, non-driven wheels. Therefore, slip of the implemented TCS-controllers is calculated as:

$$\lambda = \frac{\omega\, r_e}{v_x} - 1$$

2.1.2.2. Tire characteristics

Tires transfer the forces (or torques) from the engine (powertrain) onto the road. During braking the forces are also transferred from the brakes to the road. The maximum transferable longitudinal force (F_x in [N]) depends on the friction (μ in [1]), vertical contact force (F_z in [N]), and the current slip (λ in [%]).
Figure 2.3 gives examples of F_x in relation to λ for several constant F_z-values. Other influencing parameters (like tire-pressure and velocity) are kept constant.

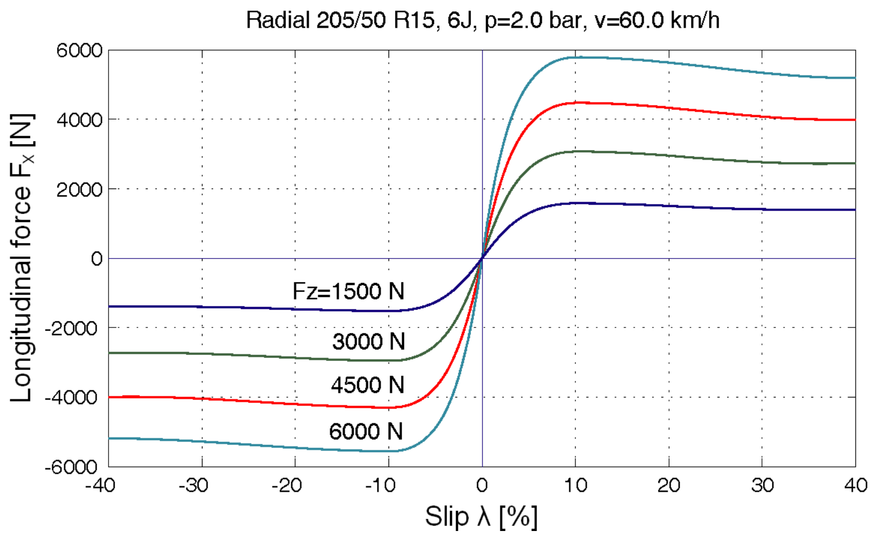


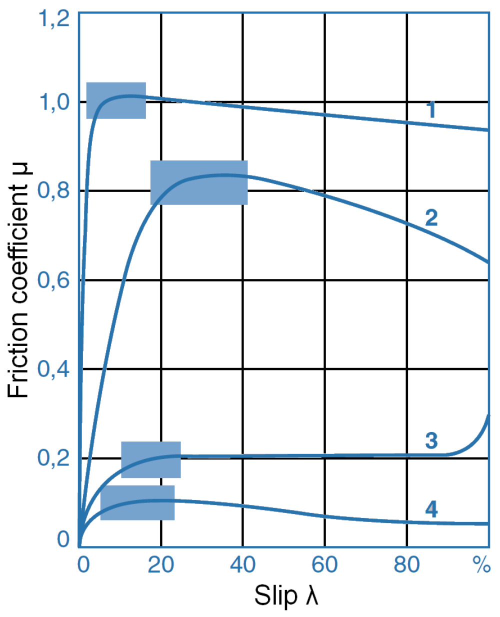
Figure 2.3.: Longitudinal force characteristics, [5], [6]

The friction-coefficient can be evaluated at the maximum as:

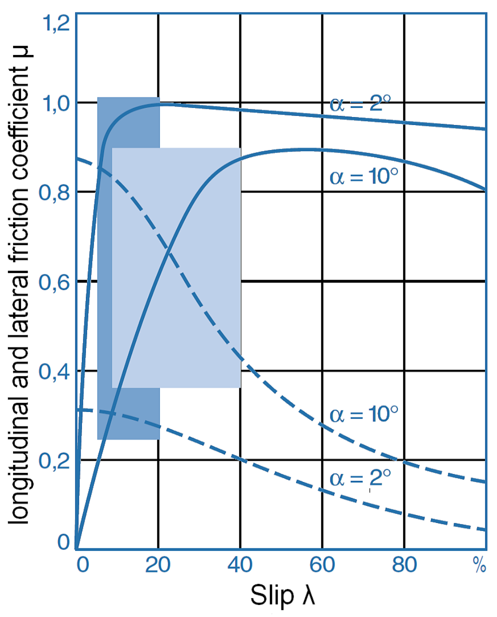
$$\mu = \frac{F_{x,max}}{F_z}.$$

The friction-coefficient (μ) results from the tire-road-combination and conditions. A lower friction means lower maximum transferable forces and, therefore easier wheel-spin or lock-up. For instance, braking on ice (low friction coefficient, μ_{ice}=0.2) results in longer braking distances compared to dry tarmac.

The tire's lateral force is similar to the longitudinal force characteristics. The resulting circle of forces (combining longitudinal and lateral forces) is shown in figure 2.5.



(a) road-tire-combinations



(b) drift angles

Figure 2.4.: Brake slip characteristics, [17]

Figure 2.4a provides friction over brake-slip curves for different road-surfaces and tires:

1. Radial tire on dry concrete
2. Bias winter-tire on wet tarmac
3. Radial tire on loose snow
4. Radial tire on icy roads

At first, with increasing slip also friction-values increases, reaching their maximum at the blue-shadowed areas. Thereafter, the friction and, therefore, the maximum transferable longitudinal force decrease again. For braking on loose snow or gravel, the resulting wedges increase friction during blocking, sliding wheels (see figure 2.4a, line 3).

Figure 2.4b combines longitudinal (solid lines) and lateral (dashed lines) friction coefficients at different slip angles (α in [°]) and slips (λ in [%]). The blue-shadowed areas present the optimal ABS-areas. At higher slip angles the longitudinal friction decreases and the optimal control-area has to be extended.

ABS-systems try to keep the tires braking within the blue-shadowed (optimal) areas. There are very similar curves at acceleration providing optimal slip-values for TCS-controllers.

2.1.2.3. Combined tire-forces

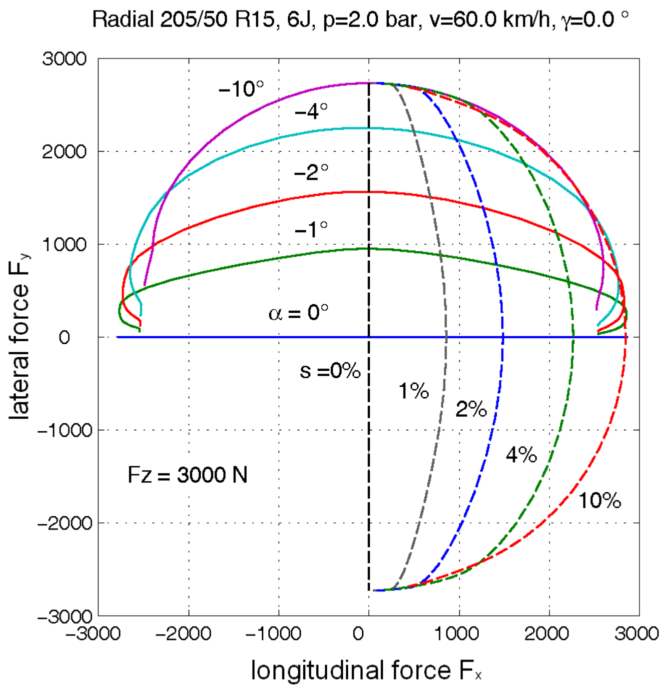


Figure 2.5.: Circle of forces (*"Kammscher Kreis"*), [5], [6]

Combining longitudinal and lateral tire forces results in a circle of forces, which is shown in figure 2.5.

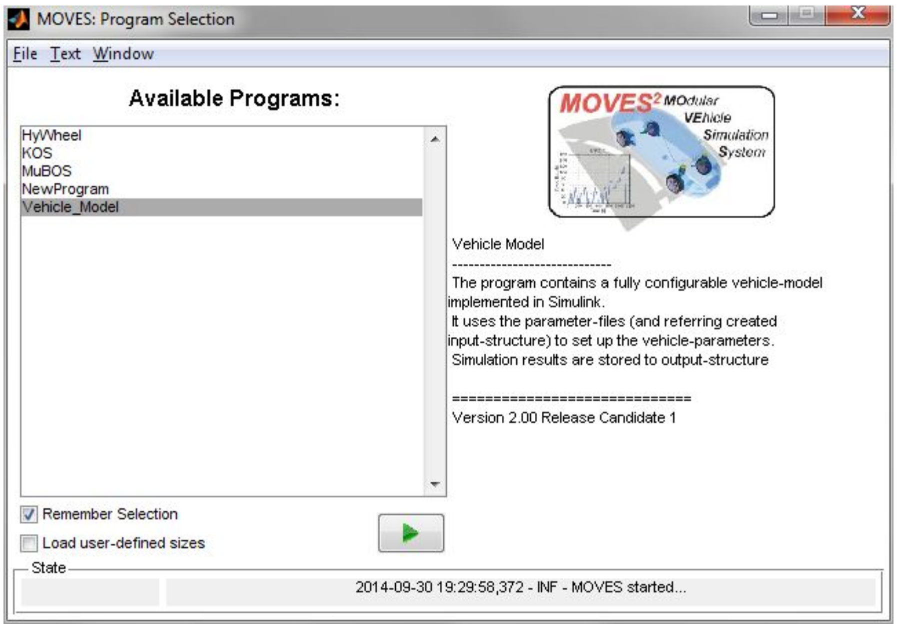
During normal driving conditions (e.g. when braking or driving through curves) there is always a combination of longitudinal (F_x in [N]) and lateral (F_y in [N]) forces. The maximum transferable lateral forces depend on the slip angle (α in [°]), whereas the maximum transferable longitudinal forces depend on the slip (λ in [%]).

The wrapping circle of forces defines stable driving conditions. It is also known as *"Kammscher Kreis"*, named after Wunibald Kamm. ABS- and TCS-controllers start working, when the tire-forces tend to leave this circle, trying to hold them or bring them back inside stable conditions.

2.2. MOVES^2 (Vehicle Model)

As mentioned before MOVES can be seen as two kinds of programs (see chapter 1.1).

First, there is the GUI which allows to configure and set up several applications, dialogs, programs and functions, [14]. And there is the initial application [18], which includes several vehicles models implemented in Simulink (see appendix, figure A.1).



Vehicle Model

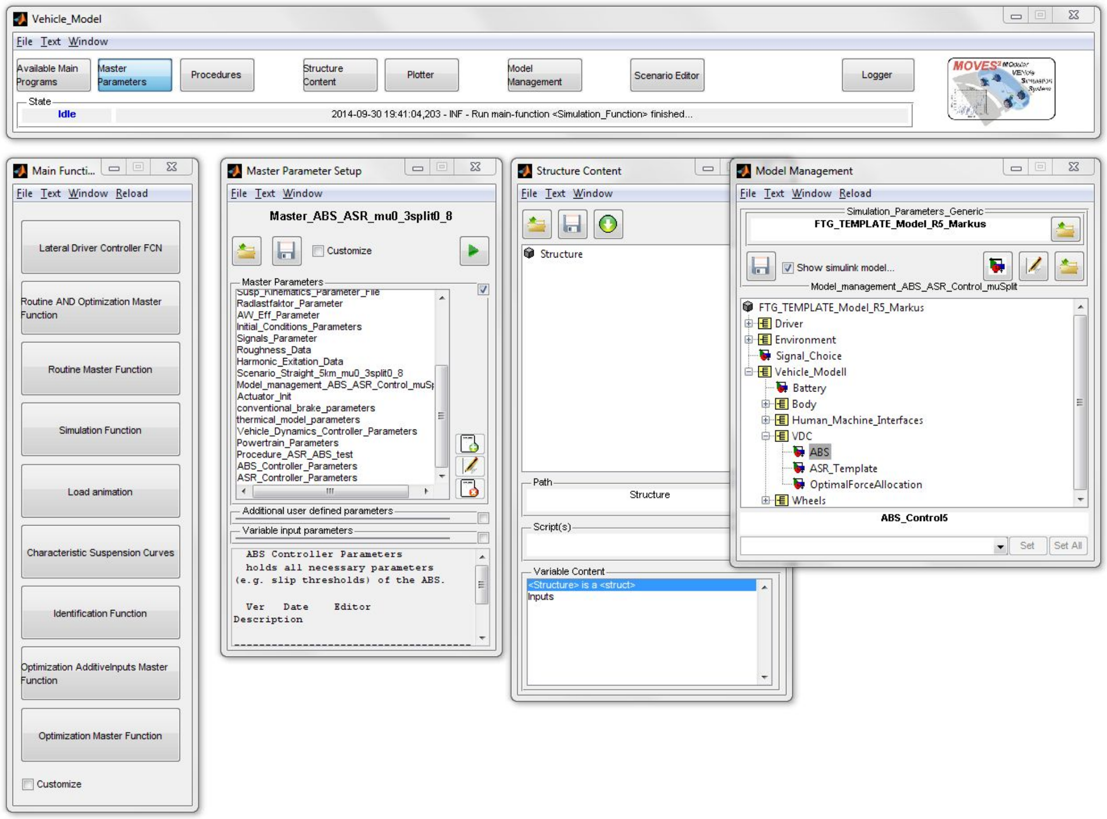
The program contains a fully configurable vehicle-model implemented in Simulink.
It uses the parameter-files (and referring created input-structure) to set up the vehicle-parameters.
Simulation results are stored to output-structure

=============================
Version 2.00 Release Candidate 1

2014-09-30 19:29:58,372 - INF - MOVES started...

Figure 2.6.: MOVES: Program Selection

Figure 2.6 shows the program selection window after MOVES has been started. The application, *Vehicle Model*, is currently selected. Besides this initial, name-giving, application, some further programs have been developed at the FTG, like KOS or MuBOS [12].



```
Idle    2014-09-30 19:41:04,203 - INF - Run main-function <Simulation_Function> finished...
```

Master_ABS_ASR_mu0_3split0_8

```
Susp_Kinematics_Parameter_File
Radlastfaktor_Parameter
AW_Eff_Parameter
Initial_Conditions_Parameters
Signals_Parameter
Roughness_Data
Harmonic_Exitation_Data
Scenario_Straight_5km_mu0_3split0_8
Model_management_ABS_ASR_Control_muSp
Actuator_Init
conventional_brake_parameters
thermical_model_parameters
Vehicle_Dynamics_Controller_Parameters
Powertrain_Parameters
Procedure_ASR_ABS_test
ABS_Controller_Parameters
ASR_Controller_Parameters
```

```
  ABS Controller Parameters
  holds all necessary parameters
(e.g. slip thresholds) of the ABS.

  Ver   Date    Editor
Description
------------------------------------
```

```
<Structure> is a <struct>
Inputs
```

FTG_TEMPLATE_Model_R5_Markus

Model_management_ABS_ASR_Control_muSplit

```
FTG_TEMPLATE_Model_R5_Markus
├ Driver
├ Environment
├ Signal_Choice
└ Vehicle_Modell
   ├ Battery
   ├ Body
   ├ Human_Machine_Interfaces
   ├ VDC
   │  ├ ABS
   │  ├ ASR_Template
   │  └ OptimalForceAllocation
   └ Wheels
```

ABS_Control5

Figure 2.7.: Overview: Vehicle Model

An overview of a possible scenario of opened dialogs and functions is shown in figure 2.7. In this case there is the main program-window at top of the screen, holding the buttons to open the referring sub-dialogs and programs. Below, from left to right, there are the following sub-dialogs opened:

Available Main Functions
: holds the program-specific functions (e.g. simulation function).

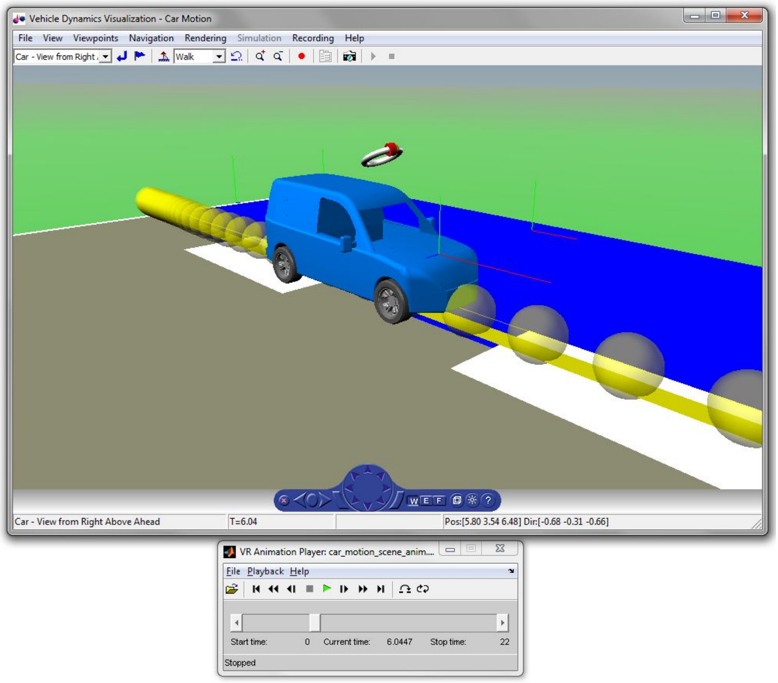
Master Parameter Setup
: collects all necessary parameter-files and creates the input-structure.

Structure Content
: holds the input- and resulting output-data-structures.

Model Management
: is used to configure the underlying Simulink-model and its block-choice selections.

A description of used dialogs will be presented in the following sections. The complete description of all dialogs can be found in [14].

After running a simulation, the result can be viewed as animated movie shown in figure 2.8. This provides a more imaginable, seizable output, than just arrays of numbers.


Vehicle Dynamics Visualization - Car Motion
File View Viewpoints Navigation Rendering Simulation Recording Help
Car - View from Right...
Walk
VR Animation Player: car_motion_scene_anim...
File Playback Help

Car - View from Right Above Ahead T=6.04 Pos:[5.80 3.54 6.48] Dir:[-0.68 -0.31 -0.66]

Start time: 0 Current time: 6.0447 Stop time: 22

Stopped

Figure 2.8.: Animated simulation result

The underlying Simulink-model is displayed in appendix, figure A.1.

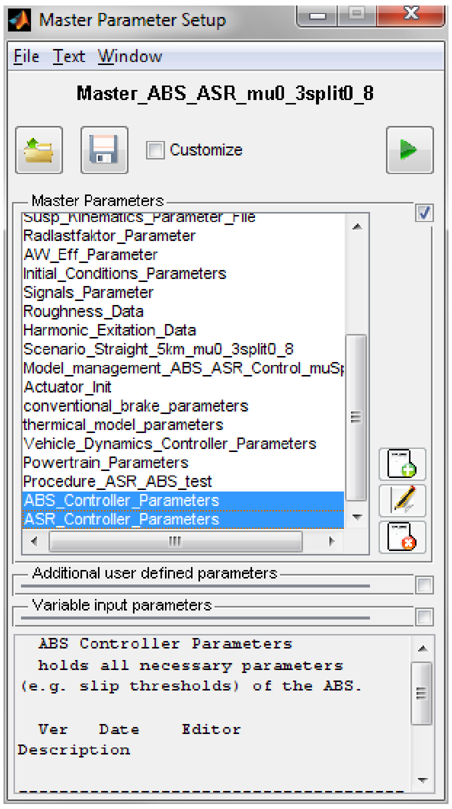
2.2.1. Master Parameter Setup

The dialog, *Master Parameter Setup*, collects all parameter-files referring to a simulation-scenario within a so called master-parameter-file. It is used to create and initialize the input-structure for the simulation.

Figure 2.9a shows the dialog, with the currently loaded master-parameter-file, `Master_ABS_ASR_mu0_3split0_8.m`, displayed at the top. This master-parameter-file simulates a combined ABS- and TCS-test-scenario on a μ-split surface (left side low friction (μ=0.3), right side high friction (μ=0.8))

All necessary parameters needed for the ABS- and TCS-models and controllers are stored in referring parameter-files, `ABS_Controller_Parameters.m` and `ASR_Controller_Parameters.m`.
Those two parameter-files were additionally added to the existing ones.

See appendix for complete file listings A.1 and A.2



Master_ABS_ASR_mu0_3split0_8

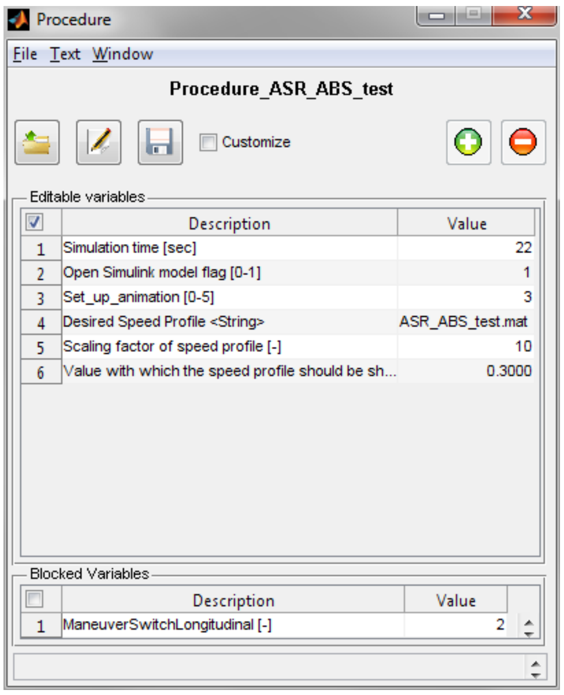
- Susp_Kinematics_Parameter_File
- Radlastfaktor_Parameter
- AW_Eff_Parameter
- Initial_Conditions_Parameters
- Signals_Parameter
- Roughness_Data
- Harmonic_Exitation_Data
- Scenario_Straight_5km_mu0_3split0_8
- Model_management_ABS_ASR_Control_muSp
- Actuator_Init
- conventional_brake_parameters
- thermical_model_parameters
- Vehicle_Dynamics_Controller_Parameters
- Powertrain_Parameters
- Procedure_ASR_ABS_test
- ABS_Controller_Parameters
- ASR_Controller_Parameters

```
  ABS Controller Parameters
  holds all necessary parameters
(e.g. slip thresholds) of the ABS.

  Ver   Date    Editor
Description

----------------------------------------
```

(a) Master Parameter Setup



Procedure_ASR_ABS_test

	Description	Value
1	Simulation time [sec]	22
2	Open Simulink model flag [0-1]	1
3	Set_up_animation [0-5]	3
4	Desired Speed Profile <String>	ASR_ABS_test.mat
5	Scaling factor of speed profile [-]	10
6	Value with which the speed profile should be sh...	0.3000

	Description	Value
1	ManeuverSwitchLongitudinal [-]	2

(b) Procedure

Figure 2.9.: Dialogs

2.2.1.1. Parameter-Files

Listing 2.1: Parameter-file (header)

```
% ABS Controller Parameters
% holds all necessary parameters (e.g. slip thresholds) of the ABS.
%
% Ver   Date    Editor                   Description
%---------------------------------------------------------------------
% 0.1   140221  Manfred Großmann         initial version

g = 9.81; % [m/s^2] gravity

%% Parameter Fields List
Structure_Position = 'Parameters.Vehicle_Dynamics_Controller';
Structure_Fields   = {'ABS'...
                      };

PreProcessing_FCN  = '';
```

Parameter-files store all necessary simulation-parameters. As an example listing 2.1 shows the header of the `ABS_Controller_Parameters.m`-file. The whole file can be found in appendix, listing A.1.

Line 1-6: Comments at the top describe the functionality of the current file.

Line 8: Some temporary (auxiliary) variable is defined (gravity-constant g).

Line 11: `Structure_Position` defines the position within the input-structure.

Line 12-13: `Structure_Fields` define several fields within the superior structure position.

Line 15: `PreProcessing_FCN` defines a function, which is run at structure-initialization (prior to simulation).

2.2.1.2. Procedures

The *Procedure*-dialog allows simple changes of the simulation-parameters without changing the underlying parameter-files (e.g. simulation-time).
Figure 2.9b shows an example of a loaded procedure-script.

2.2.1.3. Scenarios

Scenarios are collecting the road-layout and -friction. Furthermore, a sub-dialog is used to set up speed-profiles.

2.2.1.4. Model Management

This dialog is used to select, change, and adapt the used Simulink-model. Block-choices allow to set up the vehicle-model to any detail needed for the current simulation-scenario.

Block-Choices Complex models, such as the vehicle-model, demand a powerful simulation-computer to produce results in reasonable time. There are several ways to decrease the simulation time:

- Use a more powerful computer:
  That's usually limited by costs.
- Use parallel computing:
  That's a good possibility for optimizations (when several results for similar simulations are run at the same time) but might not work for a single simulation.
- Increase the simulation-step time:
  This decreases the output-precision and might have side effects as aliasing (and, therefore, cause wrong results).

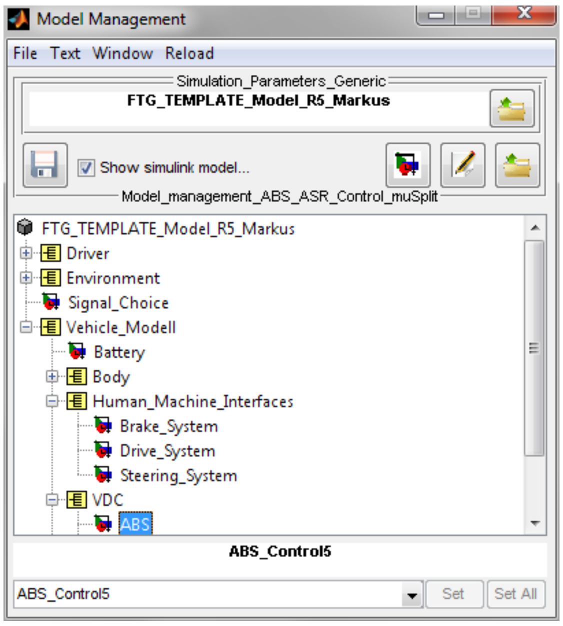


Figure 2.10.: Model Management

- Simplify the (whole) model:
  This might also decrease the output-accuracy.
- Keep the (whole) model as simple as possible, but increase the complexity of the important, examined, parts.

The latter provides the necessary output-precision where needed, while the rest of the model stays reasonable simple and, therefore, fast. This seems to be a good compromise.

Simulink provides the possibility to build configurable subsystems, so called block-choices. This easily allows to switch sub-systems and, hence, define the whole model according to the selected block-choices. The model is adaptable to the specific simulation-scenario requirements (e.g. when testing only acceleration behavior there is no need to set up brakes - or just use a simplified model in this case). Thus, the complexity of the model is adaptable.

All the block-choice selections of the complete model are stored in parameter-files.

2.2.2. Vehicle Model

The vehicle-block is part of the whole Simulink-model, see appendix, figure A.1 (with highlighted green vehicle-model). Figure 2.1 gives a schematic overview of the vehicle as part of control-system.

The vehicle is the system which is controlled primary by the driver (outer loop). At the inner control-loop, vehicle dynamic controllers assist the driver by keeping the car stable and steerable.

Figure 2.11 shows the generic layout of the vehicle-model.
The complete model implemented in Simulink can be found in the appendix, figure A.2.

2.2.2.1. Design

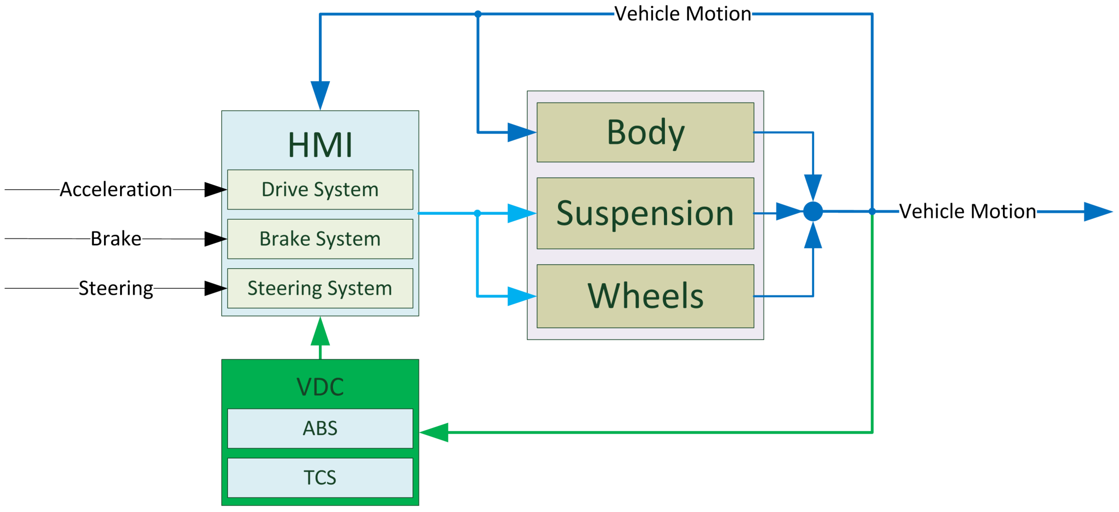


Figure 2.11.: Vehicle Model (schematic)

The model consists of three main blocks, which form the vehicle. The Human Machine Interface (HMI) serves as interface between driver and vehicle and transforms the provided input-signals to forces and torques onto the wheels. Wheels, suspension and body are connected together. Thus, when forces are applied to wheels, the whole vehicle moves accordingly.

The output-signal, vehicle motion, is a collection of movement-information, such as e.g. tire-forces, and -torques, coordinates of the vehicle, speeds, accelerations, etc. The VDC (Vehicle Dynamic Controller) tries to detect and avoid unstable driving-conditions and directly interact with the vehicle (like e.g. hold brake-pressure or reduce engine-power).

Two signals are not displayed in this schematic model, but are not less important:

First, the (outer) environment influences the whole vehicle (e.g. change of road-friction, tight curves,...).
Second, the VDCs inform the driver, when active. Thus, the driver might notice a critical situation and react by himself (e.g. by reducing the speed).

Input signals (driver-commands):
Acceleration: position of the acceleration-pedal.
Brake: position of the braking- (deceleration-) pedal.
Steering: steering-wheel angle.

Human Machine Interface - HMI:
The HMI calculates steering-, powertrain- and brake-signals. It consists of three blocks as shown in figure 2.11. The complete Simulink-model can be found in the appendix, figure A.3.

Drive System simulates the current drive-torque (M_d) per wheel (green in figure A.3).

Brake System creates the current brake-torque (M_b) for each wheel (red in figure A.3).

Steering System calculates the steer-angle (δ) per wheel.

Additionally to the driver's input-commands, the current vehicle-motion and interacting VDCs influence the results.

Vehicle Dynamics Controller - VDC:
The VDC is interacting with the HMI, when critical situations are detected to avoid e.g. locking wheels during braking. It consists of two blocks, ABS and TCS, as shown in figure 2.11. The complete Simulink-model can be found in the appendix, figure A.4. It holds a third block, `|OptimalForceAllocation|`, which refers to torque-vectoring, which is not part of this thesis.

ABS interacts with the brake system to avoid blocking wheels during braking.

TCS interacts with the powertrain and brake system to avoid spinning wheels.

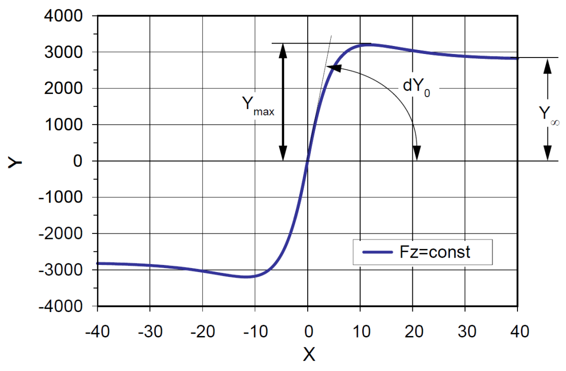
Input-parameters to ABS are usually only wheel-speeds for each tire. Additionally, the TCS needs the current engine-torque. Further signals and parameters are necessary to de-/activate the currently useful controller (e.g. ABS is only activated during braking, if the vehicle's speed is higher than 2.5km/h, [17]).

The VDC-outputs are directly linked to the HMI input. ABS signalizes to the hydro-unit of the brake-system to increase (follow input), hold, or decrease brake-pressures for each wheel. TCS also influences the hydro-unit, but additionally decrease the engine-torque in the powertrain-model, if needed.

Body, Suspension, and Wheels
Those blocks create the resulting motion (of the complete vehicle).

2.2.2.2. Tire-model

The used tire-model, TMsimple, relies on a mathematical approximation of real tire-curves, [5].



$$Y = K \cdot \sin\left(B\left(1 - e^{\frac{-|X|}{A}}\right)\text{sign}\,(X)\right)$$
$$K = Y_{max}$$
$$B = \pi - \sin^{-1}\left(\frac{Y_\infty}{Y_{max}}\right) \qquad (Y_\infty \leq Y_{max})$$
$$A = \frac{1}{\mathrm{d}Y_0} \cdot K \cdot B$$

Figure 2.12.: Tire-model: TMsimple, [17], [21]

Figure 2.12 shows a typical tire-curve for longitudinal slip-variable (X in $[\%]$[1]) and force-variable (Y in [N]) at constant vertical contact force (F_z in [N]). Initially, while slip increases, the maximum transferable force increases (almost linear), too. After the force reaches its maximum (Y_{max} in [N]), with increasing X, it decreases again and approaches its saturation value (Y_∞ in [N]).

The parameter K equals the maximum of the force-variable (Y_{max}). Variable B is defined by the ratio between Y_∞ and Y_{max}. Variable A depends on K, B an the initial derivation ($\mathrm{d}Y_0$).

The current wheel-speed (ω in [rad/s]) is simulated using following equation:

$$J_{rot} \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} = M_d + M_b + M_y$$

M_d	...drive torque in [Nm] provided by the engine trough transmission and final drive.
M_b	...brake torque in [Nm] from driver and TCS.
M_y	...current torque in [Nm] is provided by the tire-model.
J_{rot}	...is the rotary inertia in [kgm^2].

[1]In case of lateral slip X in [rad]

2.3. Modeling

This section describes the points of intervention within MOVES. The referring models of ABS, TCS and the adapted brake- and drive-systems are explained in detail.

2.3.1. Antilock Braking System - ABS

As already described in chapter 1, ABS-controllers are used to avoid locking wheels during braking and keep the car stable.

The descriptions of typical ABS-cycles (in [3], [17]) serve as basis for the implementation in MOVES.

ABS-controllers interact with the brake-system. In normal braking situations, the driver pushes the brake-pedal, which increases the brake-pressure. It is further increased or multiplied by a booster and transferred to the brakes. At the brakes, the hydraulic brake-pressure presses the pads against the disks. This results in a brake-torque against the current direction of tire-movement.
When the ABS detects a possible locking condition, the pressure is adapted to the current situation. This is done by switching valve positions within the hydro-unit. In real-world an electric driven pump is used to increase or decrease the pressure by ABS (and TCS). As simplification the implemented model of the hydro-unit just consist of hydraulic valves, which allows the ABS to either increase (follow the input-), hold, or decrease the pressure for each wheel.

The implementation itself is done using Stateflow-models, [1], to keep close to state-control-logic of electronic controllers.

2.3.1.1. Typical ABS cycle

Figure 2.13 shows at top the velocities of the car (v_F) and wheel (v_R). The reference-speed (v_{Ref}) and referring slip-threshold (λ_1) are calculated by the ABS. In the middle part of the figure, the wheel's de-/acceleration (b_t) and according thresholds ($-a$, $+a$, and $+A$) are plotted. At bottom the referring brake pressure (p_{hydr}) is shown.

Description of the steps:

1. When starting to brake, p_{hydr} increases, the wheel and the car decelerate.
   The difference between v_R and v_F increases.

2. b_t decreases abruptly.
   When it falls below threshold $-a$, ABS holds the pressure.
   Now the v_{Ref} has to be extrapolated.

3. After v_R falls below the λ_1, p_{hydr} is decreased by the ABS.
   A (possible) locking has been avoided, as b_t starts to increase again.

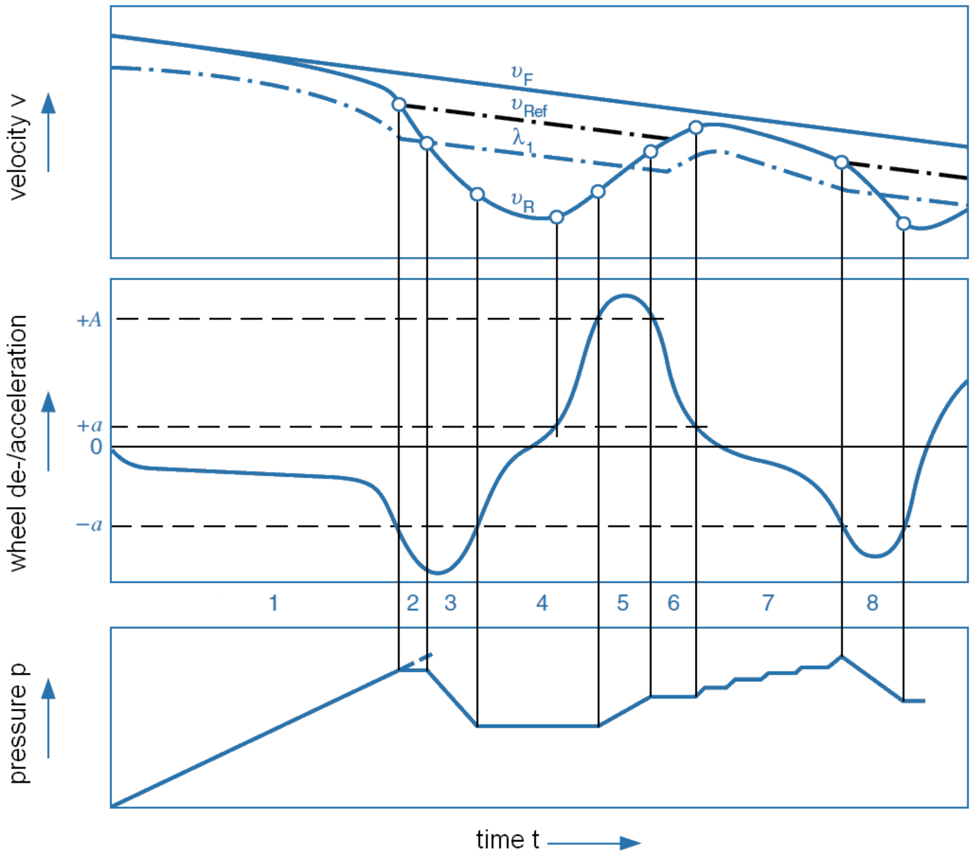


Figure 2.13.: Typical ABS cycle, [17]

4. When b_t rises above $-a$, p_{hydr} is held again. The wheel starts to accelerate.
5. b_t reaches the $+A$ threshold, and v_R approaches v_{Ref}.
   p_{hydr} is increased, until the b_t falls below $+A$ again.
6. Now ABS holds p_{hydr}, until b_t falls also below $+a$.
   v_R is near to v_F. The situation stabilizes.
7. During this state, p_{hydr} is efficiently increased stepwise, until b_t falls below $-a$. The increase-pressure-time is adjustable and recalculated at each iteration according to previous amount of steps.
8. Now p_{hydr} is immediately decreased (like in step 3), the ABS-cycle restarts itself.

2.3.1.2. Slip- or locking- detection

ABS-controllers usually rely only on the measured wheel-speeds. The vehicle's real velocity is needed to calculate the current slip-values. But it cannot be easily measured, or rather there are no reasonable, low-cost sensors to measure the current vehicle's velocity. Thus, it is only estimated through reference-speed, which is usually only the mean-value of two crossover wheels. During braking, the difference between the real vehicle-velocity and the reference-speed increases. If tires are slipping or locking, the mean value would result in insufficient precise values of the reference-speed to calculate proper slip-values. Therefore, another, measurable unit has to be used to detect and avoid blocking wheels.

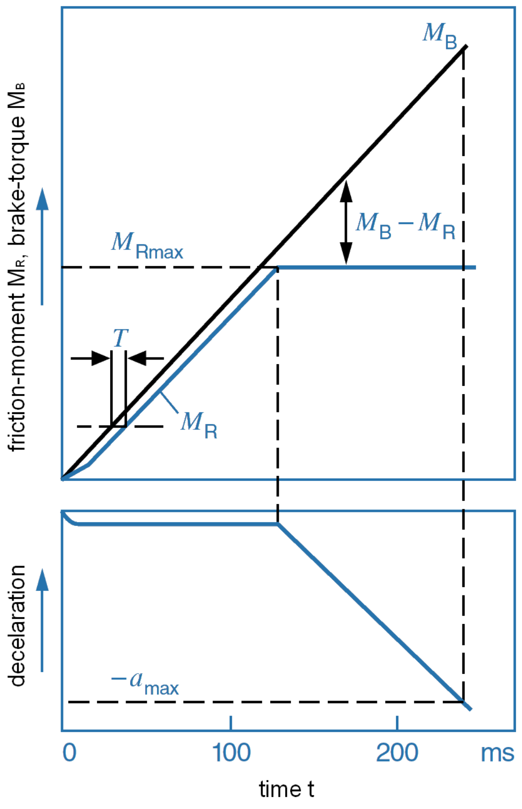


Figure 2.14.: Slip- or locking- detection (concept), [17]

Figure 2.14 shows the concept of the slip- or locking-detection:
At top, the brake-torque (M_B) increases. The friction torque (M_R) between tires and road follows delayed. The wheel at bottom decelerates accordingly, keeping deceleration-values (b_t) reasonable low and stable.
At a certain point, depending on the tire-road-conditions, the M_R reaches its maximum (M_{Rmax}) and, therefore, cannot increase any further. If M_B is still increasing, b_t abruptly decreases, until the wheels completely lock up (reaching $-a_{max}$).

This behavior of b_t is in contrast to the slip-value, which increases to its maximum and stays there (this is a simplification, see figures 2.3, 2.4, and 2.12 for realistic curves). Therefore, the properties of b_t are perfect to start the ABS-cycles.
The combination of b_t and the calculated reference-speed with its referring slip-thresholds is precise enough to control braking.

2.3.1.3. Implementation in Simulink

In appendix, figure A.4, the ABS is implemented as a block-choice-model within the VDC-block. There exists already a template and some previous models, but they use and control the simulated slip-values, which are usually not accessible in real-world. In appendix, figure A.5, the newly implemented ABS is shown completely.

Following figure 2.15 shows the input-signals and the first two of four ABS-controllers for the front-left and -right wheels.

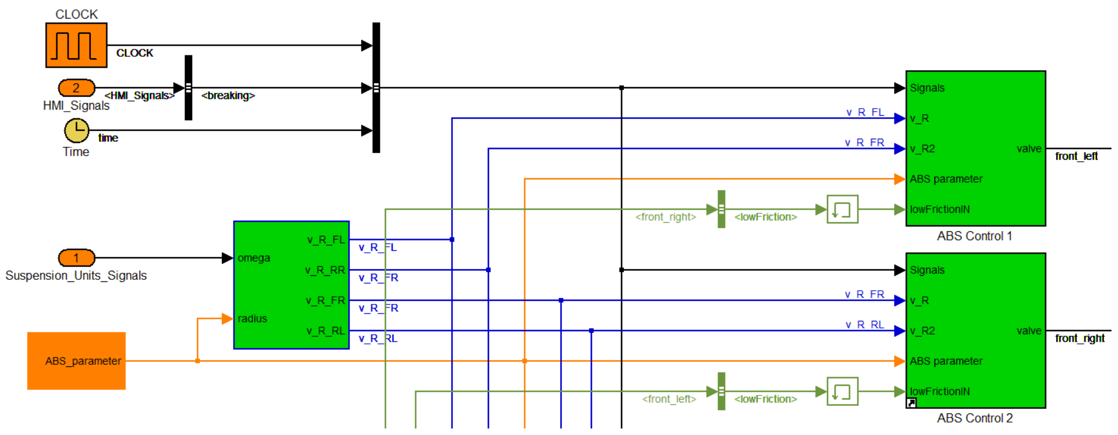


Figure 2.15.: ABS: selection

Inputs

Two input-signals are used: From the `<HMI_Signals>` (bus-signal) the input `<braking>` is used to turn the ABS-controller on and off. The `<Suspension_Unit_Signals>` (bus-signal) provide the wheel-speeds (`<omega>`) for all wheels.
Additionally, the current simulation-time (`<time>`) is needed to calculate de-/acceleration values and other derivations. The `<CLOCK>`-signal triggers the discrete Stateflow-controller. The `|ABS_parameter|`-block provides the needed/initial parameters.

Blocks
In the only additional (nameless) block the wheel speeds (`[v_R_XY]`) are calculated and filtered for each wheel.

$$v_R = \omega \cdot r_{eff}$$

v_R	...wheel speed in [rad/s].
r_{eff}	...effective tire radius in [m].

As a simplification of the effective (dynamic) tire radius a constant parameter is used. The filter is necessary to get rid of higher frequencies and disturbances.
`X` indexes the longitudinal direction of the vehicle, where `F` refers to the front and `R` to the rear wheels. `Y` indexes the lateral direction of the vehicle, where `L` refers to the left and `R` to the right side.

The four remaining blocks are the ABS-controllers (`|ABS Control n|`, n=1-4).

Outputs
As a single bus-signal, `[valve_control_ABS]` combines all signals from all four wheels to the only used output.
The existing outputs from previous models have been grounded, as they couldn't be reused for new purposes.

2.3.1.4. Implementation in Stateflow

Figure 2.16 looks inside the front-left wheel's ABS-block (`|ABS Control 1|`). The provided input-structures (bus-signals) are divided and forwarded as single input-signals to the Stateflow-model.

Table 2.2.: ABS controller: input-signals

<Input>	Description
`<CLOCK>`	This event triggers the state-machine. It is used to switch between states.
`<braking>`	This Boolean value signals the start and end of braking. It starts and stops the ABS-controller.
`<time>, <t>`	Simulation time is needed to calculate derivations.
`<v_R>`	The longitudinal wheel-speed of the controlled tire is used to calculate wheel's de-/acceleration (`b_t`).
`<v_R2>`	Crossover tire's wheel-speed is needed to estimate the reference-speed (`v_Ref`).
`<lowFrictionIN>`	Signals this current controller an actively working ABS-controller at the opposite wheel. This is needed during initial braking to decrease yaw-rate at μ-split-conditions.
	End of table

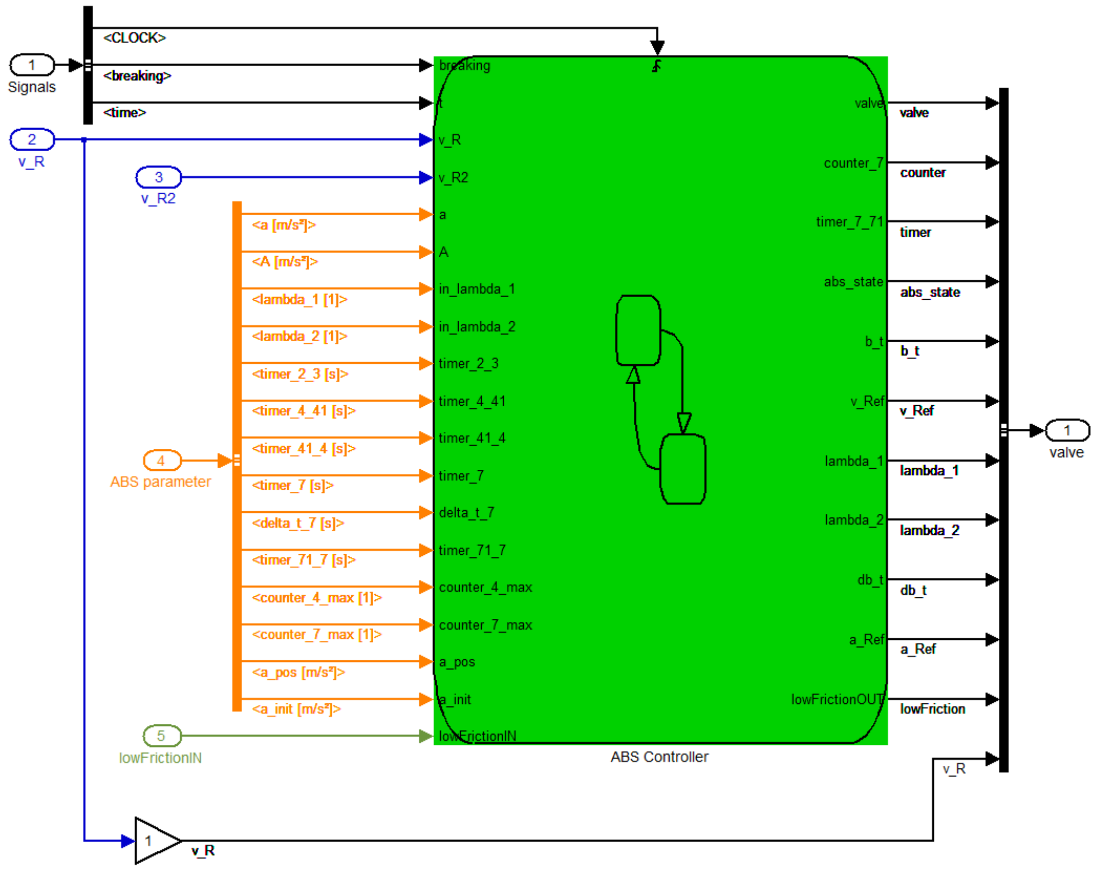


Figure 2.16.: ABS: Control 1 (front left wheel)

Table 2.3.: ABS controller: input-parameters

Parameter	Description
`a_init`	The initial (negative) wheel-deceleration threshold to start controlling in the first place.
`a`	(Negative) wheel-deceleration threshold to continue/restart the ABS-cycles.
`a_pos`	(Positive) wheel-acceleration threshold indicating stable conditions after anti-lock-phase.
`A`	The higher (positive) wheel-acceleration threshold.
`in_lambda_1`	The lower slip-threshold initiates start of ABS-cycles.
`in_lambda_2`	The higher slip-threshold indicates low-friction conditions.

Continued on next page

Parameter	Description
timer_2_3	Time to wait in state |hold_pressure_2| before, unconditionally, switch to state |decrease_pressure_3| and, therefore, start the ABS-cycle.
timer_4_41	Time to wait in state hold_pressure_4 before switch to state decrease_pressure_4_1 and, therefore, stepwise decrease the pressure.
timer_41_4	Time to wait in state decrease_pressure_4_1 before switch back to state hold_pressure_4.
counter_4_max	Inner-cycle's counter-maximum. When reached, switch to next outer state.
timer_7	Initializes variable timer_7_71, which is the time to wait in state |increase_pressure_7| before switching to state |hold_pressure_7_1|.
delta_t_7	Time used to calculate the variable timer_7_71 at each iteration based upon the previous number of (inner) cycles at the puls-step-control.
timer_71_7	Time to wait in state |hold_pressure_7_1| before switch to state |increase_pressure_7| and, therefore, stepwise increase the pressure.
counter_7_max	Inner-cycle's counter-maximum. When reached, switch to next outer state.
	End of table

Table 2.4.: ABS controller: (output-) variables

Variable [Output]	Description
[valve]	Valve-position of the hydro-unit. Switches between increase or follow input-pressure, hold, and decrease brake-pressure.
[abs_state]	The state of the ABS controller to monitor its behavior.
counter_4	Counter-value of the (inner) |Anti-lock|-loop. Needed to escape the loop when counter_4_max is reached.
counter_7 [counter]	Counter-value of the puls-step-control. Needed to in- or decrease the step-size (timer) at the next iteration and breaks the loop when counter_7_max is reached.
timer_7_71 [timer]	The calculated timer-value of the puls-step-control depending on previous count of (inner) cycles.
[b_t] b1 b2	Calculated wheel-de-/acceleration value ($b_t = \Delta v_R / \Delta t$). Temporary variable to calculate mean value of b_t. Temporary variable to calculate mean value of b_t.
	Continued on next page

Variable [Output]	Description
v_R0	Temporary variable holds the previous value of v_R. Needed at the next iteration to calculate Δv_R.
[db_t] b_t0	Derivation of b_t, to signal rising or falling progress of the wheel's de-/acceleration. Temporary variable holds the previous value of b_t. Needed at the next iteration to check rising of falling b_t.
[v_Ref] v0	Estimated reference-speed v_Ref. It is extrapolated during |Anti-lock|-state. Temporary variable holds the previous value of v_Ref. Needed at the next iteration to calculate a1 or a2.
[a_Ref] a1 a2	The car's estimated deceleration is used to extrapolate v_Ref during ABS-cycle. Temporary variable to calculate mean value of a_Ref. Temporary variable to calculate mean value of a_Ref.
[lambda_1]	Estimated lower slip-threshold, which starts the ABS-cycle.
[lambda_2]	Estimated higher slip-threshold, which indicates low-friction conditions.
lowFriction	The internal variable is used to bypass several state. This results in faster cycles at low-friction-conditions.
lowFrictionOUT [lowFriction]	This output signals the opposite controller that this wheel is currently controlled by ABS to decrease yaw-moment at μ-split-conditions.
t0, t00	Temporary variable holds the previous simulation-time t. Needed to calculate Δt.
	End of table

The parameters in table 2.3 are created during initialization of the *Master Parameter Setup*. In the parameter-file ABS_Controller_Parameters.m (see appendix, listing A.1) the referring values are stored. The position within the resulting input-structure is: Inputs.Parameters.Vehicle_Dynamics_Controller.ABS.

Tables 2.2 and 2.4 describe the purpose of the input-signals, variables, and output-signals. The main output-signal is [valve], as it controls directly the position of the referring hydro-unit (for each wheel individually). The other outputs are only used for monitoring the ABS control-flow.

ABS Controller

Figure 2.17 shows the basic layout of the ABS-controller in Stateflow. There are two states: `|NotBraking|` and `|Braking|`. The input-signal `<braking>` switches between those two states, thus, turns the ABS controller on and off.

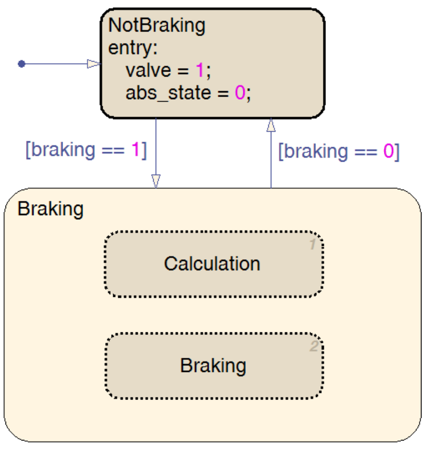


Figure 2.17.: Stateflow: ABS Controller

The `|Braking|` state consists of two parallel running states:
`|Calculation|` sets the necessary variables for each iteration.
`|Braking|` is the main state-machine of the ABS-controller.

Braking

Figure 2.18 shows the schematic Stateflow-model of the `|Braking|`-state. A complete overview can be found in appendix, figure A.11.

It is based on the initial layout as seen in appendix, figure A.10, which relies on the common ABS-cycle (see section 2.3.1.1, figure 2.13, [17]). When the driver starts braking, the `|initial braking|`-state is entered, where the pressure increases (follows the input-pressure according to the brake-pedal position). In case the ABS detects a possible locking-situation, it switches to the next state `|hold pressure|` to monitor the behavior. It might be a false positive detection (e.g. slipping on few gravels) and, therefore, it switches back to `|initial braking|`, if situation returns to normal. In case that the possible locking situation continues or worsens, the ABS control-cycle is started by entering `|decrease pressure|`-state. During the following `|Anti-lock|`-state the pressure is decreased until the wheel spins freely again. In the following `|Pulse Step Control|` the pressure is stepwise increased again, staying close to the optimal slip and returning to normal conditions (`|restart braking|`). When the ABS detects another locking-situation, it restarts its cycle.

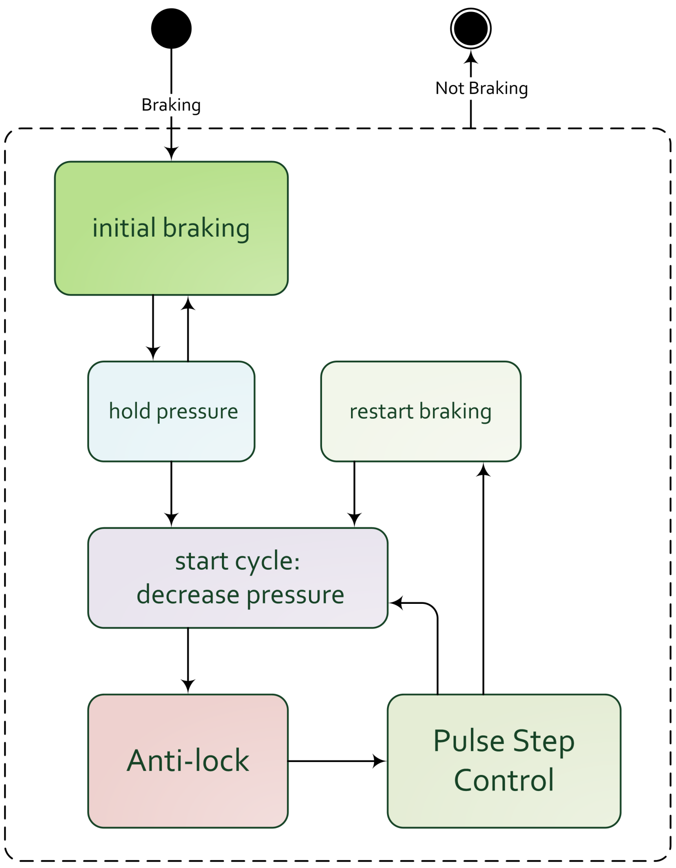


Figure 2.18.: Stateflow: ABS Controller - Braking (schematic)

Initial braking (see figure 2.19)

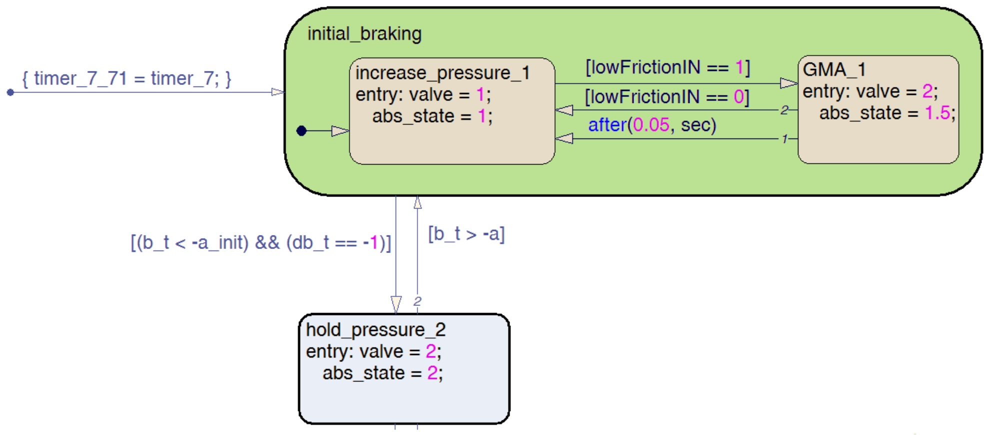


Figure 2.19.: Stateflow: ABS Controller - initial braking

The |initial_braking| state is entered, when <braking> is applied.
Variable timer_7_71 is initialized with parameter timer_7.

It consists of two internal states: State |increase_pressure_1| is entered at the beginning, the output [valve] is set to 1 (increase pressure/follow input pressure). In case that the input-signal <lowFrictionIN> is set, the state switches to |GMA_1|, where [valve] is set to 2 (hold pressure). If the signal is turned off again, or after 0.05s, the state returns to |increase_pressure_1|.

The above circle of inner states is used to avoid high yaw-moments. If the opposite wheels' ABS controller detects locking-conditions, it signals [lowFrictionOUT] to the other wheel's controller. By stepwise increasing the pressure at the high-friction sided wheel, the yaw-moment increases slower compared to full-pressure-braking. This gives the driver enough time to react and counter-steer against the upcoming yaw-moment.

If the wheel-deceleration (b_t) falls below the initial threshold (-a_init), and if its derivation (db_t) is negative, the state switches to |hold_pressure_2|.

(Re-)start cycle: hold pressure, decrease pressure, restart braking (see figure 2.20)

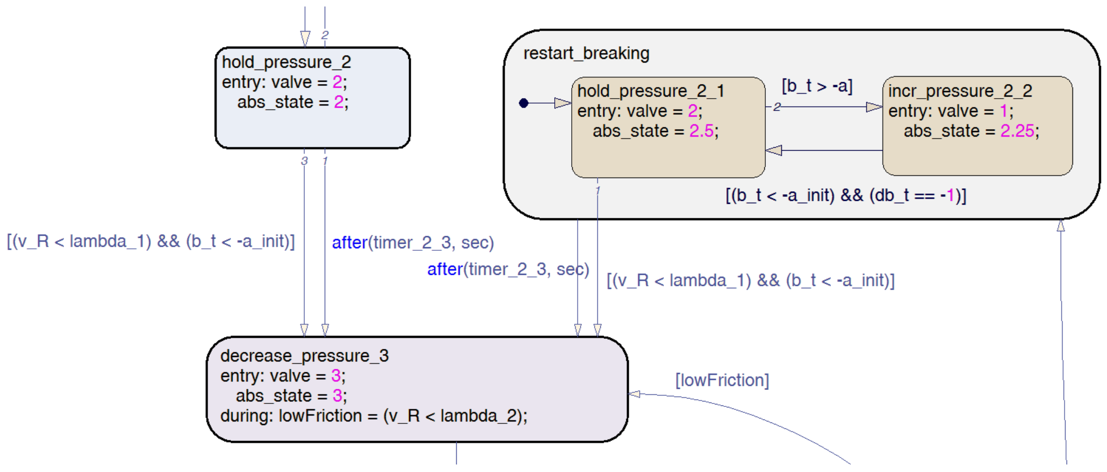


Figure 2.20.: Stateflow: ABS Controller - (re-)start cycle

The state is entered by |initial_braking|. Output [valve] is set to 2, holding the current pressure. This allows the wheel to either continue deceleration (abruptly) and therefore possible lock-up. In case that b_t rises above -a, it returns to the former state |initial_braking|.

Otherwise, if the wheel-speed (v_R) falls below the slip-threshold (lambda_1), while still meeting the former criteria (b_t<-a_init), or after waiting timer_2_3 seconds for changes, it switches to state |decrease_pressure_3|.

As ABS has detected a (possible) locking-situation, or locking is imminent, the ABS-cycle is started. Entering state |decrease_pressure_3|, the output [valve] is set to 3, decreasing the pressure. Thus, the brake-torque is reduced to avoid blocking wheels. When entering `|decrease_pressure_3|`, there is no further exit of the ABS-cycle, besides stopping braking (lifting the brake-pedal, see figure 2.17). During this state the internal variable `lowFriction` is set.

The state `|restart_braking|` consists of a inner loop switching between states `|hold_pressure_2_1|` and `|incr_pressure_2_2|`. The conditions for the transition to state `|decrease_pressure_3|` are the same as for `|hold_pressure_2|`. This constellation seems to be copied but ignores the input-signal `<lowFrictionIN>` for an already running ABS-cycle. Otherwise, ABS-controllers are unintentional influencing each other, creating an unwanted yaw-moment at homogenous friction-conditions.

Anti-lock (see figure 2.21)

If the wheel starts to accelerate and passes the threshold (`-a`) again, the `|Antilock|`-state is entered. In case `lowFriction` was set before and `b_t` is already passing by `a_pos`, it immediately switches to the next state.

During the `|Antilock|` state, the pressure is held and further, stepwise, decreased, until `b_t` passes by the positive threshold (`a_pos`), or the (inner) cycle-counter reaches its maximum. It is necessary to reduce the pressure, until the wheel starts to accelerate again.

The next state combines the initially created states of the common ABS-cycle phases four (last part), five and six (see figures 2.13 or appendix, figure A.10). It can also be directly entered from previous `|decrease_pressure_3|`-state in case of low friction. Depending on the given circumstances, the proper inner state is entered.
The intended flow would be: First, `|hold_pressure_4_2|` is entered to hold the pressure until `b_t` passes by the high threshold (`A`). Then switch to `|increase_pressure_5|` to start increasing the pressure, again. The wheel-speed narrows to the real car's velocity again. When `b_t` falls below `A` again, the pressure is held in state `|hold_pressure_6|`.

The critical situation has stabilized again. After `b_t` falls below `a_pos`, the state switches to `|PulsStepControl|`.

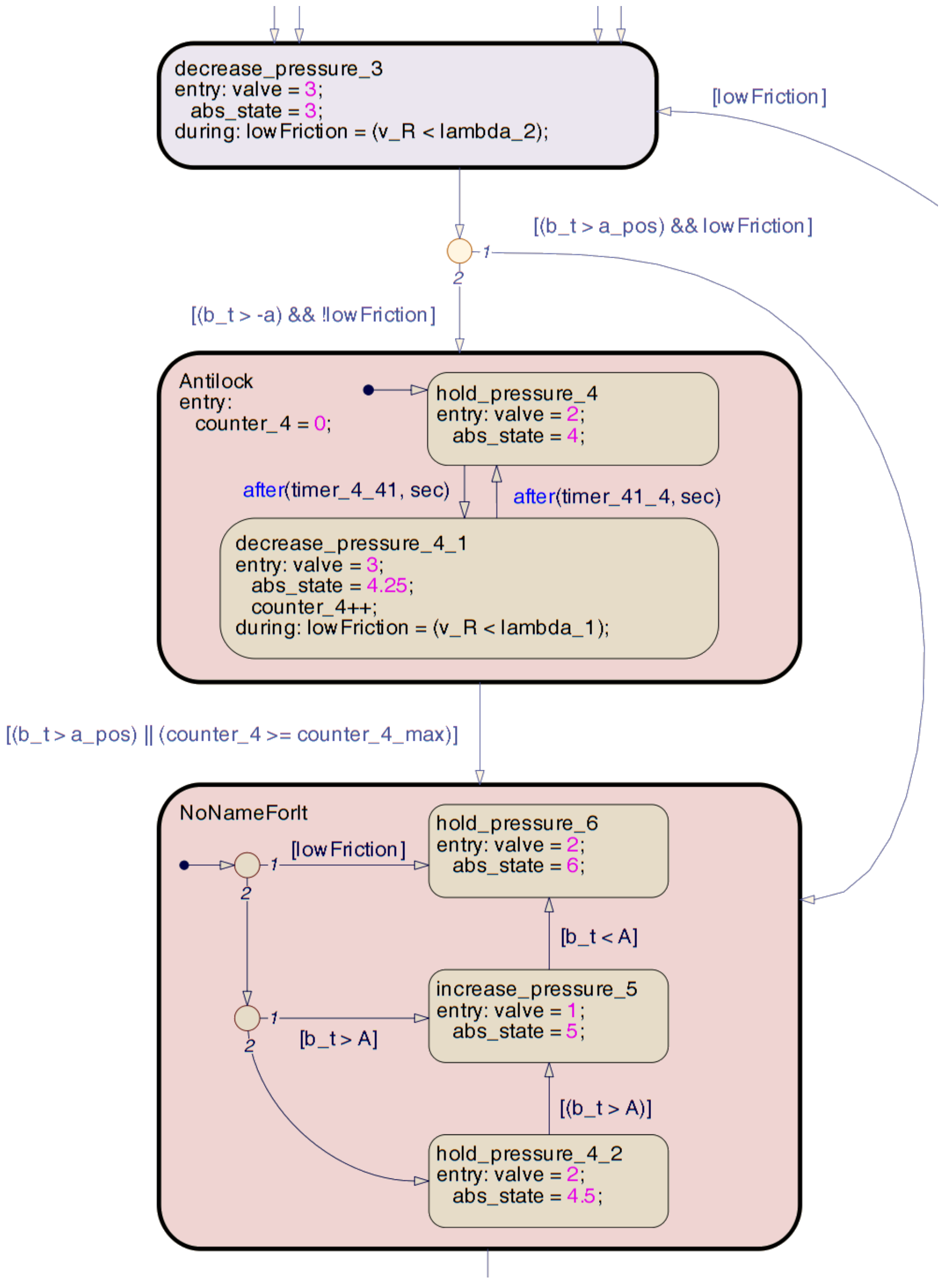


Figure 2.21.: Stateflow: ABS Controller - Anti-lock

Pulse Step Control (see figure 2.22)

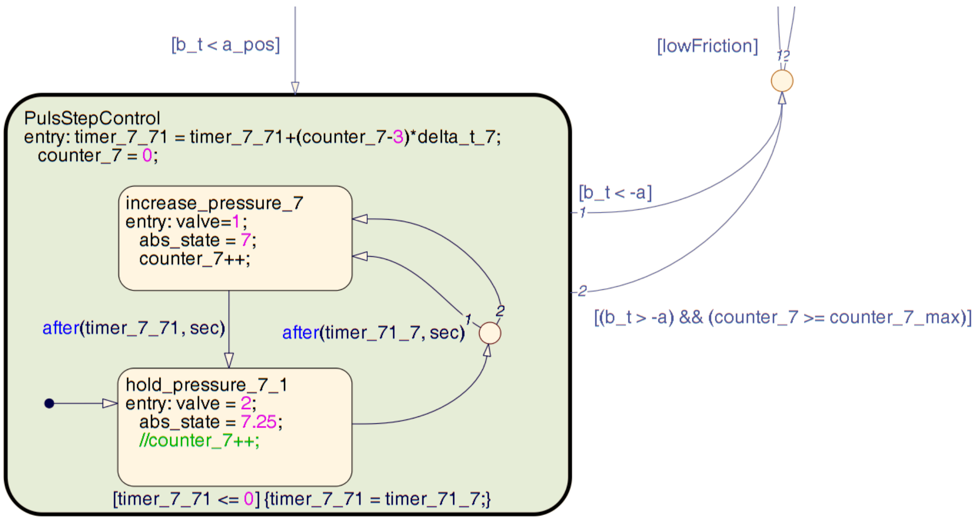


Figure 2.22.: Stateflow: ABS Controller - PulsStepControl

When the wheel-acceleration (`b_t`) falls below the positive threshold (`a_pos`) again, the `|PulsStepControl|`-state is entered. It stepwise increases the pressure again. The goal is to smoothly approach the optimal slip.

During this state, the inner cycles are counted, which are used at the next iteration to recalculate the timer `timer_7_71`, that defines the increase-pressure time-step. Ideally, there are three steps needed to increase the pressure, before the wheel starts locking again, [3], [17].

When the `b_t` falls below `-a` the ABS-cycle restarts.
Depending on the current situation, the state switches to `|restart braking|` or, in case of low friction, to `|decrease_pressure_3|`.
Also, if the inner counter reaches its maximum, `|PulsStepControl|` leaves to the next state.

Calculation

Figure 2.23 shows the `|Calculation|`-state which consists of six parallel running states, to calculate the necessary variables.

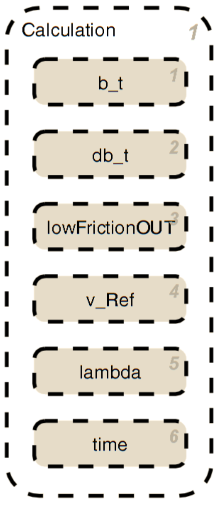


Figure 2.23.: Stateflow: ABS Controller - Calculation

Wheel de-/acceleration: b_t

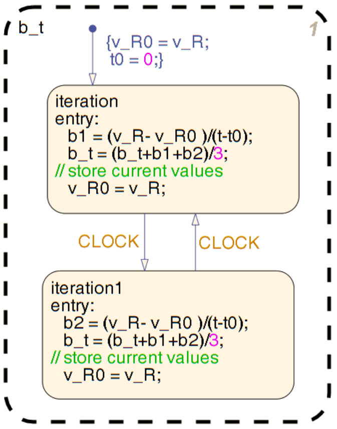
As shown in figure 2.24, the wheel de-/acceleration (`b_t`) is calculated using the mean values over the current and past values. This evens the transition at discontinuous signal-changes. Equation (2.1a) calculates the current derivation ($b(t)$) of the wheel-speed ($v_R(t)$). In equation (2.1b) the average of the current ($b(t)$) and last ($b(t-1)$) derivation and the former result ($b_t(t-1)$) smoothes the current result of $b_t(t)$.

Derivation of wheel de-/acceleration: db_t

As shown in figure 2.25 the derivation of `b_t` is a comparison between its current and previous value. Depending on the difference the result is either set to positive (+1) or negative (-1). It is needed at certain states to evaluate the current behavior (change of direction) of `b_t` and react accordingly.

Output-signal lowFrictionOUT

Output-signal `[lowFrictionOUT]` is set (=1), when `abs_state` is greater or equal than 2 (see figure 2.26). It is reset, when `abs_state` switches back to 1. Thus, immediately when ABS-cycle is started for one wheel, this is signaled to the opposite wheel's ABS to reduce possible yaw-moments at μ-split-conditions. Only during `|initial_braking|` (`abs_state==1`) this output-signal is not set.



$$b(t) = \frac{v_R(t) - v_R(t-1)}{\Delta t} \tag{2.1a}$$

$$b_t(t) = \frac{b_t(t-1) + b(t) + b(t-1)}{3} \tag{2.1b}$$

Figure 2.24.: Stateflow: Calculation - b_t

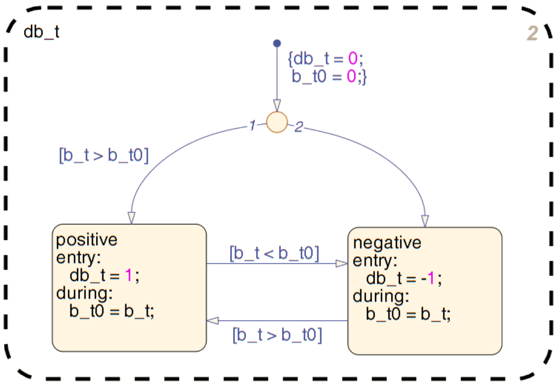


Figure 2.25.: Stateflow: Calculation - db_t

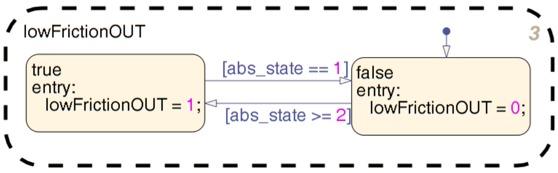


Figure 2.26.: Stateflow: Calculation - lowFrictionOUT

Reference-speed: v_Ref

The vehicle's reference speed (`v_Ref`) is an estimated value. Figure 2.27 shows the three used states.

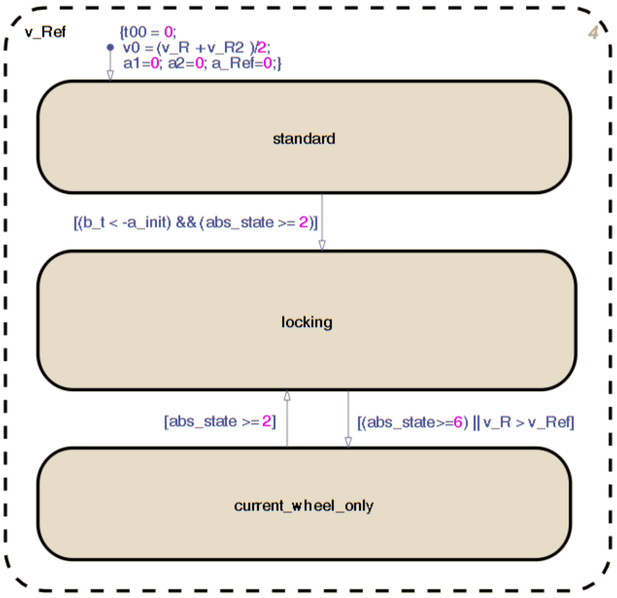


Figure 2.27.: Stateflow: Calculation - v_Ref (overview)

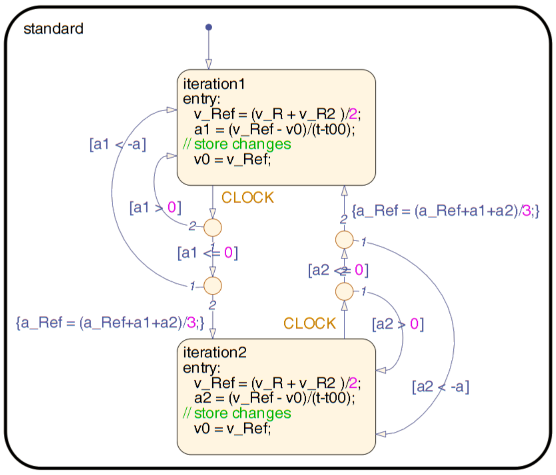
During `|initial_braking|` (ABS-cycle) its values are calculated within `|Standard|` (see figure 2.28). In equation (2.2a) the current reference-speed (v_{Ref}) is calculated as average of current wheel-speed ($v_R(t)$) and its crossover wheel-speed ($v_{R2}(t)$).

Equation (2.2b) calculates its current derivation ($a(t)$) and in equation (2.2c) the mean value of this current and past derivation plus the former result equals the current reference-deceleration (a_{Ref}).

When ABS starts controlling and wheels might be locking, `v_Ref` (v_{Ref}) is extrapolated within `locking` (see figure 2.29). In equation (2.3) current $v_{Ref}(t)$ is calculated using the former value ($v_{Ref}(t-1)$) and the reference deceleration (a_{Ref}) times the time-difference (Δt).

After the wheels becomes stable again, only the current wheel is considered to estimate `v_Ref` (see figure 2.30). In equation (2.4) the current $v_{Ref}(t)$ is defined as the current wheel-speed $v_R(t)$. The crossover wheel might be currently locking and, therefore, falsify the result.

For both calculation-states, `standard` and `current_wheel_only`, only negative vehicle-accelerations (only decelerations) are considered. Furthermore, the car's deceleration value (`a_Ref`) has to be above the wheel's deceleration threshold `-a`.



```
iteration1
entry:
  v_Ref = (v_R + v_R2 )/2;
  a1 = (v_Ref - v0)/(t-t00);
// store changes
  v0 = v_Ref;
```

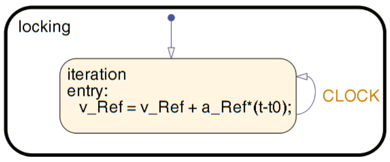
```
iteration2
entry:
  v_Ref = (v_R + v_R2 )/2;
  a2 = (v_Ref - v0)/(t-t00);
// store changes
  v0 = v_Ref;
```

$$v_{Ref}(t) = \frac{v_R(t) + v_{R2}(t)}{2} \qquad (2.2a)$$

$$a(t) = \frac{v_{Ref}(t) - v_{Ref}(t-1)}{\Delta t} \qquad (2.2b)$$

$$a_{Ref}(t) = \frac{a_{Ref}(t-1) + a(t) + a(t-1)}{3} \qquad (2.2c)$$

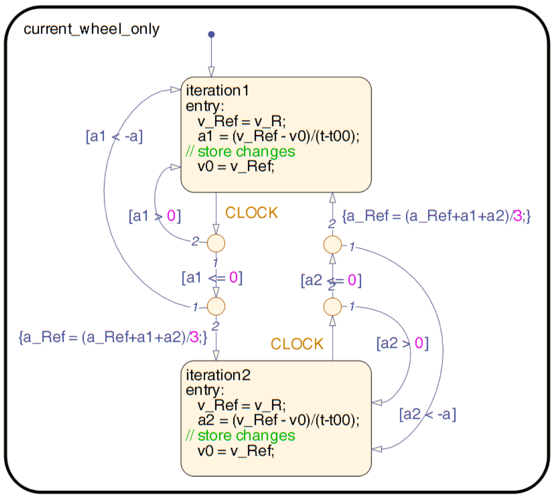
Figure 2.28.: Stateflow: Calculation - v_Ref (standard)



```
iteration
entry:
  v_Ref = v_Ref + a_Ref*(t-t0);
```

$$v_{Ref}(t) = v_{Ref}(t-1) + a_{Ref} \cdot \Delta t \qquad (2.3)$$

Figure 2.29.: Stateflow: Calculation - v_Ref (locking)



```
iteration1
entry:
  v_Ref = v_R;
  a1 = (v_Ref - v0)/(t-t00);
// store changes
  v0 = v_Ref;
```

```
iteration2
entry:
  v_Ref = v_R;
  a2 = (v_Ref - v0)/(t-t00);
// store changes
  v0 = v_Ref;
```

$$v_{Ref}(t) = v_R(t) \qquad (2.4)$$

Figure 2.30.: Stateflow: Calculation - v_Ref (current wheel only)

2.3.2. Traction Control System - TCS

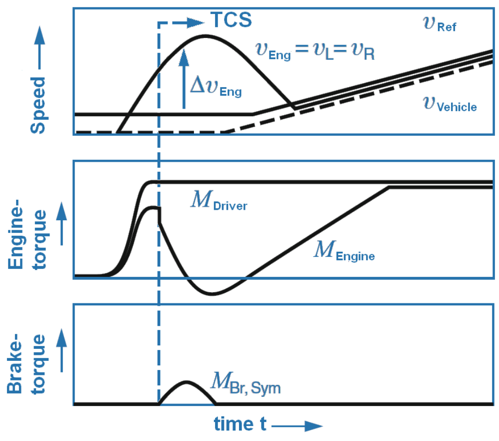
As already described in chapter 1, a TCS is used to avoid locking wheels and keeps the car stable during acceleration.

The description of typical TCS-cycles in (in [3] and [17]) serves as basis for the implementation in MOVES.

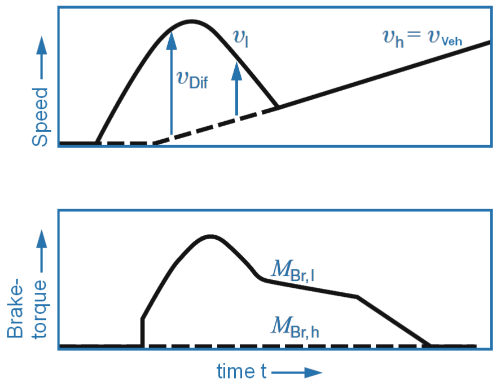
TCS-controllers interact with the brake- and the drive-system. When the driver pushes the acceleration-pedal, the engine produces (more) power and transfers the engine-torque through the whole powertrain to the driven wheels. If the resulting drive-torque is higher than the currently maximum transferable torque, the wheels start to spin. The maximum transferable torque depends (among others) on tire-road-friction-conditions (see figures 2.3, 2.4, and 2.5)
There are two basic scenarios where spinning occurs and which needs to be addressed by TCS-controllers. In case of low-friction surfaces, both driven wheels might spin, if the driver pushes the pedal to hard. There is an excess of current engine torque (especially during first gear), which needs to be reduced by the TCS-controller. In the second scenario there is a μ-split scenario, where one side has lower friction than the other. The low-friction sided, driven wheel will start spinning, whereas the other one won't accelerate. Most of the engine-torque is transferred through the final drive and wasted onto the spinning wheel. In this case the TCS-controller applies brake-torque onto low-friction sided wheel. Therefore, an equal amount of torque is transferred through the final drive onto the high-friction sided wheel. Thus, the car starts to accelerate without spinning wheels.

The implementation itself was done using Stateflow-models [1], to keep close to state-control-logic of electronic controllers.

2.3.2.1. Typical TCS cycles



(a) traction controller



(b) final drive controller

Figure 2.31.: Typical TCS cycles, [3], [17]

Figure 2.31a (beginning from top) shows the velocities of the car ($v_{Vehicle}$), the reference speed (v_{Ref}), and the engine-speed (v_{Eng}), which is the same for both driven wheels (v_L left- and v_R right-side of the vehicle). Velocity v_{Ref} is the mean value over the rear, non-driven, wheels.

The friction between tires and road-surface is low. Thus, when the driver starts to accelerate, the speed-difference (Δv_{Eng}) increases, the driven wheels start to slip or spin. If Δv_{Eng} rises above a given threshold, the TCS starts to reduce the engine-torque (M_{Engine}). As reducing M_{Engine} is not as fast as braking, additional brake-torque ($M_{Br,Sym}$) is applied symmetrically onto the driven wheels to respond faster. When Δv_{Eng} decreases again, M_{Engine} is slowly increased to approach the optimum (slip-)value and meet the drivers demands.

Figure 2.31b displays the reaction of the TCS-controller at a μ-split-scenario. The speed-difference (v_{Dif}, between low- (v_l) and high-friction (v_h) sided wheels) increases during acceleration. If v_{Dif} passes a given threshold, TCS applies brake-torque ($M_{Br,l}$) onto the low-friction sided wheel. Thus, the same amount of torque is transferred through the final drive onto the high-friction sided wheel and the car starts to accelerate. When v_{Dif} decreases again, $M_{Br,l}$ is decreased accordingly.

2.3.2.2. Implementation in Simulink

In appendix, figure A.4, the TCS is implemented as a block-choice-model within the VDC-block. There already exists a template and some previous models, but they use and control the simulated slip-values, which are usually not accessible in real-world. Figure 2.32 shows the newly implemented TCS. MOVES is a mixture of implementations using English- as well as in German-language and its abbreviations. Thus, TCS-controllers were implemented as ASR, which is the German abbreviation for *"Antriebsschlupfregelung"*. Fortunately, for ABS the English and German abbreviations are equal.

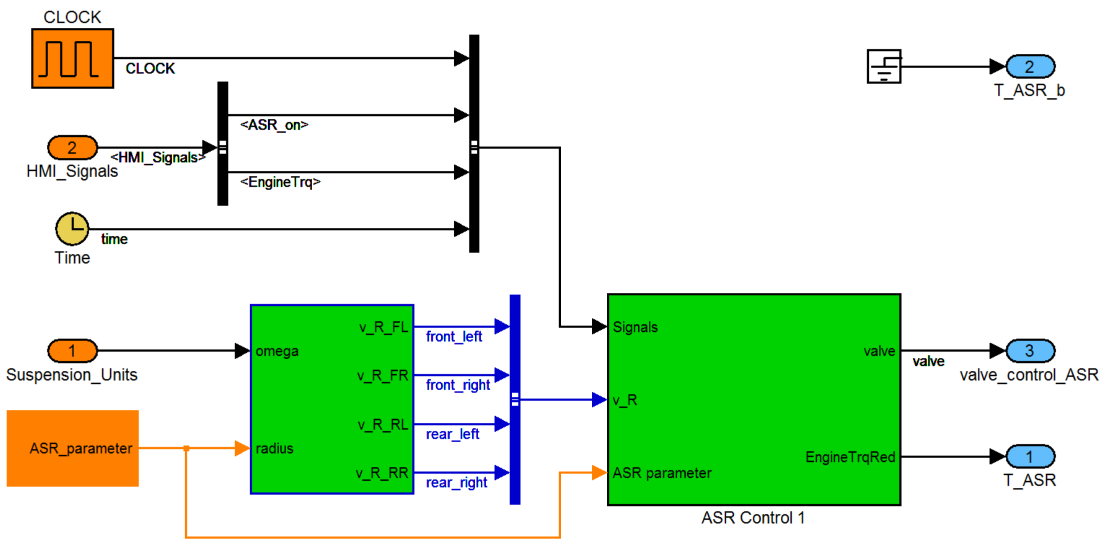


Figure 2.32.: TCS: overview

Inputs

From the `<HMI_Signals>` (bus) the input `<ASR_on>` is used to turn the TCS-controller on or off. The current engine-torque (`<EngineTrq>`) is also provided through the HMI. The `<Suspension_Unit_Signals>` (bus) provide the wheel-speeds (`<omega>`) for all wheels. Additionally, the `<CLOCK>`-signal triggers the discrete Stateflow-controller. The `|ASR-parameter|`-block provides the needed/initial TCS-parameters.

Blocks

In the only additional (nameless) block the wheel speeds (`[v_R_XY]`) are calculated and filtered for each wheel. This block has been reused from ABS (see figure 2.15).

The TCS-controller (`|ASR Control 1|`) is kept in the remaining block (one controller for both driven (front) wheels).

Outputs

The existing output ([T_ASR_b]) has been grounded, as it couldn't be reused for new purposes. Variable [T_ASR] holds the reduced engine-torque ([EngineTrqRed]), which needs to be subtracted from the current engine-torque.

The bus-signal, [valve_control_ASR], combines all signals for both driven wheels to the remaining output. It controls the valve-position of the hydro-unit to increase, hold, or reduce brake-pressure for both driven wheels.

2.3.2.3. Implementation in Stateflow

Figure 2.33 looks inside the TCS-block. The provided input-structures (bus-signals) are divided and forwarded as single input-signals to the Stateflow-model.

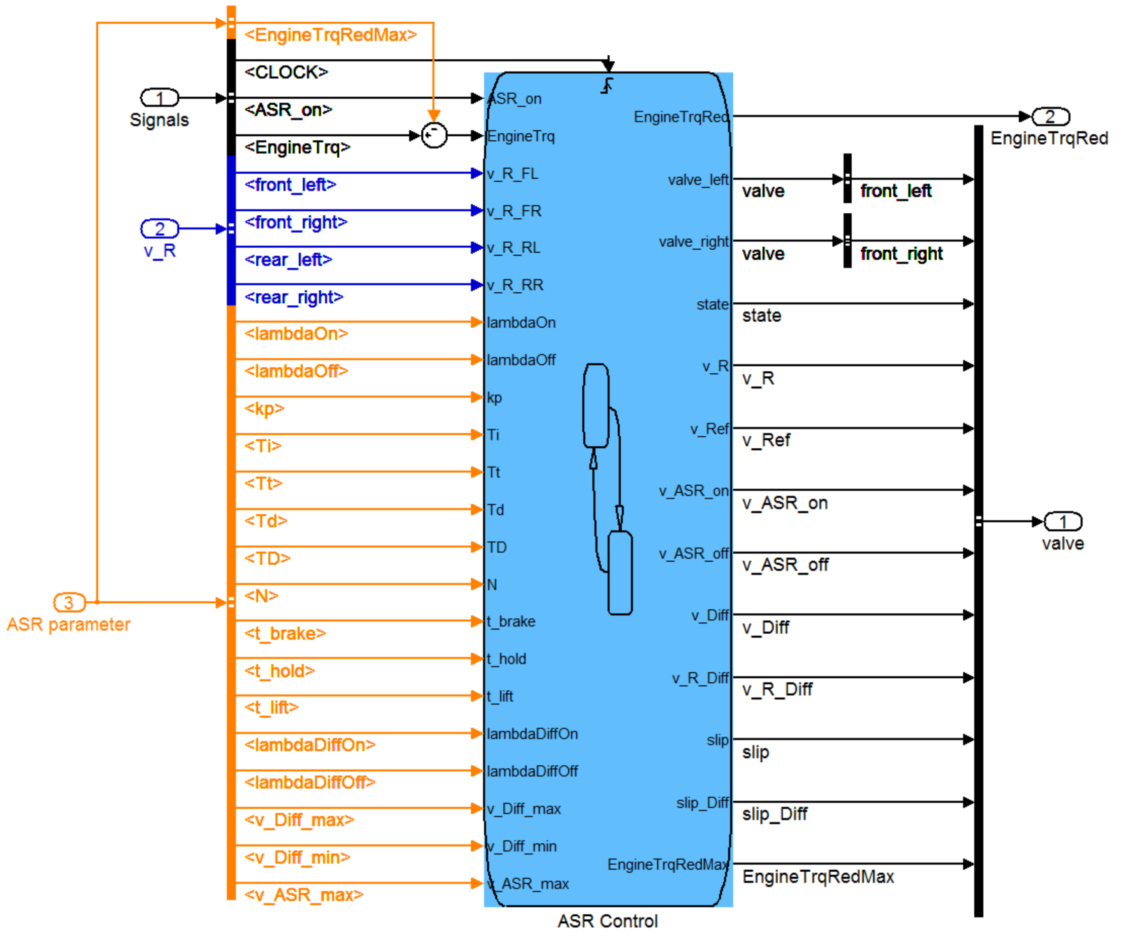


Figure 2.33.: ASR: Control 1

Table 2.5.: ASR controller: input-signals

<Input>	Description
`<CLOCK>`	This event triggers the state-machine. It is used to switch between states.
`<ASR_on>`	This Boolean value signals start and end of acceleration to turn TCS-controller on or off.
`<EngineTrq>`	The current engine-torque is needed as a maximum limit for the controller to reduce.
`<v_R_FL>`	The longitudinal wheel-speed of the front left tire.
`<v_R_FR>`	The longitudinal wheel-speed of the front right tire.
`<v_R_RL>`	The longitudinal wheel-speed of the rear left tire.
`<v_R_RR>`	The longitudinal wheel-speed of the rear right tire.
	End of table

Table 2.6.: ASR controller: input-parameters

Parameter	Description
`EngineTrqRedmax`	The minimum current engine-torque.
`lambdaOn`	This slip-threshold turns the traction- (slip-) control on.
`lambdaOff`	Slip-threshold turns the traction-control off.
`kp`	The basis for the proportional gain-factor of the PID-controller (traction control).
`Ti`	Integral-time of the PID-controller (traction control).
`Tt`	Anti-windup-time of the PID-controller (traction control).
`Td`	Discretization-time of the PID-controller (traction control).
`TD`	Derivative time of the PID-controller (traction control).
`N`	Number of time-steps to look back at the derivative part of the PID-controller.
`t_brake`	Time to increase the pressure at the final drive-controller.
`t_hold`	Time to hold the pressure at the final drive-controller.
`t_lift`	Time to decrease the pressure at the final drive-controller.
`lambdaDiffOn`	The first threshold to start the final drive control cycle (switches to state `|Brake_Left|` or `|Brake_Right|`).
`v_Diff_max`	The second threshold to start the final drive control-cycle.
`v_ASR_max`	Speed limitation of the final drive control-cycle.
`lambdaDiffOff`	The first threshold to stop the final drive control-cycle (switches to state `|Decrease_Pressure|`).
`v_Diff_min`	The second threshold to stop the final drive control-cycle.
	End of table

Table 2.7.: ASR controller: (output-) variables

Variable [Output]	Description
`[EngineTrqRed]`	The current torque that needs to be reduced from the engine.
`[EngineTrqRedMax]`	Current maximum reducible engine-torque (limited by current engine-torque `<EngineTrq>` and zero).
`valve_left` `[front_left.valve]`	Valve-position of the hydro-unit. Switches between increase or follow input-pressure, hold, and decrease brake-pressure.
`valve_right` `[front_right.valve]`	Valve-position of the hydro-unit. Switches between increase or follow input-pressure, hold, and decrease brake-pressure.
`[state]`	The state of the TCS-controller to monitor its behavior.
`[v_Ref]`	The reference-speed calculated as average velocity of the non-driven wheels ($v_{Ref} = (v_{R,RL} + v_{R,RR})/2$).
`[v_ASR_on]`	The calculated speed-threshold to start the traction control ($v_{ASR,on} = v_{Ref}/(1 - \lambda_{On})$).
`[v_ASR_off]`	The calculated speed-threshold to stop the traction control ($v_{ASR,off} = v_{Ref}/(1 - \lambda_{Off})$).
`[v_R]`	The driven wheels average velocity ($v_R = (v_{R,FL} + v_{R,FR})/2$).
`[v_Diff]`	Calculated absolute speed-difference between driven wheels and reference-speed ($v_{Diff} = v_R - v_{ASR,on}$).
`[slip]`	Calculated slip-value between driven wheels and reference-speed ($\lambda = v_R/v_{Ref} - 1$).
`[v_R_Diff]`	Calculated absolute difference between driven left and right wheel-speed ($v_{R,Diff} = v_{R,FL} - v_{R,FR}$).
`[slip_Diff]`	Calculated slip-value between speed-difference and reference-speed ($\lambda_{Diff} = v_{R,Diff}/v_{Ref}$).
	End of table

The parameters in table 2.6 are created during initialization of the *Master Parameter Setup*. In the parameter-file `ASR_Controller_Parameters.m` (see appendix, listing A.2) the referring values are stored. The position within the resulting input-structure is: `Inputs.Parameters.Vehicle_Dynamics_Controller.ASR`.

Tables 2.5 and 2.7 describe the purpose of the input-signals, variables, and output-signals. The main output-signals are `[valve_left]`, `[valve_right]`, and `[EngineTrqRed]`. Former two directly control the position of the hydro-unit of the referring wheels.
Output `EngineTrqRed` (always a negative value) is directly added to the current engine-torque. The other outputs are just used to monitor the control-flow.

TCS Controller

In figure 2.34 the basic layout of the TCS is presented.

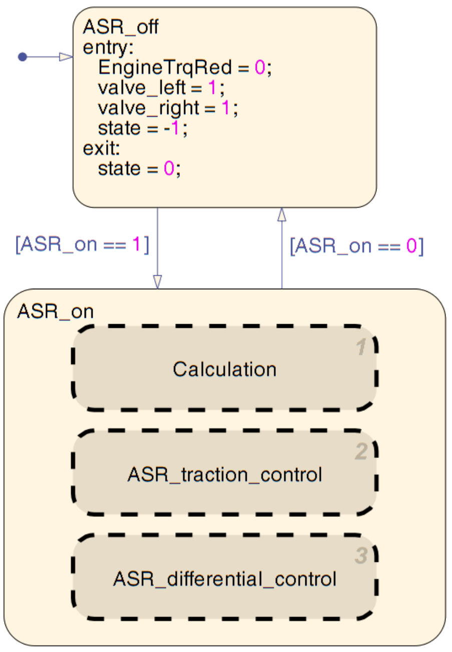


Figure 2.34.: Stateflow: TCS Controller

There are two states: `|ASR_off|` and `|ASR_on|`. The input-signal `<ASR_on>` switches between those two states, turns the controller on or off.

`|ASR_on|` consists of three parallel running states:
`|Calculation|` sets the necessary variables for each iteration.
`|ASR_traction_control|` controls the slip by reducing engine torque.
`|ASR_differential_control|` controls μ-split-scenarios (final drive control).

Traction Control

Figure 2.35 shows the content of the first (of two) TCS-controllers.

The traction-control regulates the slip of the driven (front) wheels by reducing the engine-torque (`[EngineTrqRed]` is set accordingly), when wheels start spinning. Initially the controller is set to `|Idle_State|` (`[EngineTrqRed]`=0Nm, full available current engine-torque is transmitted). When the calculated `slip` exceeds `lambdaOn` and the speed-difference (`v_Diff`) between driven wheels and reference-speed is higher than 0.1m/s, it switches to state `|ReduceEngineTorque|`.

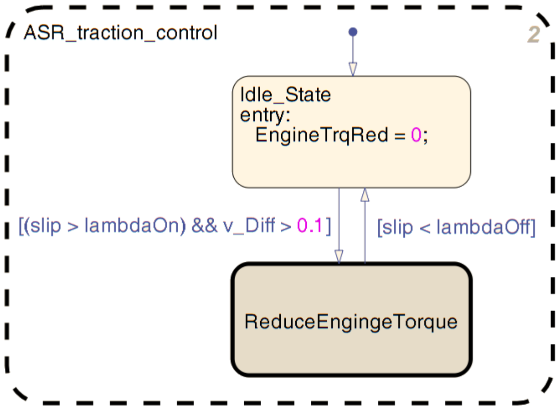


Figure 2.35.: Stateflow: TCS Controller - traction control

Reduce Engine Torque (see figure 2.36)

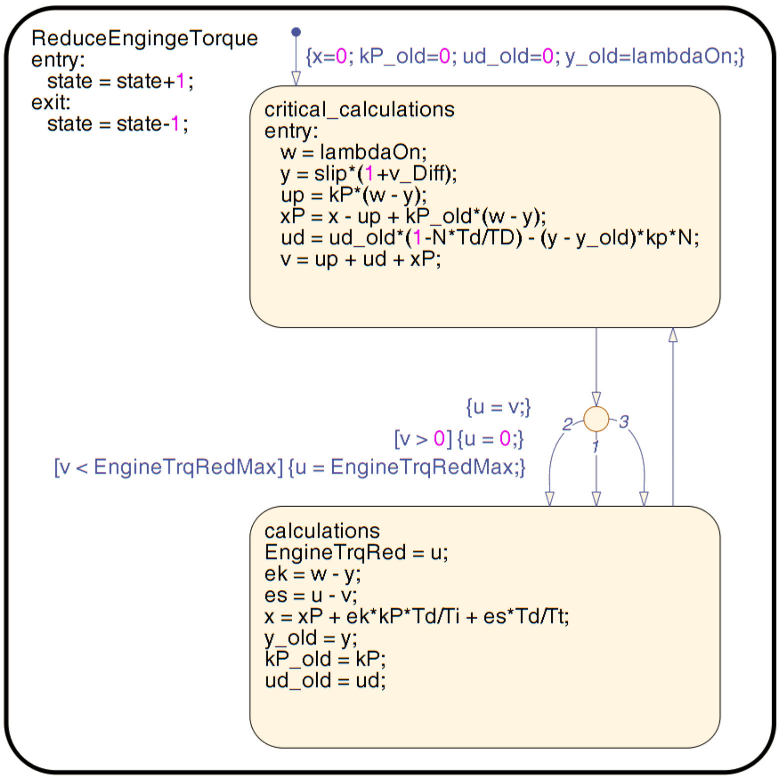


Figure 2.36.: Stateflow: TCS Controller - Reduce Engine Torque

This state is entered, when slip-conditions worsen. Thus, the traction control has to reduce the engine torque to avoid spinning wheels.

A common PID-controller has been implemented using two states, [7], which switches at each `<CLOCK>`-trigger-input.

State `|critical_calculations|` pre-calculates all important values in advance.

$$
\begin{aligned}
w_k &= \lambda_{On} && \text{(desired value)} \quad (2.5a)\\
y_k &= \lambda \cdot (1 + v_{Diff}) && \text{(actual value)} \quad (2.5b)\\
u_p &= k_{p,k} \cdot (w_k - y_k) && (2.5c)\\
\tilde{x}_k &= x_k - u_p + k_{p,k-1} \cdot (w_k - y_k) && (2.5d)\\
u_{d,k} &= u_{d,k-1} \cdot (1 - N \cdot T_d/T_D) - (y_k - y_{k-1}) \cdot k_p \cdot N && (2.5e)\\
v_k &= u_p + \tilde{x}_k + u_{d,k} && (2.5f)
\end{aligned}
$$

When switching to `|calculations|`, limitations of `[EngineTrqRed]` are checked (between 0Nm and `EngineTrqRedMax`). During `|calculations|` the output `[EngineTrqRed]` is set and further equations are calculated in advance of next iteration.

$$
\begin{aligned}
EngineTrqRed &= u && \text{(output value)} \quad (2.6a)\\
e_k &= w_k - y_k && (2.6b)\\
e_s &= u_k - v_k && (2.6c)\\
x_{k+1} &= \tilde{x}_k + e_k \cdot k_{p,k} \cdot T_d/T_i + e_s \cdot T_d/T_t && (2.6d)
\end{aligned}
$$

Equations (2.5) calculate the time-critical values:

(a) reads the desired input-variable (w_k). In this case the slip-value (λ_{On}).
(b) reads the current output-value (y_k), which is calculated using current slip (λ) and absolute speed-difference (v_{Diff}).
(c) calculates the proportional regulating variable (u_p).
(d) calculates an intermediate state ($\tilde{x}_k$) from previously calculated state (x_k) and current deviation between desired and actual value.
(e) calculates the differential regulating variable ($u_{d,k}$).
(f) the unlimited regulating variable (v_k) results as sum of u_p, $\tilde{x}_k$ and $u_{d,k}$.

When transition to `|calculations|`, v_k is trimmed to its limitations, resulting in (limited) regulation variable (u).
Equations (2.6) calculate uncritical values in advance:

(a) sets the output `[EngineTrqRed]` to the limited regulation variable (u).
(b) reads the current error (e_k) between desired (w_k) and actual value (y_k).
(c) reads the current error (e_s) of limited (u_k) and unlimited (v_k) regulating variable.
(d) in advance calculates the next state (x_{k+1}) from given errors using PID-parameters.

Final-drive Control

This is the second (of two) TCS-controllers (see figure 2.38). It controls the speed-difference between left and right driven wheel by applying brake torque onto the faster spinning wheel. Therefore, an equal amount of drive-torque is transferred through the final drive onto the other wheel.

Initially, the controller is set to |`Idle_State`|, where both output signals (`[valve_left]` and `[valve_right]`) are set to their defaults (=1). Depending on positive or negative values of `slip_Diff` and `v_R_Diff`, either the left or right branch of the controller is used.

If `slip_Diff` exceeds threshold `lambdaDiffOn` and the speed-difference (`v_R_Diff`) is higher than `v_Diff_max`, while the reference-speed (`v_Ref`) is smaller than `v_ASR_max`, the left branch (controlling the front left wheel) is used and the state switches to |`Brake_Left`| (see figure 2.37).

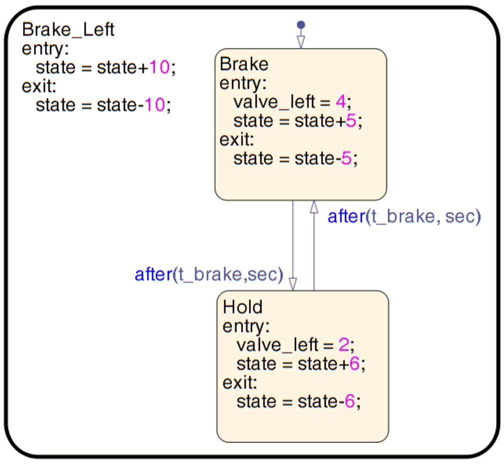


Figure 2.37.: Stateflow: TCS Controller - brake left

Within |`Brake_Left`| the states switch between |`Brake`| (increasing the pressure for `t_brake` seconds, output `[valve_left]` is set to 4) and |`Hold`| (holding the pressure for `t_hold` seconds, output `[valve_left]` is set to 2). This increases the pressure step-wise.
If the front left wheel-speed decreases again, the controller switches to |`Hold_Left`|. The pressure is held until `slip_Diff` falls below threshold `lamdaDiffOff` and `v_R_Diff` falls below `v_Diff_min` again. After waiting `t_hold` seconds, the state either returns to |`Brake_Left`| or switches to |`Decrease_Pressure`| (depending on the current derivation of the wheel-speed).
At |`Decrease_Pressure`| both branches (left and right side) are joined again and both valves (`[valve_left]` and `[valve_right]`) are set to decrease the brake-pressure (=3). The cycle restarts itself.

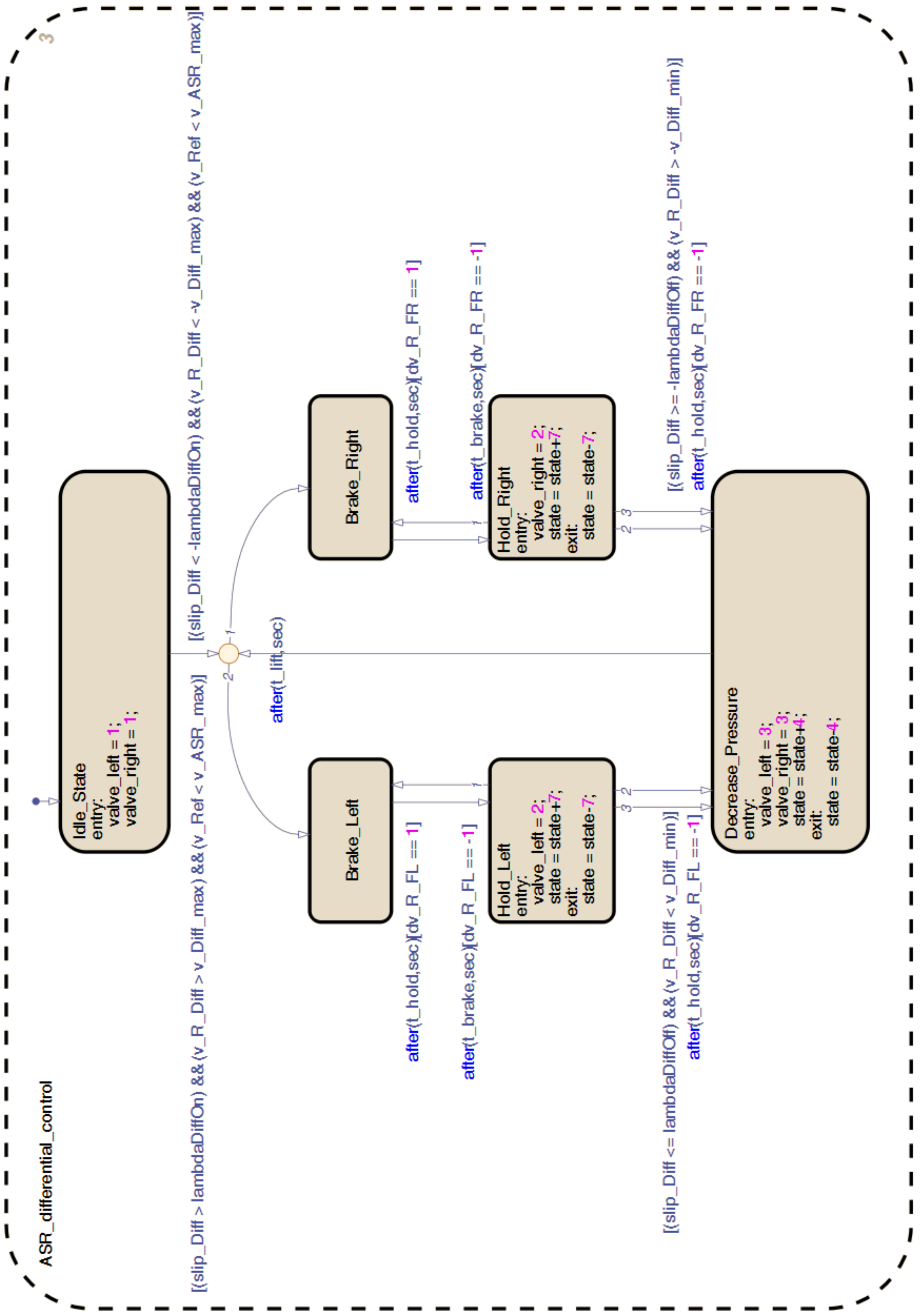

ASR_differential_control
3
Idle_State
entry:
valve_left = 1;
valve_right = 1;
[(slip_Diff < -lambdaDiffOn) && (v_R_Diff < -v_Diff_max) && (v_Ref < v_ASR_max)]
[(slip_Diff > lambdaDiffOn) && (v_R_Diff > v_Diff_max) && (v_Ref < v_ASR_max)]
1
2
after(t_lift,sec)
Brake_Right
Brake_Left
after(t_hold,sec)[dv_R_FR == 1]
after(t_brake,sec)[dv_R_FR == -1]
Hold_Right
entry:
valve_right = 2;
state = state+7;
exit:
state = state-7;
after(t_hold,sec)[dv_R_FL == 1]
after(t_brake,sec)[dv_R_FL == -1]
Hold_Left
entry:
valve_left = 2;
state = state+7;
exit:
state = state-7;
[(slip_Diff >= -lambdaDiffOff) && (v_R_Diff > -v_Diff_min)]
after(t_hold,sec)[dv_R_FR == -1]
[(slip_Diff <= lambdaDiffOff) && (v_R_Diff < v_Diff_min)]
after(t_hold,sec)[dv_R_FL == -1]
Decrease_Pressure
entry:
valve_left = 3;
valve_right = 3;
state = state+4;
exit:
state = state-4;

Figure 2.38.: Stateflow: TCS Controller - Final-drive Control

Calculation

Figure 2.39 shows the |Calculation|-state which consists of six parallel running states, to calculate the necessary variables.

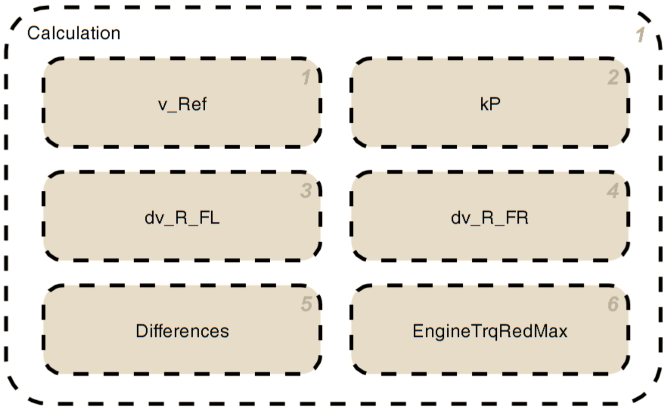


Figure 2.39.: Stateflow: TCS Controller - Calculation

Reference-speed: v_Ref, Differences

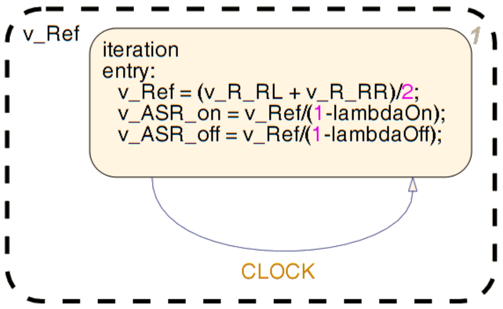
Equations (2.7) calculate the reference speed (`v_Ref` or v_{Ref}) as average value over the two rear, non-driven wheels (see figure 2.40). It serves as basis to calculate the thresholds `v_ASR_on` ($v_{ASR,on}$) and `v_ASR_off` ($v_{ASR,off}$).

(a) calculates current v_{Ref} as average over $v_{R,RL}$ and $v_{R,RR}$.
(b),(c) calculate the thresholds $v_{ASR,on}$ and $v_{ASR,off}$ according to current v_{Ref} and slip-parameters λ_{On} and λ_{Off}.

Equations (2.8) calculate the average front wheel-speed (`v_R` or v_R), which is needed to calculate the speed-difference (`v_Diff` or v_{Diff}) and `slip` (λ). Absolute value `v_Diff` is used to detect spinning wheels. The commented code in figure 2.40b shows several other possibilities to calculate according slip-values.

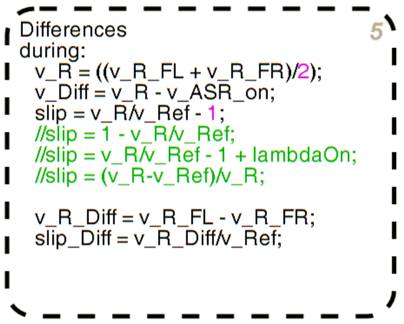
(a) calculates current v_R as average over $v_{R,FL}$ and $v_{R,FR}$.
(b) calculates absolute difference between current v_R and TCS-parameter $v_{ASR,on}$.
(c) calculates the current slip λ (compare section 2.1.2.1, table 2.1).

The difference between driven left and right wheel (`v_R_Diff` or $v_{R,Diff}$) is needed to calculate `slip_Diff` (λ_{Diff}) as slip-value between left and right side. Equation (2.9a) calculates $v_{R,Diff}$, equation (2.9b) calculates λ_{Diff} as ratio of $v_{R,Diff}$ to v_{Ref}.



```
entry:
  v_Ref = (v_R_RL + v_R_RR)/2;
  v_ASR_on = v_Ref/(1-lambdaOn);
  v_ASR_off = v_Ref/(1-lambdaOff);
```

(a) v_Ref



```
during:
  v_R = ((v_R_FL + v_R_FR)/2);
  v_Diff = v_R - v_ASR_on;
  slip = v_R/v_Ref - 1;
  //slip = 1 - v_R/v_Ref;
  //slip = v_R/v_Ref - 1 + lambdaOn;
  //slip = (v_R-v_Ref)/v_R;

  v_R_Diff = v_R_FL - v_R_FR;
  slip_Diff = v_R_Diff/v_Ref;
```

(b) Differences

$$v_{Ref} = \frac{v_{R,RL} + v_{R,RR}}{2} \tag{2.7a}$$

$$v_{ASR,on} = \frac{v_{Ref}}{1 - \lambda_{On}} \tag{2.7b}$$

$$v_{ASR,off} = \frac{v_{Ref}}{1 - \lambda_{Off}} \tag{2.7c}$$

$$v_R = \frac{v_{R,FL} + v_{R,FR}}{2} \tag{2.8a}$$

$$v_{Diff} = v_R - v_{ASR,on} \tag{2.8b}$$

$$\lambda = \frac{v_R}{v_{Ref}} - 1 \tag{2.8c}$$

$$v_{R,Diff} = v_{R,FL} - v_{R,FR} \tag{2.9a}$$

$$\lambda_{Diff} = \frac{v_{R,Diff}}{v_{Ref}} \tag{2.9b}$$

Figure 2.40.: Stateflow: Calculation - v_Ref, differences

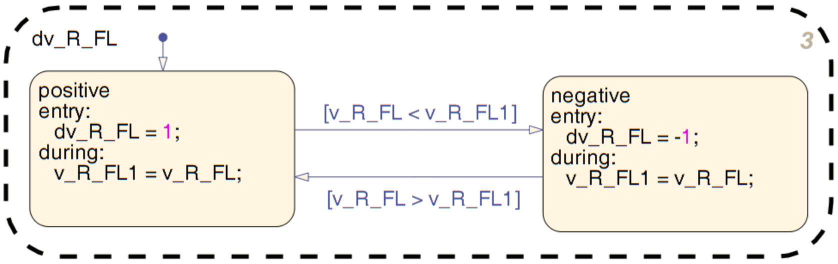
Derivations of driven wheels: dv_R_FL and dv_R_FR

Figure 2.41 shows how the speed-derivations for the front left and right wheel (`dv_R_FL` and `dv_R_FR`) are set. If the previous value is higher than the current one, the state switches to `|negative|`, where the variable is set to -1. In case the previous value is smaller than the current one, the state returns to `|positive|`, setting the variable to 1.

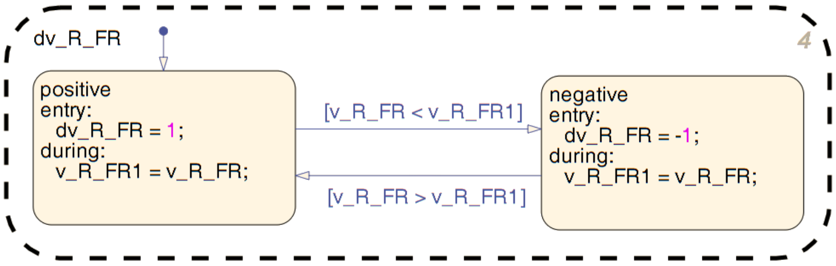
Those variables help decide whether the pressure needs to be further increased or already decreased at the final drive control (see figure 2.38).

PID-traction-controller: kP

The proportional factor (`kP` or k_p) of the PID-traction-controller is calculated depending on the absolute speed-difference (`v_Diff` or v_{Diff}) and parameter `kp` ($k_{p,in}$) (see figures 2.35 and 2.42). Equation (2.10) defines the current $k_{p,k}$ as $k_{p,in}$ plus muliplication of $k_{p,in}$ to the current square of v_{Diff}.
Using a constant parameter didn't work for the range of velocity (starting from 0km/h) and slip-values are relative to the current speed. Therefore, in an own approach the absolute value (`v_Diff`) is added to calculate `kP` depending on the current speed-difference.

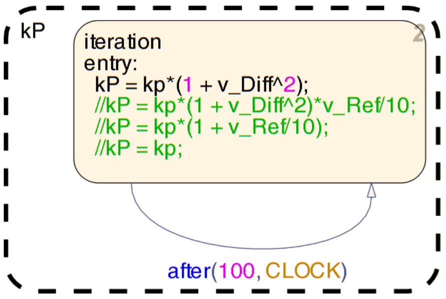


(a) dv_R_FL



(b) dv_R_FR

Figure 2.41.: Stateflow: Calculation - speed-derivations



$$k_{p,k} = k_{p,in} \cdot \left(1 + v_{Diff}^2\right) \tag{2.10}$$

Figure 2.42.: Stateflow: Calculation - kP

Maximum Reduced Engine Torque

The maximum reducible engine-torque (`EngineTrqRedMax`) depends on the current available engine-torque (`<EngineTrq>`). As long as this input-signal is greater than zero, `EngineTrqRedMax` is defined as negative input-value (see figure 2.43). If the input is lower or equal to zero, `EngineTrqRedMax` is set to zero, meaning that no further reduction of `<EngineTrq>` is possible.
As seen in figure 2.33, `<EngineTrq>` is already reduced by parameter `EngineTrqRedMax`. This is necessary to avoid a shutdown of the engine because of TCS-interaction.

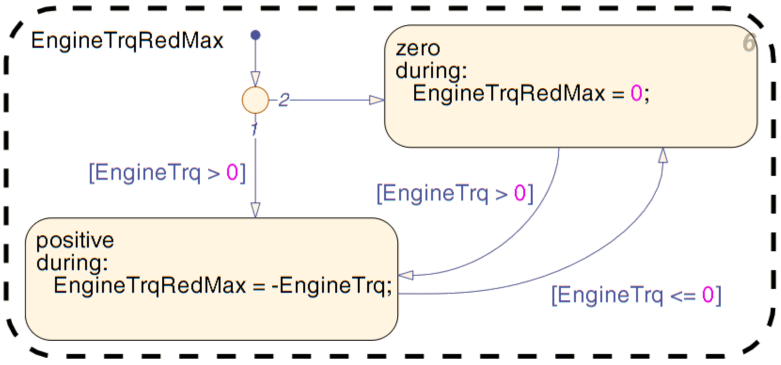


Figure 2.43.: Stateflow: Calculation - maximum reduced engine torque

2.3.3. Brake-System

Brakes are controlled by both newly implemented systems, ABS and TCS. They regulate the current brake-torque at the wheels.

The formerly existing brake-system consists of following components with referring input- and output-signals:

Deceleration pedal: The brake-pedal position is the only input-signal provided by the driver.

Mechanical brake-booster: Converts the pedal-position to a force, further to a hydraulic pressure (`p_hydraulic`) and finally increases and multiplies `p_hydraulic` through the brake-booster.

Hardware: `p_hydraulic` is transformed with provided hydraulic parameters to the brake-torque (`[M_brake]`), which is applied onto the wheels.

The brake-torque M_b (`M_brake`) is calculated as

$$-M_b = p_{hydr} \cdot r_b \cdot C^* \cdot A_b \cdot \text{sign}(\omega) .$$

p_{hydr}	is the hydraulic pressure in [Pa] provided by the hydro-unit.
r_b	is the mean brake-radius in [m].
A_b	is the friction surface of brake pads on brake disk in [m^2].
C^*	is the brake-coefficient in [1].
ω	is the current wheel-speed (only the sign-value is used).

The transition from input `<Deceleration_Pedal_Input>` to output `[M_brake]` is instantly done (without delays). Discontinuous changes of the input-signal are immediately forwarded to the output, too.

This is no natural behavior. Usually, the pressure builds up following a continuous natural curve. The pressure needs to be transported through lines (pipes) to the brakes.

In the brakes the pads have to travel a distance between to the disks, before the pressure produces a force that causes its final brake-torque. This hysteresis is also not covered by the previous brake-system.

Therefore, the former system has been extended by a hydro-unit. Its valves allow to individually control the current-pressure for each wheel, which is necessary for ABS- and TCS-controllers. Former abnormalities have been addressed by adding a simply transfer-function to phenomenologically rebuild a normal behavior.

The implementation itself was done using Simulink-models.

2.3.3.1. Implementation in Simulink

In appendix, figure A.3, the brake-system is implemented as a block-choice-model within the HMI-block. There already exists a template and some previous implemented models. Figure 2.44 shows the newly implemented brake-system, which relies on the former existing |Hydraulic_Brake_System|.

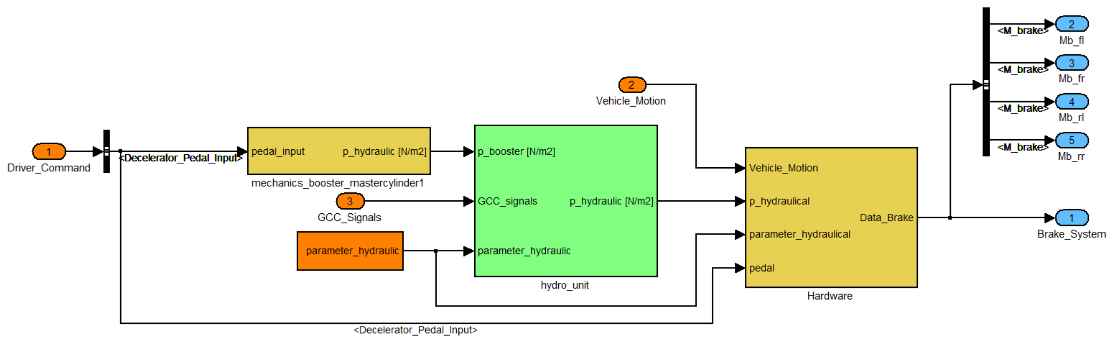


Figure 2.44.: Brake-system: overview

Inputs

There are three input-signals. The bus-signal <Driver_Command> provides the system with <Decelerator_Pedal_Input>, the position of the brake-pedal.
Input <Vehicle_Motion> (bus-signal) is needed at the |Hardware|-block.
The <GCC_Signals>-bus contains the ABS- and TCS-<valve_control>-signals, which are controlling the |hydro_unit|.
The |parameter_hydraulic|-block provides the necessary hydraulic parameters, like the brake-dimensions.

Blocks

The already existing blocks |`mechanics_booster_mastercylinder1`| and |`Hardware`| (see appendix, figures A.6 and A.7) have been described before. First one transforms `<Decelerator_Pedal_Input>` into the brake-pressure (`p_hydraulic`). Later one converts `p_hydraulic` into the brake-torque (`M_brake`) .

The |`hydro_unit`| has been newly implemented. It is controlled by ABS and TCS controllers and it smooths `p_hydraulic` to gain natural behavior.

Outputs

The bus-signal, `[Brake_System]` combines all output-data of the brake-system. The single brake-torques (`[M_brake]`) of each wheel are additionally set as outputs.

Hydro-Unit

In figure 2.45 the layout of the hydro-unit is shown. The input `<p_booster>` is the current pressure after the brake-booster. It is equal for all four wheels (|`valves`|).

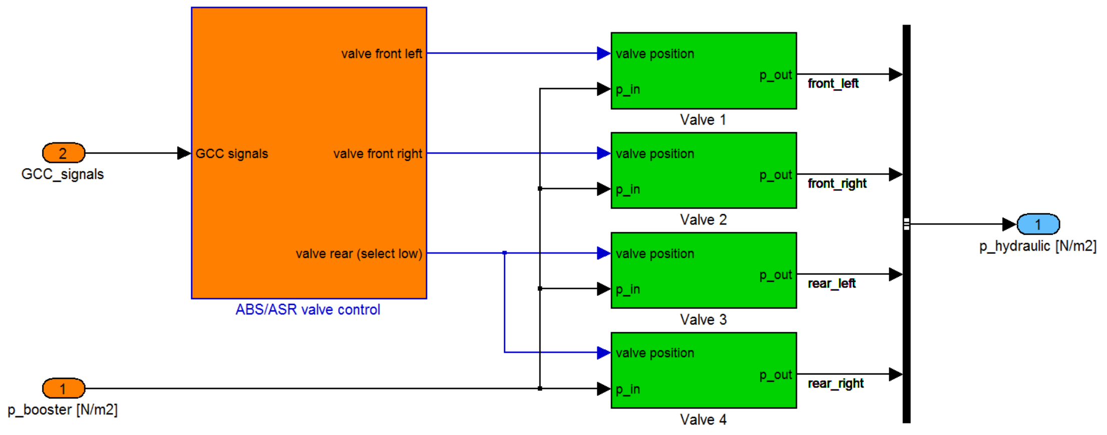


Figure 2.45.: Brake-system: hydro-unit

The bus-signal `<GCC_signals>` holds the `<valve_control>`-signals of the ABS- and TCS-controllers. Within the block |`ABS/ASR valve control`| the individual signals of ABS and TCS are merged for both front wheels. The ABS-signals of the rear-wheels are also combined, to select the (current) low-friction controller. This is done to avoid instability at μ-split-conditions, decreasing an upcoming yaw-moment, [3] and [17].

Each wheel's pressure is controlled by its referring |`valve`|.

Valve-Control

Figure 2.46 displays the content of a `|valve|` within the hydro-unit.

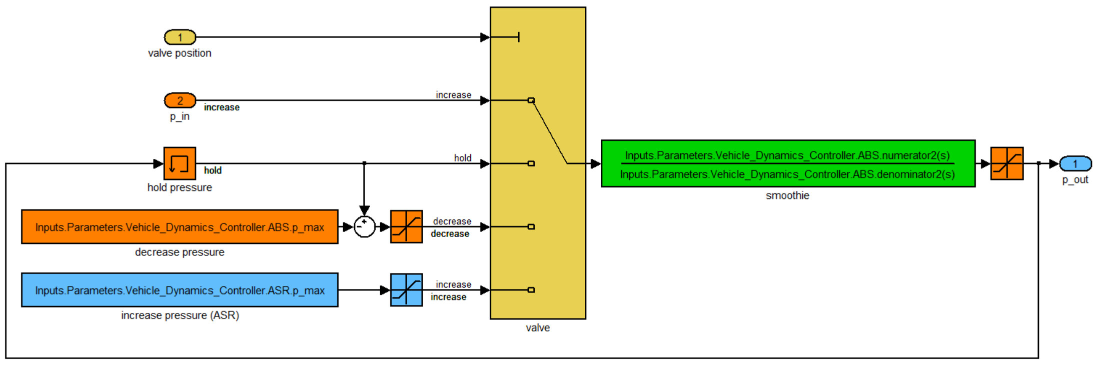

1
valve position
2
p_in
increase
increase
hold
hold
hold pressure
Inputs.Parameters.Vehicle_Dynamics_Controller.ABS.p_max
decrease pressure
decrease
decrease
Inputs.Parameters.Vehicle_Dynamics_Controller.ASR.p_max
increase pressure (ASR)
increase
increase
valve
Inputs.Parameters.Vehicle_Dynamics_Controller.ABS.numerator2(s)
Inputs.Parameters.Vehicle_Dynamics_Controller.ABS.denominator2(s)
smoothie
1
p_out

Figure 2.46.: Brake-system: valve-control

Depending on the given input-signal (`<valve position>`) the switch `|valve|` selects one of the following inputs.

1: p_in increases (or follows) the pressure according to the driver's input. This is also the default-position.

2: hold pressure holds the current output-pressure of the valve as input.

3: decrease pressure decreases the current-pressure by parameter `ABS.p_max` with according limitations. This is done to gain similar decrease-rates compared to measured test-data. In real-world this pressure is decreased by a hydraulic pump.

4: increase pressure TCS increases the pressure up to `ASR.p_max` with according limitations. In real-world this pressure is generated by a hydraulic pump.

The output of the valve-switch (`[p_out]`) is smoothed by a transfer-function and limited accordingly. This phenomenologically models the physical background of pipes, response times, hysteresis, and so on. It is set up to match the pressure-in- and decrease-rates of measured test-data, [13], but also complies with referring literature [11].

2.3.4. Drive-System

The drive-system (or powertrain) moves the wheels by applying a positive torque onto them. Basically, the driver's input (`<Accelerator_Pedal_Input>`) is converted into the output `[M_drive]` for each wheel.

The TCS-controller reduces the engine-torque if spinning wheels are detected.

The implementation itself was done adapting existing Simulink-models to the new circumstances.

2.3.4.1. Implementation in Simulink

In appendix, figure A.3, the drive-system is implemented as a block-choice-model within the HMI-block. There exist already some previous models and templates which have been reused and slightly adapted.

In appendix, figures A.8 and A.9 show the implemented ideal and real drive-systems used in MOVES. The additionally needed signals `[EngineTrq]` and `[ASR_on]` are highlighted.

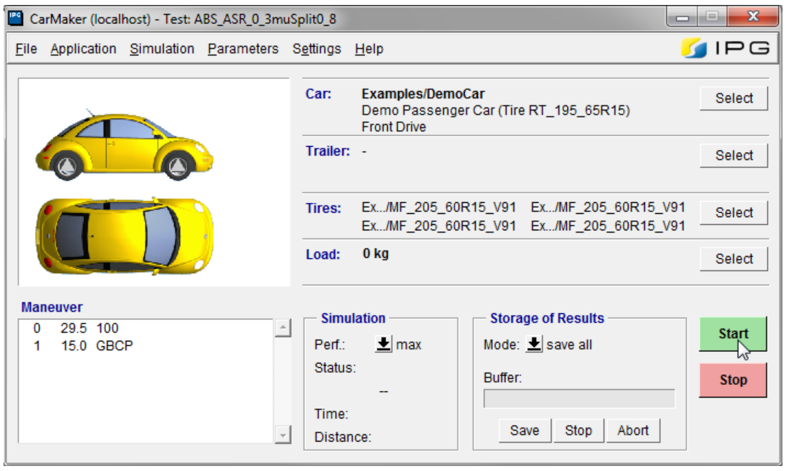
The ideal drive-system transfers the input `<Accelerator_Pedal_Input>` directly into a torque (`[EngineTrq]`), which is split in half to supply the driven (front) wheels.
The real drive-system produces `[EngineTrq]` within the `|Engine|`. It is transferred and modified through `|Clutch|`, `|Transmission|`, and `|Final_drive|` onto the driven wheels, [4].

2.4. IPG Carmaker

Carmaker is another vehicle-simulation-environment. It is a commercial software produced by the company IPG, [10].

Its intended purpose is to validate and check the provided results of MOVES. But after unexpected problems with vertical vibrations, both system switched places. Carmaker is the implementation platform, using MOVES models and interfaces. Thereafter, resulting models were reintegrated in MOVES.

2.4.1. Program and GUI


CarMaker (localhost) - Test: ABS_ASR_0_3muSplit0_8

Car: Examples/DemoCar
Demo Passenger Car (Tire RT_195_65R15)
Front Drive

Trailer: -

Tires: Ex.../MF_205_60R15_V91 Ex.../MF_205_60R15_V91
Ex.../MF_205_60R15_V91 Ex.../MF_205_60R15_V91

Load: 0 kg

Maneuver
0 29.5 100
1 15.0 GBCP

Simulation
Perf.: max
Status: --
Time:
Distance:

Storage of Results
Mode: save all
Buffer:

Figure 2.47.: IPG: GUI

Carmaker runs with a similar GUI (see figure 2.47) to MOVES. It is used to select, create, or edit simulation-scenarios and its details, [9].

The simulation itself is done by the underlying program, which uses its own models and interfaces. It is able to be integrated in Simulink, which allows the user to build own extensions and test own models. A simulation is started using the GUI's or the Simulink's start-buttons.

Running simulations can be visualized in real-time (see figure 2.48).

2.4.2. Implementation in Simulink

Carmaker is able to use Simulink for extensions and user-specific functions and systems. Therefore, the goal is to use existing MOVES-models, its layouts and conventions, and integrate them as extensions of Carmaker in Simulink, [8].

Figure 2.48.: IPG: Visualization of simulation

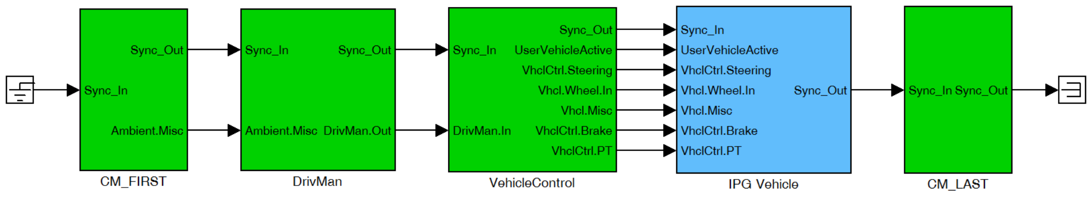


Figure 2.49.: IPG: Overview

Figure 2.49 shows the basic layout of Carmaker-components within Simulink. The blocks |CM_FIRST| and |CM_LAST| are interfaces to the simulation program. |DrivMan| holds the input of the driver and the maneuver. |VehicleControl| prepares the input for the upcoming |Vehicle|-block.

2.4.2.1. Vehicle

In figure 2.50 the components of Carmaker's |Vehicle|-block are shown.
|CarAndTrailer| contains the vehicle's body and attachments, and related components as e.g. the steering-system. |Brake|- and |PowerTrain|-systems are equal to already known brake- and drive-systems from MOVES.

Additionally, the block |User Brake + ABS + ASR| has been added as extension and replacement of Carmaker's components.

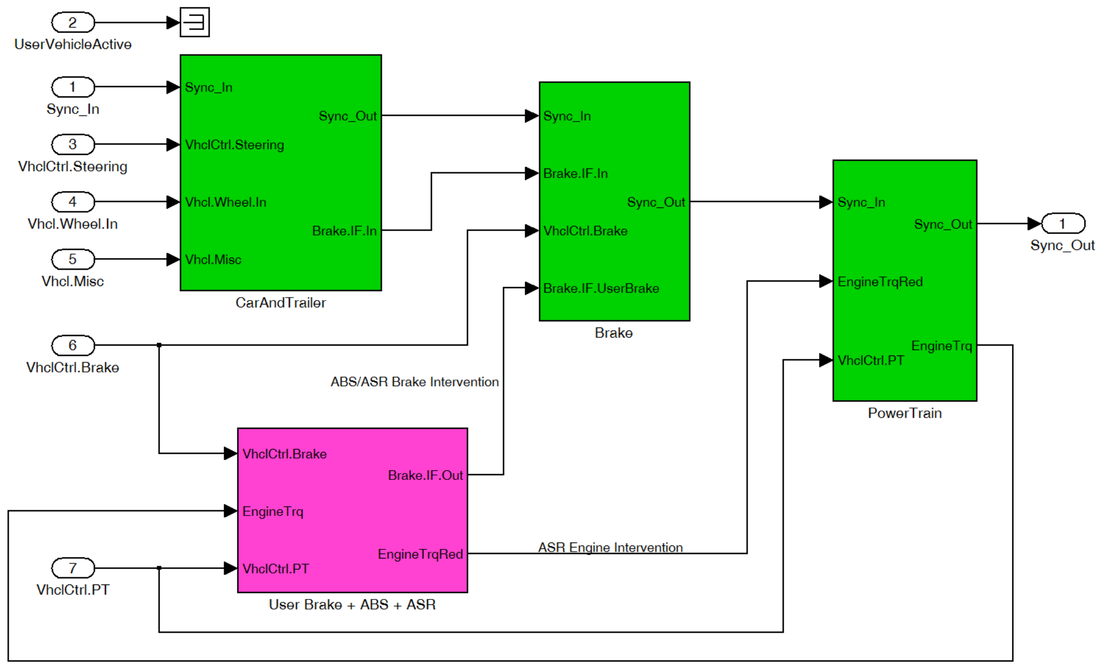


Figure 2.50.: IPG: Vehicle

2.4.2.2. Integration of MOVES-models into Carmaker

Figure 2.51 shows the integration of MOVES-models into Carmaker.

Inputs

Inputs `<VehicleCtrl.Brake>` and `<VehickeCtrl.PT>` provide the necessary powertrain- and brake-signals from Carmaker. In this case only the `<Deceleration_Pedal_Input>` and the signals `<ABS_on>` and `<ASR_on>` are transferred and evaluated.
Additionally, the current engine torque `<EngineTrq>` is provided by the `|PowerTrain|`-block.

Input `<Vhcl.v>` equals current vehicle-speed, which is needed for the brake-system's `|Hardware|`-block to set the output-signal `[braking]`.
Inputs `<Car.WheelSpd_FL>` to `<Car.WheelSpd_RR>` are the current wheel-speeds (of all four wheels) and equivalent to MOVES' `<omega>`-signals.

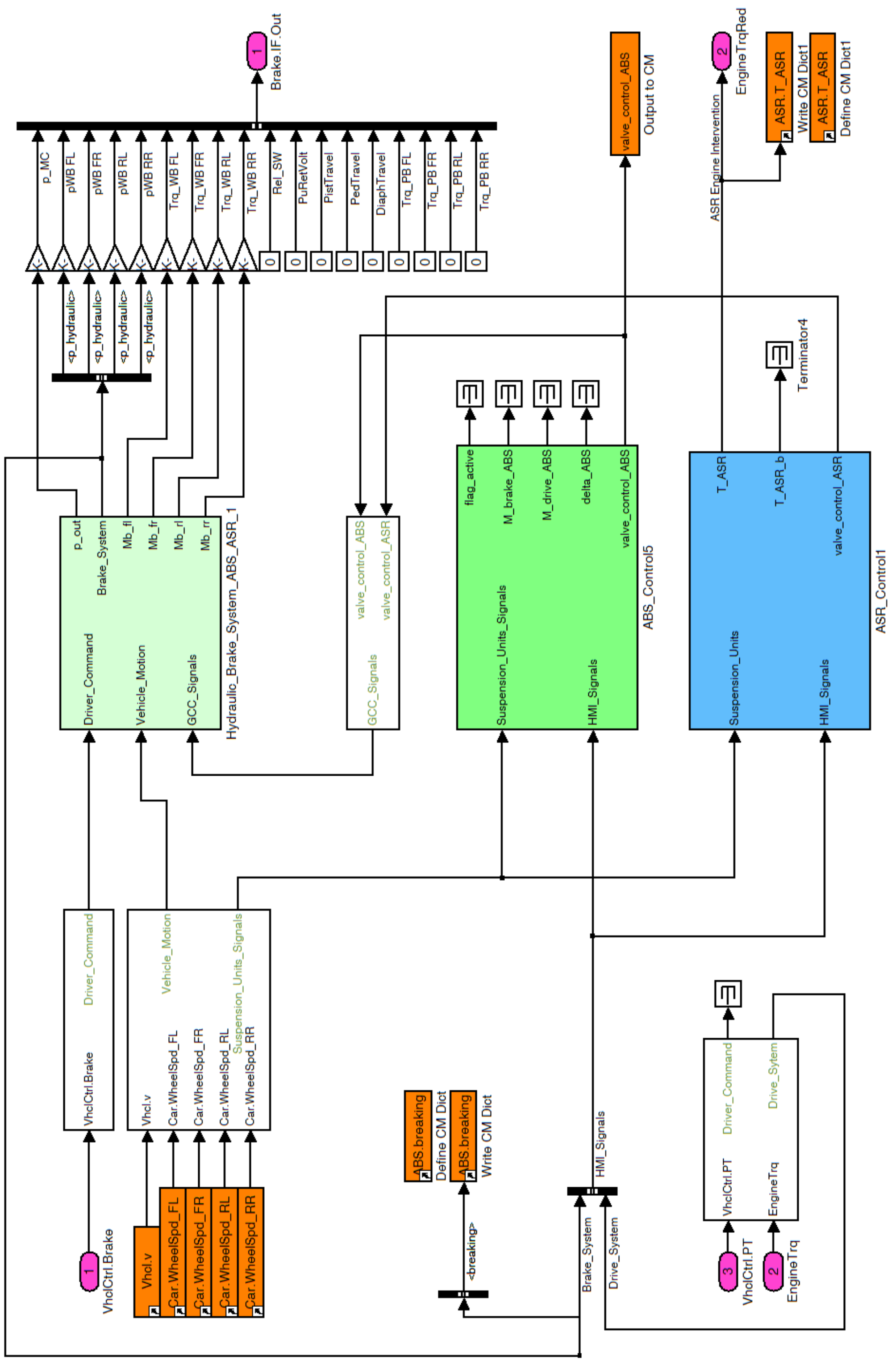


Figure 2.51.: IPG: User Brake + ABS + ASR

Blocks

The left-side (white and nameless) blocks are used to transfer the Carmaker-signals into MOVES' bus-structures.

In the middle there are the already known blocks:
|Hydraulic_Brake_System_ABS_ASR_1|,|ABS_Control5|, and |ASR_Control1|.
They have been described in previous sections (see figures 2.32, 2.15, and 2.44).

In-between there is another (white and nameless) conversion-block, which puts the outputs of the ABS- and TCS-controllers into proper bus-structures.

Outputs

All outputs from the brake-system are accordingly converted to fit Carmaker's brake-interface [Brake.IF.Out].

Inactive outputs of the ABS-controller are terminated. The [valve_control_ABS]-bus-signal is forwarded to the brake-system and for monitoring reasons also stored in Carmaker's simulation-output-variables (within block |Output to CM|).

The bus-signal, [valve_control_ASR] is also forwarded to the brake-system. Output [T_ASR] is forwarded as [EngineTrqRed] to the |PowerTrain|-block, where it directly reduces the current engine torque. This value is also written to Carmaker's dictionary (holding all simulation results) for monitoring purposes. The inactive output is terminated.

3. Results

Contents

The following pages show plotted results of several simulated scenarios (on the left page) and referring discussion on the following (right) page.

Simulations were mainly done in Carmaker.

3.1. Antilock Braking System - ABS

The functionality of the ABS-controller is shown with plots of full-braking maneuvers on different surfaces (different friction-values). Furthermore, tests were carried out on μ-split-surfaces (low- and high-friction side of road, different friction conditions for left and right ride of the vehicle) and, so called, zebra-surfaces (alternating high- and low-friction stripes).

Some of the test-results are plotted in detail, others are compared in terms of braking-distance, slip, or curve-characteristics.
The detailed plots contain three figures.

Top:

At top there are the velocities of the car (v_x in [m/s]) and the wheel (v_R in [m/s]), the reference-speed (v_{Ref} in [m/s]) and its referring slip-thresholds (λ_1, λ_2 in [m/s]).

Velocity v_x is a simulated value, which is not accessible to the ABS-controller.
Input v_R is a filtered sensor-signal measuring the current wheel-speed.
Velocity v_{Ref} is the calculated reference-speed. As long as wheels are not blocking, the average of two cross-over wheel-speeds is used. When blocking of a wheel is detected the reference-speed is extrapolated. The slip-thresholds λ_1 and λ_2 are calculated from referring reference-speed v_{Ref}.

Middle:

In the middle figure there are plots of the car's de-/acceleration (a_x in [g]) and the wheel (b_t in [g]), as well as the negative threshold (a_{init} in [g]). Additionally, the current slip (in [%]) is plotted (scale on the right side of the figure).

Slip and a_x are simulated values which are not accessible to the ABS-controller.
Deceleration b_t is calculated and smoothed from its wheel-speed (v_R).
Threshold a_{init} is a constant parameter of the ABS-controller, initially starting the ABS-cycle.

Bottom:

At bottom there is the current brake-pressure (p_{hydr} in [bar]). Also the internal state of the ABS-controller and its according valve-position of the hydro-unit (both in [1]) are monitored. For each state the hydro-unit's valve-position is set by the ABS-controller.

3.1.1. ABS-braking: straight, high friction, high speed

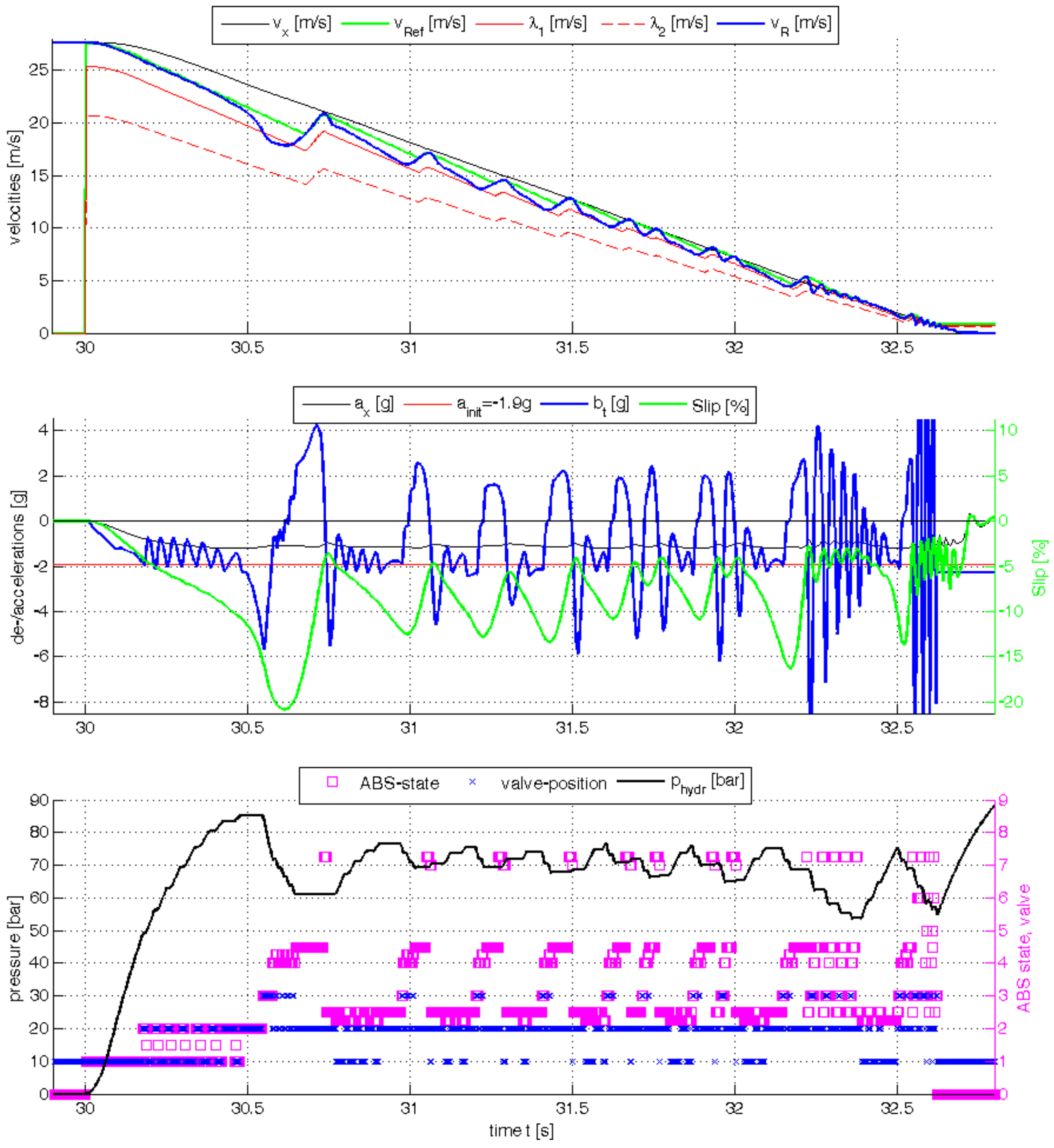


Figure 3.1.: ABS-braking: straight, high friction, high speed:
Full braking from 100km/h down to 0km/h within 2.7s and 38.7m on a high-friction surface (μ=1.0).
Slip between 3.5 and 13%, average car-deceleration around -1.2g.
Mean pressure around 70bar.

Discussion

Figure 3.1 shows an ABS-controlled full-braking maneuver from 100km/h down to 0km/h on a high-friction, straight-line road (μ=1.0). Braking starts at time t=30s and is finished after 2.7s. The brake-distance is 38.7m.

After full-brake is applied, the hydraulic pressure (p_{hydr})rises naturally, the wheel-speed (v_R) and according deceleration (b_t) are decreasing. Before locking is imminent, the pressure is increased stepwise. Thereafter, the ABS-controller cycles through its states, controlling the brake-pressure and -torque.

Velocity v_{Ref} and according slip-thresholds (λ_1, λ_2) are strongly depending on the characteristics of v_R. Especially at the beginning the reference-speed differs a lot compared to the simulated car-velocity v_x (see figure 3.2).
Later, the gaps between reference-speed and its thresholds become smaller. This causes more or faster cycles of the ABS-controller and an oscillating behavior towards the end (see figure 3.3).

At the beginning there is an oscillating trend of b_t. This is because the pressure is momentary held, when b_t falls below threshold a_{init}, allowing the wheel to accelerate. When b_t rises above threshold $-a$, the pressure is increased again. Thus, the pressure is automatically increased stepwise.
Just before the wheel locks up, its deceleration (b_t) falls rapidly below the threshold a_{init}, followed by v_R falling below threshold λ_1. This starts the ABS-control-cycle by switching to state 3 and decreasing the pressure. After the initial-process to start the ABS-control-cycle, its pressure is controlled around an ideal slip-value.
The slip follows phase-delayed to the wheel-deceleration.

At the beginning the reference-speed (v_{Ref}) is very similar to the wheel-speed (v_R). Because none of the wheels started locking, yet. Until then, the average of two crossover wheels is used to calculate v_{Ref} and its deceleration value (a_{Ref}), which is used during ABS-cycle to extrapolate the reference-speed.
The ABS-state switches between 1 and 2, increasing and holding the pressure, according to the characteristics of b_t (compare also figure 2.19). This results in a stepwise increase of brake-pressure and -torque until locking is finally imminent.

At the end the wheel-speed (v_R) and therefore also the reference-speed (v_{Ref}) exceed the current velocity of the car (v_x). This is due to the use of a constant effective wheel-radius r_{eff} (which was done as a simplification of the implementation, see section 2.3.1.3).
As an effect the ABS-controller runs faster through its cycle, while the pressure is further decreased.

There are no ABS-states 5 or 6 during the most of the braking-maneuver. This is, because the higher positive acceleration-threshold (A=5.0g) isn't reached by b_t. Parameter variations show that decreasing A to 4.0g would result in shorter brake-distances and better slip-curves (see section 3.1.10).

Following figures 3.2 and 3.3 zoom in the beginning and end of the braking-maneuver.

Zoom in beginning

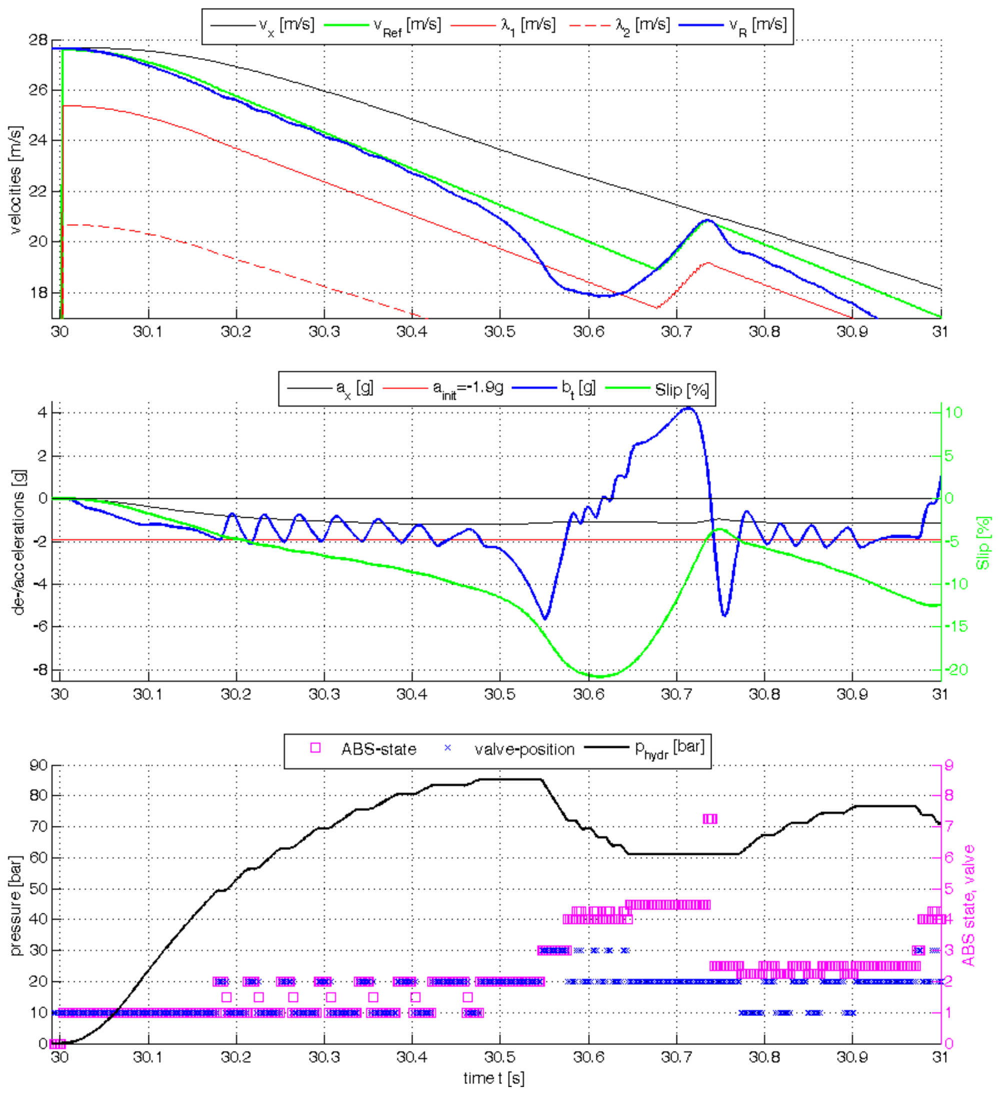


Figure 3.2.: ABS-braking: straight, high friction, high speed (zoom in beginning):
Difference between simulated v_x and calculated v_{Ref}, λ_1, and λ_2.
Oscillation of b_t before locking is imminent.
Phase-delay between b_t and slip.
State switches between 1 and 2 until v_R falls below λ_1.
No state 5 or 6 due to low positive b_t (reaches only 4.0g at positive curve).

Zoom in end

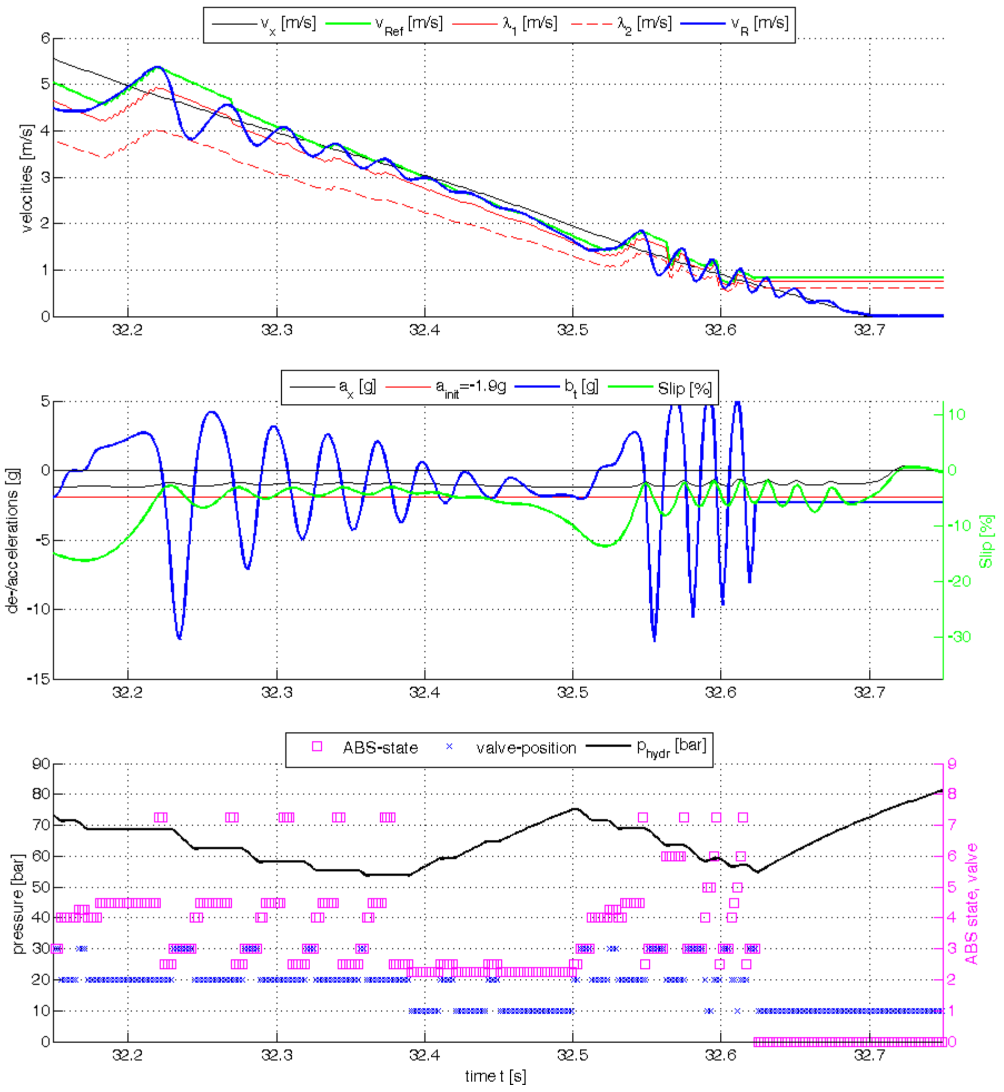


Figure 3.3.: ABS-braking: straight, high friction, high speed (zoom in end):
Velocities v_R and v_{Ref} exceed v_x (due to use of static effective radius r_{eff}).
Smaller limits λ_1 and λ_2 due to smaller velocities.
Falling oscillation of b_t with slip values below 10%.
Stepwise decreasing p_{hydr} (running faster through ABS-cycles).

3.1.2. ABS-braking: straight, low friction, high speed

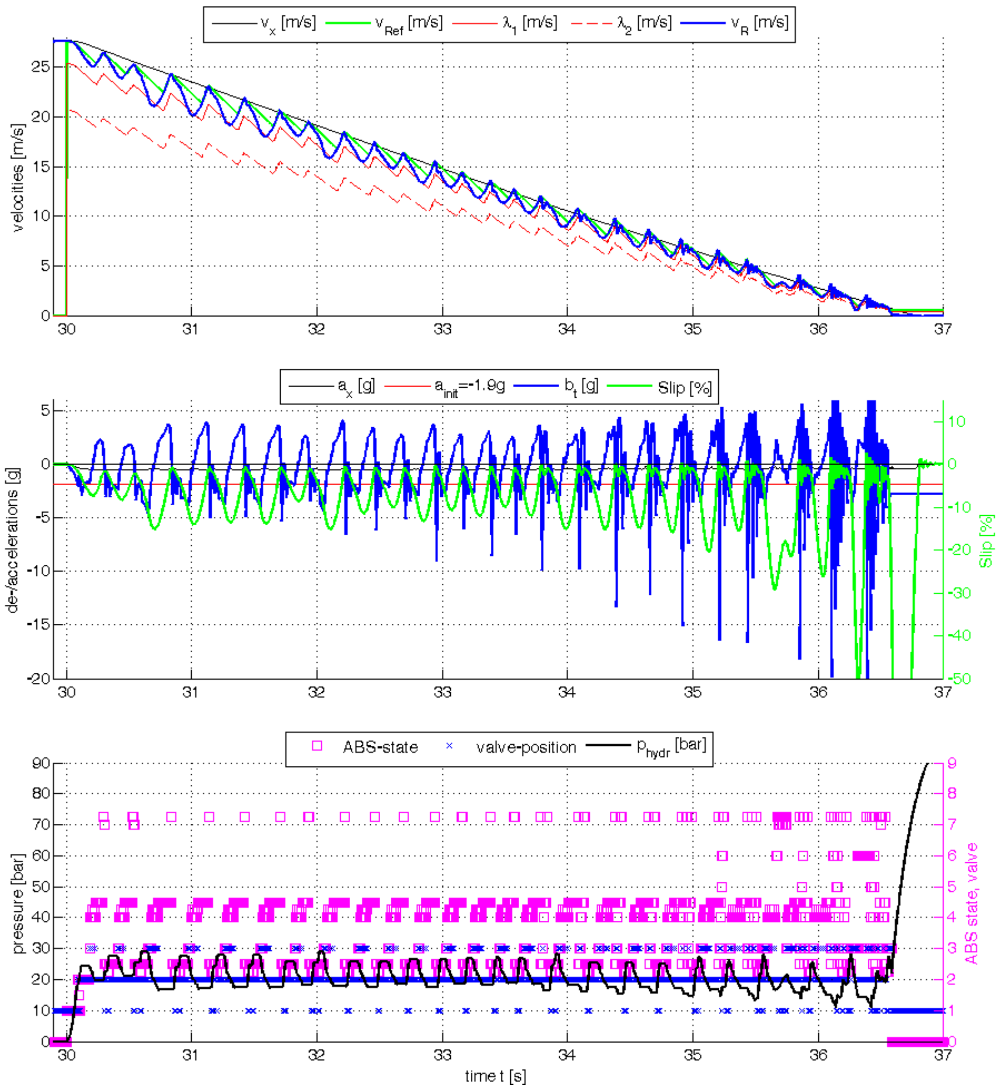


Figure 3.4.: ABS-braking: straight, low friction, high speed:
Full braking from 100km/h down to 0km/h within 6.8s and 90.9m on a low-friction surface (μ=0.4).
Slip between 0.8 and 15%, average car-deceleration around -0.5g.
Mean pressure around 20bar.
Locking is much more probable than on high-friction surfaces. As a result the ABS controller regulates around a much lower ideal pressure. Deceleration b_t and slip-values oscillate in a similar way as on high friction before but increase towards the end.

3.1.3. ABS-braking: straight, low friction, low speed

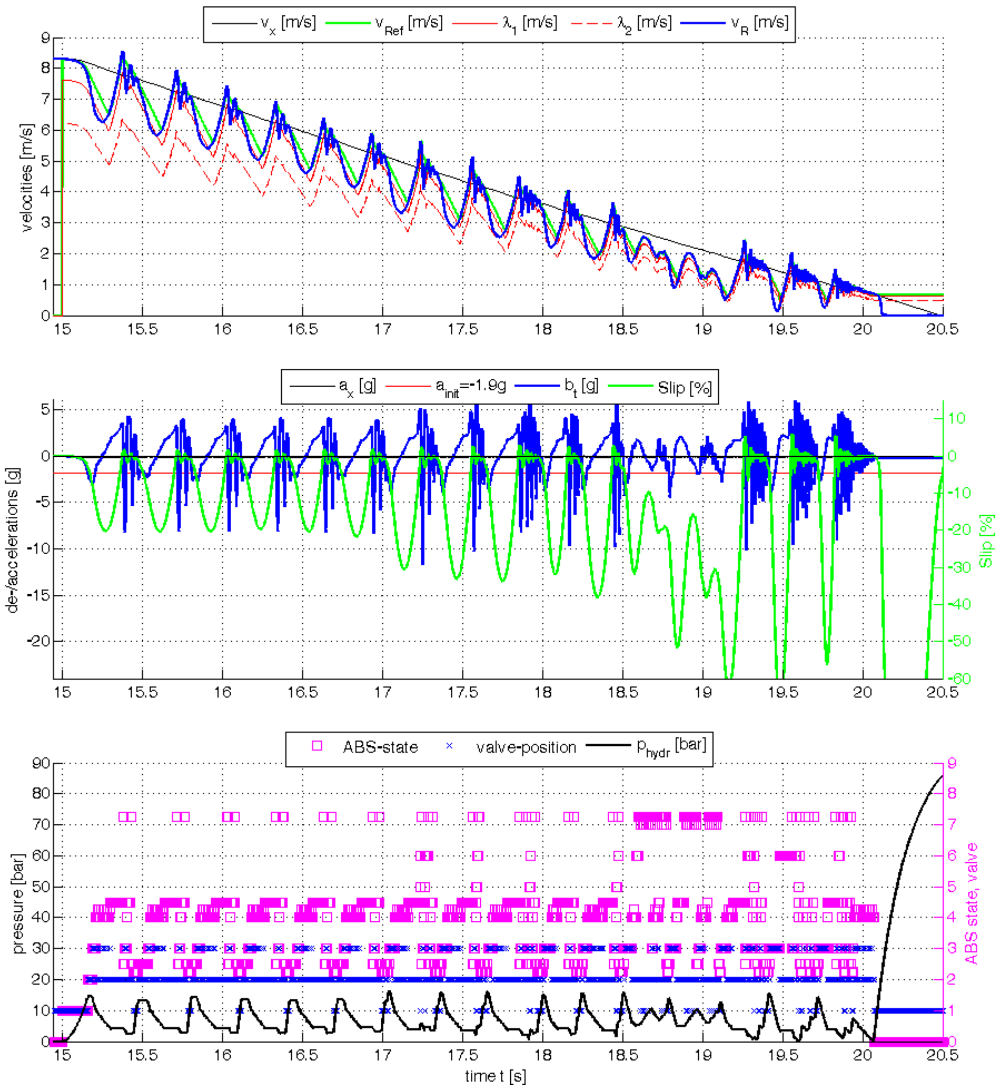


Figure 3.5.: ABS-braking: straight, low friction, low speed:
Full braking from 30km/h down to 0km/h within 5.5s and 22.4m at μ-low-surface (μ=0.2).
Slip between 0 and 30%, average car-deceleration around -0.25g.
Mean pressure below 10bar.
Velocity v_{Ref} differs a lot from simulated v_x.
Falling oscillation of b_t for each ABS-cycle.
Slip-values with increasing oscillation (lower frequency than b_t) towards the end.

3.1.4. ABS off, full-braking: straight, high friction, high speed

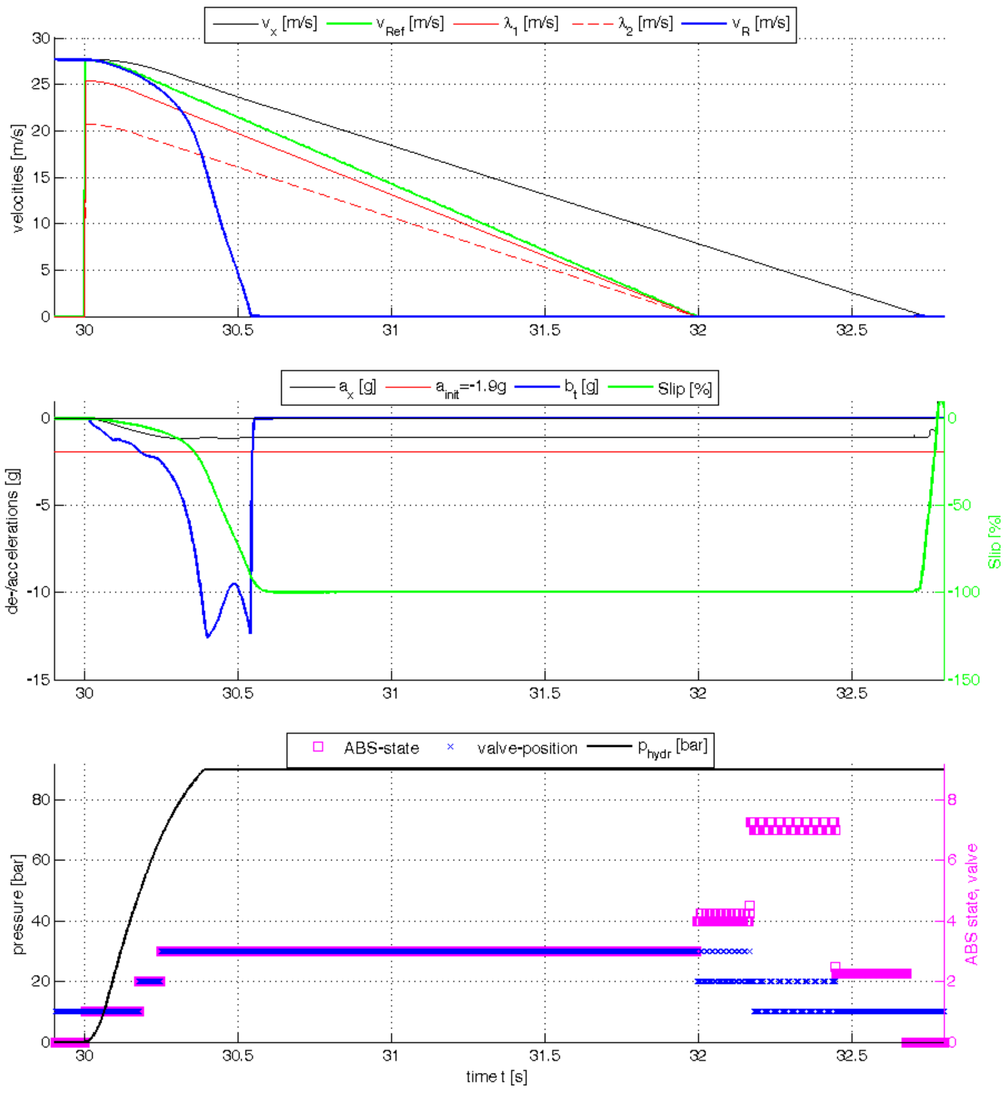


Figure 3.6.: ABS off, full-braking: straight, high friction, high speed:
Full braking from 100km/h down to 0km/h within 2.7s and 39.7m at μ-high-surface (μ=1.0).
Blocking wheels after 0.5s (slip at 100%), average car-deceleration around -1g.
Maximum pressure (90bar) reached after 0.4s.
The deviation between simulated v_x and calculated v_{Ref} become more visible with disabled ABS-control. The brake-pressure follows a natural course. The discontinuity when p_{hydr} reached its maximum shortly causes an increasing but soon again decreasing b_t-course. When slip reaches 100% (completely blocking wheels, v_R=0m/s), b_t drops to 0. ABS-state and valve-position signals stay ineffective.

3.1.5. Comparison brake distances: straight

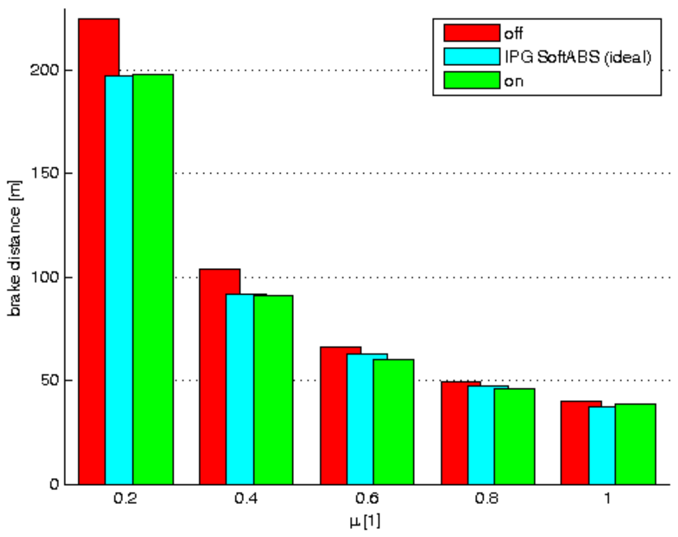


Figure 3.7.: Comparison brake distances (ABS on/off and ideal friction controller):
Full braking from 100km/h down to 0km/h on different friction conditions (from μ=0.2 up to μ=1.0).
This was done with (green) and without (red) ABS-controller and also using the IPG SoftABS (cyan), which serves as an ideal slip-controller, [8].

Results show that there is a progressive increase in brake-distance at decreasing friction-value. Furthermore, ABS-systems are always reducing the brake-distance compared to vehicles without ABS. This is only true for this ideal circumstances: straight line braking on roads with single friction-conditions.

The implemented ABS-controller works slightly better on moderate friction surfaces than the ideal controller. This is due to changing slip-tire-characteristics. The optimal slip-value (maximum transferable torque) depends on the friction (see figure 2.4). The implemented ABS-controller is able to handle such changes, as slip is generally unknown to it and wheel's de-/acceleration is unaffected. Whereas the ideal slip-controller always regulates around a single configured slip-value (optimal for a single friction-conditions, μ=1.0).

At low friction (μ=0.2) the brake-distance is again longer for the implemented against the ideal ABS-controller. This is due to rising oscillations of slip towards the end of braking.

3.1.6. Comparison brake distances/course: curve

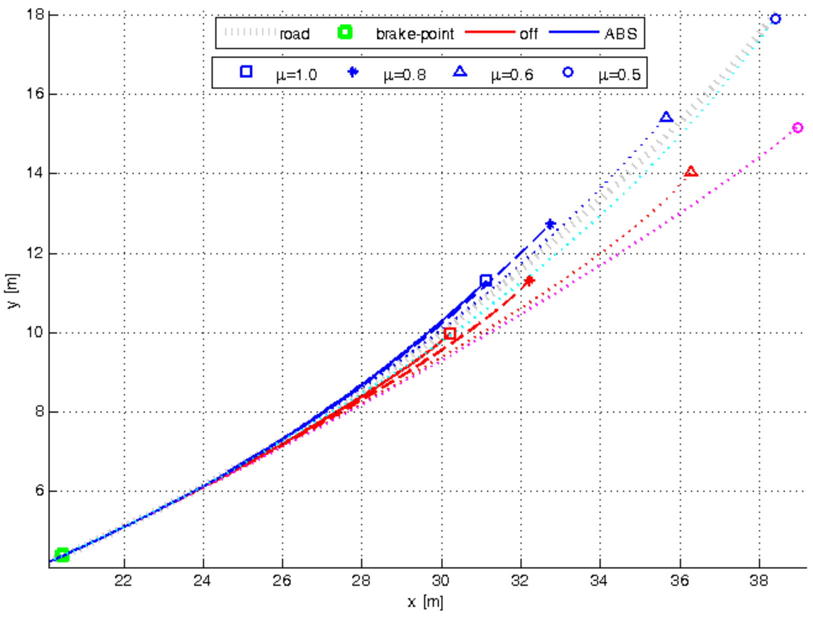


Figure 3.8.: Comparison brake distances/course (ABS on/off): curve, moderate speed: Full braking from 50km/h down to 0km/h on different friction conditions (from μ=0.5 up to μ=1.0) on a road-circle (radius of 50m) with (blue) and without (red) ABS-controller.

Results show that ABS-controlled braking maneuvers tend to understeer, whereas braking without ABS produces oversteering of the vehicle. With decreasing friction the brake-distance increases. For ABS-controlled braking steering stays possible even for low-friction values (μ=0.5, cyan).

In case of blocking wheels, the driver looses the ability to steer through the curve. As visualized in figure 3.8 the course of such a braking-maneuver (magenta) is almost straight, tangential to the course of the road at the brake-point.

3.1.7. ABS-braking: straight, μ-split, high speed

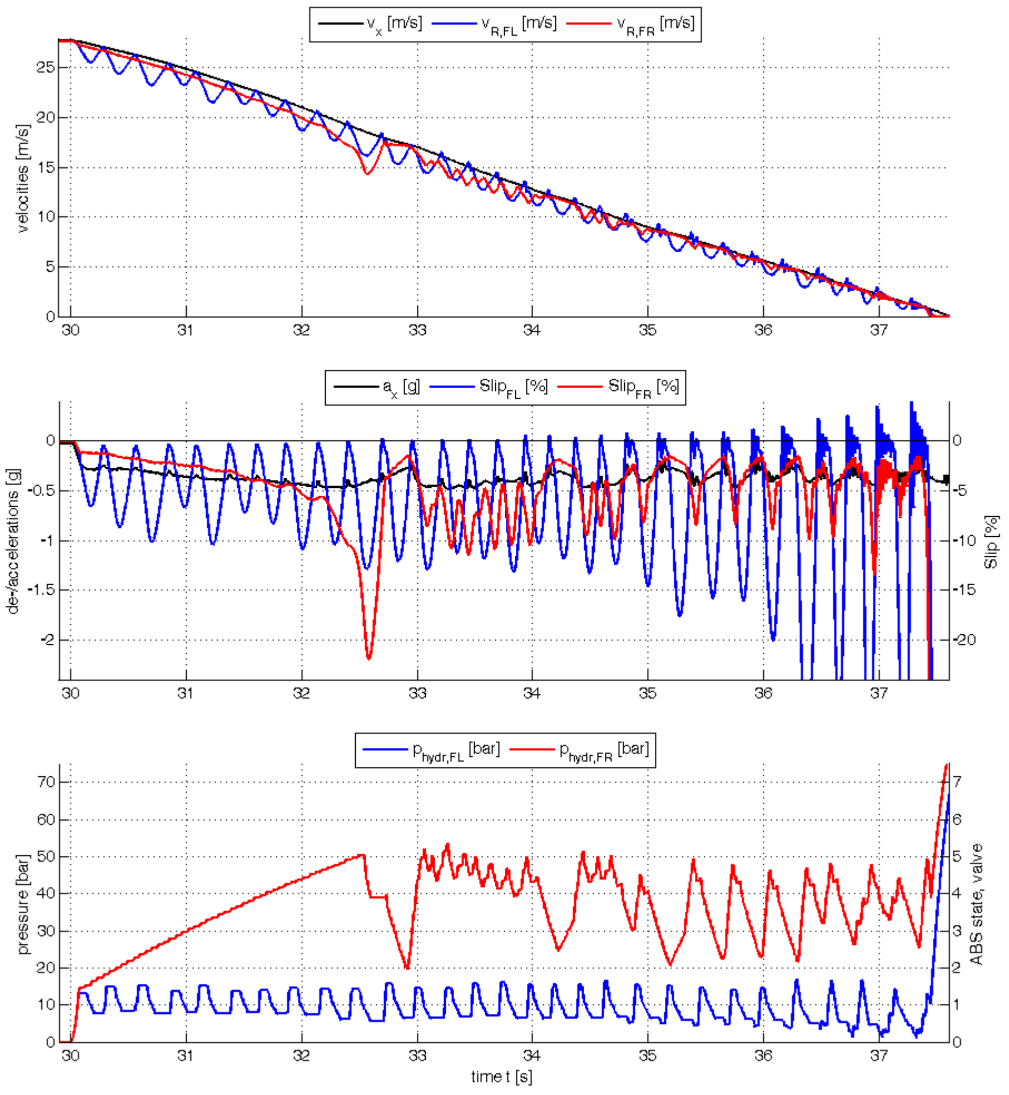


Figure 3.9.: ABS-braking: straight, μ-split, high speed:
Full braking from 100km/h down to 0 within 7.5s and 105.8m at μ-split-surface (left-side μ=0.2, right-side μ=0.8).
When the ABS-controller starts working for the left-side wheel, the pressure is stepwise increased for the opposite right-side wheel. This decreases the rate of building up the yaw-moment due to different friction-values for left and right side. As result the car stays controllable by the driver.
Simulations without the implemented ABS-controller and also using the ideal IPG SoftABS-controller made the vehicle uncontrollable. The immediate up-building yaw-moment causes the vehicle to spin of the road.

3.1.8. ABS-braking: straight, zebra friction, high speed

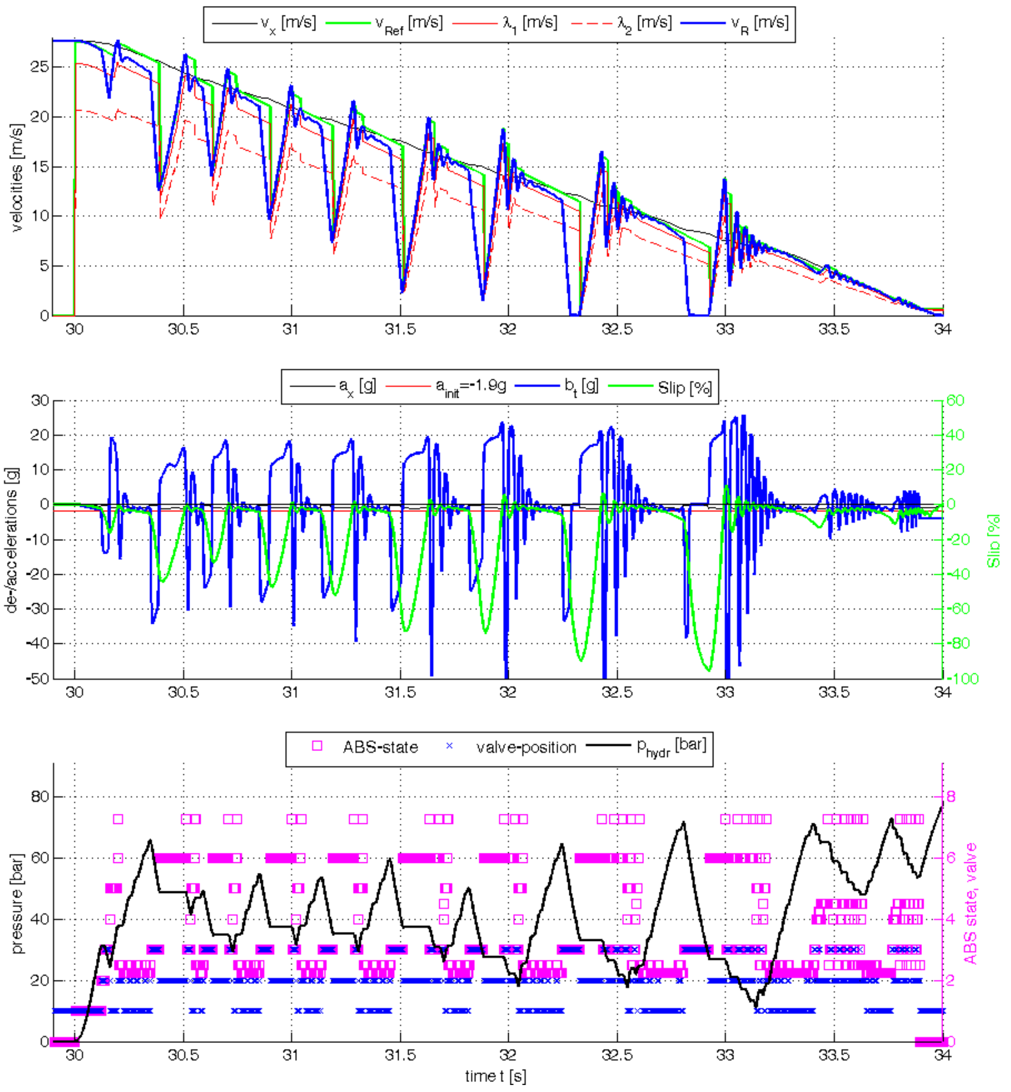


Figure 3.10.: ABS-braking: straight, zebra friction, high speed:
Full braking from 100km/h down to 0km/h within 4s and 57m on a zebra-friction surface (alternating low- (μ=0.2) and high- (μ=1.0) friction stripes).
The ABS-controller reacts to the alternating changes of friction-conditions. Compared to the following maneuver without ABS (see figure 3.11), it seems that ABS has a disadvantage in terms of brake distance. But the ABS-controller reacts to the changing friction-conditions in time to avoid blocking wheels (as opposed to following maneuver without ABS). The car stays controllable during the whole braking-maneuver.

3.1.9. ABS off, full-braking: straight, zebra friction, high speed

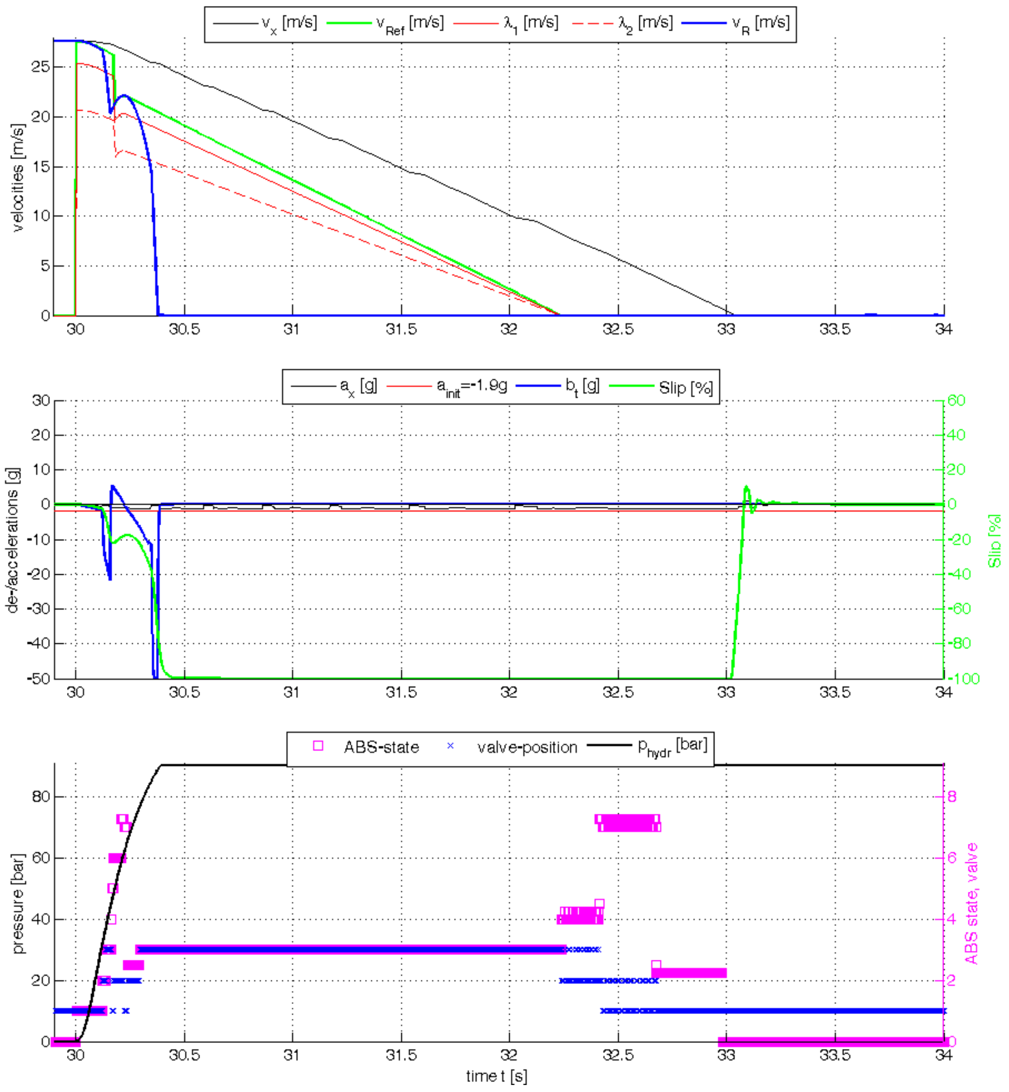


Figure 3.11.: ABS off, full-braking: straight, zebra friction, high speed:
Full braking from 100km/h down to 0km/h within 3s and 44.7m on a zebra-friction surface (alternating low- (μ=0.2) and high- (μ=1.0) friction stripes).
Blocking wheels after 0.4s (slip at 100%).
Maximum pressure (90bar) reached after 0.4s.
Stripes with low friction are flat steps in the otherwise steady descending vehicle-speed (v_x). Locking (slip at 100%) occurs faster than without zebra-profile (compare braking without ABS, figure 3.6).
ABS-state and valve-position signals stay ineffective.

3.1.10. Parameters-Variation

All configurable parameters of the ABS-controller are described in section 2.3.1 and table 2.3. The parameters were created using parameter-file `ABS_Controller_Parameters.m` (see listing A.1 in appendix).

Additionally to those controller-parameters some vehicle-specific parameters (e.g. vehicle weight) were analyzed. The variation has been done for one scenario at high speed with single high-friction condition (μ=1.0) straight-line road (see figure 3.1). For selected variations simulations were carried out for lower frictions, too.

The parameter variation shows that many parameters are depending among each other and that it isn't easy to find an optimum by manually checking one by one (it takes too much time). Instead an automatic optimization might be necessary to vary all possible parameters within reasonable boundaries and find the settings for minimal brake-distance, keeping the slip-characteristics within optimal limits. Then again, the optimized result might only be best for a single vehicle- and wheel-model combination.

The implemented ABS-controller works within limits for all simulated scenarios and model-configurations. Improvements are possible, but this is not part of this thesis.

Following table (3.1 show the range of parameter-variation and according impact (considered are slip-course and brake-distance).

Table 3.1.: ABS controller: parameters-variation

Parameter	Value	Range	Impact
`CLOCK`	0.002s	0.001s - 0.005s	Variation of `CLOCK` causes longer brake-distances. The trigger-signal of the ABS-controller has to match pressure-rates. It also influences calculation of mean-values
`p_max`	90bar	60bar - 110bar	Variation of p_{max} causes longer brake-distances. The maximum pressure determines its pressure-rates, which affects `CLOCK`. Those two parameters need to be set up initially according to system-specifications.
`a_init`	1.9g	1.4g - 2.4g	This is one of the most important parameters, as it initially starts the ABS-cycle. The parameter-value 1.9g seems to be a good compromise of nearly-optimal slip-values for higher frictions and still functioning ABS-control at low frictions.

Continued on next page

Parameter	Value	Range	Impact
	2.2g 1.7g		The best brake-distance of 38.3m uses 2.2g as parameter, but causes worse slip-curves at lower frictions. Using 1.7g as threshold causes a very nice slip-curve for high-friction and even better brake-distances at lower frictions. There are some disadvantages during moderate friction-conditions, but in sum this could be considered for further optimizations.
`a`	1.5g 1.6g	1.2g - 2.0g	All variations (besides 1.6g) cause longer brake-distances and worsening of slip-course. Literature implies thresholds of 1.4g to 1.5g, [20]. The parameter value 1.6g decreases or holds brake-distances for all friction-conditions (besides μ=0.8), which implies further optimizations. A combined test of a=1.6g and a_{init}=1.7g cause again worse results compared to the reference. This shows the dependence of several (or all) used parameters among themselves.
`a_pos`	1.5g	1.2g - 2.0g	All variations cause longer brake-distances and worse slip-courses.
`A`	5.0g 4.0g	3.0g - 6.0g	Higher values are not changing results (for high-friction conditions). Decreasing the parameter to 4.0g causes shorter brake-distances for almost all friction conditions and better slip-characteristics. It should be considered for further optimizations.
`in_lambda_1`	0.08 0.07	0.05 - 0.10	Best brake-distance at λ_1=0.07. A combined test of λ_1=0.07 and A=4.0g is significantly better at lower frictions. This should be considered in future optimizations.
`in_lambda_2`	0.25 0.30	0.10 - 0.30	Similar results for high frictions. Increasing the parameter to 0.30 causes shorter brake-distances for lower frictions.
			End of table

Parameters-Variation: vehicle mass

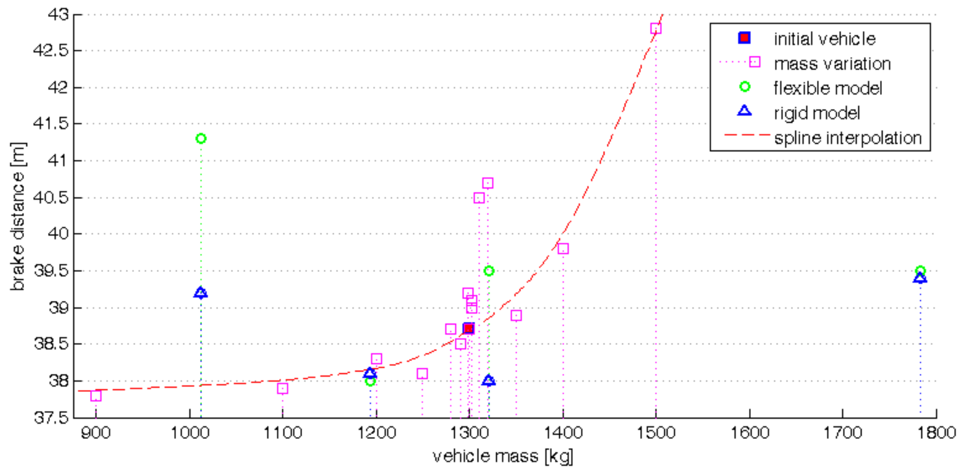


Figure 3.12.: Comparison brake distances (different vehicle-masses/-models): straight, high friction, high speed:
Full braking from 100km/h down to 0km/h on a high-friction surface (μ=1.0)

Purple squares show the different brake-distances for mass-variation of the initial (rigid) car-model. The red, dashed line implies the interpolation of the brake-distance according to current vehicle-mass. As expected, lower total weight reduces the brake-distance, whereas higher weight causes longer brake-distances. Furthermore, the ABS-parameters are optimized for a single vehicle-weight, as small deviations cause longer brake-distances than the interpolation predicts.

Blue triangles are drawn for additional rigid car-models created using Carmaker's Vehicle Data Set Generator [9], for small (m=1012kg), compact (m=1194kg), medium (m=1321kg), and luxury (m=1783kg) cars. Surprisingly, the newly created compact and medium model are using shorter brake-distances (compared to the initial vehicle-model).

Green circles are additional flexible models. As mentioned in [9], simulation results depend on detailed measurement of the vehicle. Otherwise, outcome of simulations may be distorted, as visible in figure 3.12.

Parameters-Variation: brake-force

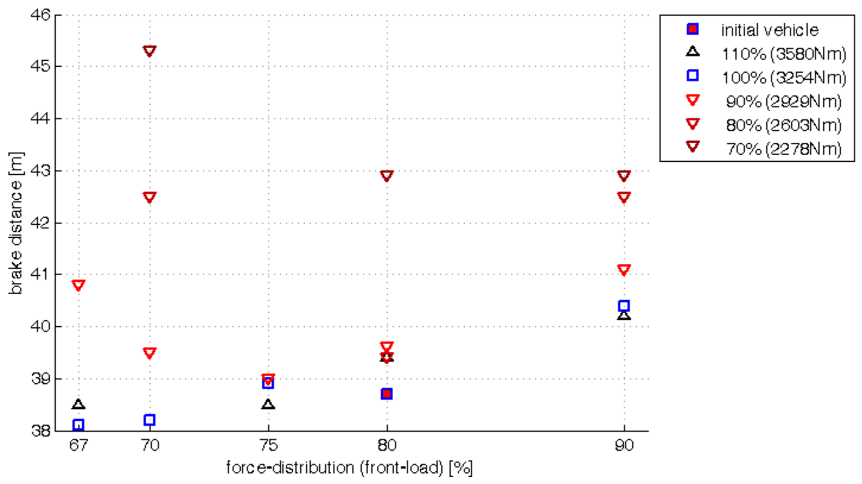


Figure 3.13.: Comparison brake distances (different brake-forces):
straight, high friction, high speed:
Full braking from 100km/h down to 0km/h on a high-friction surface (μ=1.0)
The initial vehicle has a 80:20 force-distribution between front and rear brakes.
This is due to geometrical differences in the dimension of the brake disks and pads.
The overall brake-torque is 3254Nm.

Looking at initial brake-distribution 80:20, increasing or decreasing the overall total brake-torque results in immediate more brake-distance. Thus, the brake-force and distribution are already optimized for this pairing.

Moving the brake-distribution even further to the front-wheels would leave the whole work to them. Rear wheels won't build up enough forces for the ABS to control them around optimal slip, whereas front wheels would need constant regulation to avoid blocking.
Decreasing the brake-distribution (while keeping the overall force constant), causes shorter brake-distances, which seems to be a better solution. But if the force is shifted from front to rear brakes, front-wheels loose ABS-control. Forces at the front wheels are too low for an ABS-interaction, whereas rear wheels tend to lock. This conditions needs to be avoided at any chance as blocking rear wheels mean immediate loss of steerability and stability of the vehicle.

Decreasing the overall brake-force causes longer brake-distances. A possible start for further optimization would be an increase of forces while changing the distribution towards the rear wheels.

3.1.11. Discussion

Results show a stable working ABS-controller. Parameter variation imply that it is nearly optimized for a single vehicle-model, its mass, and the used tire-model. In deviating conditions it still provides reasonable results.

On surfaces with a single friction-condition the implemented controller has smaller brake-distances than without ABS . The ABS controller tries to keep the slip-values below or near the optimal value. Whereas, braking without ABS can result in blocking wheels, which reduces the maximum transferable brake-torque (compare figures 2.4 and 2.12).
Additionally, all forces are spent in longitudinal direction, which makes steering uncontrollable. Simulating braking-maneuvers during curves, shows the advantage of ABS controlled vehicles against blocking wheels (see figure 3.8). Furthermore, continuous braking with blocking wheels will cause brake-spots on tires which can irrevocable damage or destroy them.

Comparing the implemented ABS controller to an ideal slip-controller, the ideal controller might have an advantage under certain conditions. But the ideal controller is not adaptable to different conditions. It will always regulate the pressure according to a given slip-parameter. This slip-parameter might be optimal for some tire-models and surface-conditions resulting in the shortest brake-distances. But if conditions are different or changing, the implemented ABS-controller gains advantage as it is independent from actual slip-values.
The controller has been implemented without simulated values, like slip or car-velocity, to stay close to real-world conditions. Those values have to be estimated using measured wheel-speeds. As mentioned before, this doesn't have to be a disadvantage, as the controller will automatically adapt to different friction-conditions with different optimal slip-values.
But the controller lacks a certain degree of precision. Therefore, it needs to be set up with low thresholds to avoid blocking of wheels for all conditions. This results in suboptimal brake-distances compared to ideal slip-controllers (at optimal slip).

On μ-split surfaces (one side has low-, the other one high-friction) the implemented ABS controller confirms its purpose. Where not-ABS-controlled vehicles start spinning, the implemented ABS-controller keeps the yaw-moment controllable. Also the used ideal ABS becomes unsteerable.
The difference in brake-forces between left and right side of the car results in a yaw-moment which has to be handled by the driver. Without proper countermeasures of the ABS-controller the slew-rate is to high for the driver to react in time.

Changing friction-condition (from low to high and back) during braking need also be addressed by ABS-controllers. This is automatically handled by the implemented one, as it relies on (unaffected) wheel-decelerations to regulate brake-pressure and control slip. It causes much longer brake-distances compared to blocking wheels (braking without ABS), as the brake is always lifted, when changing from high to low friction. The benefit is again the steer-ability during the whole braking maneuver.

3.2. Traction Control System - TCS

Following plots show full-acceleration maneuvers with and without TCS. Simulations were done on low-friction and μ-split-surfaces (μ-high and μ-low side of road).

The plots contain three figures.

Top:

At top there are the velocities of the car (v_x in [m/s]) and the wheel-speeds (ω_{XY} in [rad/s]). X indexes the longitudinal direction of the vehicle, where F refers to the front and R to the rear wheels. Y indexes the lateral direction of the vehicle, where L refers to the left and R to the right side.
Velocity v_x is a simulated value, which is not accessible to the TCS-controller.
Wheel speed ω_{XY} is a filtered sensor-signal measuring the current wheel-speed.

Middle:

In the middle figure there are plots of current engine torque ($EngineTrq$ in [Nm]) and TCS-controlled reduced engine torque ($EngineTrqRed$ in [Nm]). Furthermore, the current drive- ($M_{d,XY}$ in [Nm]) and brake-torques ($M_{b,XY}$ in [Nm]) are plotted for referring wheels.
Input $EngineTrq$ is provided by the powertrain-unit as upper limit for the TCS to reduce when wheels are spinning.
Torque $M_{d,XY}$ are simulated results, depending on the current friction between tire and road, and distribution of torques according to the current powertrain-settings (e.g. gear-selection and final drive-distribution). Drive-torques are only plotted for the driven (front) wheels.
Torque $M_{b,XY}$ are also simulated results, depending on the interaction of TCS- (and ABS-) controllers and on current friction conditions.

Bottom (low friction):

For low-friction surfaces the longitudinal slip values for the driven (front) wheels ($slip_{XY}$ in [1]) are plotted at the bottom. This is a simulated value to show the (avoidance of) spinning of wheels. Opposed to brake-maneuvers slip can exceed 100% (or 1). Spinning wheels can rotate n-times faster than their reference, whereas blocking wheels can only decrease their wheel-speed down to 0km/h (which equals 100% brake-slip).

Bottom (μ-split):

In case of μ-split conditions the current steering wheel torque (in [Nm]) and angle (in [rad]) are plotted instead of slip. Both values are simulated results, to show the driver's (necessary) interaction and load.

3.2.1. TCS-controlled full-acceleration: straight, low friction

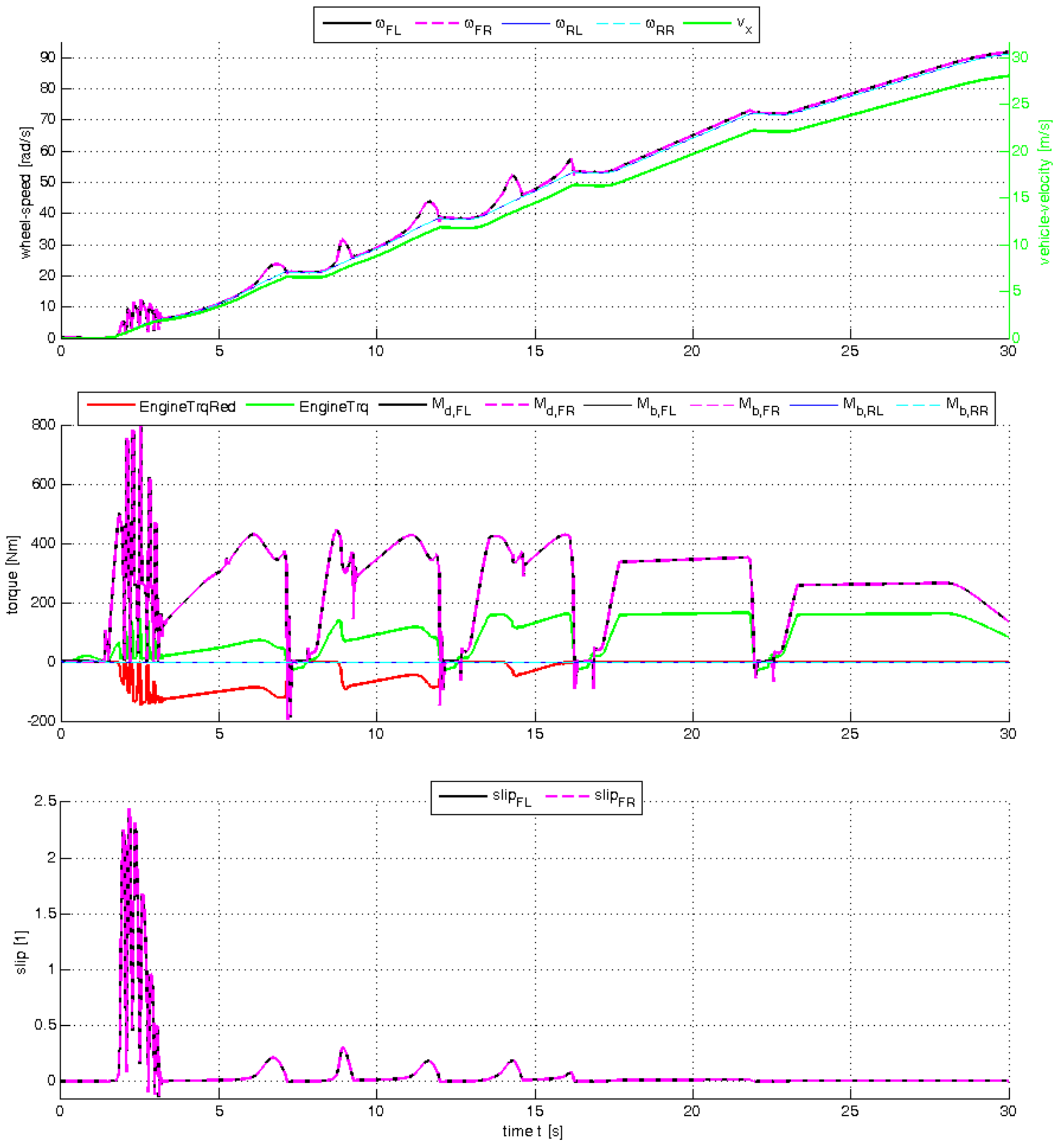


Figure 3.14.: TCS-acceleration: straight, low friction:
Full acceleration from 0km/h to almost 30km/h with gear changes on a low-friction, straight-line road (μ=0.3)

Discussion

Figure 3.14 shows a TCS-controlled full-acceleration maneuver from 0km/h to almost 30km/h on a low-friction, straight-line road (μ=0.3). Acceleration starts around 2s (after the engine is turned on, first gear is selected, and the clutch is closed).

Top: The simulated vehicle-velocity (v_x) visualizes also the acceleration of the car. During gear-shifts the velocity stays equal or decreases slightly before the clutch is closed again and acceleration continues. One shifting-maneuver takes one second. The reference-speed is calculated as mean value over the non-driven (rear) wheels. TCS regulates the wheel-speeds of the driven (front) wheels to stay close to the optimal slip, trying to avoid spinning.

Middle: Positive values are the current engine torque (green) and its referring drive-torques applied to the driven wheels. Negative values show the TCS-controlled reduced engine torque (red). Brake-torques are not applied, if there is no friction-difference between left an right side of the vehicle.
During the first three gears the provided engine torque exceeds the maximum transferable drive-torques, which causes spinning wheels. Thus, the TCS reduces the engine-torque, resulting in the current engine torque.

Bottom: Bottom figure show slip-values of driven wheels.

During first gear the excess torque causes the driven (front) wheels to exceed slip-limit and even spin (see figure 3.15). Driven wheels start immediately spinning. Thus, engine torque is reduced to its minimum.
Afterward, the engine torque is increased again to regulate slip around its optimum. This happens usually slowly and controlled as during gear two (t=8-12s) and three (t=13-17s). Only during first gear there is too much excessive torque and the vehicle needs to overcome the inertia before acceleration is starting. With TCS-controller trying to regulate an optimal slip-value, this results in oscillations of torque and slip at the beginning. This is due to the initial on-off-behavior of the traction-controller.
Afterwards, the controller works as intended.

Even tough, there are spinning wheels and oscillation at the beginning, the slip doesn't exceed 2.5. Later, during following acceleration it stays even below 0.3. Compared to following maneuver without TCS, this is a real improvement (see figure 3.16). The driven wheels are spinning until after the third gear-change, when drive-torques are already low and vehicle speed is already high enough. The current engine torque is almost completely wasted onto the spinning of wheels instead of acceleration of the vehicle. This is a big disadvantage compared to TCS-controlled acceleration.

Following plotted results show the same simulation-scenario without TCS (see figure 3.16) and zoom in at the beginning (see figures 3.15. A comparison of different friction conditions follow afterwards (see figure 3.17).

TCS-controlled full-acceleration: zoom in beginning

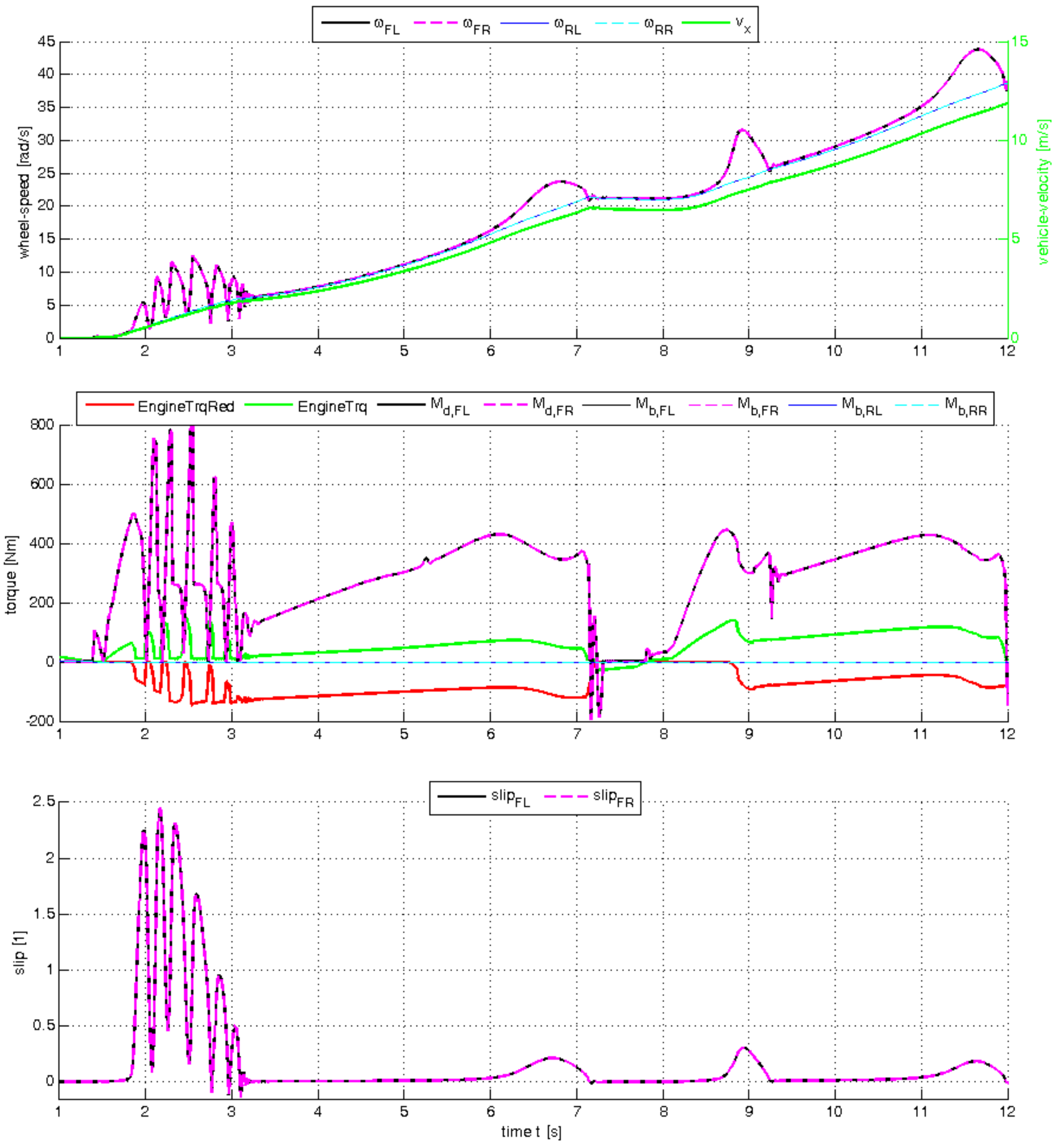


Figure 3.15.: TCS-acceleration: straight, low friction (zoom in beginning):
Full acceleration from 0km/h to almost 30km/h with gear changes on a low-friction, straight-line road (μ=0.3).
On-off-behavior for *EngineTrqRed* (red) at the beginning.

TCS off, full-acceleration: straight, low friction

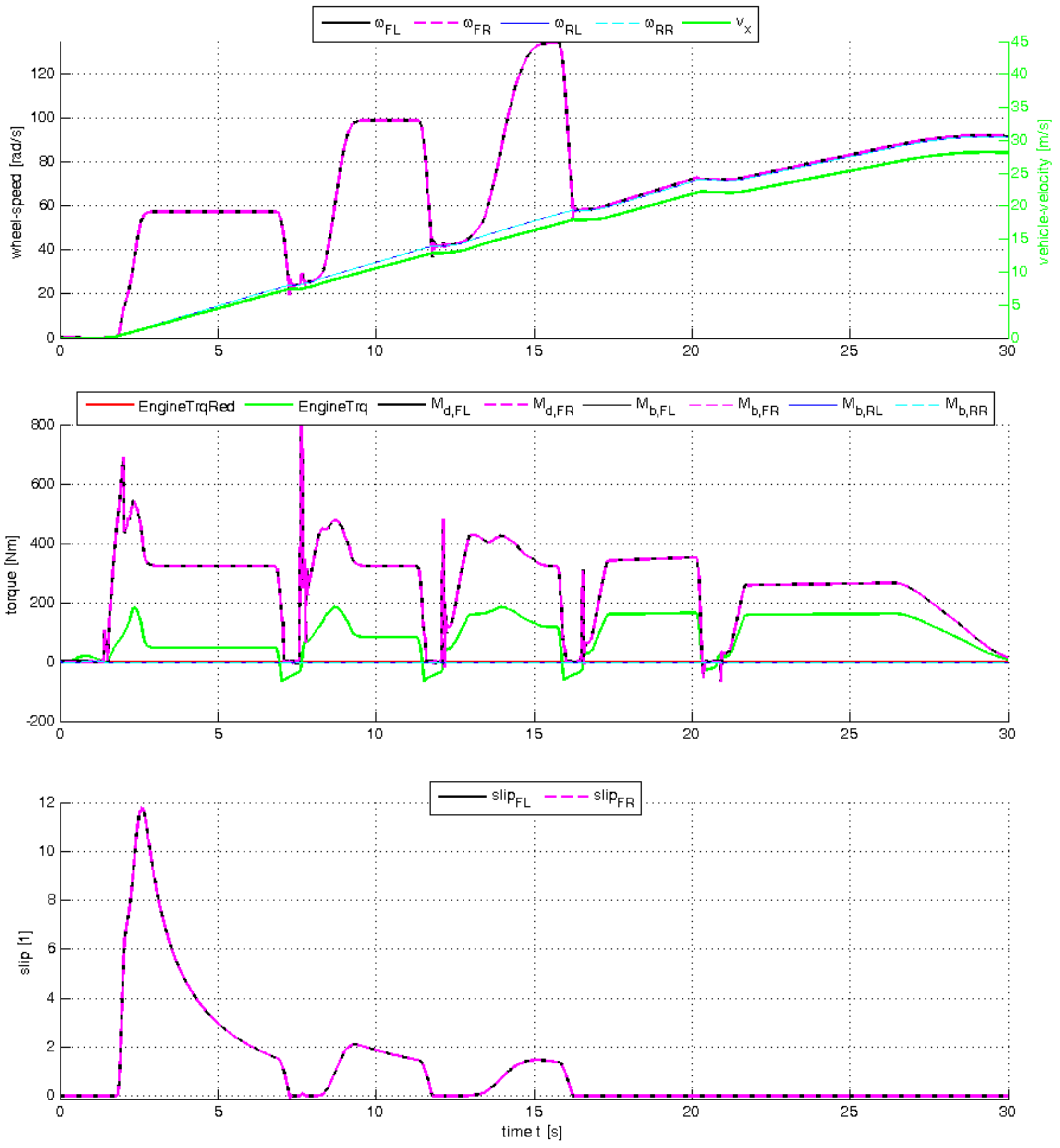


Figure 3.16.: TCS off, full-acceleration: straight, low friction:
Full acceleration from 0km/h to almost 30km/h with gear changes on a low-friction, straight-line road (μ=0.3).
ω_{FL} and ω_{FR} rise above twelve-times the current reference-speed.
Torque is wasted on spinning wheels. The vehicle accelerates only slightly faster compared to previous TCS-controlled maneuver.

3.2.2. Compare slip values (TCS on/off): low friction

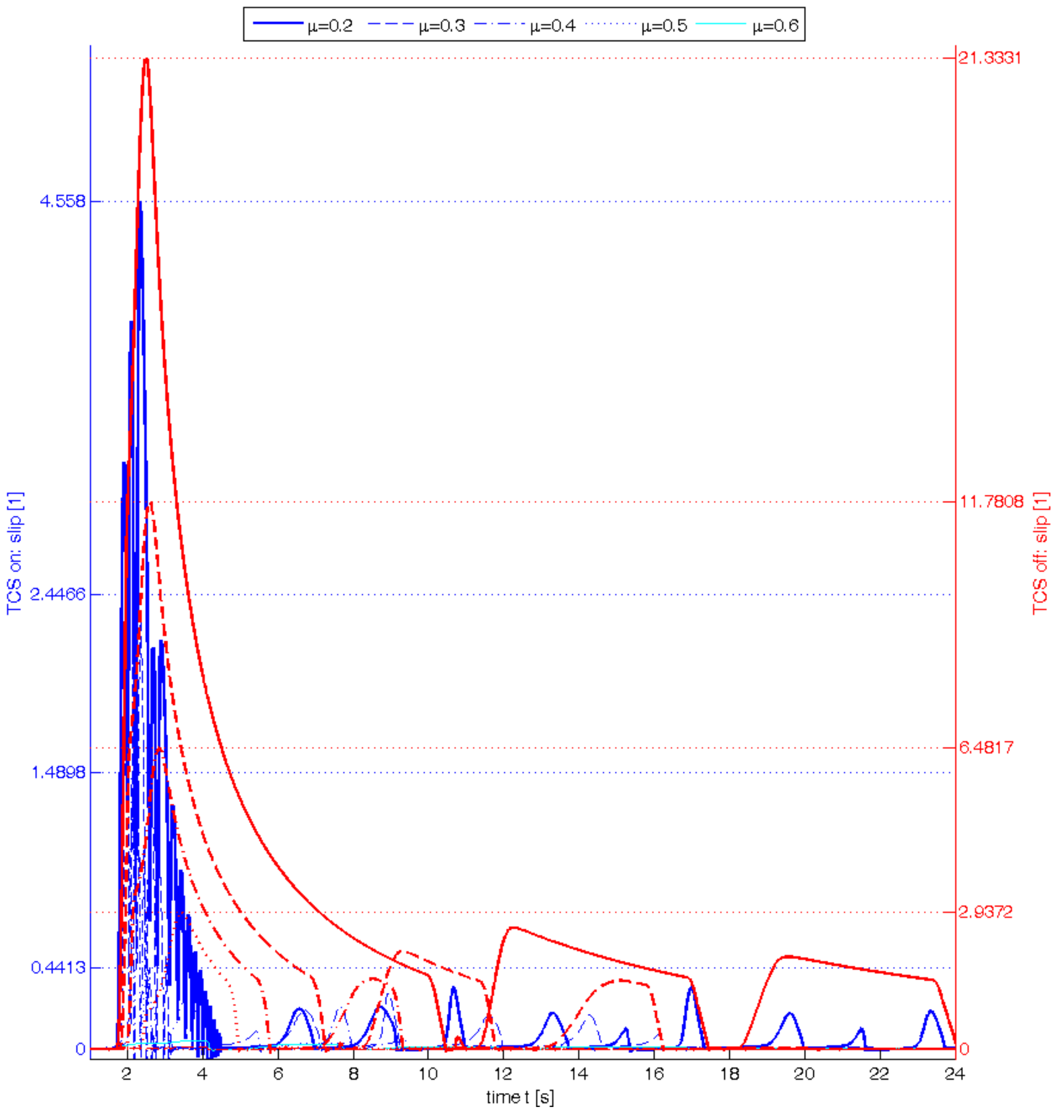


Figure 3.17.: Compare slip values (TCS on/off): low friction:
Full acceleration with gear changes on a low-friction, straight-line road (μ=0.2-0.6) with (blue) and without (red) TCS.

Discussion

Figure 3.17 compares full-acceleration maneuvers on a straight-line road under different friction conditions with (blue) and without (red) TCS-controller turned on. Acceleration starts around 2s (after the engine is turned on, first gear is selected, clutch is closed).

In contrast to braking, acceleration slip-values can achieve values greater than 1, as spinning wheels can still accelerate their wheel-speed, whereas blocking wheels can only decrease to 0. A slip-value of 1 equals 100% slip, which means that the wheel is spinning or turning twice as fast as comparable non-driven wheels.
Slip values rise up to 4.5 for TCS-controlled and 21 for non-controlled full-acceleration on low friction (μ=0.2) at the beginning. Thereafter, slip values stays well below 0.5 for TCS-controlled maneuvers for all different friction-conditions.

In case of non-controlled full-acceleration the achieved slip-values depend on the current friction-condition. With increasing friction the slip decreases. For moderate friction μ=0.5 there is still spinning with slip of around 3 for the first gear. For higher frictions there is no more spinning.

The first gear has an excess of torque which leads to the slip-peaks at the beginning of the maneuver (shown in figure 3.17). For maneuvers without TCS, the peak decreases and slowly approaches zero, while car is slowly accelerating. Before it could actually reach zero, the driver needs to shift to the next gear.
In case of TCS-controlled acceleration the initial peak during first gear is reduced by regulating the current engine torque. As the car needs to start accelerating and as the engine-torque must not be reduced to zero, it results in an oscillating behavior. This is due to the currently low velocities and referring low thresholds which start or stop the TCS-controller (see figure 3.15).
Still the TCS-controlled slip-values are more than four-times lower than without TCS.

After shifting to higher gears the slip-values are regulated to an optimum. Short peaks appear shortly after gear changes and during acceleration as TCS-controller is turned on or off when passing thresholds. But the slip-values stay well below 0.5 which is still close to the optimal value and periods are kept short. In contrast, slip-values of maneuvers without TCS are still higher than 1 and the periods of spinning wheels are for a couple of seconds. Constantly spinning wheels cause tire-degradation.

3.2.3. TCS-controlled full-acceleration: straight, μ-split

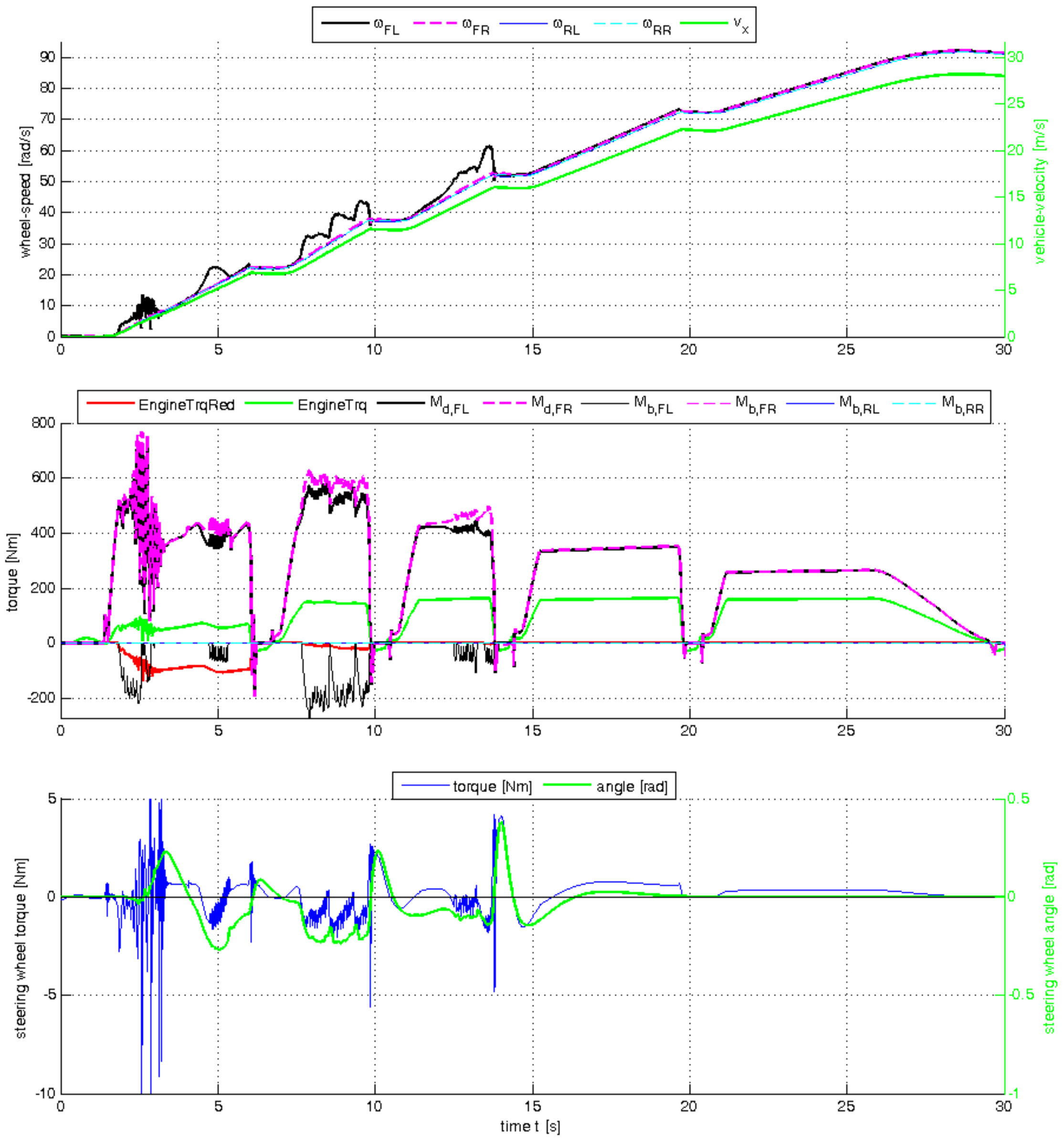


Figure 3.18.: TCS-acceleration: straight, μ-split:
Full acceleration from 0km/h to almost 30km/h with gear changes on a μ-split surface (left side low friction with μ=0.3, right side high friction with μ=0.8).

Discussion

Figure 3.18 shows a TCS-controlled full-acceleration maneuver from 0km/h to almost 30km/h on a μ-split, straight-line road (left side low friction with μ=0.3, right side high friction with μ=0.8). Acceleration starts around 2s (after the engine is turned on, first gear is selected, and the clutch is closed).

Top: The simulated vehicle-velocity (v_x) visualizes also the acceleration of the car. During gear-shifts the velocity stays equal or decreases slightly before the clutch is closed again and acceleration continues. One shifting-maneuver takes one second. The reference-speed is calculated as mean value over the non-driven (rear) wheels. TCS regulates the wheel-speeds of the driven (front) wheels to stay close to the reference-speed and optimal slip, and avoid spinning.

Middle: Positive values are the current engine torque ($EngineTrq$), and its referring drive-torques applied to the driven wheels. Negative values show the TCS-controlled reduced engine torque ($EngineTrqRed$) and applied brake-torques.
At the beginning the provided engine torque is too high, which would cause spinning wheels. As left side has a low-friction surface, the front-left wheel tends to spin. Therefore brake-torque is applied to this wheel, to maintain distribution of torque through the final drive onto the high-friction sided wheel.
Also the TCS reduces the engine-torque during the first two gears.

Bottom: Bottom figure show load and (necessary) interaction of the driver during the maneuver.

Initially, there is oscillating behavior of both driven wheels, as drive-torque of first gear exceeds maximum transferable torques by far (see figure 3.19). This conditions causes torque-peaks at the steering wheel and the driver has to counter-steer against upcoming yaw-moment. After this initial turbulence the vehicle steadily accelerates without spinning wheels. The engine torque is slowly and steadily increased again as long as slip-values stay within boundaries.
The front left wheel starts to slip again, which is immediately countered by brake-torque until slip-values return to optimal values. This is also noticeable for the driver at the steering-wheel.

In comparison to TCS-controlled acceleration, the vehicle without TCS accelerates faster (see figure 3.20). But the front left wheel is spinning until the third gear-change. This a huge disadvantages compared to the TCS-controlled acceleration, as the current engine torque is almost completely wasted onto the spinning, low-friction sided wheel. Moreover, continuous spinning causes tire-degradation.
Also steering wheel torque and angle reach higher values and have to be longer applied. Thus, the load and interaction of the driver is higher or more in terms of torque, angle and duration.

The benefit of faster acceleration without TCS is outnumbered by the disadvantages of (constantly) spinning wheels and driver-interaction.

3.2.3.1. TCS-controlled full-acceleration: straight, μ-split (zoom in beginning)

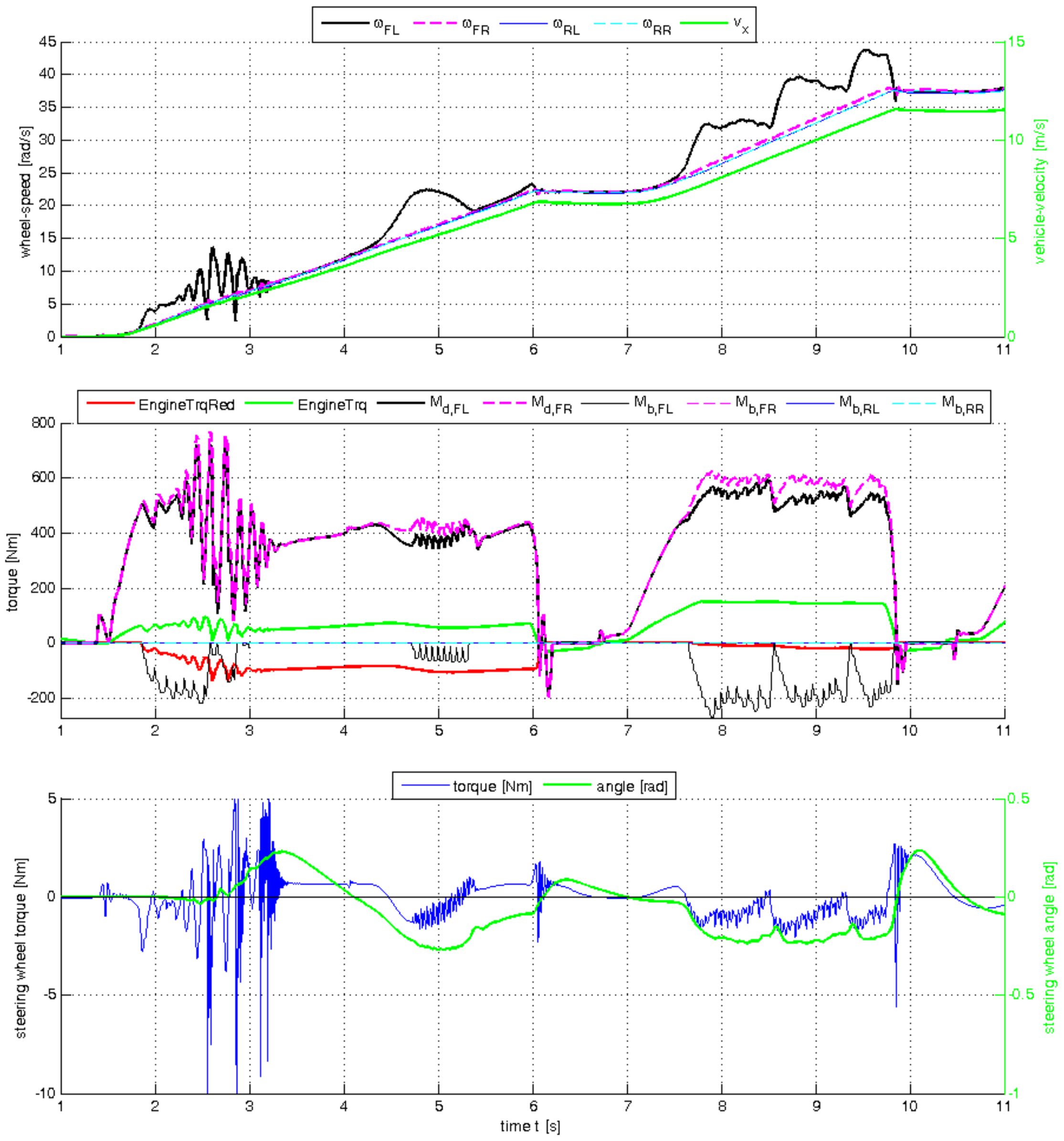


Figure 3.19.: TCS-acceleration: straight, μ-split:
Full acceleration from 0km/h to almost 30km/h with gear changes on a μ-split surface (left side low friction with μ=0.3, right side high friction with μ=0.8).

3.2.3.2. TCS off, full-acceleration: straight, μ-split

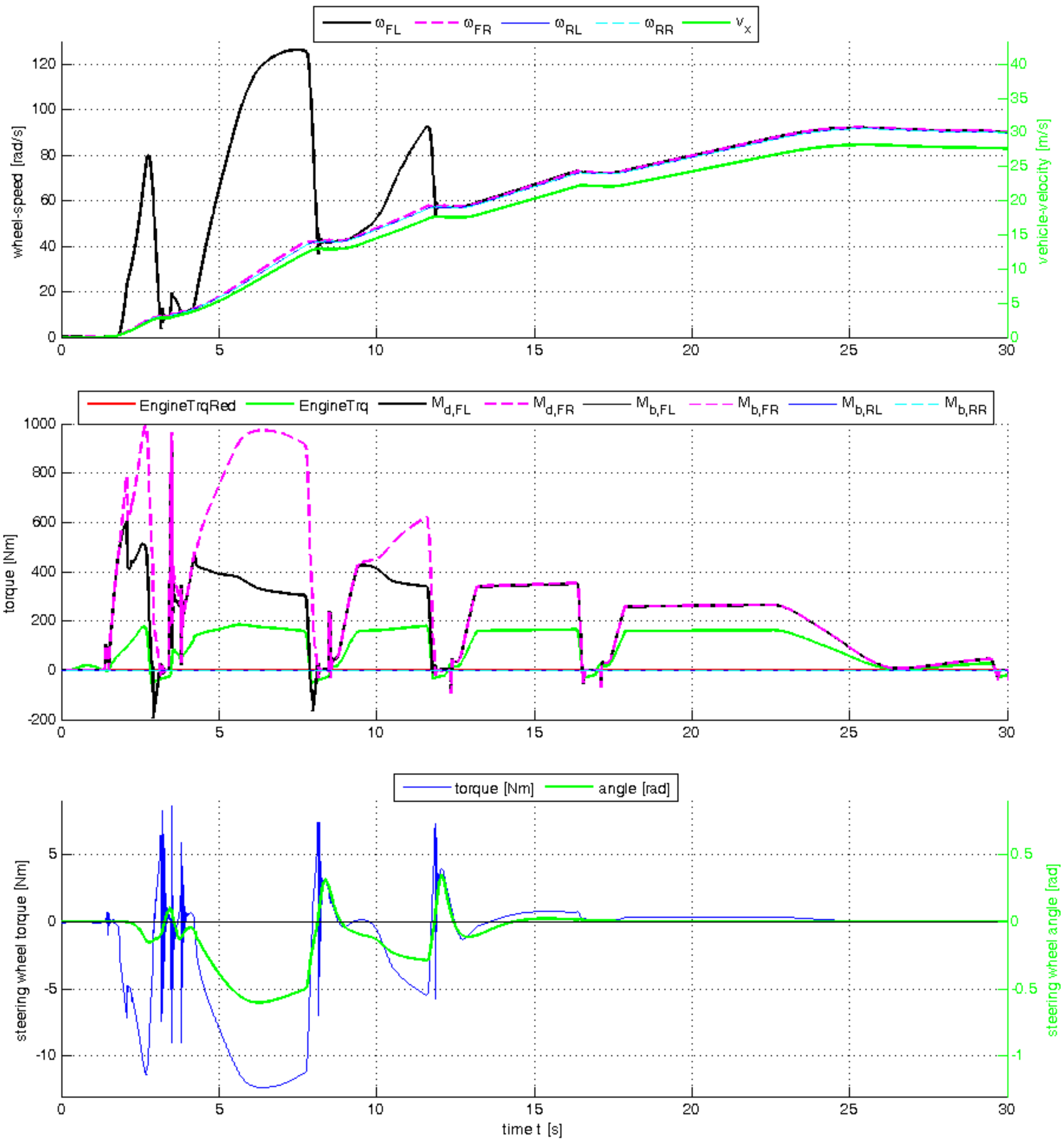


Figure 3.20.: TCS off, full-acceleration: straight, μ-split:
Full acceleration from 0km/h to almost 30km/h with gear changes on a μ-split surface (left side low friction with μ=0.3, right side high friction with μ=0.8).
Wheel speed ω_{FL} rises above ten-times the current reference-speed.

3.2.4. TCS & ABS combined test: straight, μ-split

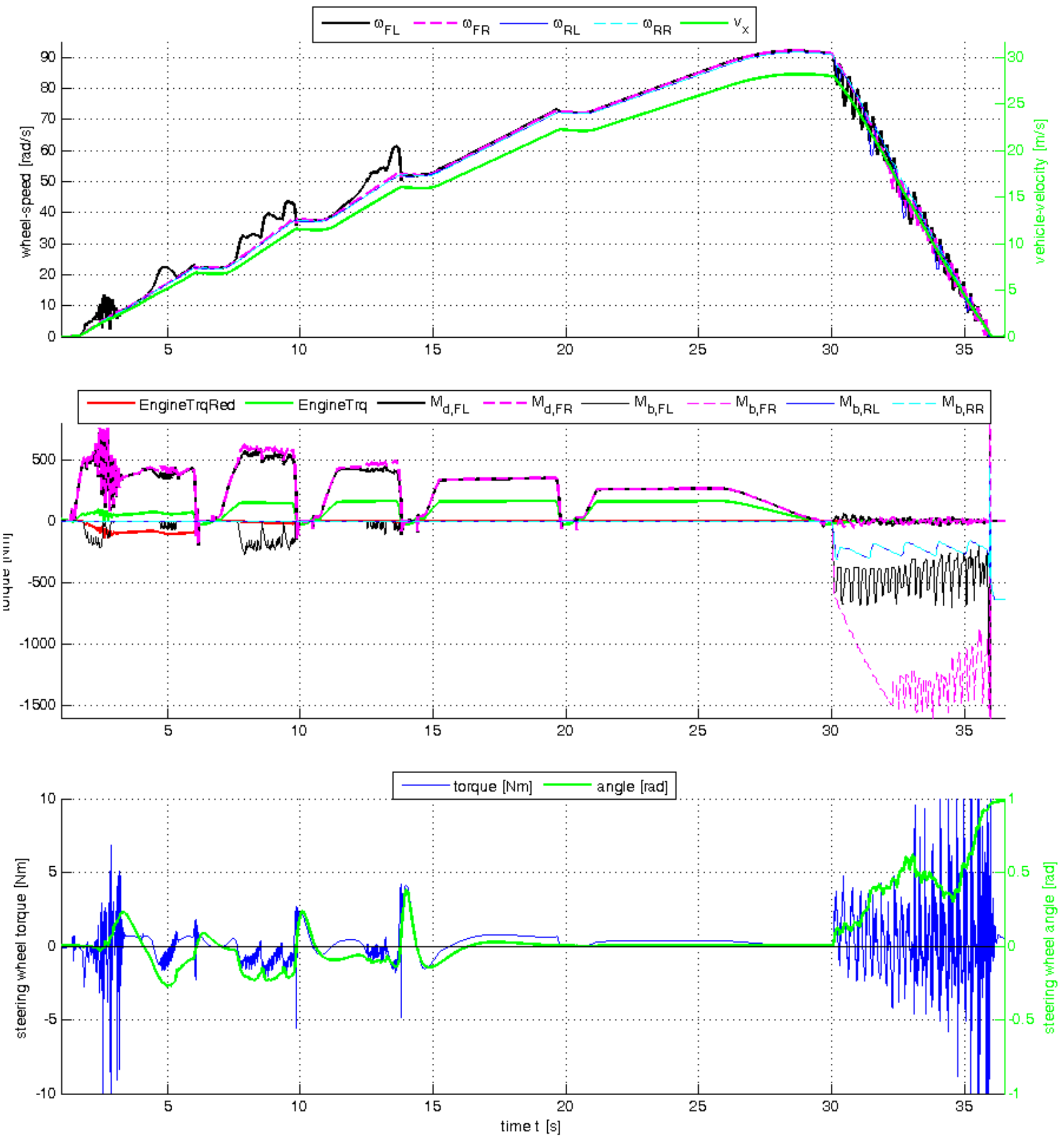


Figure 3.21.: TCS & ABS combined test: straight, μ-split:
Full acceleration from 0km/h to almost 30km/h with gear changes on a μ-split surface (left side low friction with μ=0.3, right side high friction with μ=0.8).
Full braking at time $t = 30s$ until vehicle has stopped.
Compared to the acceleration, there are high vibrations and more load during braking. The car tends to turn to the high-friction side due to the yaw-moment.

3.2.4.1. TCS & ABS off: straight, μ-split

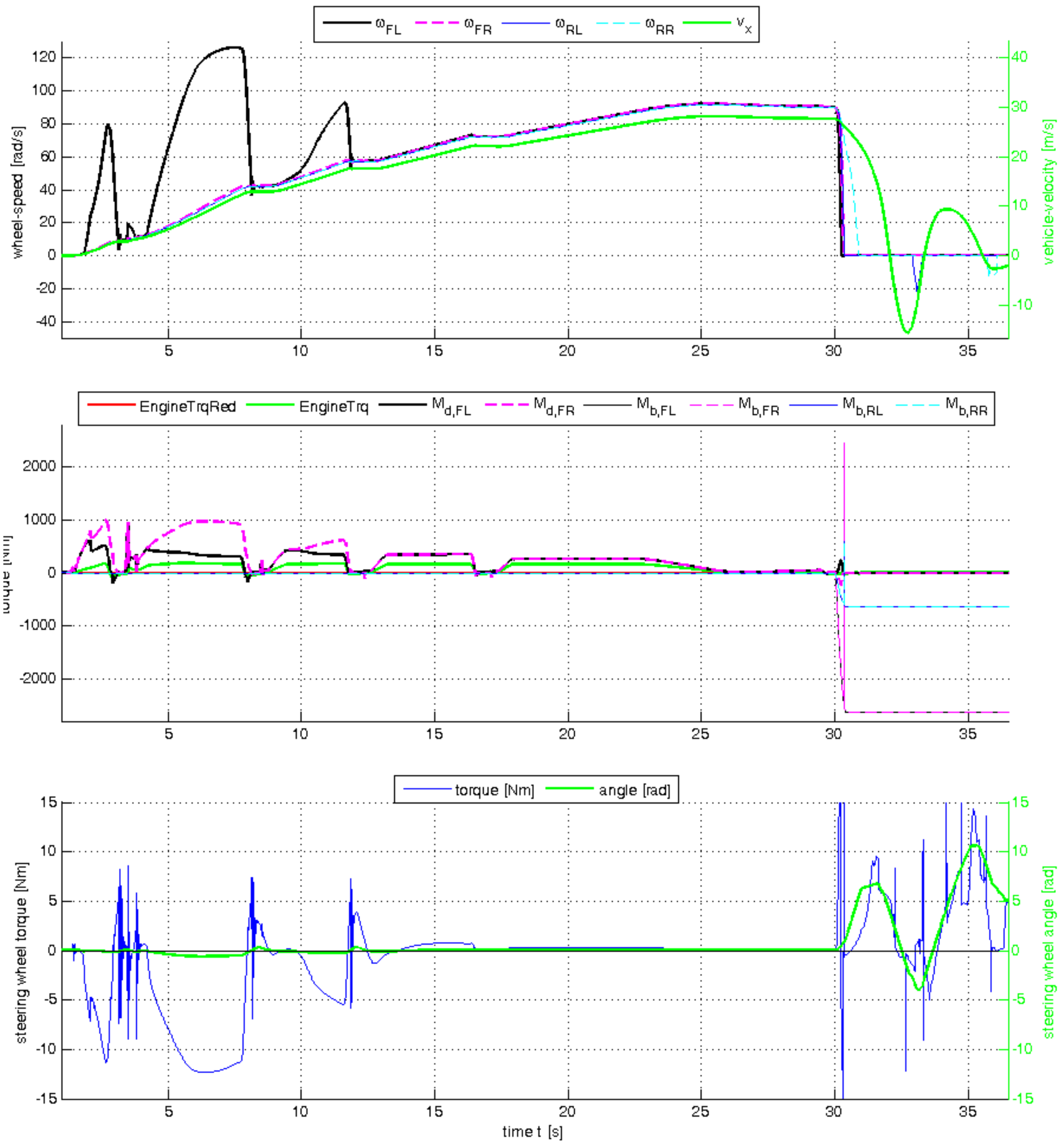


Figure 3.22.: TCS & ABS off: straight, μ-split:
Full acceleration from 0km/h to almost 30km/h with gear changes on a μ-split surface (left side low friction with μ=0.3, right side high friction with μ=0.8).
Full braking at time $t = 30s$ until vehicle has stopped.
All wheels lock up immediately and, due to the increasing yaw-moment, the car starts spinning, uncontrollable for the driver.

3.2.5. Parameters

Following values of controller-parameters (see table 3.2) have been estimated and optimized during implementation until results were satisfying enough. Further optimizations or parameter-variations are possible but not part of this thesis.

Variation of vehicle-mass showed no affect on the TCS-control-algorithms.

Table 3.2.: ASR controller: parameters

Parameter	Value	Description
`CLOCK`	0.004s	Influences the reaction-time of the controller.
`p_max`	20bar	Maximum hydraulic pressure of hydro-unit, when TCS-controller increases brake-pressure. This pressure has to be adjusted together with `CLOCK` and valve-timers.
`EngineTrqRedmax`	10Nm	The minimum current engine-torque before engine stalls. A shutdown of the engine must be avoided. This minimum engine-torque is always present.
`lambdaOn`	0.08	The slip-threshold to turn the traction- (slip-) control on.
`lambdaOff`	-1	The slip-threshold to turn the traction-control off. As value is set to -1, the controller is never turned off (by this threshold). Thus, the controller regulates the slip during the whole acceleration maneuver (until the next gear change).
`kp`	20	The basis for the proportional gain-factor of the PID-controller (traction control) [7]. This value is re-calculated according to current speed-difference.
`Ti`	0.1s	The integral-time of the PID-controller (traction control) [7].
`Tt`	sqrt(`Ti`·`TD`)	The anti-windup-time of the PID-controller (traction control) [7].
`Td`	2·`CLOCK`	The discretization-time of the PID-controller (traction control) [7].
`TD`	6·N·`Td`	The derivative time of the PID-controller (traction control) [7].

Continued on next page

Parameter	Value	Description
`N`	2	The number of time-steps to look back at the derivative part of the PID-controller [7].
`t_brake`	0.02s	Time to increase the pressure at the final drive-controller.
`t_hold`	0.03s	Time to hold the pressure at the final drive-controller.
`t_lift`	0.01s	Time to decrease the pressure at the final drive-controller.
`lambdaDiffOn`	0.1	The first (of three) thresholds to start the final drive control cycle (switches to state |`Brake_Left`| or |`Brake_Right`|).
`v_Diff_max`	0.2m/s	The second threshold to start the final drive control cycle.
`v_ASR_max`	200km/h	The third threshold (in this case boundary) to start the final drive control cycle. As this threshold exceeds maximum reachable velocities of simulations, it doesn't affect the controller. TCS is turned on during the whole simulated maneuvers.
`lambdaDiffOff`	0.05	The first (of two) thresholds to stop the final drive control cycle (switches to state |`Decrease_Pressure`|).
`v_Diff_min`	0.1	The second threshold to stop the final drive control cycle.
		End of table

3.2.6. Discussion

Results show working TCS-controllers. During implementation parameters have been set up and optimized to gain reasonable outputs.

Especially, the initial start of vehicle's acceleration causes suboptimal slip-values or spinning wheels during first gear. Two issues have to be addressed there. On one side there is an excess of torque provided through the first gear. On the other side the vehicle is still standing or slowly starting to accelerate.
Now, the maximum transferable torque is limited by the tire-road-combination and resulting friction-curves. Additionally, as velocities are near zero, the related thresholds to start or stop TCS-controllers are very low at the beginning. As result, initially, the TCS-controller causes oscillating slip-courses until velocities and thresholds are accordingly increased.
After the initial on-off-behavior, the engine-torque is reduced to an optimal value and slowly increased according to current slip. If spinning is imminent, engine-torque is reduced again. This can be seen in referring figures from 3.14 to 3.19.

Slip values are controlled around their optimum. If wheels tend to spin, TCS immediately reacts in time, keeping slip always well below 0.5.

In contrast, non-TCS-controlled accelerations cause continuous spinning of driven wheels (see figure 3.16). This lasts until drive-torques (current engine-torque transferred through transmissions and final drive onto driven wheels) are below the current maximum transferable torques. Besides wasting engine-torque this constant spinning destroys tires over time. Moreover, the car becomes unsteerable by the driver as available forces are completely spent onto longitudinal direction (see figure 2.5).

In case of μ-split conditions (e.g. one side icy or wet lane grooves) the final drive transfers the main torque automatically onto the low-friction sided wheel. Thus, the low-friction sided wheel starts spinning, wasting most of the provided engine-torque, while the high-friction sided wheel runs with minimum load. Additionally, to former disadvantages the vehicle builds up a yaw-moment, which needs proper driver-attendance. In case of non-TCS-controlled acceleration the driver has to constantly steer against the yaw-moment.

TCS controllers counter the spinning of low-friction sided driven wheel by applying brake-torque. As result, the brake torque is transferred through the final drive onto the high-friction sided driven wheel. Again, the slip is kept near its optimum. Furthermore, the driver's load is decreased and the car stays steerable throughout the acceleration.

4. Summary

At the starting point of this thesis, there was a newly redesigned GUI for the institute's vehicle simulation system, MOVES, [14]. MOVES contains an already highly complex double-track vehicle-model with many possible variations in terms of parametrization and block-choice selections, [18]. A previous master thesis already implemented a fully functional real powertrain, containing selectable engine, clutch, transmission, and final drive, [4]. There were also some ideal TCS-controllers implemented, [4].

On this basis, new ABS- and TCS-algorithms should be implemented for MOVES. The goal is to stay close to real-world, using only measurable vehicle-variables as input for the controllers. This allows more realistic simulations of different maneuvers and scenarios. Moreover, the controllers might be implemented on real hardware and tested on track. This would allow to match the simulation with the real-world and, therefore, create further, more realistic simulations.

As previous implementations already created templates for ABS- and TCS-controllers, the decision was to stay backwards-compatible and reuse those templates. First, the existing brake-system had to be adapted, as both, ABS and TCS, need to control the brake-torque and, therefore, the brake-pressure.
The previous brakes were directly converting the input of the driver (position of the brake-pedal) into pressure and torque. This transformation happened without delay and discontinuous changes at the brake-pedal were immediately resulting in discontinuous changes of brake-torque at the wheels.
Now for the new system this transformation of driver's input into brake-torques should be close to real-world. There are no discontinuities and increasing or decreasing brake-pressure follows a natural curve with delays before the pressure is transferred into torques.

Thus, a hydro-unit has been implemented inside the brake-system. The main purpose of this hydro-unit is to allow ABS and TCS to control the hydraulic pressure. As in real-world, electrohydraulic valves (and an electric driven pump) control the hydraulic flow and, therefore, the hydraulic pressure output. To gain a realistic behavior, a transfer-function smooths any discontinuities and simulates delays or hysteresis of real-world brake-systems, [11]. Thus, a phenomenological approach has been implemented. The parameters of the transfer-function were adapted to match real-world measured test-data, [13].
For ABS the hydro-unit has three possible valve-positions. First (default) position of the hydro-unit just puts the input through to the output. Therefore, the driver still regulates the pressure. The second position holds the current pressure and the third decreases the

pressure. Additionally, for TCS-controllers, a fourth position allows the hydro-unit to build-up output-pressure without driver's interaction, meaning without input-pressure.

Now an ABS-algorithm had to be implemented. Basis was the common ABS-cycle by BOSCH, [17], [3]. Stateflow was the best implementation-choice, as single cycle-phases are directly map-able to states of a state-machine, [1].
The initial implementation was done quite fast, but proper parametrization was missing. There is a lack of specific values for almost all parameters in given literature. A first approach to gain certain parameter-values from above mentioned real-world test-data failed, as test-data did not match the common ABS-cycle. Therefore, the parametrization was estimated and optimized until results were satisfying enough.

Unfortunately, simulations in MOVES resulted in uncontrollable vertical vibrations, which couldn't be fixed (it is not part of this thesis). Those disturbances slowed the progress of optimizing the ABS-controller and its parametrization. As fixing the cause for the disturbances was unsuccessful and already took a lot of time, the decision was made to switch to another simulation-environment.
Thus, Carmaker became the new implementation platform, trading places with MOVES. First, MOVES' models had to be integrated in Carmaker's environment. Initial tests showed even worse results than in MOVES. This was due to the used unrealistic tire-model, which allowed even higher forces for higher slip-values. Switching to a realistic tire-model brought the change. Now, ABS-controlled braking finally decreased the brake-distance significantly compared to locking wheels.

Further optimizations and parameter-variations improved the ABS-model to current state. The resulting ABS works for a wide range of simulation-scenarios.

The TCS-controllers didn't cause any of above experienced problems. Implementation was immediately done in Carmaker, using again Stateflow and MOVES' templates for later easier re-integration.
The traction-controller has been implemented as PID-controller. Parametrization was approximately estimated and further optimized. Proper values were found fast.
As final drive-controller a simple on-off-controller using thresholds fulfills its purpose. Timing-values, to increase, hold, or decrease brake-pressures, were optimized. Satisfying results came in fast.
Variation of vehicle-mass didn't affect the outcome at all. The resulting TCS works for all tested simulation-scenarios.

4.1. Conclusion

The initial goal - to create ABS- and TCS-algorithms within MOVES - has been achieved in a different way as intended. Vertical oscillations at braking forced the ABS-controller to be finalized using Carmaker instead of MOVES. Subsequently, the implementation of TCS was also done in Carmaker.

Finally, results were reintegrated in MOVES and verified, showing similar behavior and, therefore, confirming the goal. Thus, Carmaker and MOVES switched places in terms of implementation-platform and verification-system.

Results heavily depend on used configuration of driver-, vehicle-, and tire-model. The driver works as an outer controller, who might disguise errors of the inner controllers. The vehicle-model defines the behavior of vertical forces, when braking. The tires build the contact to the road, simulating slip depending on friction-conditions, current forces and velocities.

Resulting controllers work for a wide range of different conditions and scenarios. There is still space for improvements or (automatized) optimizations.

4.2. Outlooks

Both, ABS and TCS, are properly working now. Still, especially for the ABS-controller with its numerous parameters, there is potential for optimization. As manually varying parameters and simulating several tests for different conditions takes too long an automated approach might make sense.

In contrast to TCS, ABS doesn't control the slip directly, as the reference-speed has to be estimated. For this thesis the reference-speed is calculated from two cross-over wheels and extrapolated when ABS-cycle starts. Results show huge deviations from simulated vehicle-velocity. Slip-thresholds and, therefore, reaction or response-time of ABS-controller, depend on this estimated reference-speed. Due to wide deviations for different scenarios, those thresholds have to be set up to gain a working ABS at all circumstances.
Providing a better estimation of vehicle-velocity might dramatically improve ABS' performance. The approaches of two master-theses might be implemented in future optimizations, [2], [16].
Moreover, today's cars' sensor-systems are increasing in number and functionality. Acceleration sensors (in longitudinal direction) would significantly help to extrapolate the reference-speed during ABS-cycles.

Even without optimization, the current system allows further test-scenarios and comparisons between the implemented systems and others (e.g. ideal controllers or uncontrolled maneuvers).

Furthermore, both controllers already made the basis for ESP-controllers, which basically use the same inputs and outputs.

Abbreviations

ABS	Anti-lock braking system (*"Antiblockiersystem"*)
TCS	Traction control system
ASR	*"Antriebsschlupfregelung"*
VDC	Vehicle Dynamics Controller
HMI	Human Machine Interface
FTG	Institute of Automotive Engineering (*"Fahrzeugtechnik Graz"*)
MOVES	MOdular VEhicle Simulation System
Carmaker	IPG CarMaker®
GUI	Graphical User Interface
KOS	Kinematic Optimization System
MuBOS	Multi Body Optimization System
MATLAB	MathWorks MATLAB®
Simulink	MATLAB Simulink®
Stateflow	MATLAB Stateflow®
GUIDE	MATLAB Graphical User Interface Development Environment

Symbols

Text-formatting

text	Common text formatting.
ABBR	ABBReviation of programs, models, or controllers.
"Deutsch"	German equivalent or translation.
Dialog	A dialog itself, its functions, or an element.
`Code`	(MATLAB) program-code, function names, variables or parameters.
`<Input>`	An input-signal or -variable.
`[Output]`	An output-signal or -variable.
`|Block|`	Simulink-blocks or Stateflow-states.

Variables, Parameters and Constants

$var_{dsc,idx}$	`var_dsc_idx`	variable with lowered description and index.
a	`a`	accelerations in [m/s^2]
C		center
F		forces in [N]
g	`g`	gravity-constant g=9.81m/s^2
J		moment of inertia in [kgm^2]
M	`M`	torques or moments in [Nm]
p		pressure in [Pa]
r		radius in [m]
t	`t`	(simulation-) time in [s]
v	`v`	velocities in [m/s]
W		wheelpoint
α		slip-angle in [rad]
γ		camber-angle in [rad]
δ		steer-angle in [rad]
λ	`slip`	slip in [1] or [%]
μ	`mu`	friction-coefficient in [1]
ω	`omega`	wheel-speed in [rad/s]
π		circle-constant π=3.14159...

b_t	`b_t`	wheel de-/acceleration in [m/s^2]
F_x		longitudinal tire force in [N]
F_y		lateral tire force in [N]
F_z		contact (vertical) tire force in [N]
J_{rot}		rotational moment of inertia in [kgm^2]
M_b, M_B	`M_b`	brake-torque in [Nm]
M_d	`M_d`	drive-torque in [Nm]
M_R		friction moment in [Nm]
p_{hydr}	`p_hydraulic`	hydraulic (brake-) pressure in [Pa]
r_0		undeformed radius in [m]
r_e, r_{eff}		effective radius in [m]
r_s		static radius in [m]
v_F		vehicle-velocity in [m/s]
v_R	`v_R`, `v_R_XY`	wheel-speed in [m/s]
v_{Ref}	`v_Ref`	reference-speed in [m/s]
$v_{Vehicle}$		vehicle-velocity in [m/s]
v_{Sx}		longitudinal slip-velocity in [m/s]
v_{Sy}		lateral slip-velocity in [m/s]
v_x		longitudinal vehicle-velocity in [m/s]
λ_A		traction-slip in [1] or [%]
λ_B		brake-slip in [1] or [%]

Indeces

x, X	longitudinal direction
y, Y	lateral direction
z, Z	vertical direction
fl, FL (vl, VL)	front left (*"vorne links"*)
fr, FR (vr, VR)	front right (*"vorne rechts"*)
rl, RL (hl, HL)	rear left (*"hinten links"*)
rr, RR (hr, HR)	rear right (*"hinten rechts"*)
F	*"Fahrzeug"*
R	*"Rad"*

List of Figures

List of Tables

Bibliography

[1] A. Angermann, M. Beuschel, M. Rau, and U. Wohlfahrth. *Matlab-Simulink-Stateflow.* Oldenburg Verlag, 2003.

[2] D. Brenner. Methoden zur Online-Schätzung fahrdynamischer Kenngrößen, 2011.

[3] Robert Bosch GmbH. *Kraftfahrtechnisches Taschenbuch.* Vieweg+Teubner, 2007.

[4] M. Haas. Modellbildung und Simulation eines PKW Antriebsstrangs, 2013.

[5] W. Hirschberg and H. Waser. Fahrzeugdynamik. Skriptum, 2010. Institut für Fahrzeugtechnik, TU Graz.

[6] W. Hirschberg and H. M. Waser. Kraftfahrzeugtechnik. Lecture notes, Institute of Automotive Engineering (Graz University of Technology), 9. Oktober 2012.

[7] M. Hofbaur. Automatisierung mechatronischer Systeme. Skriptum, 2012. Institut für Regelungstechnik, TU Graz.

[8] IPG Automotive GmbH. CarMaker® Programmer's Guide, 2012. Version 4.0.3.

[9] IPG Automotive GmbH. CarMaker® User's Guide, 2012. Version 4.0.3.

[10] IPG Automotive GmbH. CarMaker®. Available at http://ipg.de/de/simulationsolutions/carmaker/, 2013. Version 4.0.3.

[11] R. Isermann. *Fahrdynamik-Regelung.* Vieweg Verlag, 2006.

[12] Paul Karoshi. Entwicklung einer Methodik zur Optimierung von Mehrkörpersystemen. Master thesis, FTG, January 2013.

[13] C. Lex, H. . Kobialka, and A. Eichberger. Wheel-individual estimation of the friction potential for split frition and changing friction conditions for the application in an automated emergency braking system. In ATZ live, editor, *Chassis.tech plus 2013*, pages 609–621, Wiesbaden, 13.-14.06.13 2013. Springer Vieweg. Munich, Germany.

[14] M. Großmann. MOVES^2 GUI-Redesign, 2013.

[15] MATLAB. *version 7.7.0.471 (R2008b).* The MathWorks Inc., Natick, Massachusetts, 2008.

[16] T. Oberascher. Modellbasierte Ansätze zur Fahrzustandserkennung, 2011.

[17] K. Reif. *Bremsen und Bremsregelsysteme.* Springer, 2010.

[18] A. E. Rojas Rojas, H. Niederkofler, X. Bas Ferrer, and J. Duernberger. Modular modeling of vehicles with innovative powertrain systems. In *Proceedings of 13th EAEC European Automotive Congress*, June 13-16 2011. Valencia.

[19] The MathWorks, Inc. Creating Graphical User Interfaces, October 2008. Revised for MATLAB 7.7 (Release 2008b).

[20] A. T. van Zanten. Das Antiblockiersystem ABS, 2002. ESP Seminar.

[21] S. Waser and W. Hirschberg. TMeasy Application Manual. Manual, Institute of Automotive Engineering (Graz University of Technology), 2007.

[22] H. Winner, S. Hakuli, and G. Wolf. *Handbuch Fahrerassistenzsysteme Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort.* Vieweg+Teubner, Wiesbaden, 2009.

A. Appendix

Contents

A.1. Parameter-Files

A.1.1. Antilock Braking System - ABS

Listing A.1: ABS_Controller_Parameters

```
% ABS Controller Parameters
% holds all necessary parameters (e.g. slip thresholds) of the ABS.
%
% Ver   Date    Editor              Description
%---------------------------------------------------------------------
% 0.1   140221  Manfred Großmann    initial version

g = 9.81; % [m/s^2] gravity

%% Parameter Fields List
Structure_Position = 'Parameters.Vehicle_Dynamics_Controller';
Structure_Fields   = {'ABS'...
                      };

PreProcessing_FCN  = '';

%% Parameters definition
ABS.on      = 0;        % [-] ABS controller switch (1...on, 0...off)
ABS.p_max   = 90e5;     % [pa] max. hydraulic pressure
ABS.r       = 0.3061;   % [m] fixes tire-radius

%---------------------------------------------------------------------
% ABS Controller Parameters
%---------------------------------------------------------------------
ABS.v_min = 2.5/3.6; % [m/s] minimum velocity for ABS controller
ABS.t_min = 1;       % [s] minimum time (offset)
                     %     before ABS controller starts working

% accel-/deceleration thresholds:
ABS.a_init  = 1.9*g; % [m/s^2] negative threshold of wheel-deceleration
ABS.a       = 1.5*g; % [m/s^2] negative threshold of wheel-deceleration
                     %         (just before blocking)
ABS.a_pos   = 1.5*g; % [m/s^2] lower positive threshold of wheel-decel.
                     %         (regaining stability)
ABS.A       = 5*g;   % [m/s^2] higher positive threshold of wheel-decel.
                     %         (restart breaking)

% slip thresholds:
ABS.lambda_1 = 0.08;  % [1] lower threshold of brake-slip
                      %     (indicating possible locking)
ABS.lambda_2 = 0.25;  % [1] higher threshold of brake-slip
                      %     (indicating low friction)

% timer
ABS.timer_2_3   = 0.07;  % [s] time before Anti-Lock-control starts
ABS.timer_4_41  = 0.01;  % [s] time to hold pressure during Anti—Lock—state
ABS.timer_41_4  = 0.005; % [s] time to decrease pressure during AL—state
ABS.timer_7     = 0.02;  % [s] initial time to increase pressure
ABS.delta_t_7   = 0.001; % [s] calculate time to increase pressure
ABS.timer_71_7  = 0.01;  % [s] time to hold p during puls—step—control

ABS.counter_4_max =  10; % [1] max circles during state 4
ABS.counter_7_max =  10; % [1] max circles during state 7

% ABS—Controller—clock (trigger)
ABS.CLOCK = 2;         % [1] controller—clock (times simulation step time)
% IPG Carmaker:
% ABS.CLOCK = 0.002;  % [s] controller—clock

% Transfer—Functions:
% ABS.numerator1   = [1];          % numerator coefficient (front)
% ABS.denominator1 = [0.2 1];      % denumerator coefficient (front)

ABS.numerator2   = [1];          % numerator coefficient (rear)
ABS.denominator2 = [0.2 1];      % denumerator coefficient (rear)

% v_R (low—pass—filter)
ABS.numerator3   = [1];          % numerator coefficient (v_R)
ABS.denominator3 = [1e—003 1];   % denumerator coefficient (v_R)

%————————————————————————————————————————————————————————————————————————————
```

A.1.2. Traction Control System - TCS

Listing A.2: ABS_Controller_Parameters

```
% ABS Controller Parameters
% holds all necessary parameters (e.g. slip thresholds) of the ASR.
%
% Ver   Date    Editor              Description
%——————————————————————————————————————————————————————————————————
% 0.1   140827  Manfred Großmann    initial version

g = 9.81; % [m/s^2] gravity

%% Parameter Fields List
Structure_Position = 'Parameters.Vehicle_Dynamics_Controller';
Structure_Fields   = {'ASR'...
                     };

PreProcessing_FCN  = '';

%% Parameters definition

%————————————————————————————————————————————————————————————————————————
% ASR Controller Parameters
%————————————————————————————————————————————————————————————————————————
ASR.on      = 0;     % ABS controller switch (1...on, 0...off)

% ABS—Controller—clock (trigger)
ASR.CLOCK = 2;       % [1] controller—clock (times simulation step time)
% IPG Carmaker
% ASR.CLOCK = 0.004; % [s] controller—clock

ASR.p_max    = 20e5;         % [pa] max. hydraulic pressure
ASR.EngineTrqRedMax = 10;    % [Nm] maximum reduced engine—torque

%————————————————————————————————————————————————————————————————————————
% Engine Torque Controller (traction control):
%————————————————————————————————————————————————————————————————————————
% PID controller:
ASR.kp      = 20;            % [1] proportional constant
ASR.Td      = 2*ASR.CLOCK;   % Abtastperiode (sinnvollerweise CLOCK)
ASR.Ti      = 0.1;           % I—Anteil: Nachstellzeit
ASR.N       = 2;             % D—Anteil
ASR.TD      = 6*ASR.N*ASR.Td; %Vorhaltezeit (min 5*N*Td)
ASR.Tt      = sqrt(ASR.Ti*ASR.TD); % Stellgrößenbeschränkung

% thresholds:
ASR.lambdaOn  = 0.08;        % [1] slip to turn ASR control on
ASR.lambdaOff = —1;          % [1] slip threshold to turn it off (again)

%————————————————————————————————————————————————————————————————————————
% Brake Torque Controller (differential control)
```

```
%--------------------------------------------------------------------------
% valve-timings:
ASR.t_brake = 0.02;
ASR.t_hold  = 0.03;
ASR.t_lift  = 0.01;

% thresholds:
ASR.lambdaDiffOn  = 0.1;
ASR.lambdaDiffOff = 0.05;
ASR.v_ASR_max     = 200/3.6;
ASR.v_Diff_max    = 0.2;
ASR.v_Diff_min    = 0.1;

%--------------------------------------------------------------------------
```

A.2. Simulink-Models

A.2.1. Overview

Figure A.1 contains following blocks:
`|Maneuver_Inputs|` provides the simulation-maneuver for the driver.
`|Driver|` steers the car according to the given maneuver and reacts to the vehicle motion.
`|Vehicle_Modell|` simulates the vehicle-motion according to the given driver-commands.

`|Environment|` animates the vehicle in a virtual reality.
`|Log_Values|` filters data for logging.
`|Signal_Choice|` filters simulation output data.

A.2.2. Vehicle Model

Figure A.2 consists of following blocks, input-, and output-parameters:

(Input-) Parameters:
`|Vehicle_Parameters|` contains the provided parameters from the input-structure.
`<Maneuver_Parameters>` sets up the driving-maneuver.
`<Driver_Commands>` holds the driver-input (e.g. acceleration- or braking-pedal positions).

Center:
`|VDC|` The Vehicle Dynamics Controller holds the ABS- and TCS-models.
`|Human Machine Interface|` Driver-commands (e.g. braking-pedal position) are transferred to vehicle-units (e.g. brake-pressure or -torque).

Vehicle:
`|Body|` This block creates the resulting body-motion (of the complete vehicle).
`|Body_Attached_Susp_Devices|` Suspension devices (attached to the vehicle-body).
`|Wheels|` The wheels translate the provided torques and environment-parameters (e.g. road-friction) to suspension-units.
`|Battery|` This block gains only importance when simulation electrical- or hybrid drive-systems.

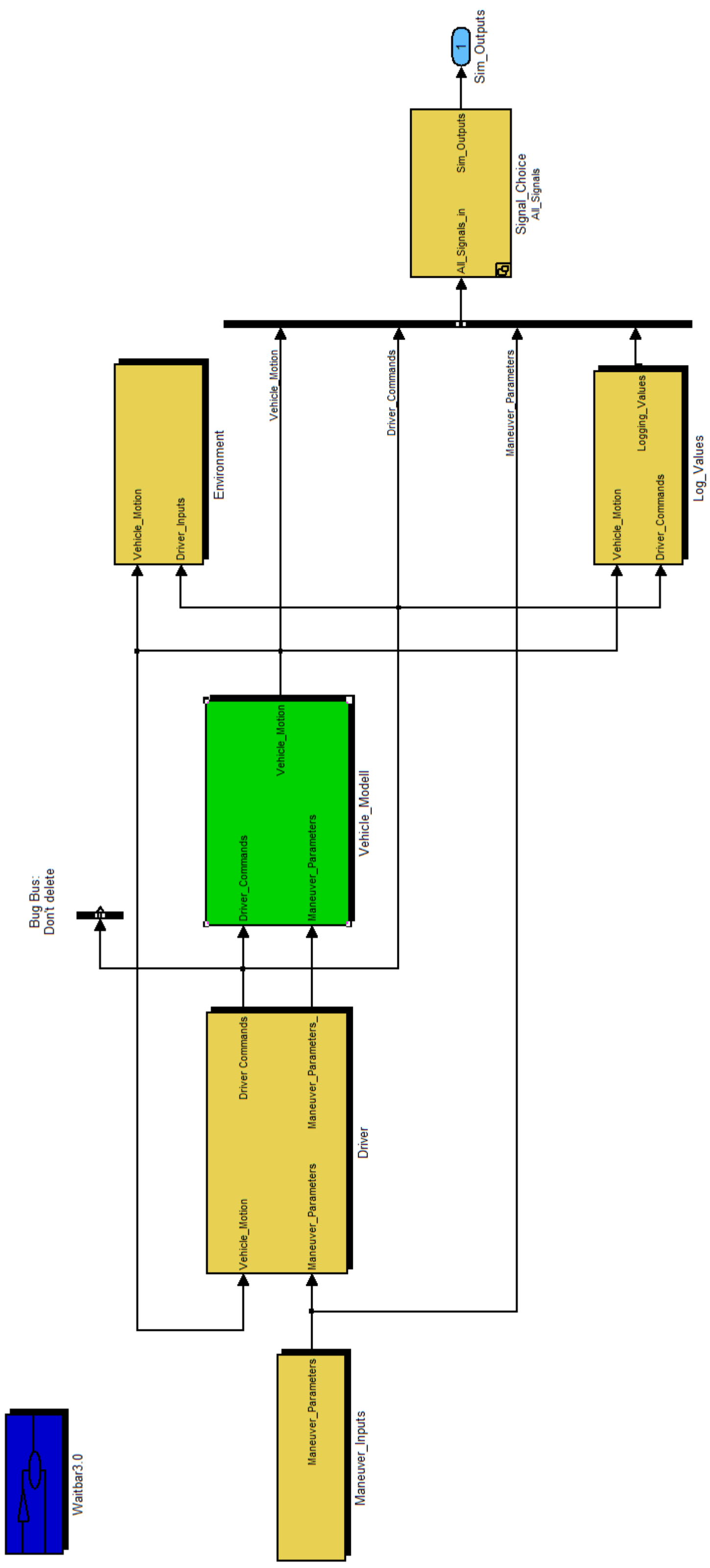


Figure A.1.: MOVES' Vehicle Model implemented Simulink

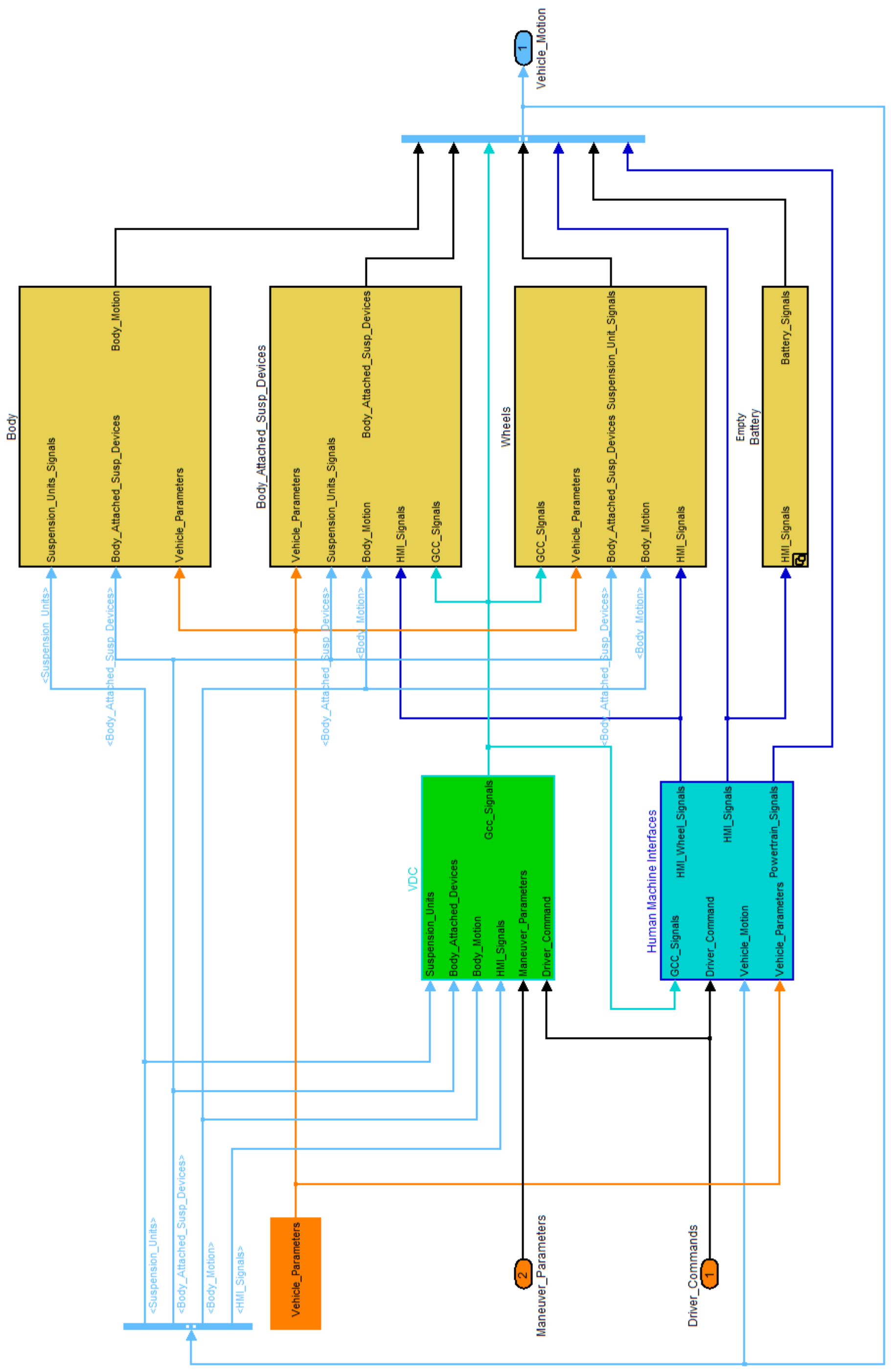


Figure A.2.: Vehicle Model implemented in MATLAB Simulink

A.2.2.1. HMI

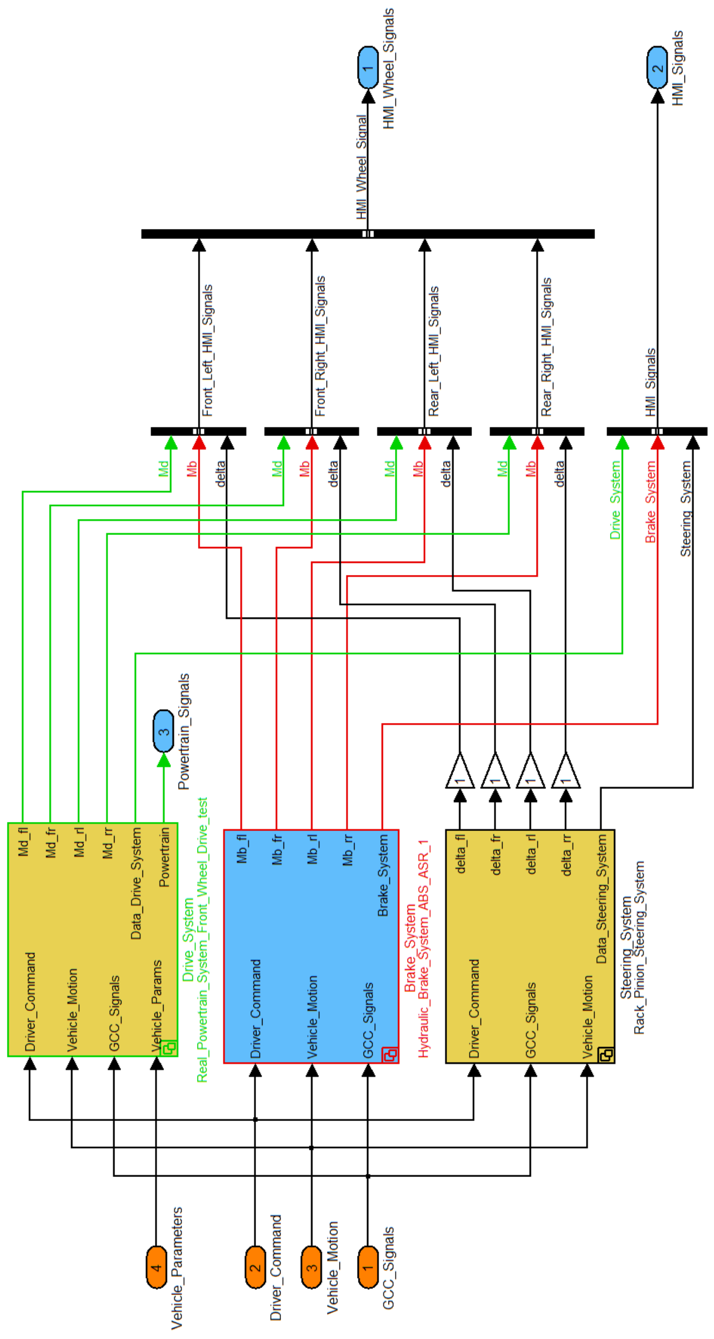


Figure A.3.: Human Machine Interface (described in chapter 2)

A.2.2.2. VDC

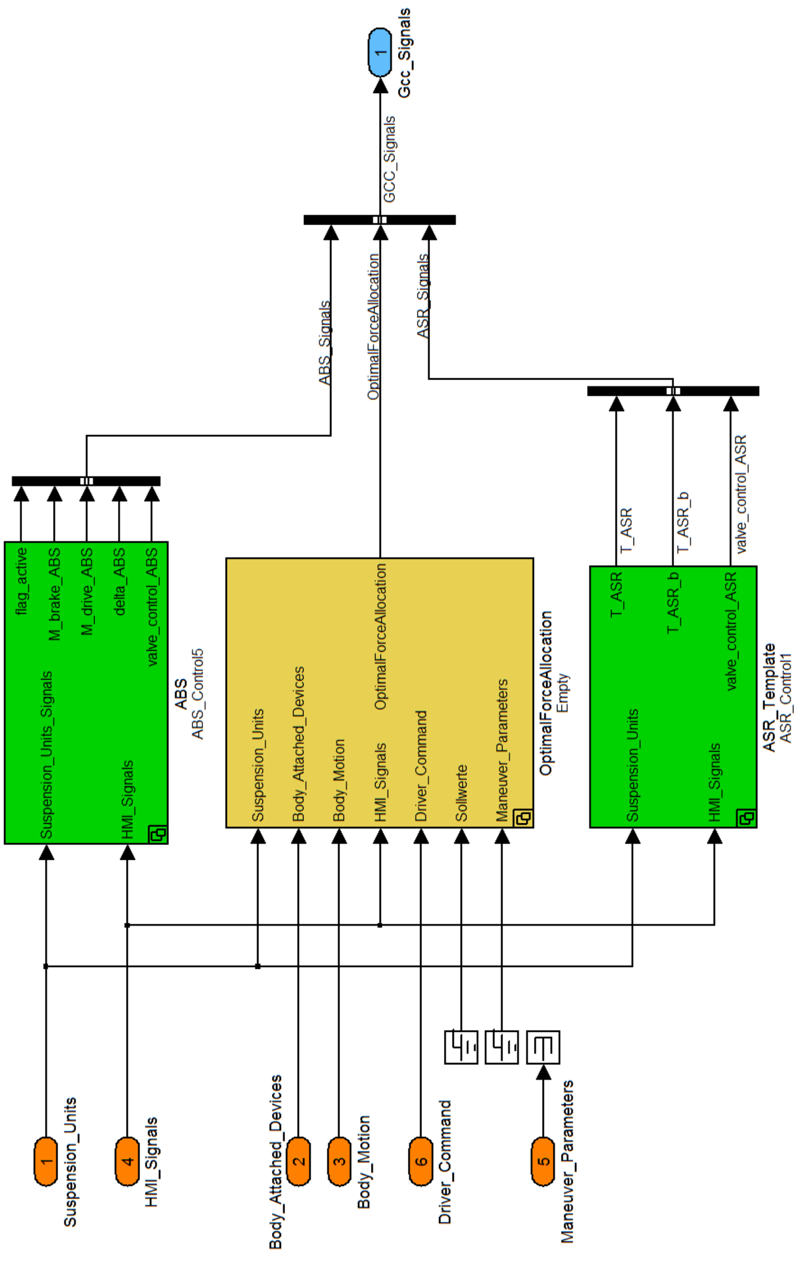


Figure A.4.: Vehicle Dynamics Controller (described in chapter 2)

A.2.2.3. ABS

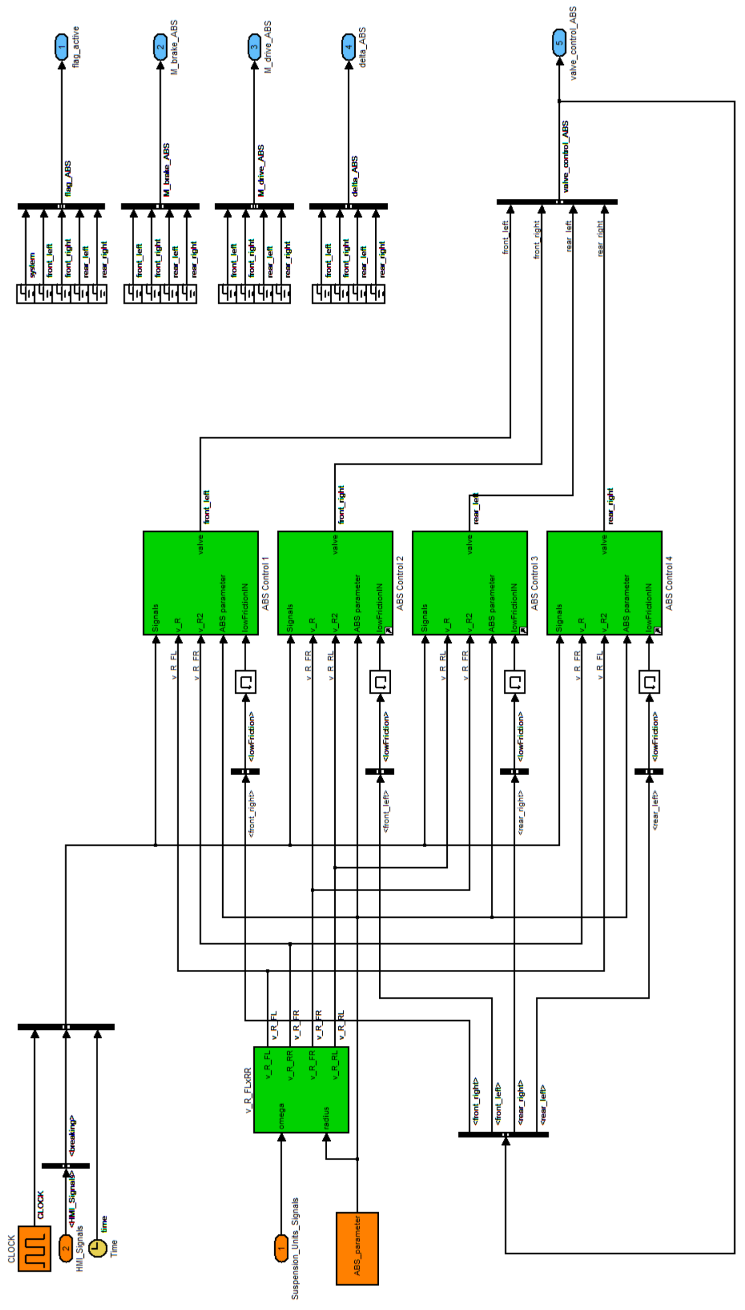


Figure A.5.: ABS: complete overview (described in chapter 2)

A.2.2.4. Brake-System

Figures A.6 and A.7 show the content of the previously existing brake-hardware-block. It has been extended by the signal `braking`, to start or stop the ABS-controller.

The description of the brake-system can be found in chapter 2, section 2.3.3.

Hardware

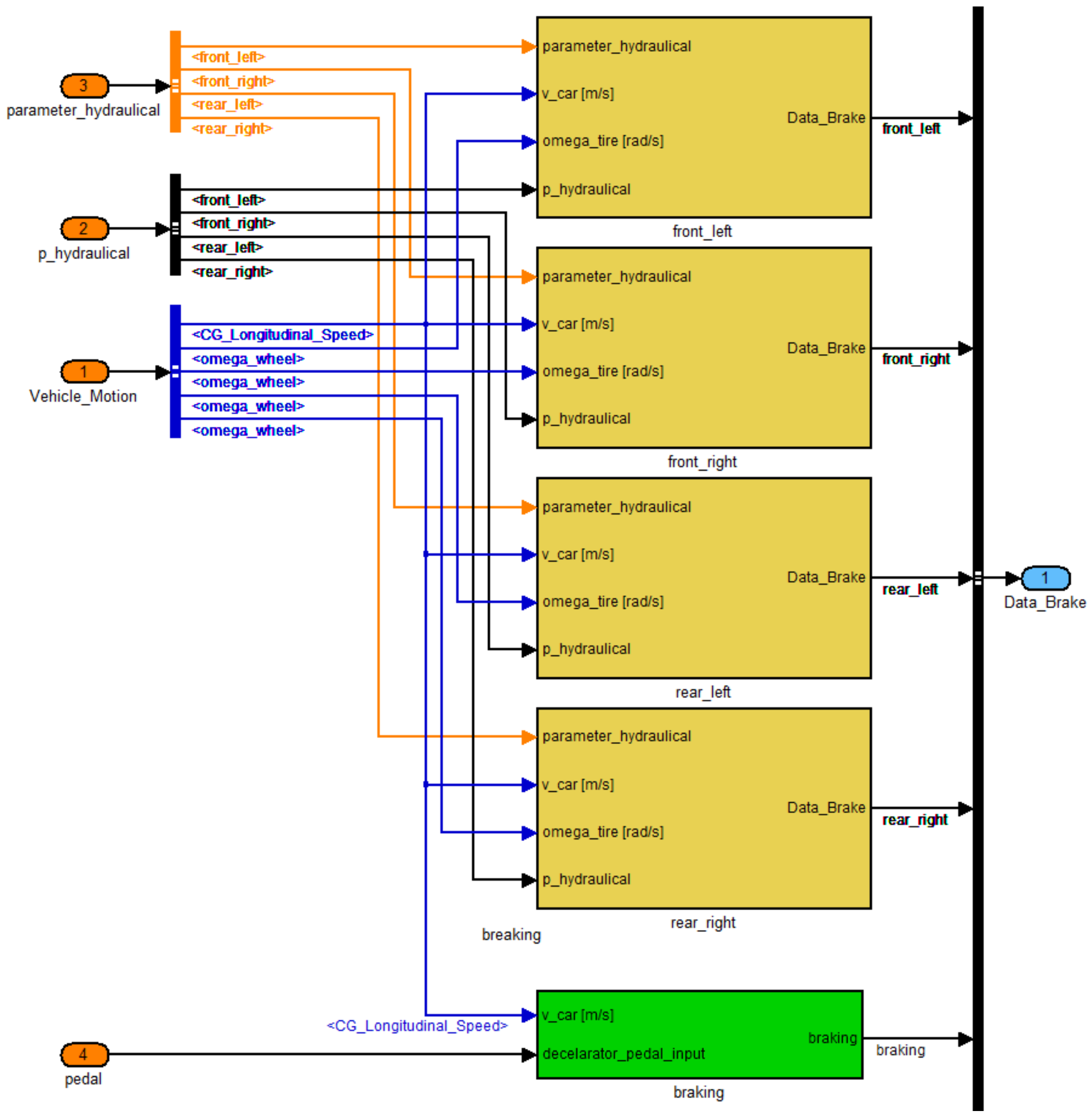


Figure A.6.: Brake-System: Hardware

Front left wheel

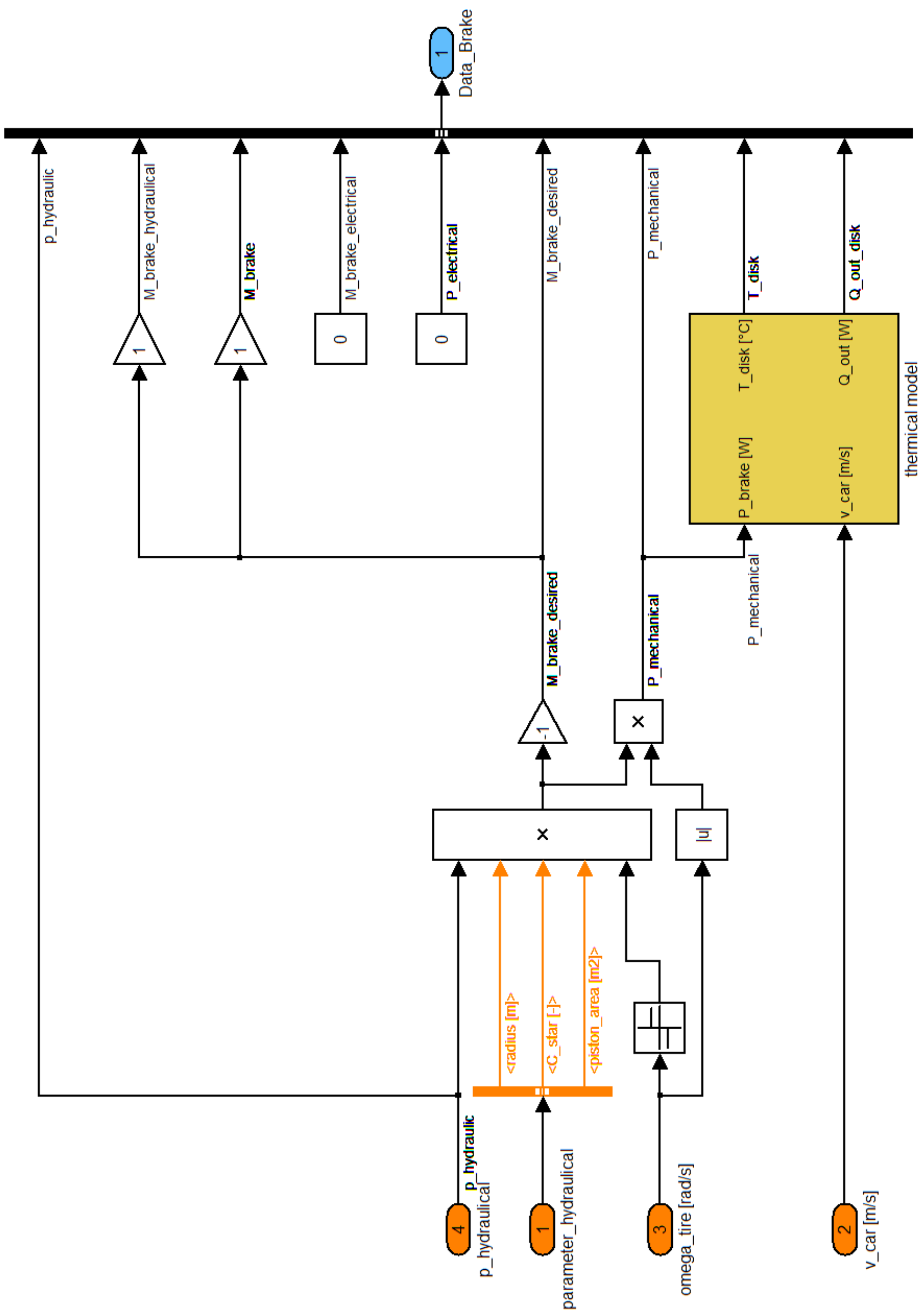


Figure A.7.: Brake-System - Hardware: front left wheel

A.2.2.5. Drive-System

Ideal

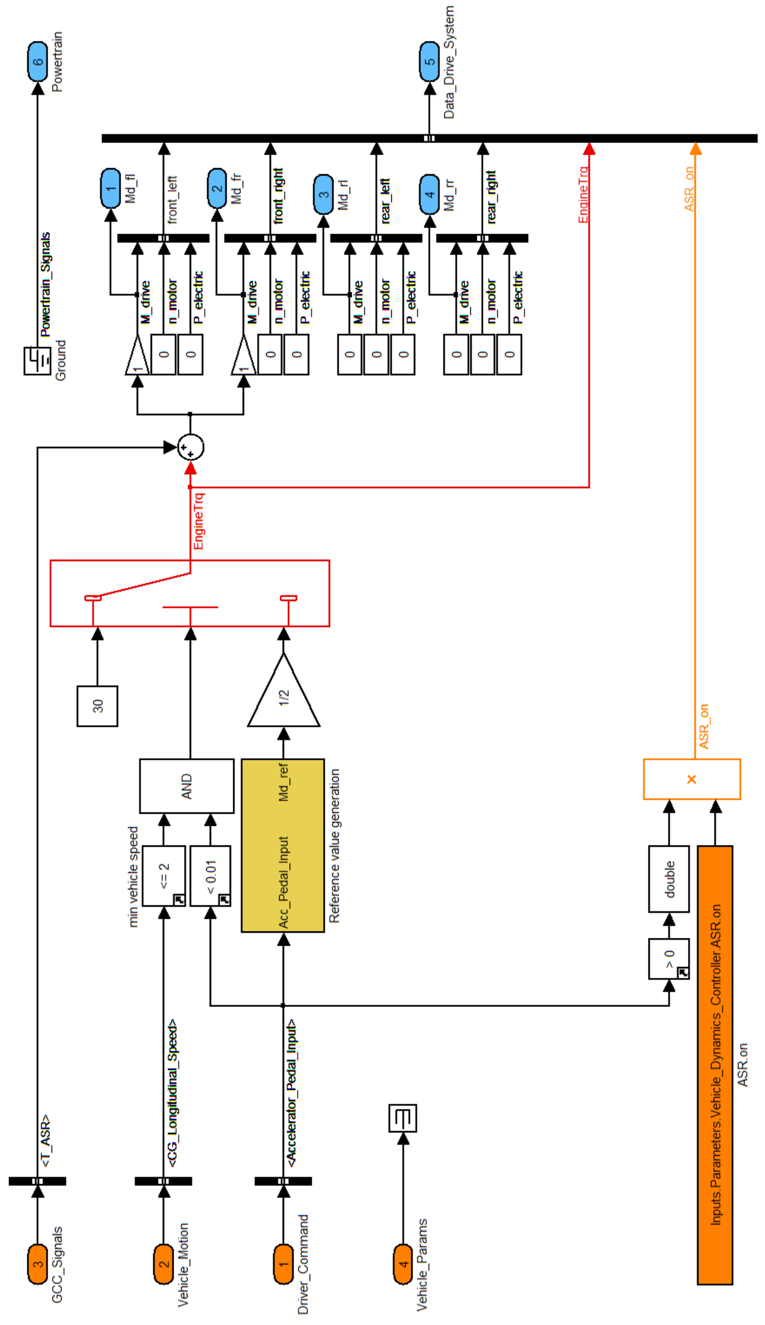


Figure A.8.: Drive-system: ideal

Real

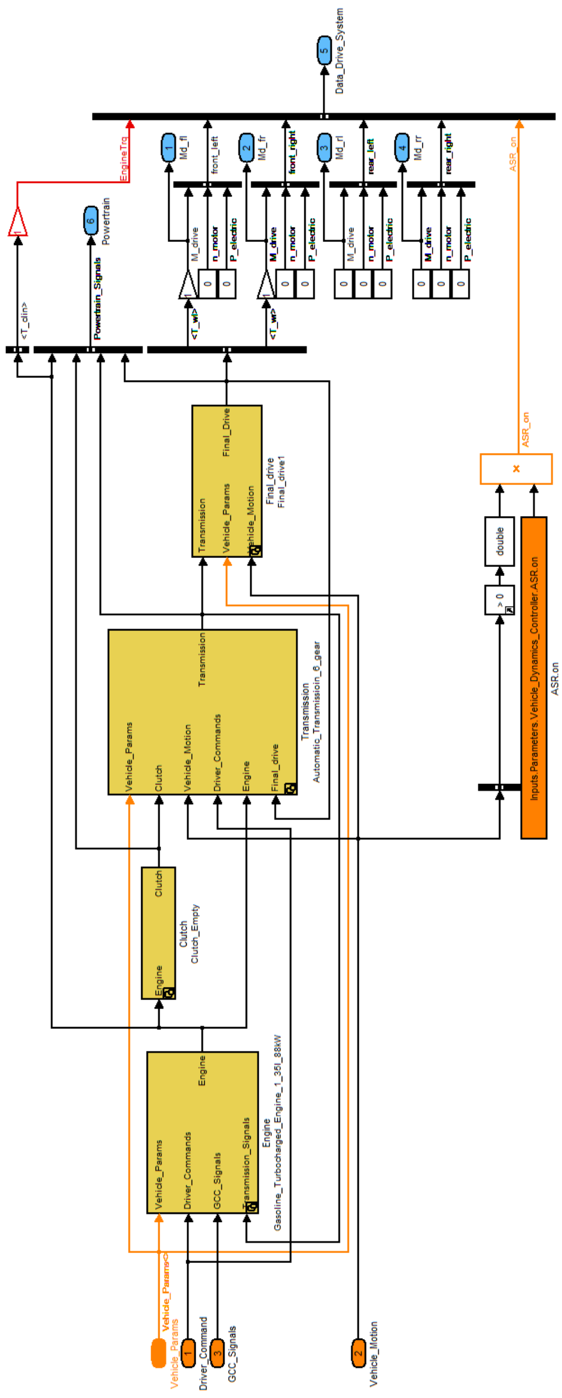


Figure A.9.: Drive-system: real

A.3. Stateflow-Models

A.3.1. ABS

Figure A.10 is the initial layout based upon the typical cycles described by BOSCH [3], [17]. They are explained in chapter 2, section 2.3.1. In figure A.11 there is the final implementation in Stateflow.

Basically, there are seven states:

initial braking: pressure is increased according to drivers input (deceleration pedal).

hold pressure 2: holds the pressure.

decrease pressure 3: ABS control-loop starts running. First, decrease the pressure.

Antilock: further decrease and hold of pressure (steps).

NoNameForIt: holds and increases the pressure, again.

PulsStepControl: increases the pressure, controlling the puls-step-times.

restart braking: uses a different approach, when ABS is already running...

A detailed description and closer look of the single states can be found in section 2.3.1.

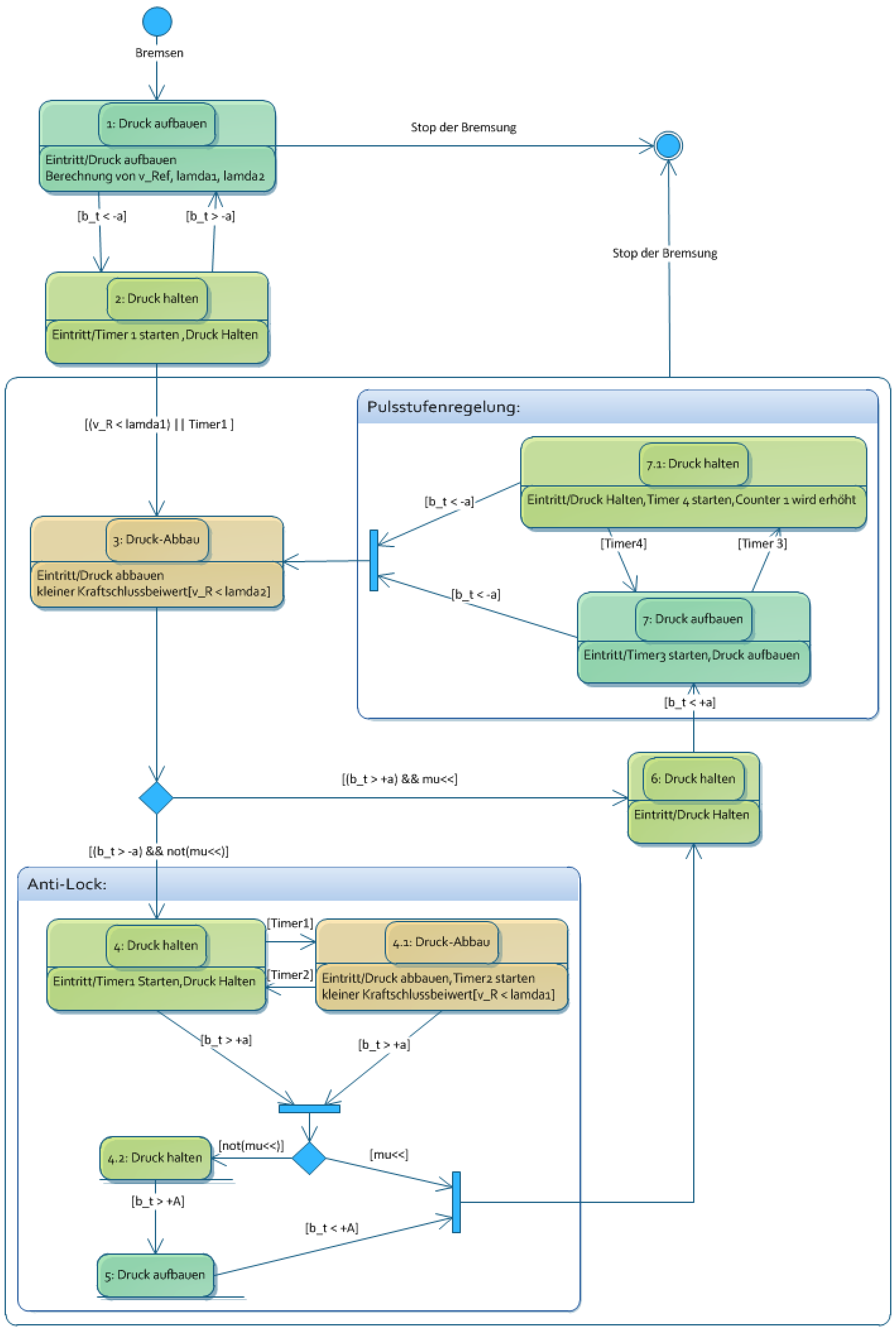


Figure A.10.: ABS: initial layout

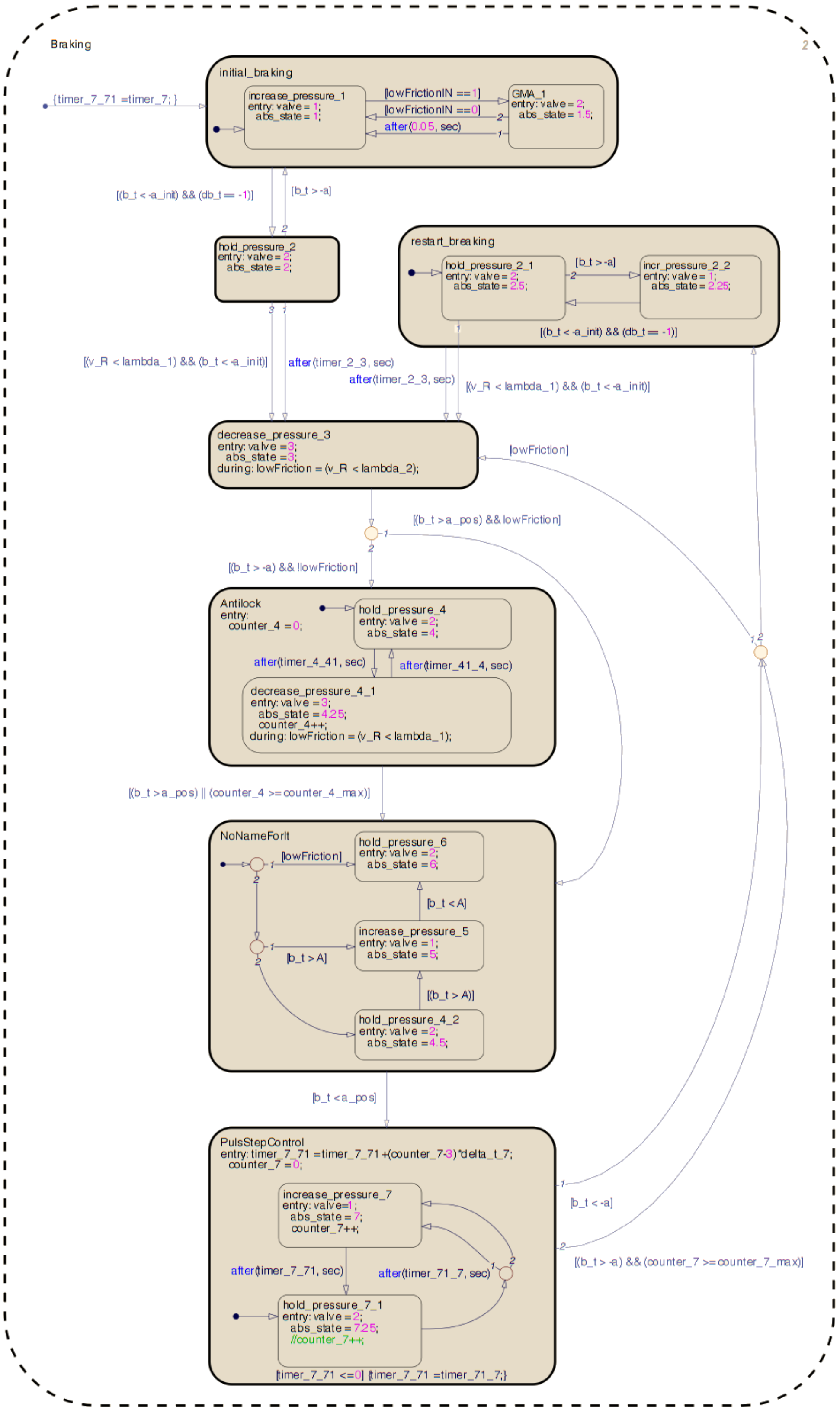


Figure A.11.: ABS: overview