Rucai Wang, BSc

KPI-model-based calibration of Model Predictive ACC function using a co-simulation platform

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mechanical Engineering

submitted to

Graz University of Technology

Supervisor

Assoc. Prof. Dipl.-Ing. Dr.techn., Arno Eichberger

Institute of Automotive Engineering
Member of [FSI]

Dr., Hans-Michael Koegeler

AVL List GmbH

Graz, February 2020

Acknowledgement

Firstly, I would like to thank my supervisor Dr. Hans-Michael Koegeler in AVL for offering me the thesis as well as his supervision during these eight months. Special thanks go as well to my university supervisor at TU Graz, Prof. Dr. Arno Eichberger, for his supervision and helpful discussions throughout this thesis.

I would also want to present my gratitude to AVL colleagues, Niklas Wikstroem, who offered the MPACC function, which laid a great basement for the start of my master thesis, and Sajin Gopi, who helped me with the set-up of the co-simulation and calibration, in addition, he also taught me a lot about the work and life.

Thanks as well to other colleagues from other departments, Klaus, Simun, Jurica, and so on, who gave me many supports with the operation of the software.

The greatest thanks is for my family, specifically my parents, only with the support of them, my study in Austria could come to reality. Additionally I would also like to thank my friends, who helped me a lot with my daily life.

Finally, thanks to AVL List GmbH, TU Graz and Tongji university for providing me the opportunity to participate in the double degree program, which is challenging but fantastic.

AFFIDAVIT

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 indicated 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.

....	....
Date	Signature

Abstract

In the USA, there were over 1,6 million people killed in motor vehicle traffic crashes over the past 40 years. 94% of these accidents were caused by human errors. In order to contribute to solve this problem, the automated driving (AD) systems are developed. However, realistic calibration and validation of highly automated driving systems requires huge efforts. With the help of the co-simulation software AVL Model.CONNECT and the calibration software AVL CAMEO, a KPI-model-based virtual calibration method for automated driving functions is performed with the example of an ACC function in this master thesis. The co-simulation is set up with IPG CarMaker, MATLAB/Simulink and AVL Model.CONNECT, then they are connected to AVL CAMEO for the calibration set-up. In order to reduce the efforts of ADAS/AD function calibration, the DoE method is firstly applied to reduce the number of simulation runs. Then the behavior model of KPIs (Key Performance Indicator) are built and the quality of the models are evaluated with the statistic methods. Finally, KPI behavior models are verified with further simulation runs and the ACC function calibration is done. The KPI-model-based virtual calibration method is demonstrated with the ACC function. The ego vehicle with the calibrated ACC function has better fuel efficiency (9% reduction of fuel consumption), better comfort performance (13,5% reduction of jerk) and safety performance (longer headway time and headway distance to the preceding vehicle) than the ACC function with base setting.

Kurzfassung

Allein in den USA sind in den letzten 40 Jahren mehr als 1,6 Millionen Menschen bei Verkehrsunfällen ums Leben gekommen. 94% der Unfällen waren auf menschliches Versagen zurückzuführen. Um dieses Problem zu lösen, werden automatische Fahrfunktionen entwickelt. Die realistische Kalibrierung und Validierung der hochautomatisierten Fahrfunktionen erfordern jedoch einen enormen Aufwand. Mit Hilfe der Co-Simulationssoftware AVL Model.CONNECT und der Kalibrierungssoftware AVL CAMEO wird in dieser Masterarbeit die KPI-Model-basierte virtuelle Kalibrierungsmethode für die automatische Fahrfunktionen am Beispiel einer ACC Funktion ausgeführt. Die Co-Simulation wird mit IPG CarMaker, MATLAB/Simulaink und AVL Model.CONNECT erstellt. Anschließend werden sie mit AVL CAMEO verbunden, um eine virtuelle Kalibrierung zu erstellen. Um den Aufwand für die ADAS-Funktionskalibrierung zu reduzieren, wird zunächst die DoE Methode angewendet, um die Anzahl der Simulationen zu reduzieren. Anschließend wird das Verhaltensmodell der KPIs erstellt und mit der statistischen Methode die Qualität der Modelle bewertet. Schließlich werden die Verhaltensmodelle mit weiteren Simulationen verifiziert und die ACC Funktion wird kalibriert. Die auf KPI-Model-basierende virtuelle Kalibrierungsmethode wird mit der ACC Funktion demonstriert. Das Ego-Fahrzeug mit der kalibrierten ACC Funktion hat bessere Kraftstoffeffizienz (um 9% Reduzierung des Kraftstoffverbrauchs), besseren Komfort (um 13,5% Reduzierung des Ruckes) und bessere Sicherheitsleistung (längere Vorlaufzeit und Abstand zum vorausfahrenden Fahrzeug) als die ACC Funktion vor der Kalibrierung.

Contents

Abbreviations

ACC	Adaptive Cruise Control
AD	Automated Driving
ADAS	Advanced Driver Assistant System
AEB	Autonomous Emergency Braking
CoG	Center of Gravity
COR DoE	Customized Output Range DoE
DIN	Deutsches Institut für Normung (The German Standardisation Institute)
DoE	Design of Experiment
Fr1	Frame One
GUI	Graphic User Interface
ICOS	Independent Co-Simulation
INN	Iterative Neural Network
ISO	International Organization for Standardization
KPI	Key Performance Indicator
LKA	Lane Keeping Assistant
MATICOS	MATLAB ICOS
MPACC	Model Predictive Adaptive Cruise Control
MPC	Model Predictive Control
NEDC	New European Driving Cycle
NRMSE	Normalized Root Mean Square Error
OEM	Original Equipment Manufacturer
OFAT	One Factor at A Time
PID	Proportional Integral Derivative
PRESS	Predicted Residual Error Sum of Squares
RMSE	Root Mean Square Error
RNN	Recurrent neural network
RSI	Relative Significance Indicator
SAE	Society of Automotive Engineers
SSE	Sum of Squares of Error
SSR	Sum of Squares of Regression
SST	Sum of Squares of Total
SI	The International System of Units
TJA	Traffic Jam Assistant
VMT	Vehicle miles traveled
WLTP	Worldwide harmonized Light vehicles Test Procedure

Symbols

y	measurements
$\bar{y}$	average value of measurements
$\hat{y}$	predicted value of measurements
$y^{\pm}$	confidence interval of y
n	number of measurements
m	number of independent model regression coefficients
e	residual / error
e^{**}	studentized residual
$e^{\pm}$	confidence interval of the residual
R^2	coefficient of determination
R^2_{pred}	predicted coefficient of determination
R^2_{adj}	adjusted coefficient of determination
h	leverage of measurement
σ	standard deviation
$\hat{\sigma}$	estimated standard deviation
df	degree of freedom
$\alpha_{0,95}$	confidence level 95%
tinv()	inverse of the student's t cumulative distribution function
u	control signal
r	reference
ω	weight of cost function
p	predictive horizon in steps of MPACC controller
v	velocity of ego vehicle
a	acceleration of ego vehicle
j	jerk of ego vehicle
J	cost function
c	the weight of cost
γ_{h}	the cost of headway time soft
γ^{Tr}	the cost of travel time
s	standard deviation of the mesurements
p	position
T_r	rise time of step response
T_s	settling time of step response
M_p	overshoot of step response
y_{ss}	steady-state value of step response

1 Introduction

1.1 Motivation

In the USA alone, there were over 1,6 Million people killed in motor vehicle traffic crashes over the past 40 years. Only in 2018, more than 36 500 people were killed and over 2 million injured in motor vehicle traffic crashes [1]. Figure 1.1 shows the fatalities and fatality rate per 100 million VMT (Vehicle miles traveled) in motor vehicle traffic crashes from 1975 to 2018. Because of the development of vehicle safety technologies, the fatality rate per 100 million VMT is reduced from 3,35 to 1,13 person. But the total number of fatalities every year does not shows a obvious reduction. Actually the total number of vehicles keeps increasing.

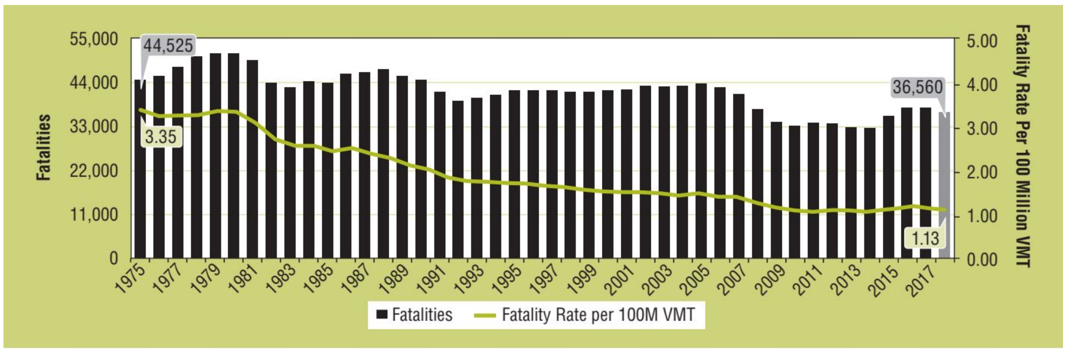


Figure 1.1: Fatalities and fatality rate per 100 million VMT in USA, 1975-2018, source: [1]

For these crashes, driver errors have been identified as the most common reasons [2]. Figure 1.2 shows the main reasons for the motor vehicle traffic crashes. There are over 2 millions motor vehicle traffic accidents collected, only 2% of the accidents are attributed to the vehicle failure, 2% to the environments and 2% remain unclear. Most of the accidents are caused by the human driver errors, up to 94% [2]. Because of limitations in perception of the traffic information, duration for decision making and other failures (e.g. distraction, impairment), the accidents happen. Figure 1.3 shows these different types of human errors for the motor vehicle traffic crashes.

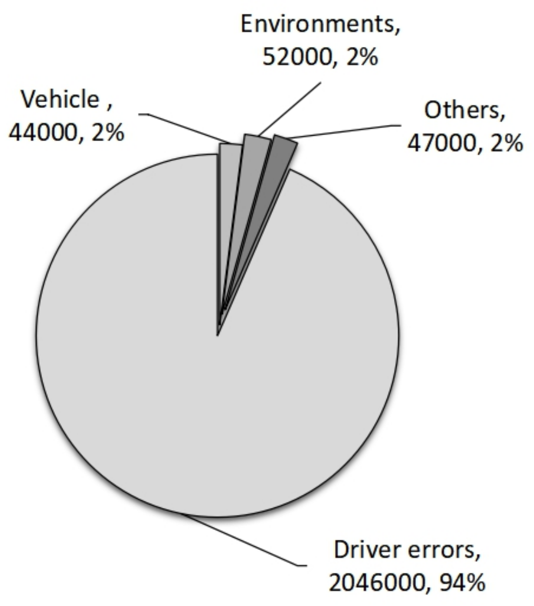


Figure 1.2: Main reasons for the motor vehicle traffic crashes, source: [2]

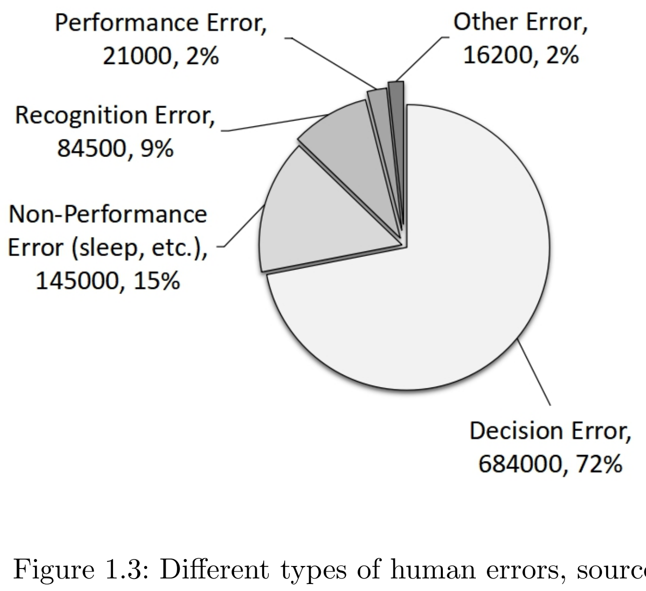


Figure 1.3: Different types of human errors, source: [2]

1.2 Advanced driver assistant system and autonomous driving

In order to avoid the motor vehicle traffic accidents and to save life and money, the degree of driving automation is increasing and the vehicle control is step by step taken over by mechatronic systems. The vision of the future is to develop a complete autonomous vehicles, which can completely avoid the accident [3] [4].

Today, production vehicles have already a long list of driver assistant systems, which partially assist in vehicle control, or even take over the control of the vehicle under some special driving conditions. For example, the Adaptive Cruise Control (ACC) system, which takes over the longitudinal control of the vehicle. It can control the vehicle to keep a certain, safe distance to the preceding vehicle [5] [6] [7]. Besides the Adaptive Cruise Control system there are also many other ADAS (Advanced Driver Assistant System) systems, such as Autonomous Emergency Braking (AEB) system, which can apply emergency braking in critical conditions without the intervention of driver [6] [7]. The Lane Keeping Assistant (LKA) system can automatically detect the lane markings and keep the vehicle in the lane, in order to avoid the accident [5] [8].

In order to classify driving automation levels of the vehicles, six levels from 0 to 5 have been defined by the SAE (Society of Automotive Engineers). These technical definition describes, on the one hand, which tasks are taken by the automation system, and on the other hand, which tasks are performed by the human drivers [9] [10]. Figure 1.4 Shows the definition of SAE Levels of driving automation and is explained in the following.

Level 0 - ”No Automation”

In level 0, there are no automated driving functions. The driver guides the vehicle in both longitudinal (acceleration or braking) and lateral (steering) direction. No intervention, but only warning from the system. The driver is fully responsible for the driving [10].

Level 1 - ”Driver Assistance”

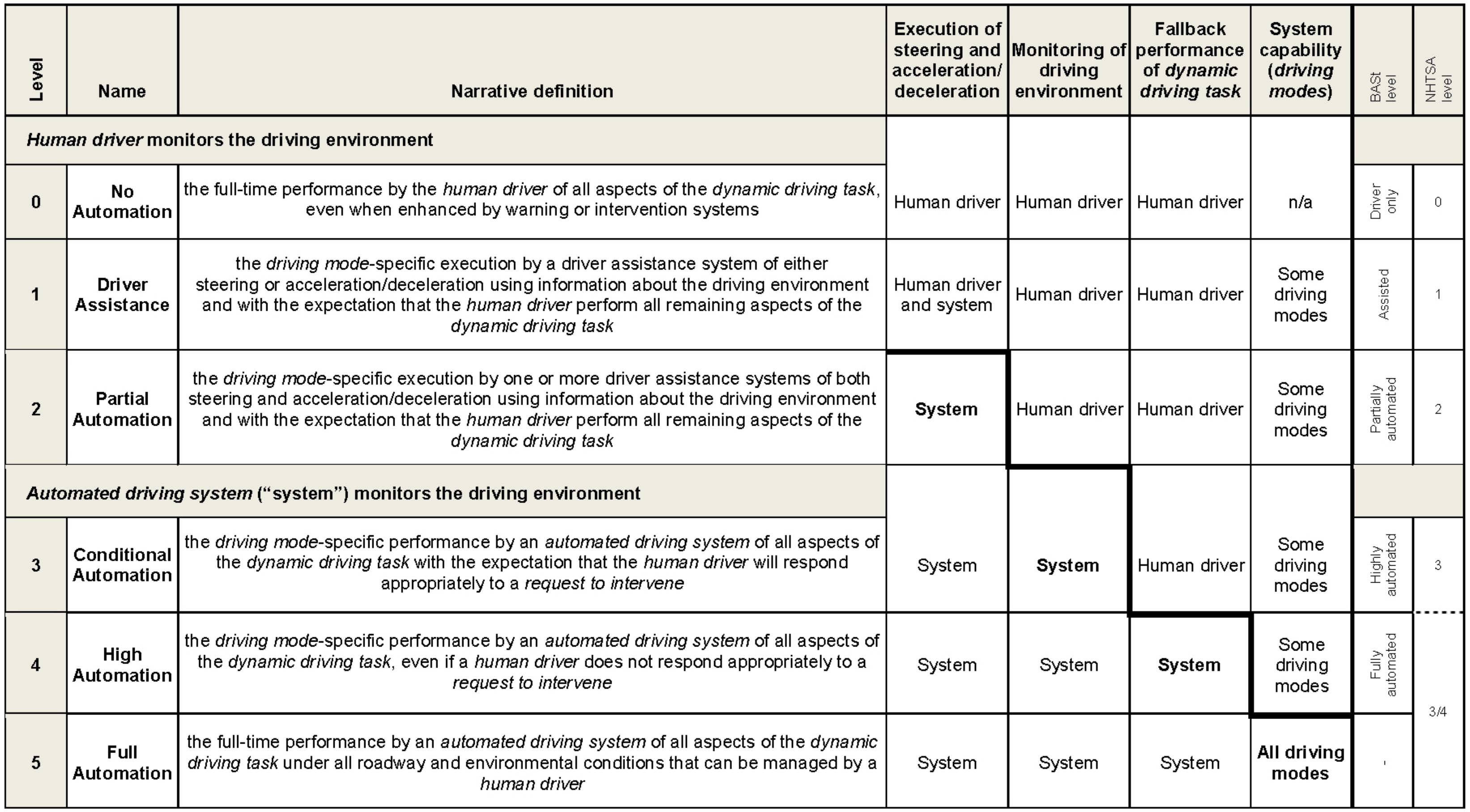
In level 1, system can support the driver by either the longitudinal or the lateral guidance of the vehicle for a certain period (e.g. ACC), while the driver can carry out the other activity. The driver still needs to observe the traffic and is fully responsible for the driving. Driver responsibility is maintained by ”hands-on” detection [10].

Level 2 - ”Partial Automation”

From level 2, onwards the vehicle is in automated control. In this level the system can take over both the longitudinal and the lateral guidance in a particular application, such as Traffic Jam Assistant (TJA) system. The driver continuously monitors the vehicle and the traffic while driving. He must be able to take control of the vehicle immediately at any time and is fully responsible for the driving. Still ”hands-on” detection is needed. The level 2 assistant system is now applied in vehicles [10] [11].

Level 3 - ”Conditional Automation”

In level 3, the automated driving system fully take over the longitudinal and lateral



Level	Name	Narrative definition	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of *dynamic driving task*	System capability (*driving modes*)	BASt level	NHTSA level
Human driver **monitors the driving environment**								
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a	Driver only	0
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes	Assisted	1
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes	Partially automated	2
Automated driving system **("system") monitors the driving environment**								
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task* with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes	Highly automated	3
4	**High Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes	Fully automated	3/4
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**	-	

Figure 1.4: SAE levels of driving automation, source: [9]

guidance of the vehicle at certain time and/or in specified driving conditions. The vehicle automatically adapts the speed and distance to the preceding vehicle, carries out the lane change automatically and activates the indicator (e.g. highway pilot). The driver can turn to other things during this time and therefore does not need to fully monitor the system and traffics. During this period, the system takes over responsibility for the vehicle. However, he must be able to take over the driving task with a certain interval after being warned by the system. So the driver is partly responsible for the driving, "hands-on" is not needed [10] [11].

Level 4 - "High Automation"

From level 4, the system can take over the complete driving task in specific applications, which are defined by the road type, the speed range and the environmental conditions. The driver doesn't need to monitor the system and traffics or to take over the steering tasks. The driver no longer has to be available compared to level 3. Therefore in the fully automated driving mode the driver is no long responsible for vehicle management [10] [11].

Level 5 - "Full Automation"

The final stage of development is driverless driving, level 5. The vehicle can carry out the entire driving task completely on all types of roads, in all speed ranges and under all environmental conditions. The vehicles are equipped without a steering wheel, accelerator pedal or brake pedal and there is no more driver in the vehicle [10] [11].

According to the current state of the art, systems of levels 1 to 2 are already ready for series production or already on the market. The latest Audi A8 is the only one equipped with level 3 autonomous driving [12]. And for the highly automated systems of levels 4 and 5 are technically already very well developed. But because of the legal framework the implementation still fails, except the application of Waymo's self-driving taxi service, "Waymo One" since 2018 in Phoenix [13] [14].

1.3 Vehicle development process - V-Model

The vehicle development process is based on the V-Model, which describes a "top-down" strategy in the design phase and a "bottom-up" strategy in the validation phase based on requirements for the overall system. In the design phase, requirements for the respective subsystems are derived based on the requirements for the overall system. And in the further step, in the validation phase in the "right branch" of the V-Model, the specifications are verified or the defined requirements are validated [15] [16]. The structural process based on the V-Model is shown in figure 1.5.

However, because of new requirements from customer and fiercely competition from the competitors, the vehicle is becoming more and more complex, but the development time is required to be shortened. Figure 1.6 shows the increasing of vehicle complexity and decreasing of the development time over time. In order to succeed in the competition,

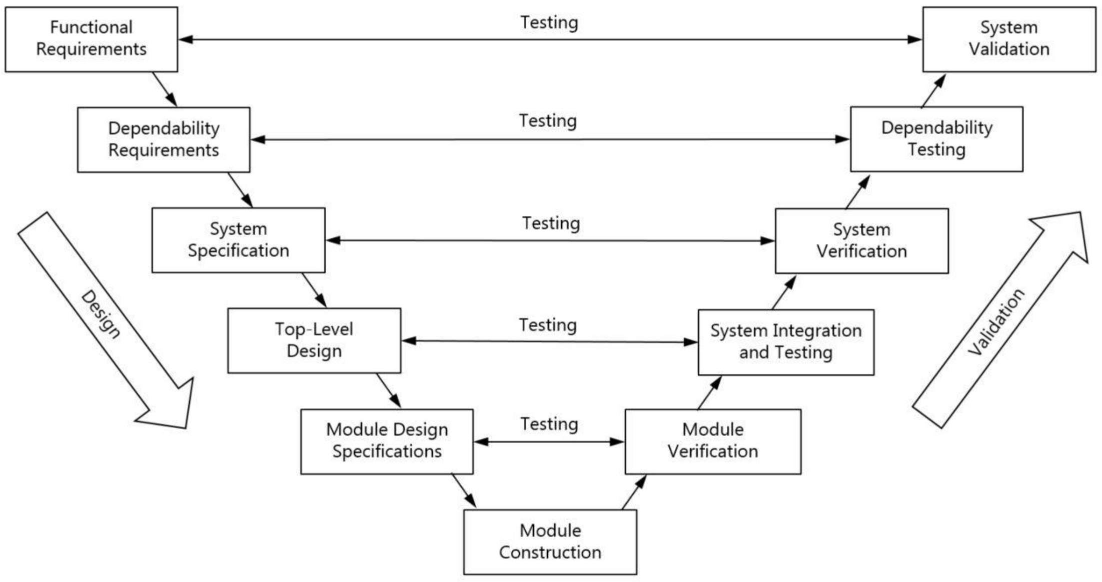


Figure 1.5: Vehicle development process based on the V-Model, source: [15]

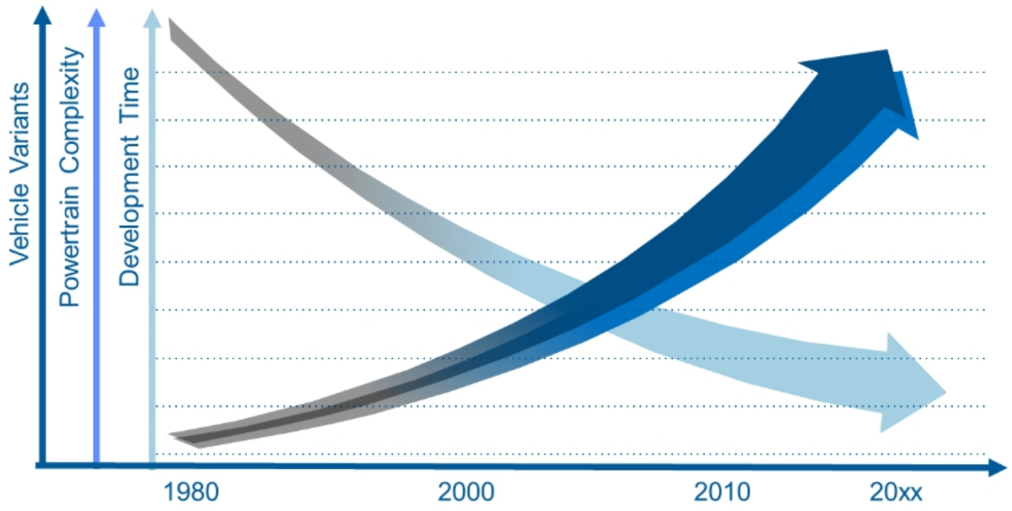


Figure 1.6: Increasing of vehicle complexity and decreasing of the development time, source: [17]

the OEMs (Original Equipment Manufacturer) as well as suppliers have to find a way, which can help them reduce the development time and costs, but remain of even increase the product quality and functional safety.

In addition, for testing of autonomous driving, the testing effort is much more than traditional vehicles. Figure 1.7 shows the autonomous driving testing distance in California in 2017 and 2018. For the autonomous driving testing, the companies Waymo and GM/Cruise contribute the most to the testing distance. And the total testing distance increased about four times form 500 000 miles in 2018 to 2 000 000 miles in 2018, which are far more efforts than the testing of traditional vehicles. But the autonomous driving vehicle is still far away from the series production, which means there more efforts are needed for the testing. This is the motivation of this master thesis to perform as well as demonstrate an KPI-model-based calibration method and contributing to real testing with virtual testing.

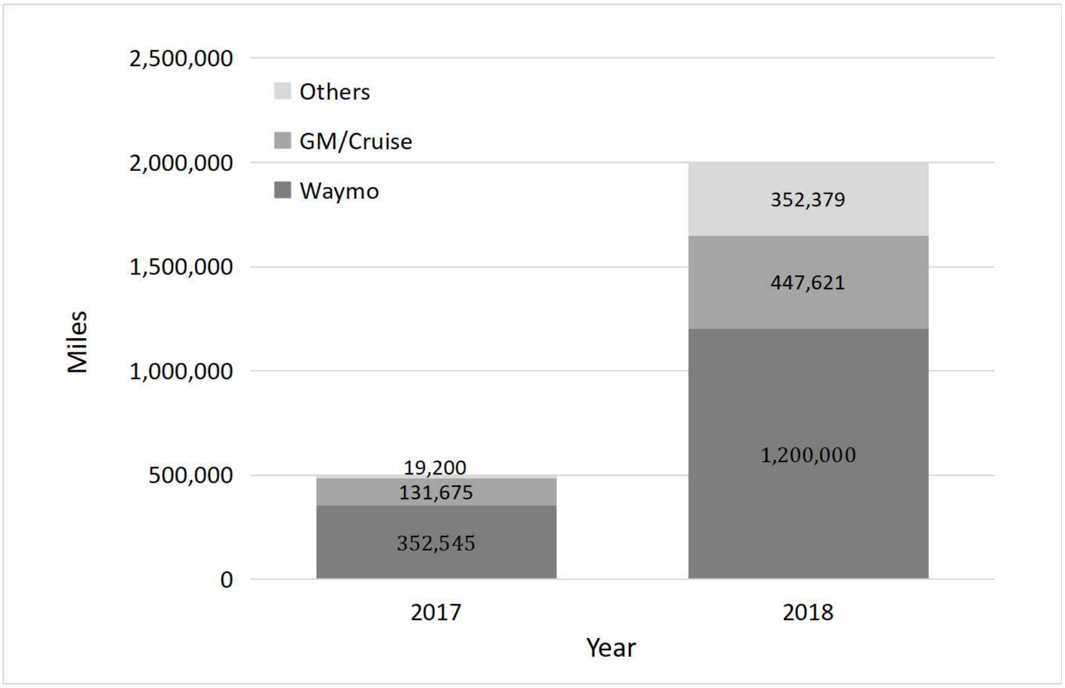


Figure 1.7: Autonomous driving testing distance in California, source: [17] [18] [19]

1.4 Simulation

Due to the decreasing development time and the increasing development costs as well as complexity of the complete vehicle, experimental test methods are increasingly being replaced by virtual simulation. With suitable simulation software, it is possible to validate many test cases early in the development process. In this section the simulation in the area of the development of driver assistance systems is considered. The requirements for simulation programs, are the following [5] [6] [21]:

- Ease of use
  - Visual input and output
  - Design of the scenarios
  - Software support service
- Functional traffic environment
  - Parameterization of the static and dynamic objects
  - Mutual influence of the transport partners
- Configurability of vehicle properties
  - Validated vehicle parameters available
  - Modifiability of vehicle parameters
  - Possibility of importing vehicle models from other simulation programs

2 Methodology

The objective of this master thesis is to develop and demonstrate a method for the virtual calibration of ADAS function with the targets of fuel efficiency, comfort and safety. In this thesis, an MPACC (Model Predictive Adaptive Cruise Control) function is calibrated as an example.To this end, a co-simulation platform is applied to simulate the function and a calibration software is employed to calibrate the function.

2.1 Co-simulation set-up

The co-simulation is controlled by AVL Model.CONNECT. In this thesis, IPG CarMaker and MATLAB/Simulink are integrated into AVL Model.CONNECT as elements. In figure 2.1, the simplified simulation environment and data exchange logic are illustrated. The single blocks will be explained in the following sections.

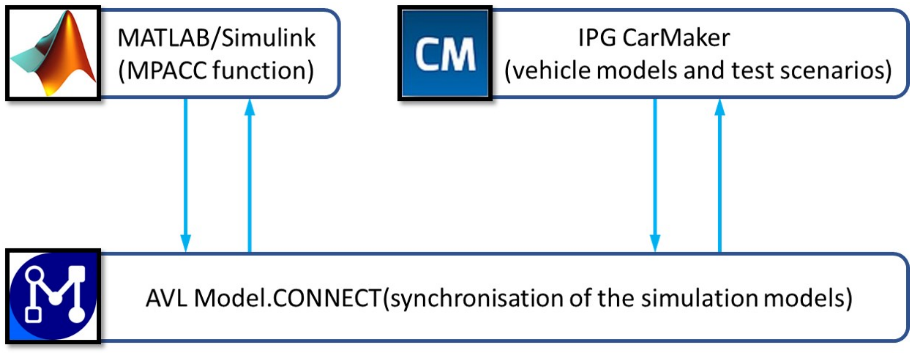

MATLAB/Simulink
(MPACC function)
CM
IPG CarMaker
(vehicle models and test scenarios)
AVL Model.CONNECT(synchronisation of the simulation models)

Figure 2.1: Simulation environment

2.1.1 AVL Model.CONNECT

AVL Model.CONNECT [22] is the co-simulation platform which is used in this master thesis. It has many different interfaces for the most of common used vehicle simulation software. These software packages can be integrated into AVL Model.CONNECT as elements. In this master thesis, the integrated software packages are IPG CarMaker and MATLAB/Simulink, which are connected to the respective simulation models with different interfaces. IPG CarMaker is connected to the specified CarMaker model with

CarMaker-Model.CONNECT interface and MATLAB/Simulink to the ICOS (Independent Co-Simulation) model with MATICOS-Model.CONNECT interface. These models are connected with the input and output ports, which are sorted by name, unit and data type. For example, the input ports of CarMaker model gas and brake pedal position are connected to the output ports of MATLAB model gas and brake pedal position. Additional with the ”signals” components is the co-simulation set up. Here the ”constant” and ”monitor” components can be added as system elements to set the model parameters and monitor the signals. In figure 2.2, the example of the co-simulation model and ports connection is illustrated.

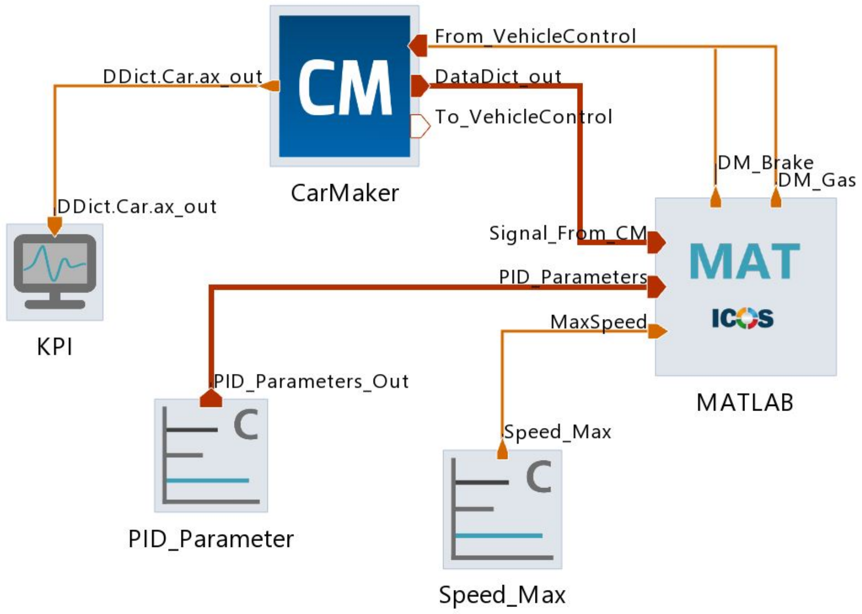


Figure 2.2: Example of co-simulation model in AVL Model.CONNECT

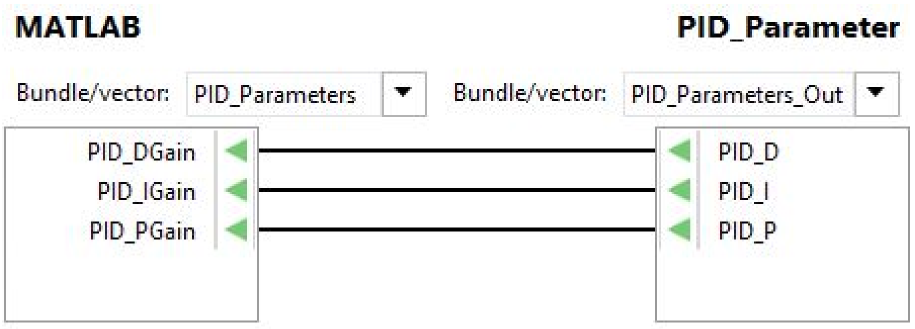

MATLAB
PID_Parameter
Bundle/vector: PID_Parameters
Bundle/vector: PID_Parameters_Out
PID_DGain
PID_IGain
PID_PGain
PID_D
PID_I
PID_P

Figure 2.3: Example of ports connection inside the bundles

In order to have a better view of the model, the input and output ports can be summarized to bundles, which have several input or output ports inside but just one port outside. In figure 2.2, the output bundle ”PID_Parameters_Out” of block ”PID_Parameter” is connected to the input bundle ”PID_Parameters” of the block ”MATLAB”. Inside the bundles, the relative signals are paired, see figure 2.3. In addition, the single input or output port can also be connected to the port inside the bundle, see figure 2.2, the output ports ”DM_Brake” and ”DM_Gas” of block ”MATLAB” are connected to the relative input ports inside the bundle ”From_VehicleControl” of block ”CarMaker”.

Attention has to be paid to the data types of the paired ports. For the connected ports with the same variable (e.g. speed), the different units (m/s and km/h) are allowed. Inside AVL Model.CONNECT the SI (The International System of Units) units are used for calculation. But for the ”signals” components (e.g. ”Constant” or ”Signal file”), AVL Model.CONNECT would just take the values of the ports without considering the units, such as the output port ”MaxSpeed” of ”Constant” block ”Speed_Max” in figure 2.2. So it is always recommended to apply the SI units for parameters and variables. At the same time, attention has to be paid to the scale of the ports. For example, the scales of output ports ”DM_Brake” and ”DM_Gas” of block ”MATLAB” and input ports ”From_VehicleControl.Brake” and ”From_VehicleControl.Gas” of block ”CarMaker” in figure 2.2. In addition, every user defined input port must be connected, such as the input port ”MaxSpeed” in figure 2.2, otherwise the simulation cannot be started. However the output ports can be unconnected.

The values of every defined ports can be visible through many kinds of graphs of table, such as ”Curve Monitor”, ”Gauge”, ”Slider”, ”Table” and so on, during the simulation. And the value of output ports in element ”Monitor” can be varied by the ”Slider” during the simulation. After the simulation, we can analyse the data with different graphs, such as ”Line Chart”, ”Pie Chart”, ”2D Surface” and so on. In addition, the simulation results of a Model.CONNECT project are stored under the project folder with the .csv format. With this format the data can also be processed by MATLAB offline, which is beneficial for the post-processing of the data.

Additionally, the parameters can be defined as variables, which can be accessed by the ”signals” components as well as the calibration software AVL CAMEO, under the ”Parameters” tab. At the same time, the pre-script , which is written in python, can also operate the variables definition in MATLAB m-file. With this we can change the settings of MPACC function automatically with the pre-script.

Finally, the responses are defined based on the simulation results under the ”Optimization” tab. These responses can be accessed by AVL CAMEO as measurements after the simulation. There are several basic statistic operations (such as minimum and maximum value, integral, the value at certain point and so on) that can be applied to the simulation results.

2.1.2 MATLAB/Simulink

As the most common used vehicle modeling and simulation software, MATLAB/Simulink has many powerful toolboxes to help the developer modeling and simulating the systems as well as the controllers. In this master thesis, the MPACC function is developed with MATLAB/Simulink. The simulation model of MATLAB/Simulink is integrated into AVL Model.CONNECT with the MATICOS-Model.CONNECT interface. The inputs and outputs are replaced by the specified ICOS inputs and outputs, which can be directly accessed as input and output ports by AVL Model.CONNECT. In figure 2.4, the ICOS input and output blocks are displayed. With the input signals (e.g. position, speed and acceleration of ego vehicle as well as preceding vehicle) from CarMaker model, the control signals gas and brake pedal position are calculated and sent back to the CarMaker model.

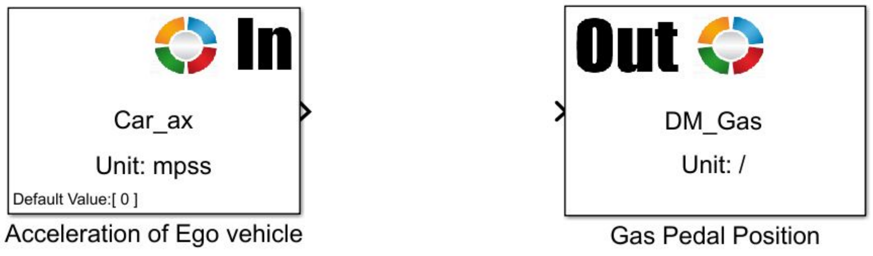
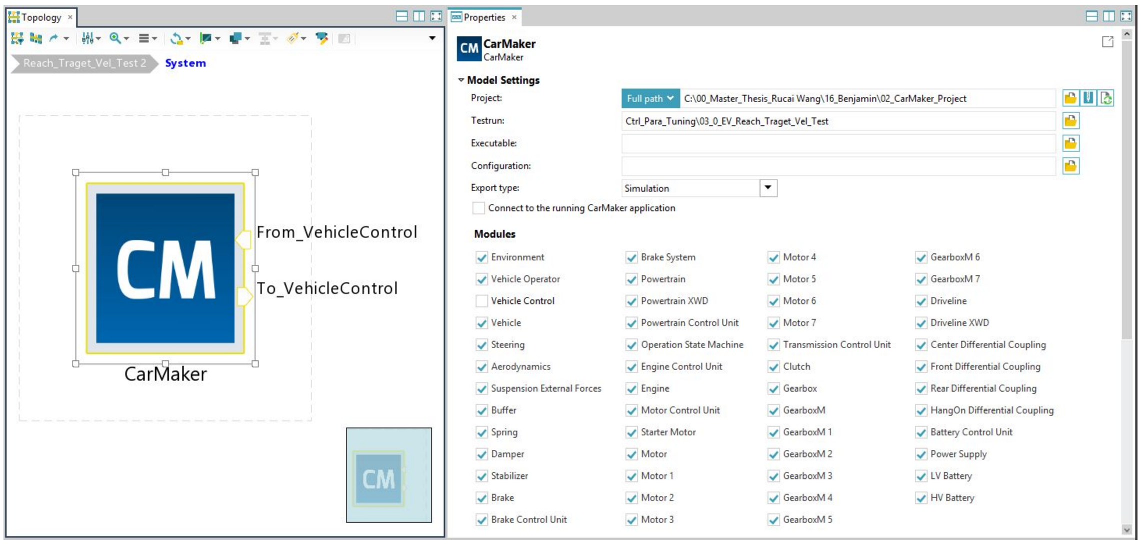


Figure 2.4: Example of ICOS input and output blocks

2.1.3 IPG CarMaker

IPG CarMaker [23], which is used in this master thesis, is widespread environment and vehicle dynamics simulation software for virtual testing. It provides the vehicle dynamic models with all necessary parameters as well as the test scenarios and environment, at the same time, the perception sensors are also provided by IPG CarMaker with simplified sensor models. The CarMaker simulation model is integrated into AVL Model.CONNECT with a specified CarMaker-Model.CONNECT interface within AVL Model.CONNECT. The input and output ports of CarMaker can be activated by deactivated the related modules inside CarMaker model. Figure 2.5 shows the specified CarMaker-Model.CONNECT interface with the ”Vehicle Control” module deactivate, the input and output ports inside the bundles ”From_VehicleControl” and ”To_VehicleControl” are activated.

In CarMaker project, firstly we need to select a car as the ego vehicle and set the parameter values for ego vehicle or just take the default values. There are plenty of vehicle parameters which can be defined by the users, from the mass and CoG (Center of Gravity) of vehicle to the maps of engine and ratio of gears. In addition, the sensors are also added to the ego vehicle here. There are several kinds of sensors available, such as slip angle sensor, acceleration sensor and so on. In this master thesis, a radar sensor, which is used for detecting of the preceding vehicle, is applied in the front of the vehicle with predefined installation position, detection range, field of view in vertical and horizontal direction and so on. In addition, there are two types of target selection can be selected, "Nearest in the path" or "Nearest". "Nearest in the path" means that only the target in the same path with ego vehicle is considered, whereas "Nearest" means all the targets on the road are considered.



Model Settings

Project:	Full path C:\00_Master_Thesis_Rucai Wang\16_Benjamin\02_CarMaker_Project
Testrun:	Ctrl_Para_Tuning\03_0_EV_Reach_Traget_Vel_Test
Executable:	
Configuration:	
Export type:	Simulation

☐ Connect to the running CarMaker application

Modules

✓ Environment	✓ Brake System	✓ Motor 4	✓ GearboxM 6
✓ Vehicle Operator	✓ Powertrain	✓ Motor 5	✓ GearboxM 7
☐ Vehicle Control	✓ Powertrain XWD	✓ Motor 6	✓ Driveline
✓ Vehicle	✓ Powertrain Control Unit	✓ Motor 7	✓ Driveline XWD
✓ Steering	✓ Operation State Machine	✓ Transmission Control Unit	✓ Center Differential Coupling
✓ Aerodynamics	✓ Engine Control Unit	✓ Clutch	✓ Front Differential Coupling
✓ Suspension External Forces	✓ Engine	✓ Gearbox	✓ Rear Differential Coupling
✓ Buffer	✓ Motor Control Unit	✓ GearboxM	✓ HangOn Differential Coupling
✓ Spring	✓ Starter Motor	✓ GearboxM 1	✓ Battery Control Unit
✓ Damper	✓ Motor	✓ GearboxM 2	✓ Power Supply
✓ Stabilizer	✓ Motor 1	✓ GearboxM 3	✓ LV Battery
✓ Brake	✓ Motor 2	✓ GearboxM 4	✓ HV Battery
✓ Brake Control Unit	✓ Motor 3	✓ GearboxM 5	

Figure 2.5: Specified CarMaker model within AVL Model.CONNECT

After the ego vehicle definition, the road is defined under the "scenario/road" tab of CarMaker. Firstly the road segment such as straight, junction, turn and so on, is defined. Then we need to define the segment lanes, which can be deleted or added as you like. In order to define the route, which is the line that vehicle follows, the reference line should be defined firstly. In addition, the road related parameters (e.g. markings and friction coefficient) and other infrastructure (e.g. traffic lights and signs) as well as legal information (e.g. speed limits) can be defined by the users. Finally the start route of ego vehicle needed to be chosen and the ego vehicle would start from this route.

The next step is to define the maneuver of the ego vehicle. Here the start velocity is defined. Under every submaneuver the longitudinal and lateral dynamics are defined with IPG driver, speed profile and so on. For the IPG driver, the characteristics and limits of the virtual driver can be defined as needed. If the maneuver of the vehicle is too complex, the speed profile can be loaded from a file. At the same time, we can also define the end condition or have some minimaneuver-commands for the submaneuver, in order to control the simulation.

For the traffic vehicle, it is defined under the "traffic" tab of CarMaker. The process is the same as the ego vehicle definition. But for the traffic vehicle, there are not so many parameters can be defined by the user. Only the size and movement related parameters are available. The next step is to define the start position of the traffic vehicle which is based on the CarMaker coordinate system and the start route which is predefined in the road definition. The last step is to define the maneuver of traffic vehicle, which is similar with the maneuver definition of ego vehicle. The speed profile of the traffic vehicle can be also imported from files, which is beneficial when the required traffic maneuver is too complicated. The only difference is to define the start condition of the traffic vehicle, otherwise it would start at the same time with ego vehicle. In addition, we can also define several traffic vehicles with the same process. The drawback of predefined traffic is that they do not interact with the ego vehicle and other traffic vehicles.

In the "IPGMovie" window of CarMaker, the simulation is visualized, see figure 2.6.

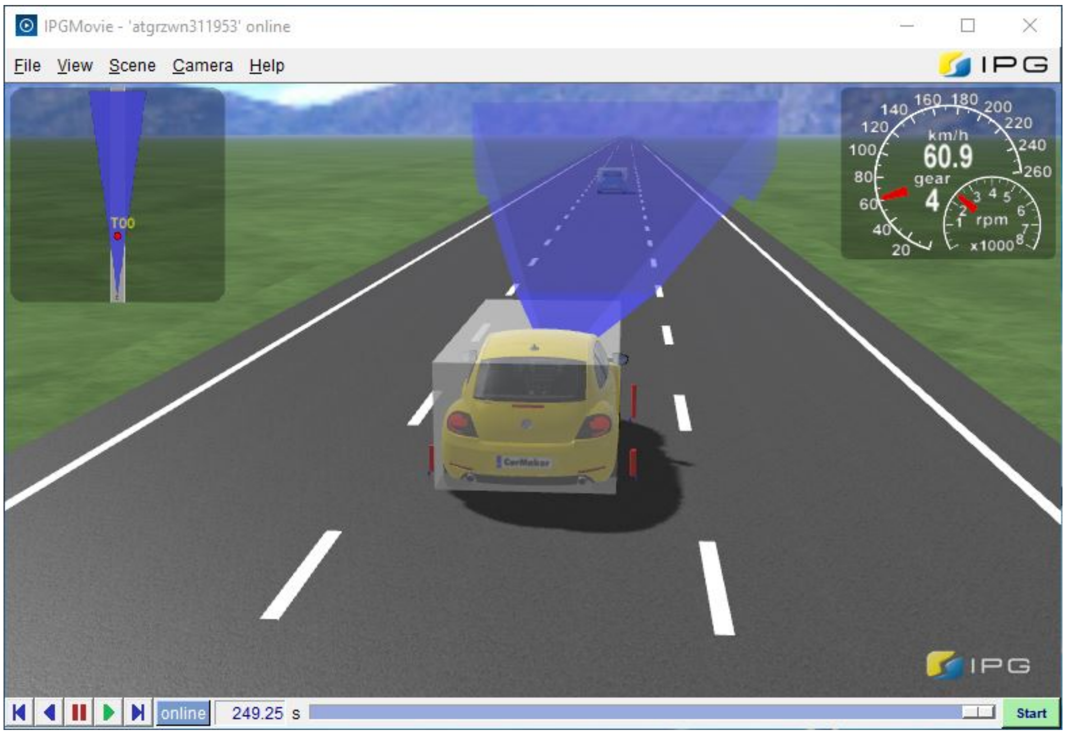

IPGMovie - 'atgrzwn311953' online
File View Scene Camera Help
IPG
T00
km/h
60.9
gear
4
rpm
x1000
online
249.25 s
Start

Figure 2.6: Visualization of the simulation

2.1.4 Coordinate system

The coordinate system, which is used in this master thesis, is based on the IPG CarMaker coordinate system. In CarMaker, coordinate systems, which are called frames, with different origins are defined [24]. However, the direction of the axes is always the same, following DIN 70000 (see figure 2.7) [26]:

- x-axis: Points in driving direction of the vehicle
- y-axis: Points 90 degree to the left side of the x axis

- z-axis: Points upwards vertically

The reference frame for parametrization of the vehicle data set is called Fr1 (Frame One). Its origin is located at the center rearmost point of the vehicle, on the ground, see figure 2.8.

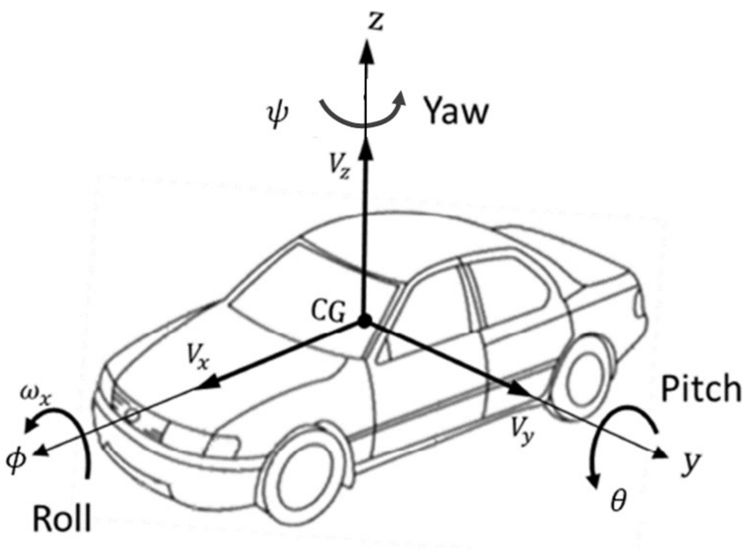


Figure 2.7: ISO 8855 [25] / DIN 70000 coordinate system, source: [26]

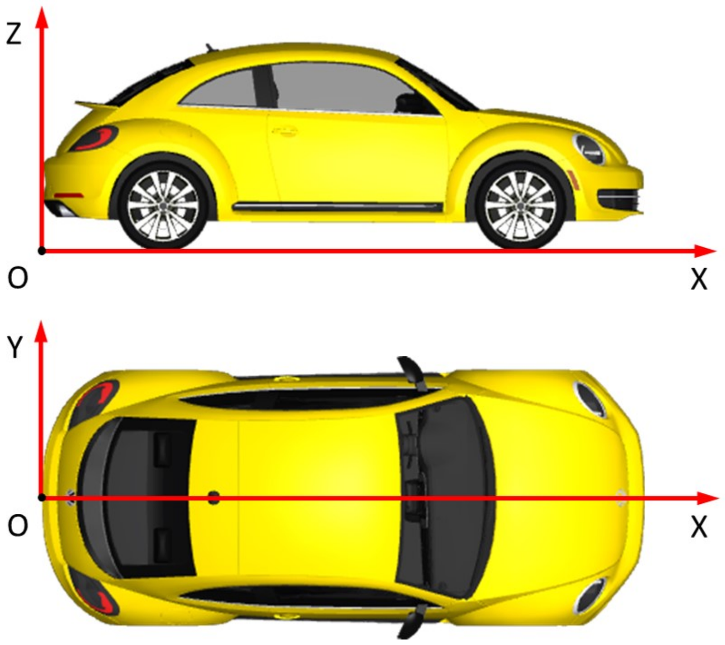
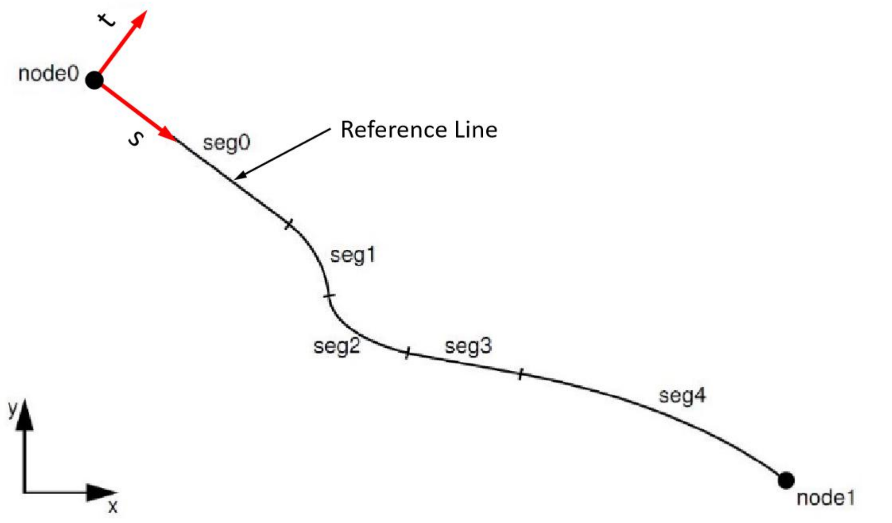
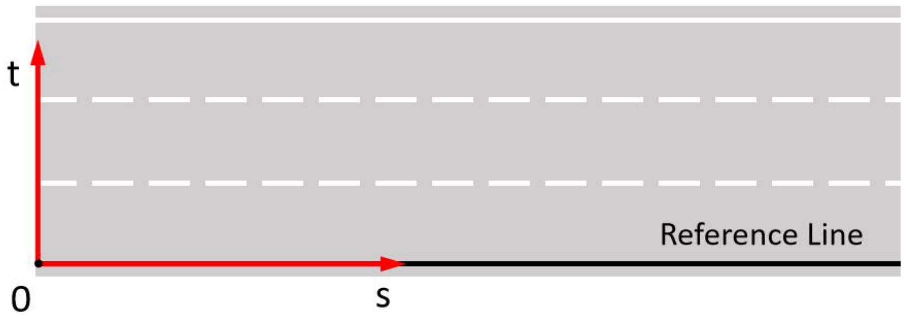


Figure 2.8: Definition of CarMaker vehicle coordinate system Fr1, source: [24]

For the entire co-simulation, the road coordinate system, which is called s-t coordinate system, is applied, see figure 2.9 (a). It travels along the "Reference Line". The origin of the s-t coordinate system lies in "node0" on the "Reference Line", the s-axis moves along the "Reference Line" and the t-axis points left of the s-axis in a right angle of 90 degree, see figure 2.9 (b) [24]. The s-axis is important when it comes to adding road obstacles. Obstacle positions are specified depending on their distance from either the beginning or end of a link or junction entrance or exit along the s-axis [24]. The t-axis is import when the traffic vehicles are added. The lateral position of the traffic participants can be based on the t-axis or the routes. It is needed to specify the lateral reference of the traffics [24].



(a) Macro



(b) Micro

Figure 2.9: Definition of CarMaker road coordinate system, source: [24]

2.2 Calibration set-up

The calibration is done with AVL CAMEO and AVL Model.CONNECT. In this thesis, AVL Model.CONNECT is connected to AVL CAMEO with a specified system interface in AVL CAMEO, which is called "AVL Simulation Desktop". The parameters and responses are imported to the AVL CAMEO project from AVL Model.CONNECT. The variation list is generated by AVL CAMEO through the internal DoE methods. After that AVL CAMEO will send the variations to AVL Model.CONNECT one by one and start the co-simulations. At the end, AVL CAMEO gets back the simulation responses, so called measurements, and calibrate the MPACC function based on the responses processing results. In figure 2.10 , the simplified calibration design is displayed.

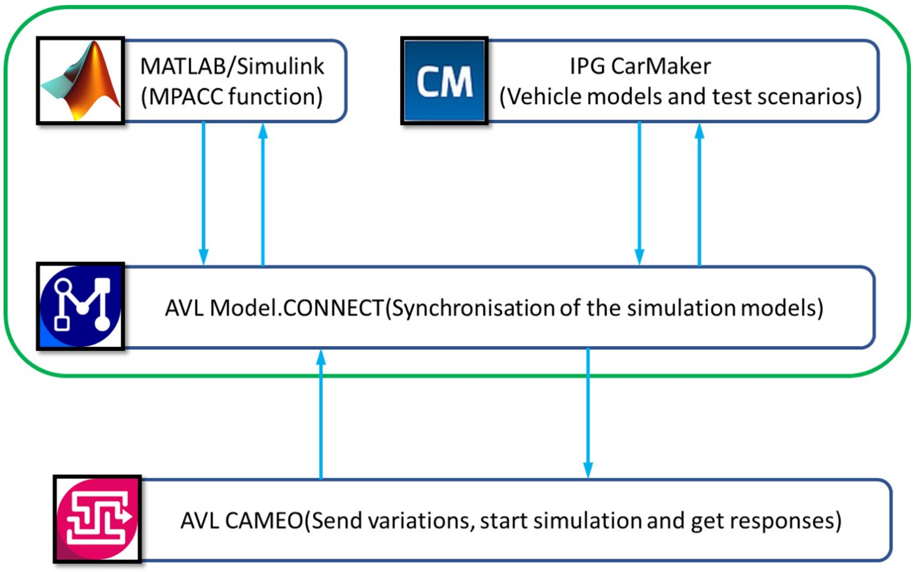

MATLAB/Simulink
(MPACC function)
CM
IPG CarMaker
(Vehicle models and test scenarios)
AVL Model.CONNECT(Synchronisation of the simulation models)
AVL CAMEO(Send variations, start simulation and get responses)

Figure 2.10: Calibration set-up

2.2.1 AVL CAMEO

AVL CAMEO [27] is the calibration software used in this master thesis. It can automatically start the simulation runs one by one based on a variation list, which is prepared by AVL CAMEO, and after every simulation the measurements are sent back to AVL CAMEO and stored, which will be processed online or offline. There are the following four main processes for the virtual calibration with AVL CAMEO:

- System set-up
- Simulation set-up and execution
- Data processing and optimization

- Verification

(1) System set-up

AVL CAMEO has a user-friendly predefined interface with AVL Model.CONNECT, which is called ”AVL Simulation Desktop”. The only thing for system set-up is to create a new ”AVL Simulation Desktop” system, choose and connect the target Model.CONNECT version, select the simulated project and import the parameters as well as responses, which are defined in AVL Model.CONNECT.

(2) Simulation set-up and execution

For the simulation set-up and execution, there are several steps to be followed. Firstly, a test is created with a specified name as well as some descriptions, and it needs to be connected to the target AVL CAMEO system, which is built in the process ”system set-up”. After the simulation is created, we can go to the next step, preparing the simulation runs. The operating point is defined, and inside operating point, the variables are defined with start value, minimum and maximum value. Then the variation list is created by using AVL CAMEO DoE (Design of Experiment) methods or just by importing the variation list that is previously prepared. The measurements are defined after the variation list. The third step is to run the simulation runs with the predefined start variation point and variation list sequentially. The last step is to check the test results after all the simulation runs have finished.

(3) Data processing and optimization

Create model evaluation and models with the test results. In this process, we can import different data sets from files, tests or other model evaluations as standard data, which is used to build the behavior models, or as verification data, which is only used for the verification of the behavior models. After importing the data set, the outliers can be detected with the predefined limits in AVL CAMEO. Before modeling, the raw data can be checked with the variation distribution graphics or measurements graphics. Then the models can be built with the internal algorithm RNN (Recurrent neural network) or ”FreePloyModel”. The quality of the models can be directly checked in the ”Measured-Predicted” graphics. After modeling, the optimization process can be applied with defined optimization target and constraints. For multi-optimization targets there would be a trade off results, which combines the different optimization targets. Then the possible optimization point can be selected automatically or manually by the user.

(4) Verification

The optimization points should then be simulated again for verification. The measured results of the optimization points should fit the predicted values from the models.

2.3 Design of Experiment

The design of experiments (DOE) is a method for statistically planning and executing tests or simulations. The purpose of DoE is to get the relationship between inputs and outputs with less effort [28] [29].

2.3.1 Full factorial

The full factorial design, which is the most popular design among engineers, is not practical or efficient enough if the calibrated parameters are more than three [28] [29]. Figure 2.11 shows the exponential increase of application efforts when increasing parameters. In this example, featuring five different values for five parameters, the full factorial design leads to more than 3000 variations.

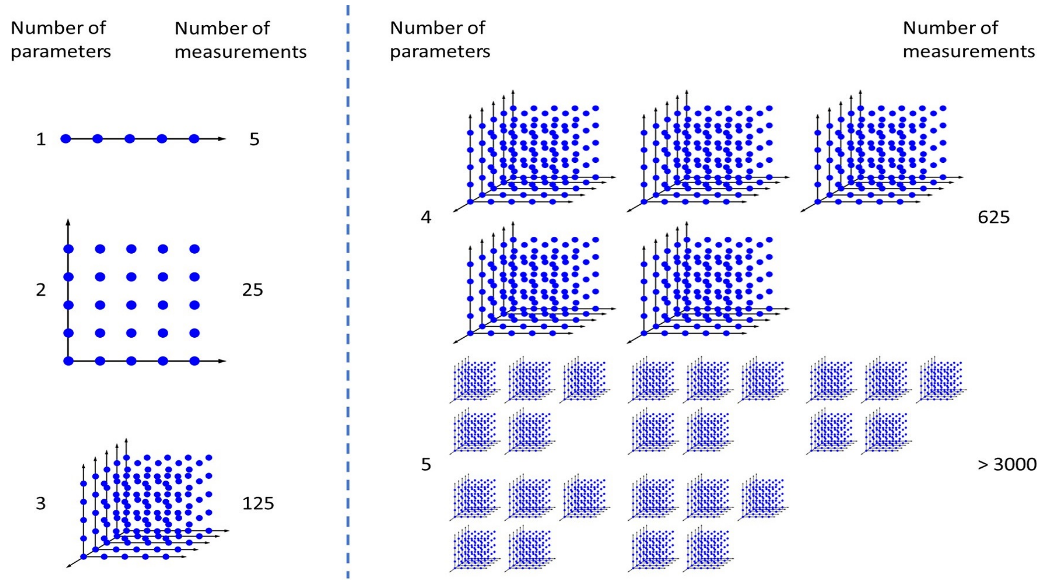


Figure 2.11: Full factorial design with exponential increase of application efforts, source: [28]

2.3.2 One factor at a time

Figure 2.12 shows an example of the principle for "One factor at a time" (OFAT) design [28] [29]. There are only two variations discussed and the response need to be minimized. Firstly, starting with the black start point, changing the value of variation 1 and fixing the variation 2. The black point with gray edge is the minimum response for variation 1. Then starting with the black point with gray edge, changing the value of variation 2 and fixing the variation 1. The gray point is the optimization point found by the OFAT design. Compared to the full factorial design, OFAT design can save many efforts, but the disadvantages are also really obvious. The optimization point is strongly depending on the start point, which requests a high level of experience and the pre-knowledge of the calibrated system. Secondly, it cannot always find the global optimization point, or even cannot find the optimization point.

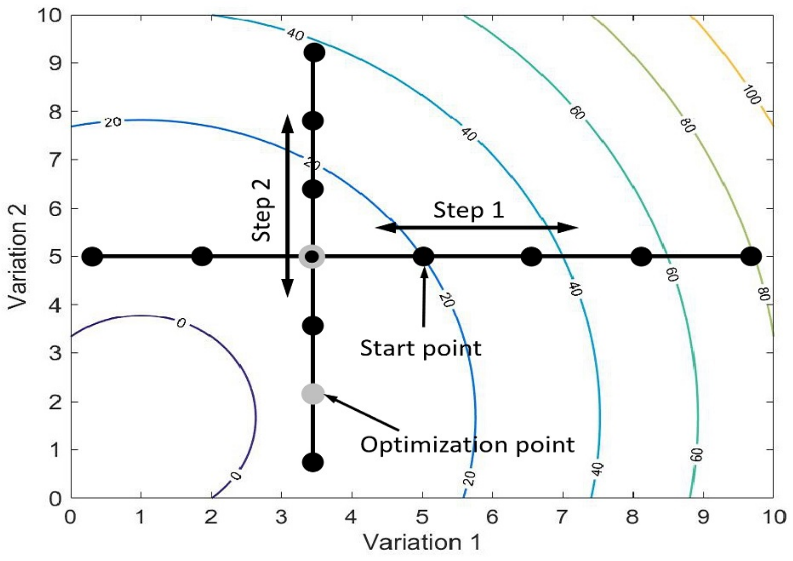


Figure 2.12: Conventional "One factor at a time" design, source: [28]

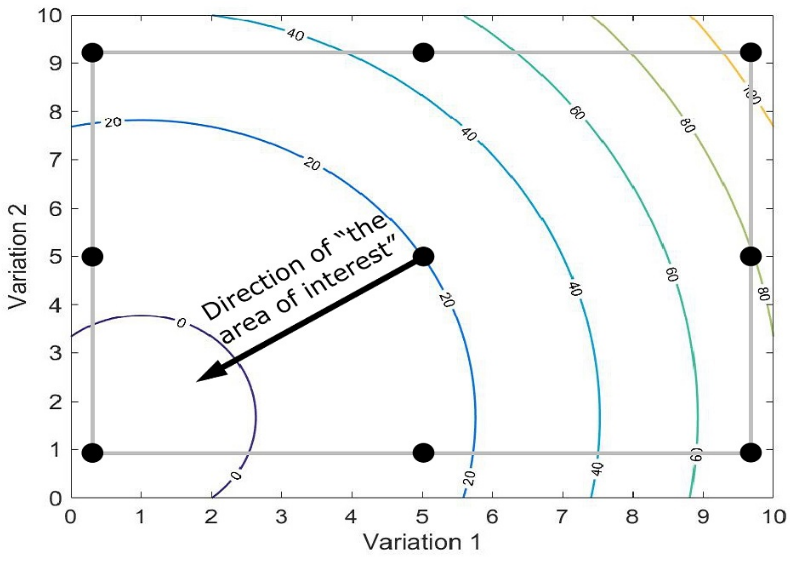


Figure 2.13: Conventional DoE, source: [28]

2.3.3 Conventional DoE

Because of the high efforts of full factorial design and the uncertain results of OFAT design, we introduce the conventional DoE method [28] [29], see figure 2.13. Firstly, the simulation runs or testing runs are executed based on a variation list (the black point in the figure 2.13) generated by a statistical method. With the results of the simulations or testings, the behavior models of measurements are build. Based on the behavior model, "the area of interest" is found with the optimization target and constraints. Finally, the optimization point is searched with adding the random points in the design space and confirmed with verification runs.

2.3.4 Active DoE

Active DoE [28] [29] [30], also called COR DoE (Customized Output Range DoE), is another DoE method, which is developed based on conventional DoE. Figure 2.14 shows the active DoE method. Based on the results of conventional DoE, the behavior model of responses are built online and "the area of interest" (the area in the dashed rectangle in figure 2.14) is found with the optimization target and constraints. After that the iteration points (the hollow points in figure 2.14) are generated in "the area of interest" based on the behavior model and simulated. With the simulation results of iteration points, the behavior model is optimized in "the area of interest". Finally, the optimization point is searched with adding the random points in the design space and confirmed with verification runs.

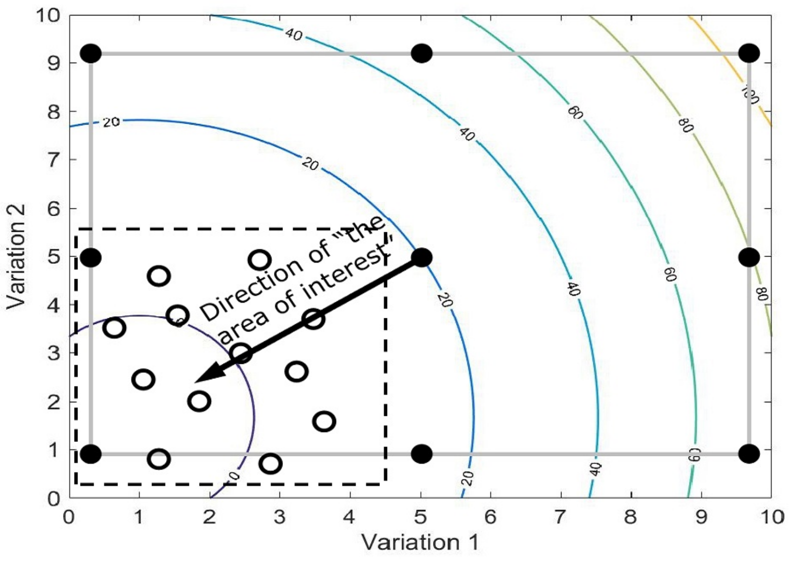


Figure 2.14: Active DoE, source: [28]

Figure 2.15 shows the process of the active DoE method. Firstly, on the basis of already measured responses, behavior models are calculated online. If the models match the desired model quality, the simulation will be stopped, if not, the next variation point, so called iteration point, will be generated based on the behavior model within "the area of interest": "Minimum Output" and "Maximum Output" of certain responses, which are set by the users. Then the simulation would be executed with the automatically generated iteration point and after the simulation the models are built again until the model quality matches a termination criterion. For the generation of the iteration points, the values of each parameters are calculated backwards with the behavior model under given response values and constraints.

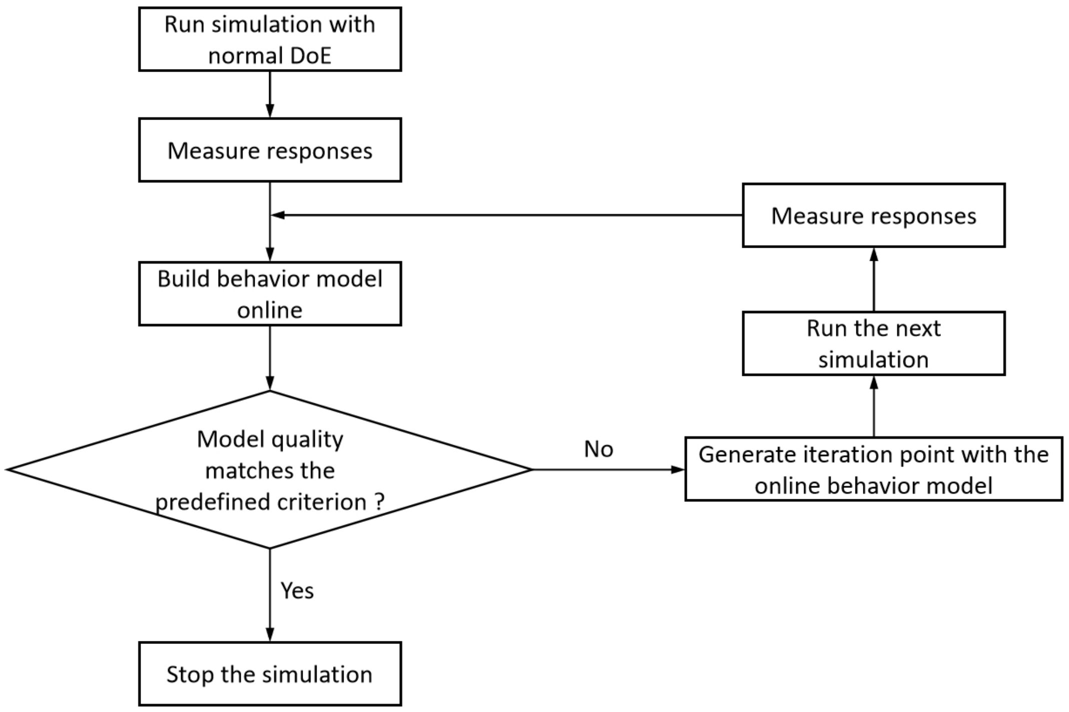

Run simulation with normal DoE
Measure responses
Build behavior model online
Model quality matches the predefined criterion ?
No
Yes
Stop the simulation
Generate iteration point with the online behavior model
Run the next simulation
Measure responses

Figure 2.15: Process of active DoE

2.4 Data processing

In this section, the main data processing methods that used in this master thesis are introduced.

2.4.1 Outliers detection

For the measurements from the simulations, there might be some outliers, which will affect the qualities of the behavior models. Figure 2.16 shows the example of measurements with outlier.

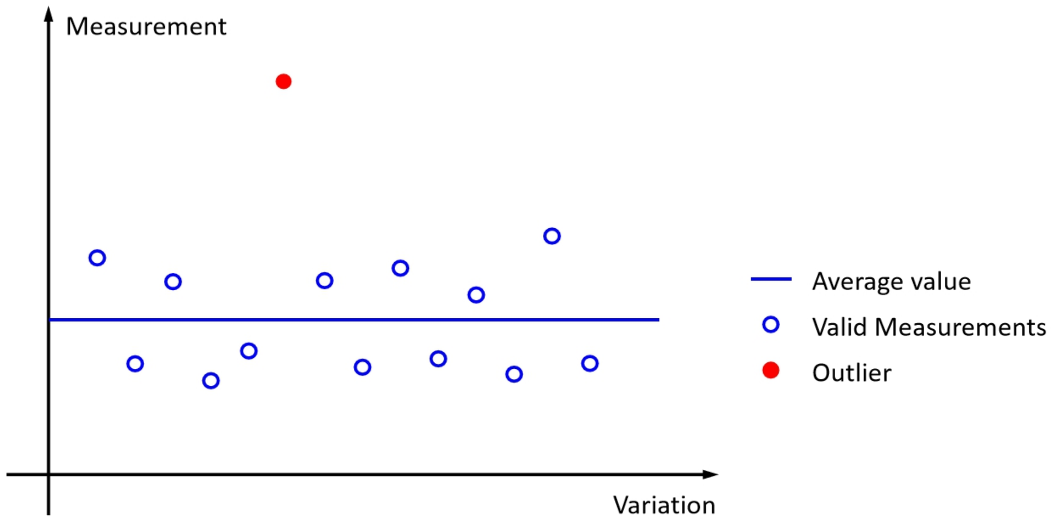


Figure 2.16: Example of measurements with outliers

For the detection of the outliers, the average value of all the measurements, including the outliers, are calculated, see equation (2.1) [30].

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{2.1}$$

With the average value, calculating the standard deviation of these measurements, see equation (2.2).

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.2}$$

$$y^{\pm} = \bar{y} \pm 3s \tag{2.3}$$

For the measurements, which are not in the range of the threshold value $y^{\pm}$ defined by equation (2.3), are treated as the outliers [30].

2.4.2 Modeling

The behavior model of measurements with the variation channels as inputs are built. There are two model types available, RNN (Recurrent Neural Network) model and free polynomial model [30]. We can choose the model type as well as the model order (up to five) for both RNN model and free polynomial model, or just build the models automatically, then the system would generate all the possible models and choose the best model. Figure 2.17 shows an example of behavior model.

The behavior model is built in a way of giving the confidence interval (space between two green lines) of the model, which means the true behavior model will lay in the interval with a specified confidence level (e.g. 95%) [30]. In section 2.4.4 is the generation of confidence interval described in detail.

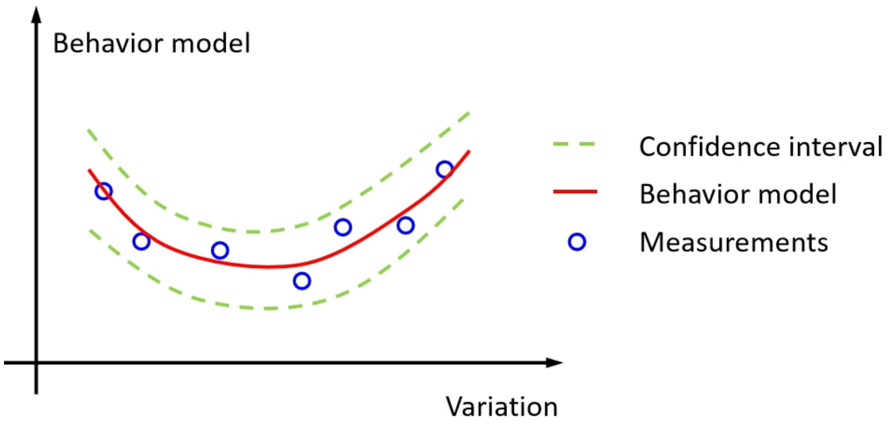


Figure 2.17: Example of behavior model

2.4.3 Goodness of tit for behavior model

The goodness of fit is evaluated by means of the statistical coefficients R^2, R^2_{adj} and R^2_{pred} [30]. R^2 is the coefficient of determination, which indicates to which degree the model explains the deviations of the measured values from a constant mean value. It shows how precisely the model fits to the measured values and is given by the equation (2.4) [30].

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} \tag{2.4}$$

Where

SSE: Deviation between measured value and predicted value
SSR: Deviation between predicted value and mean value
SST: Deviation between measured value and mean value

and

$$SSE = \sum (y_i - \hat{y}_i)^2 \tag{2.5}$$

$$SSR = \sum (\hat{y}_i - \bar{y})^2 \tag{2.6}$$

$$SST = \sum (y_i - \bar{y})^2 \tag{2.7}$$

$$SST = SSE + SSR \tag{2.8}$$

Figure 2.18 shows the definition of SST SSE and SSR.

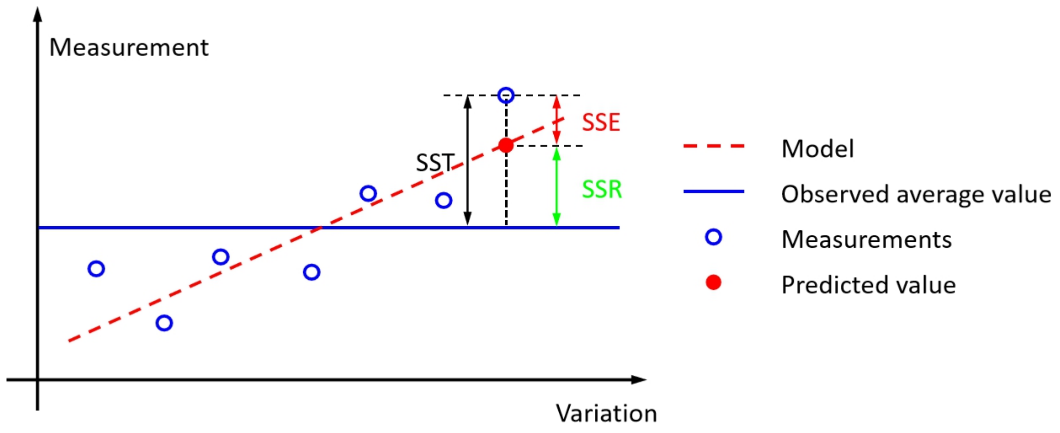


Figure 2.18: Definition of SST, SSE and SSR

R^2_{adj} is the adjusted coefficient of determination to evaluate the quality of the model calculation (R^2), additionally taking into account the degrees of freedom of the model equation [30]. It evaluates if the number of measured values is sufficient to determine the model equation (R^2_{adj}). If, for example, R^2 is too large for a higher order model and R^2_{adj} is too small, this means that the measured values lie in the vicinity of the model area, whereas the model equation is not determined precisely [30]. In this case, it would be necessary to apply a lower order model (fewer degrees of freedom) to get a better R^2_{adj}, R^2_{adj} is given by the equation (2.9).

$$R^2_{\text{adj}} = 1 - \frac{SSE/(n-m)}{SST/(n-1)} \tag{2.9}$$

Where

n indicates the number of measured values,
m indicates the number of independent model regression coefficients.

R^2_{pred} is the goodness of prediction, which is a statistic of the goodness of prediction of models [30]. R^2_{pred} indicates the degree of certainty to which it can be assumed that any point in the model area is actually true and is given by equation (2.10) [30].

$$R^2_{\text{pred}} = 1 - \frac{PRESS}{SST} \tag{2.10}$$

Where

$PRESS$: Deviation between measured values from model values, the j^{th} model does not take y_j into account), see figure 2.19.

$$PRESS = \sum (y_j - \hat{y}_{(j)})^2 \tag{2.11}$$

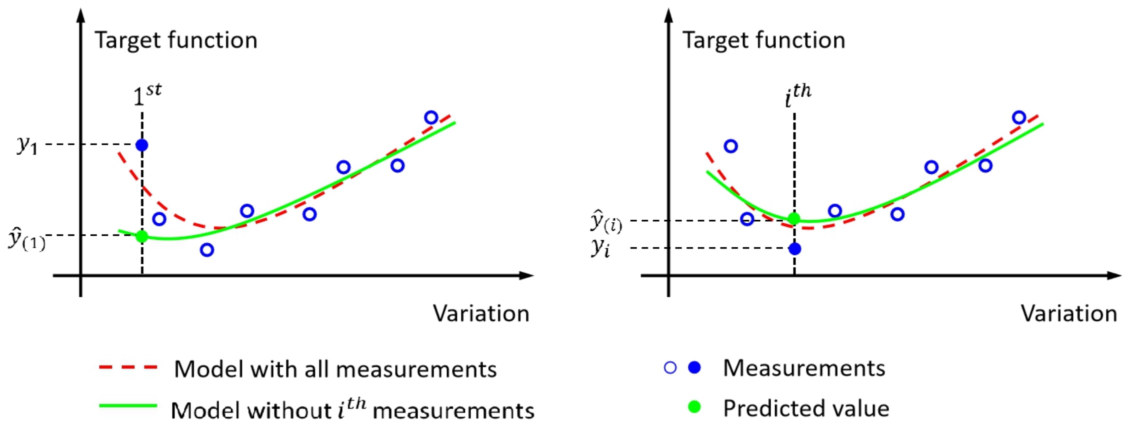


Figure 2.19: Predicted residual error sum of squares, source: [30]

The threshold values for goodness of fit for behavior model are shown in table 2.1.

Table 2.1: Threshold values for goodness of fit for behavior model, source: [30]

Coefficient	Value range	Very good	Good	Medium	Poor
R^2	0 to 1	$\geq$ 0.95	$\geq$ 0.70	$\geq$ 0.50	$<$ 0.50
R^2_{adj}	$-\infty$ to 1	$\geq$ 0.95	$\geq$ 0.70	$\geq$ 0.50	$<$ 0.50
R^2_{pred}	$-\infty$ to 1	$\geq$ 0.90	$\geq$ 0.60	$\geq$ 0.40	$<$ 0.40

In addition to the statistical coefficients R^2, R^2_{adj} and R^2_{pred}, the root mean square error ($RMSE$) and normalized root mean square error ($NRMSE$), and they are calculated with the following equations (2.12) and (2.13):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}} \tag{2.12}$$

$$NRMSE = \frac{RMSE}{y_{\max} - y_{\min}} \cdot 100\% = \frac{\sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)}{n}}}{y_{\max} - y_{\min}} \cdot 100\% \tag{2.13}$$

2.4.4 Residual analysis

In building regression models it is assumed, that residuals (i.e. deviations of the measured value from the model values, see equation (2.14)) are distributed according to the Gaussian distribution (Normal distribution). For this hypothesis the mean is 0 and the standard model deviation is constant. Normal distribution can be applied to see whether the model describes the data correctly [30].

$$e = y - \hat{y} \tag{2.14}$$

If the model is good, the residuals are consistent with the normal distribution, i.e. there is no change in the standard model deviation, the residuals are insignificantly small and lie in a horizontal band [30]. If the number of residuals increases and the standard model deviation changes, it can be assumed that either there is a poor fit or that there are outliers [30].

Residual analysis is only useful if sufficient measured values are available with regard to model complexity (number of model coefficients) [30]. In mathematical terms, residual analysis is useful if:

$$n \geq 4m \tag{2.15}$$

Where

n indicates the number of measured values
m indicates the number of independent model regression coefficients

In order to get rid of the effect of the number of measurements and the number of model coefficients, the studentized residual e^{**}, see equation (2.16), and student's t distribution are introduced, see figure 2.20 [30].

$$e^{**} = \frac{e}{\hat{\sigma}\sqrt{1 - h_{ii}}} \tag{2.16}$$

Where

h_{ii} is the leverage of i^{th} measurement

$\hat{\sigma}$ is the standard deviation of residuals

The studentized residuals should fulfill the student's t distribution with the degree of freedom of residuals $df = n - m - 1$ [31]. Figure 2.20 shows the Probability density function of normal distribution with the average value 0 and standard deviation 1 as well as the student's t distribution with different degree of freedom. It shows that when $df \to \infty$, student's t distribution $\to N(0, 1)$.

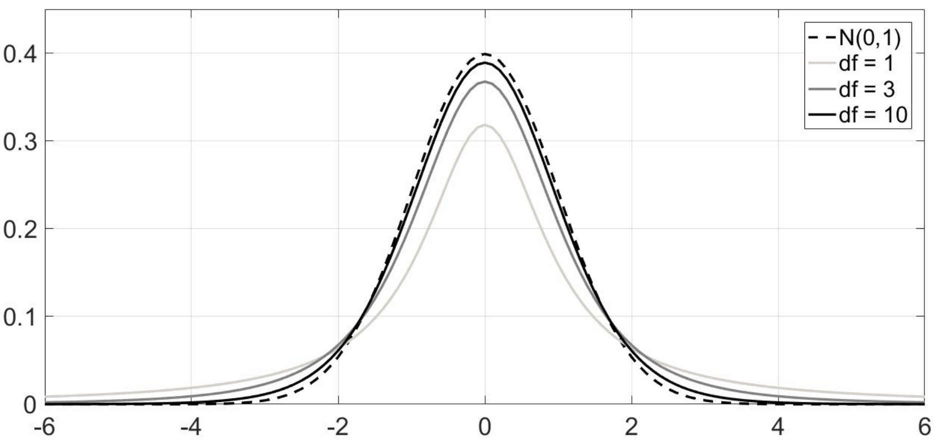


Figure 2.20: Student's t distribution, source: [31]

Equation (2.17) gives the confidence interval of the residual with specified confidence level (e.g. 95%).

$$e_i^{\pm} = e_i \pm \sigma \cdot \sqrt{1 - h_{ii}} \cdot \mathrm{tinv}(1 - \frac{\alpha_{0,95}}{2}, df) \tag{2.17}$$

Where

$\alpha_{0,95} = 0,05 = 1 - 95\%$ means the confidence level is 95%

tinv() is the inverse of the student's t cumulative distribution function [30]

If the confidence interval with confidence level 95% do not cross the zero line, that means this could be an outlier. If the confidence interval with confidence level 99% do not cross the zero line, it must be an outlier [30]. Figure 2.21 shows the residual analysis results with 95% confidence interval. The round points are the outliers, whose 99% confidence interval do not cross the zero line, and the triangle points are the suspected outlier with the 95% confidence interval do not cross the zero line but close to it.

To evaluate the model, only values out of $\pm 3\sigma$ will be used as significant residuals.

The following table 2.2 shows the threshold values of relative model quality based on the residual results.

Table 2.2: Threshold values of model quality, source: [30]

Model quality	Very good	Good	Medium	Poor
Residual out of $\pm 3\sigma$	$\leq$ 3%	$\leq$ 7%	$\leq$ 12%	$>$ 12%

For generation of the confidence interval with 95% confidence level of behavior model, the equation (2.18), which is based on the equation (2.17), are applied.

$$y_i^{\pm} = \hat{y}_i \pm \sigma \cdot \sqrt{1 - h_{ii}} \cdot \mathrm{tinv}(1 - \frac{\alpha_{0,95}}{2}, df) \tag{2.18}$$

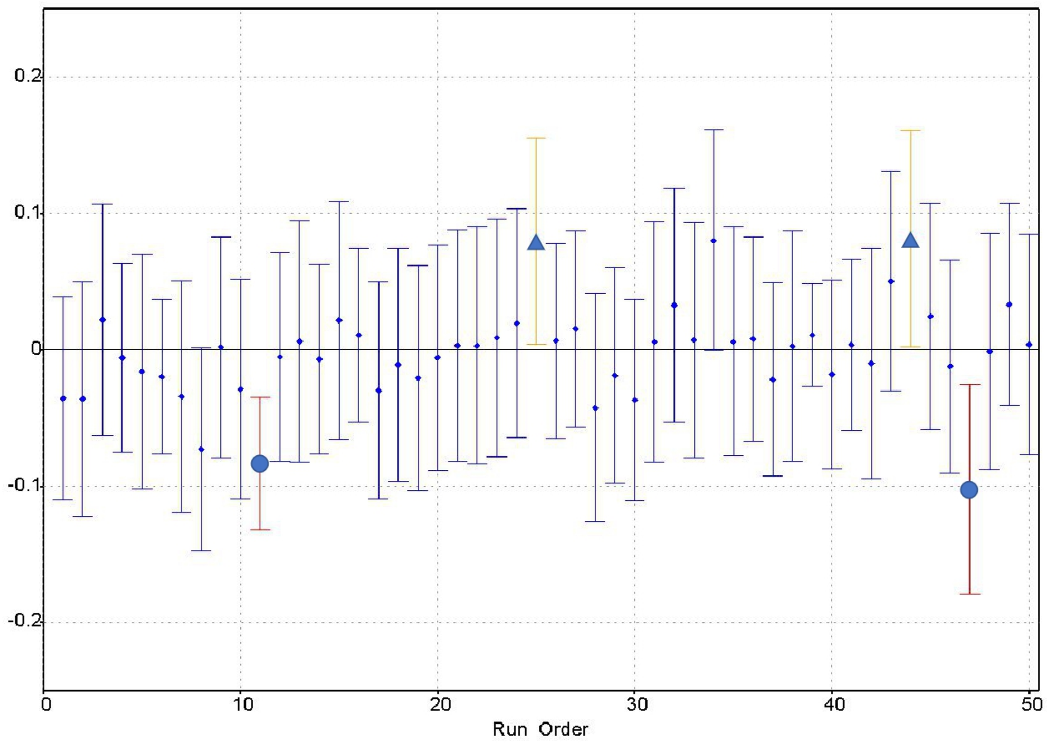


Figure 2.21: Example of residual analysis, source: [30]

2.4.5 Correlation analysis

The purpose of the correlation analysis is to identify, how much the variation channels can influence the responses. In contrary to the existing cross correlation analysis, where just linear influences are considered, this method also takes parabolic influences into consideration, or even higher nonlinear effects and interactions [32].

The sensitivity analysis algorithm, which is applied in this thesis, analyses the contribution of each individual variation channel to the quality of the calculated INN (Iterative Neural Network) behavior model. In other words, it analyses the reduction of the SSE (see figure 2.22) by an input channel on a selected target channel [32].

The algorithm starts by considering one variation channel at one time, and computes m (the number of all variation channels) individual INN-models by change the variation channel until the most significant first variation channel is detected by reaching the lowest $SSE_{\mathrm{low}}^{(1)}$ (SSE with only one variation channel, see figure 2.22). The SSE of

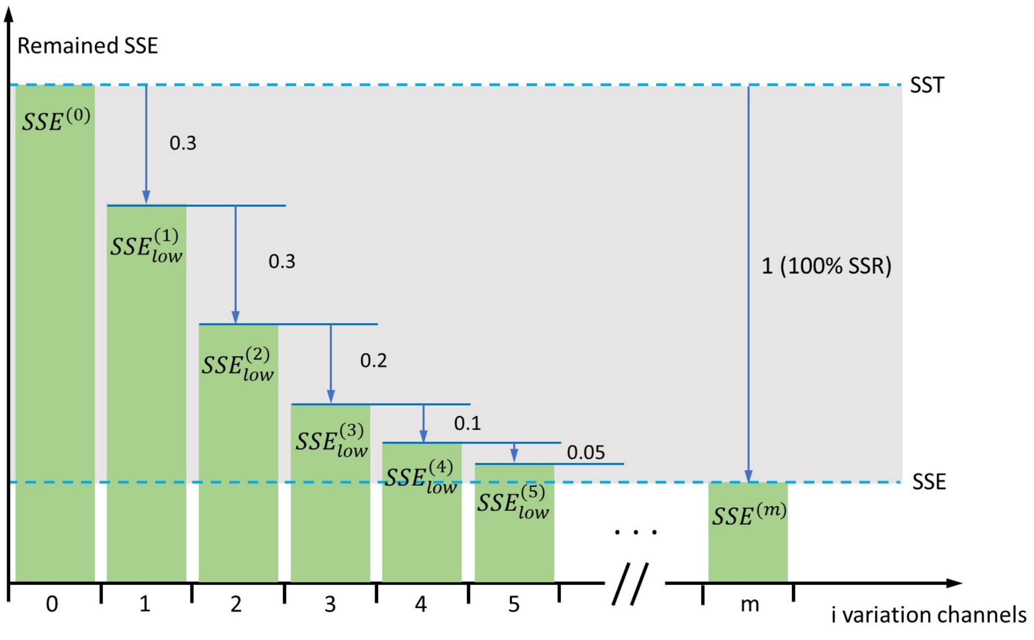


Figure 2.22: Remained SSE with i considered variation channels, source: [32]

the individual models is compared against each other. The INN model with the lowest SSE was modeled using the input variation channel having the greatest relevance [32]. And the Relative Significance Indicator (RSI) for the first channel is calculated by the following equation (2.19).

$$RSI_{1\text{st}} = \frac{SST - SSE_{\text{low}}^{(1)}}{\text{SSR}} \tag{2.19}$$

In figure 2.22, SSE , SSR and SST, see equations (2.5) to (2.7) indicate the relative valve of the INN behavior model with m variation channels.

In the next step, with the first detected most significant channel, another channel is considered additional and $m-1$ individual INN-models are built. The most significant second variation channel is detected by reaching the lowest $SSE_{\text{low}}^{(2)}$. And the Relative Significance Indicator for the second channel is calculated by the following equation (2.20).

$$RSI_{2\text{nd}} = \frac{SSE_{\text{low}}^{(1)} - SSE_{\text{low}}^{(2)}}{SSR} \tag{2.20}$$

This process is repeated iteratively until all the channels have been ranked per their contribution to the total SSE reduction of the selected target channel. The distance between the SST and the SSE of the best Model is set as 1 (100% SSR, see figure 2.22) [32].

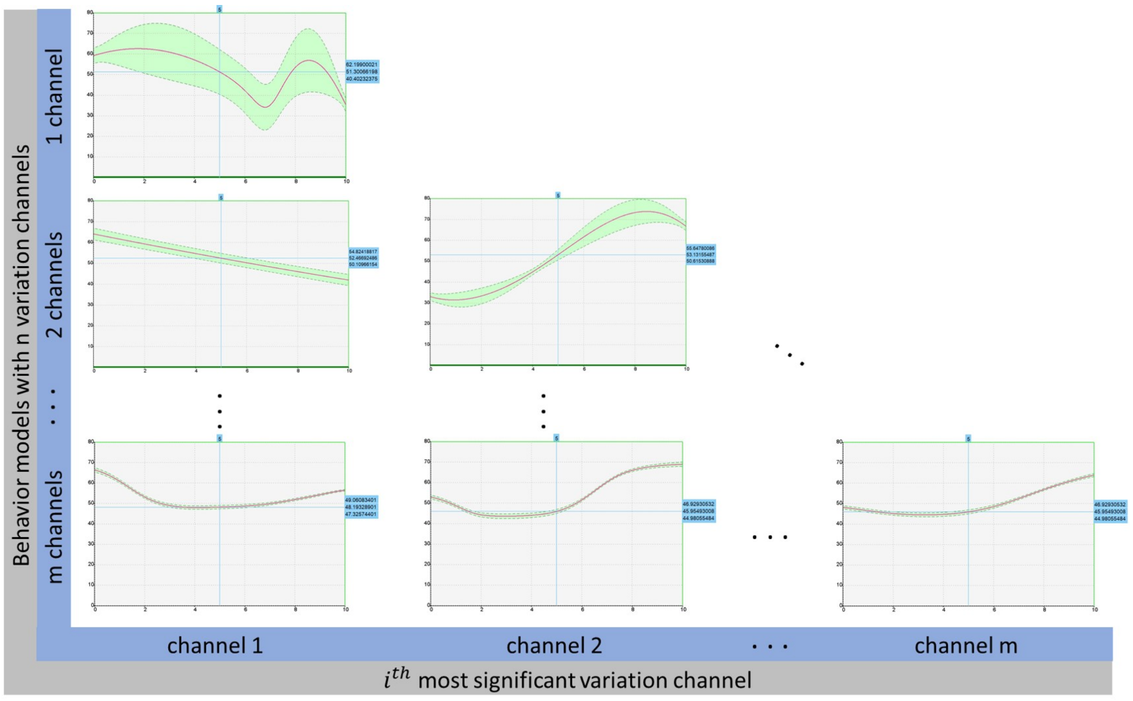


Figure 2.23: Intersection graphic of result INN Modeling with i considered variation channels, source: [32]

Figure 2.23 shows intersection graphic of result INN model with i considered variation channels in AVL CAMEO. As the increasing number of variation channel, the confidence interval of the model decrease, which means that the models are getting better.

2.5 MPACC function overview

The objective of the investigated control strategy is to allow the ego vehicle to drive with fuel efficiency, comfort and safety. For this purpose, a model predictive control algorithm is applied for the MPACC function, which will adaptively control the acceleration of the ego vehicle based on the prediction model. The model recurrently predicts the motion of preceding vehicle over a predictive horizon of 20 seconds with the traffic information, such as preceding vehicle speed and acceleration, traffic lights and so on. In addition, an PID controller is also applied, in order to stabilize the desired acceleration from the MPC controller.

2.5.1 PID

PID control is the widely employed feedback control algorithm in industrial control system [33]. An PID controller continuously calculates an error value $e(t)$ (see equation (2.22)) as the difference between a reference value $r(t)$ and a measured value $y(t)$, then applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively), until the system is stabilized. In figure 2.24, the block diagram of PID control is illustrated. The control input signal $u(t)$ for the plant is given by the equation (2.21) [33]: PID control is the widely employed feedback control algorithm in industrial control system [33]. A PID controller continuously calculates an error value $e(t)$, see equation (2.22) as the difference between a reference value $r(t)$ and a measured value $y(t)$, then a corrective term based on proportional, integral, and derivative components (denoted P, I, and D respectively) is applied, in order to stabilize the system. The error value $e(t)$ is the input to the PID controller, which outputs the variable $u(t)$, that is the input to the plant, which outputs the dynamic target $y(t)$, that is forwarded to close the control loop. In figure 2.24, the block diagram of PID control is illustrated. The control input signal $u(t)$ for the plant is given by the equation (2.21) [33]:

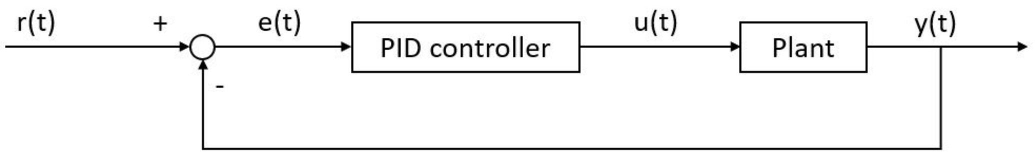


Figure 2.24: Block diagram of PID control

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t)\mathrm{d}t + K_\mathrm{d} \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{2.21}$$

Where

$$e(t) = r(t) - y(t) \tag{2.22}$$

2.5.2 MPC

Model predictive control (MPC) is an advanced method of process control that is used to control a process while satisfying a set of constraints. MPC is based on iterative, finite-horizon optimization of a plant model [34]. Figure 2.25 shows the block diagram of MPC. In comparison to the PID control, the MPC would also consider the future reaction of the system while calculating the next control input signal $u(t)$.

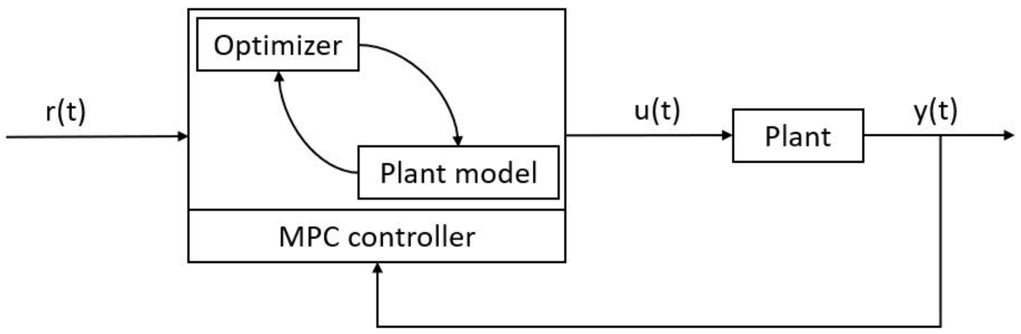


Figure 2.25: Block diagram of MPC

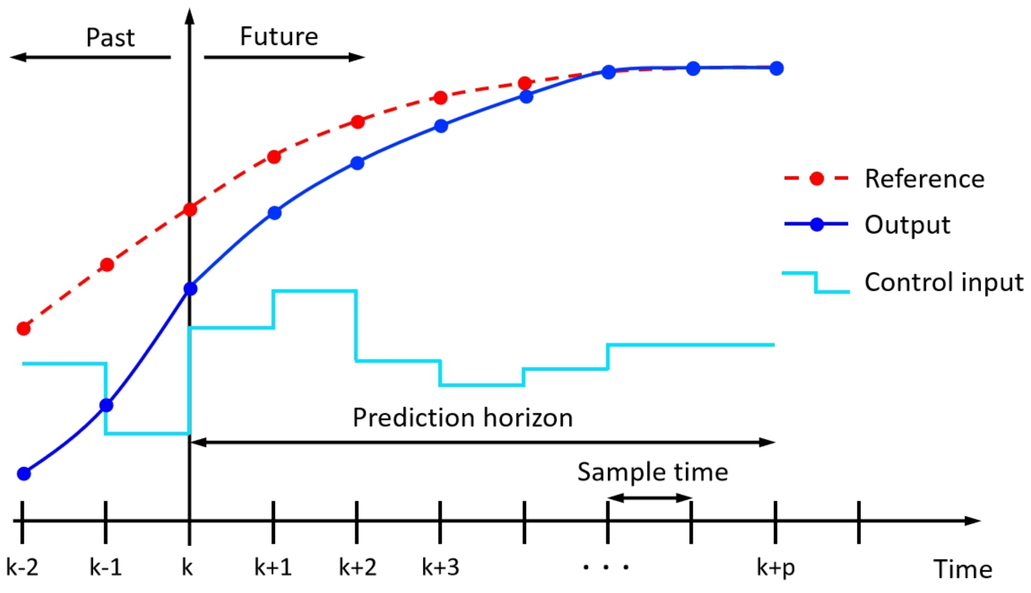


Figure 2.26: MPC control logic, source: [34]

In figure 2.26 the model predictive control logic is illustrated. At time k the current plant state is sampled and a cost minimizing control strategy is computed (via a numerical minimization algorithm) for a relatively short prediction horizon p in the future. The predicted control inputs in the time horizon $[p, k+p]$ are given to the plant model (see figure 2.25) and the predicted system outputs were calculated by the plant model. The cost function is calculated with equation 2.23 :

$$\text{Costfunction} : J(k) = \sum_{i=1}^{p} w^e e^2(k+i) + \sum_{j=0}^{p-1} w^{\Delta u} \Delta u^2(k+j) \tag{2.23}$$

Where

w^e and $w^{\Delta u}$ are the weights of e and Δu

and

$$\Delta u = u(k) - u(k-1) \tag{2.24}$$

After minimizing the cost function, only the first step $u(k)$ of the control strategy is implemented, then the plant state is sampled again, and the calculations are repeated starting from the new current state $k+1$, yielding a new control and new predicted state path.

2.5.3 MPACC function structure

In Figure 2.27, the structure of the MPACC function is displayed. It consists of four parts: inputs, MPACC controller, PI controller and outputs. For the inputs, there are several Buses, which provide the information of the ego vehicle, preceding vehicle, traffic and so on. The MPACC controller delivers the key control algorithm of the MPACC function, which would calculate the desired acceleration of ego vehicle based on the input information. The next part, PI controller would control the ego vehicle to reach the desired acceleration as fast and smooth as possible. Finally, the gas or brake pedal position represent the outputs.

(1) Input signals

As shown in figure 2.27, the input signals are sorted by Buses, such as ”Bus_EgoVehicle”, ”Bus_VehicleAhead”, ”Bus_MPACC”, and so on. The position, speed and acceleration of ego and preceding vehicle are considered as the inputs inside the ”Bus_EgoVehicle” and ”Bus_VehicleAhead”. In addition, in the ”Bus_MPACC” we have several MPACC controller settings, such as the maximum speed of the ego vehicle. For the ”Bus_RoadSegments” and ”Bus_TrafficLights”, we just set them to zero in this master thesis for the reason that they are out of interest in this thesis.

(2) MPACC controller

The MPACC controller is developed based on the model predictive control (MPC), which adaptively controls the vehicle based on the information form the predictive models. These models recurrently predict the motion of the preceding vehicle as well as the ego vehicle over a 20 seconds prediction horizon based on the input information. This information serves as input to a quadratic optimization problem, which aims to minimize the cost function, see equation (2.25), with subjection to the Vehicle constraints, traffic speed limits and headway limits. The task of this thesis is to calibrate these weights of the costs in the cost function.

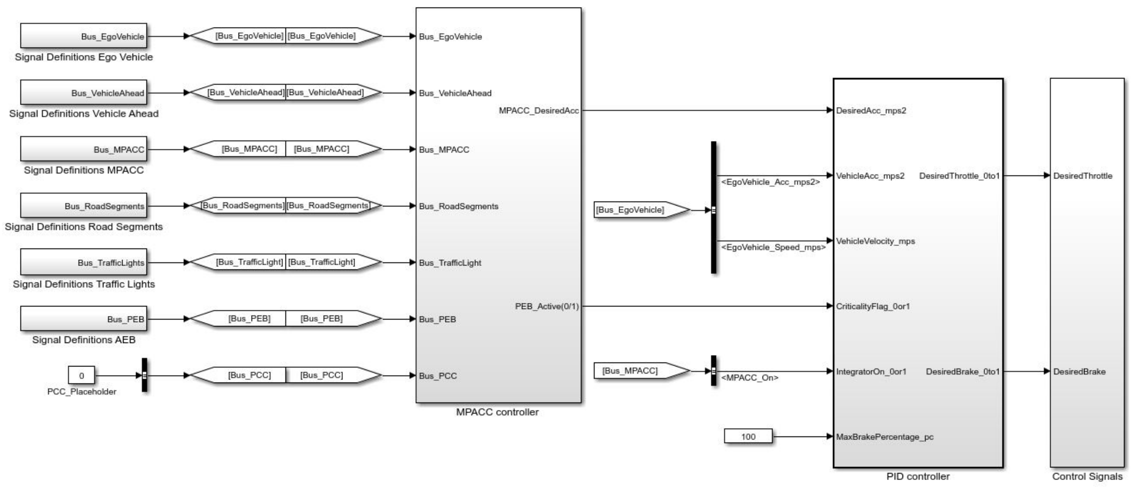


Figure 2.27: The structure of MPACC function

$$
\begin{aligned}
Cost(k) = \sum_{i=1}^{p} & c^{v1} v(k+i) + c^{v2} v^2(k+i) + c^{a1} a(k+i) + c^{a2} a^2(k+i) + c^{va} v(k+i) a(k+i) + \\
& c^{j1} j(k+i) + c^{j2} j^2(k+i) + c^{h1} \gamma_h(k+i) + c^{h2} \gamma_h^2(k+i) + c^{Tr} \gamma^{Tr}(k+i)
\end{aligned}
\tag{2.25}
$$

The Vehicle constraints are the physical limits of vehicles.

$$0 \leq v(k) \leq v_{MAX} = \min\left(v_{\mathrm{MAX}_{\mathrm{General}}}, v_{\mathrm{MAX}_{\mathrm{Legal}}}, v_{\mathrm{MAX}_{\mathrm{Curve}}}, v_{\mathrm{MAX\_TL}}\right) \tag{2.26}$$

$$a_{\mathrm{MIN}} \leq a(k) \leq a_{\mathrm{MAX}} \tag{2.27}$$

$$j_{\mathrm{MIN}} \leq j(k) \leq j_{\mathrm{MAX}} \tag{2.28}$$

The headway limits consist of minimum headway limits and maximum headway limit.

The vehicle is controlled to adapt a predefined, velocity-based headway distance. Figure 2.28 shows the minimum headway distance, which is defined by two minimum headway limits: hard and soft limits. The hard limits must never be violated because of the emergency situation related to a collision, at the same time the soft limit can be violated but with a quadratically increasing cost approaching the hard limit. The cost of the ego vehicle at the position p_2 for violating the minimum headway limits is given by the equation (2.29). For the ego vehicle at the position p_1, there is no cost.

$$Cost_{\mathrm{HW}_{\mathrm{MIN}}}(k+i) = c^{h1} \gamma_h(k+i) + c^{h2} \gamma_h^2(k+i) \tag{2.29}$$

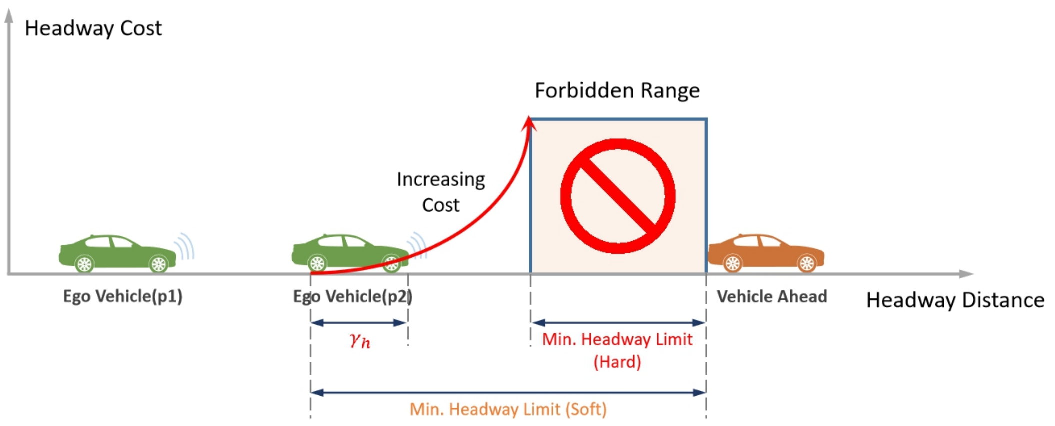


Figure 2.28: Minimum headway limits (hard and soft)

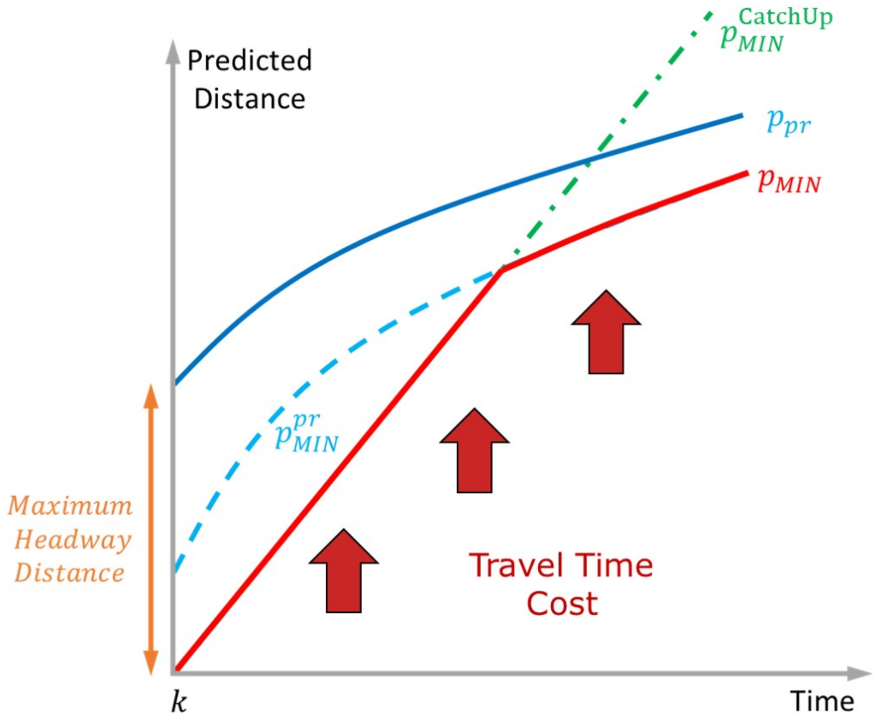


Figure 2.29: Maximum headway limits (only soft)

Figure 2.29 illustrates the maximum headway distance, the distance between the predicted position of preceding vehicle $p_{\text{pre}}^{\text{pr}}$ and the predicted minimum position of ego vehicle p_{MIN}, which is also composed of two components. The first component is proportional to the preceding vehicle speed at time k, also current speed, and scaled by a "catch-up" factor, which is defined greater than one. It works when the ego vehicle is far away behind the preceding vehicle. The second component is a predefined limit, which is a velocity-dependent distance based on the speed of preceding vehicle. It works when the ego vehicle is within the desired headway distance. Both of them are allowed to be violated with the cost of travel time cost, see equation (2.30), but not allowed to violate the speed limits on the route.

$$Cost_{\mathrm{Tr}}(k+i) = c^{Tr}\gamma^{Tr}(k+i) \tag{2.30}$$

2.6 Co-simulation model

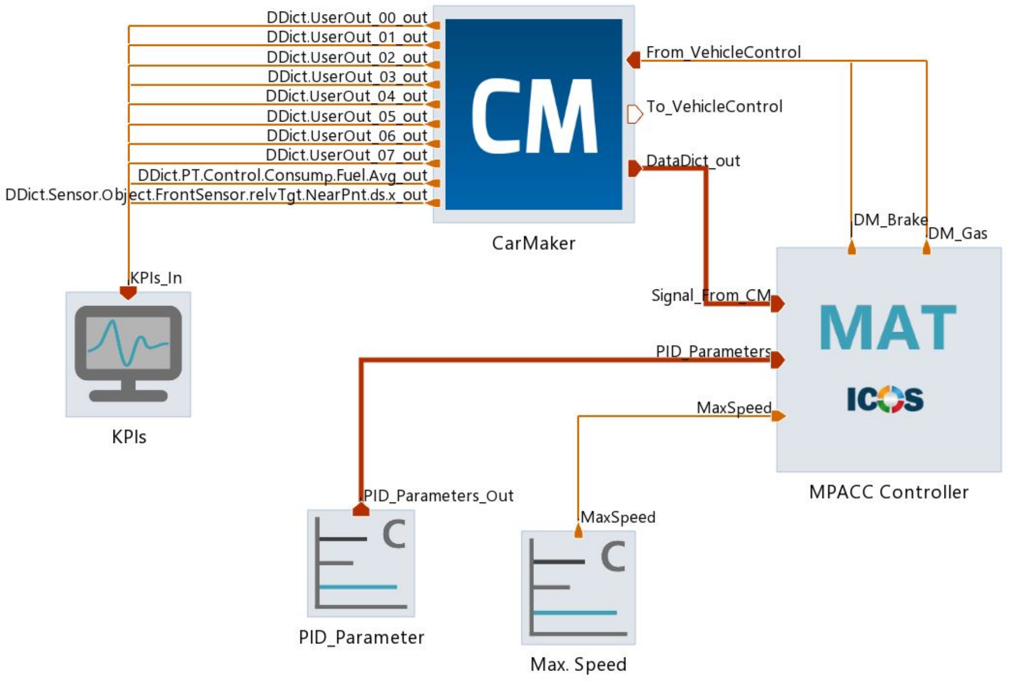


Figure 2.30: Co-simulation model in Model.CONNECT

Figure 2.30 shows the co-simulation model that is used in this master thesis. The element "MPACC Controller" contributes to the main controller in the co-simulation, whereas the element "CarMaker" offers the vehicle models and co-simulation environment. The element "KPIs" serves as a monitor for a better observation of the measurements. In the elements "PID_Parameter" and "Max. Speed", the PID controller parameters and maximum speed are given. And the inputs and outputs of the elements are illustrated in figure 2.31.

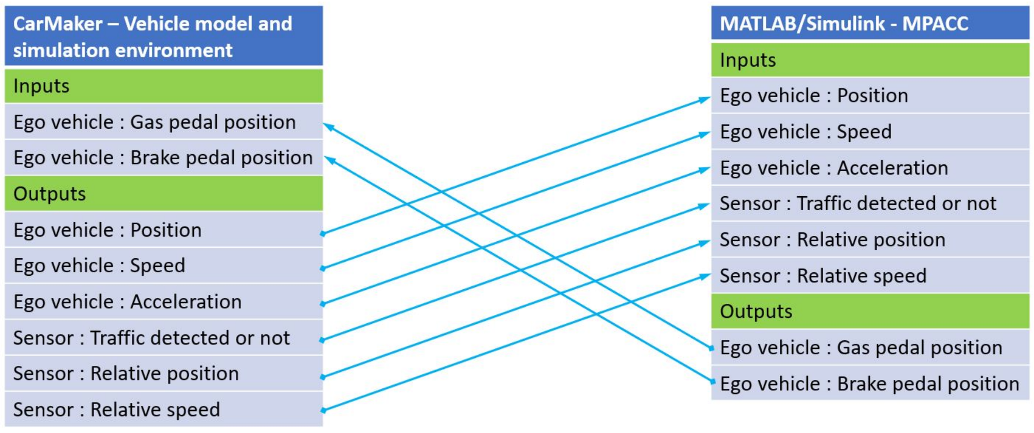

CarMaker – Vehicle model and simulation environment
Inputs
Ego vehicle : Gas pedal position
Ego vehicle : Brake pedal position
Outputs
Ego vehicle : Position
Ego vehicle : Speed
Ego vehicle : Acceleration
Sensor : Traffic detected or not
Sensor : Relative position
Sensor : Relative speed
MATLAB/Simulink - MPACC
Inputs
Ego vehicle : Position
Ego vehicle : Speed
Ego vehicle : Acceleration
Sensor : Traffic detected or not
Sensor : Relative position
Sensor : Relative speed
Outputs
Ego vehicle : Gas pedal position
Ego vehicle : Brake pedal position

Figure 2.31: Inputs and outputs of the co-simulation models

3 Simulation and calibration

In chapter 2, the co-simulation and virtual calibration set-ups are described, besides the structure of the calibrated MPACC function is explained and the data processing methods are introduced.

Target of this chapter is to implement the KPI-model-based (Key Performance Indicator) calibration of the MPACC function with the used co-simulation environment and calibration software. From the structure of the MPACC function, which is illustrated in figure 2.27, the MPACC function can be divided into MPACC controller and PI controller. The MPACC controller optimizes the desired acceleration of the ego vehicle based on the inputs, whereas the PI controller controls the actual acceleration of the ego vehicle to reach the desired acceleration and stabilizes it as fast as possible. During the MPACC controller calibration process, the problem with losing the preceding vehicle occurs. So the testing of ego vehicle reaching target speed is also implemented.

3.1 PI controller optimization

The only task of the PI controller is the control of the actual vehicle acceleration to reach the desired acceleration, which is generated by MPACC controller, as fast and precise as possible. In order to optimize the PI controller, the MPACC controller should be deactivated from the system and a step signal is applied as the desired acceleration.

3.1.1 Scenario definition

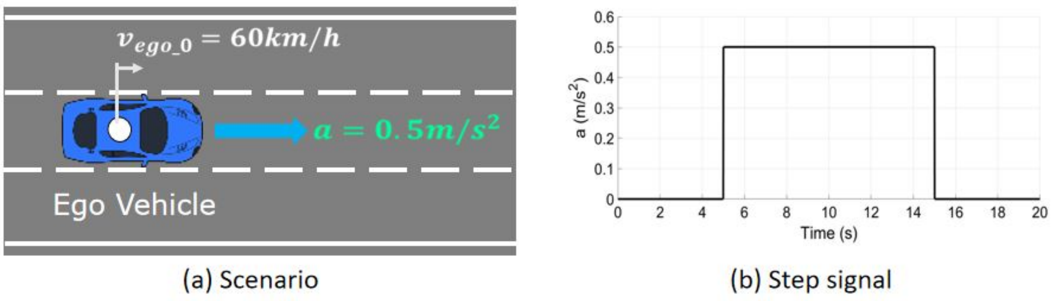


Figure 3.1: Scenario (a) and step signal (b) for the PI controller optimization

Figure 3.1 (a) shows the scenario of PI controller optimization. The ego vehicle starts with the constant speed 60km/h and is controlled to reach the target acceleration $0,5\text{m/s}^2$ with a step signal, see figure 3.1 (b), it steps up from 0 to $0,5\text{m/s}^2$ at time $t = 5\text{s}$. Because of the performance limit of the vehicle, the step signal steps down to 0 again at $t = 15\text{s}$.

3.1.2 Key performance indicator

For the optimization of the PI controller, we define three KPIs, rise time T_r, settling time T_s and overshoot M_p. Figure 3.2 shows the sample step response. The steady-state value y_{ss} of a step response is the final value of the output. The rise time T_r is the amount of time required for the signal to go from 10% to 90% of its final value. The settling time T_s is the amount of time required for the signal to stay within ±5% of its final value for all future times. Finally, the overshoot M_p is the percentage of the first pick value above the final value [35].

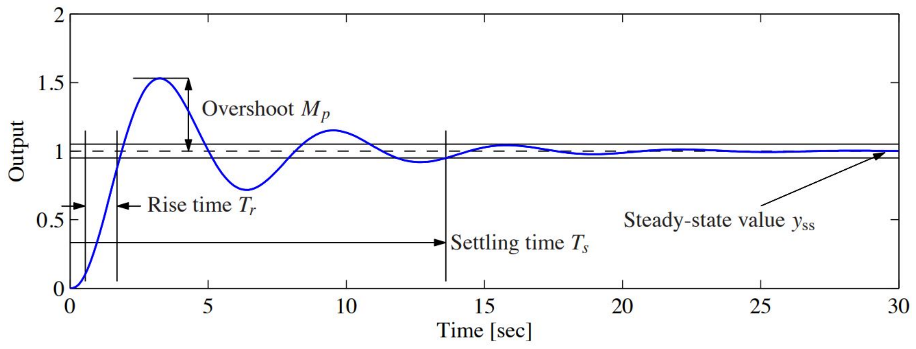


Figure 3.2: Example of step response, source: [35]

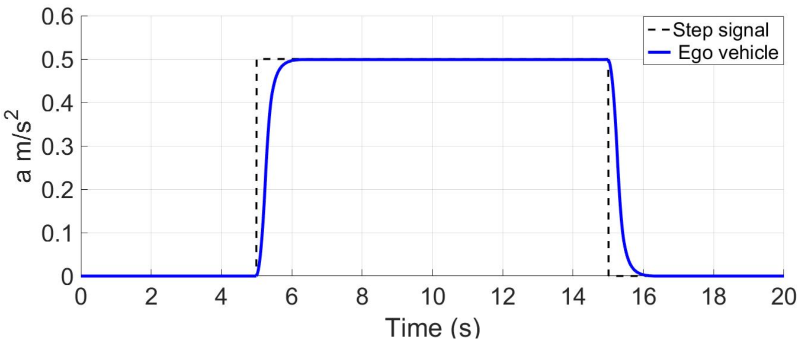


Figure 3.3: Step response of the PI controller

3.1.3 Optimization results

Figure 3.3 shows the step response of the PI controller. With the parameter $K_p = 0,01$ and $K_i = 0,78$, the PI controller has a relatively good performance with rise time $T_r = 0,42s$, settling time $T_s = 0,65s$ and the overshoot $M_p = -0,13\%$.

3.2 ”Reaching target speed” testing

Because of the speed directly related costs in the cost function, the maximum speed that the ego vehicle can reach are different. Under some conditions the ego vehicle cannot reach the maximum speed limit (e.g. on the motorway 130km/h). So before the calibration, the ”reaching target speed” testing is implemetnted, in order to make sure that the ego vehicle can reach the maximum speed limit.

3.2.1 Scenario definition

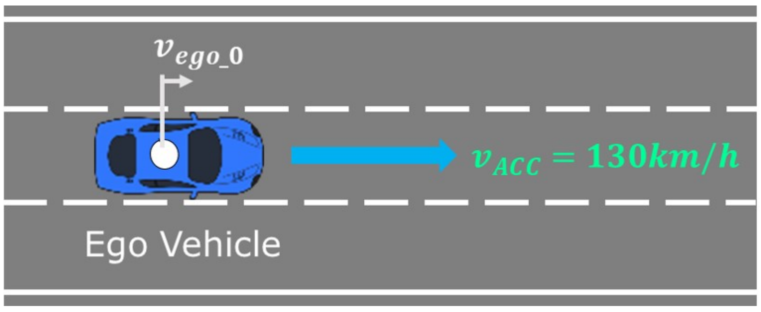


Figure 3.4: Scenario of ”reaching target speed” testing

Figure 3.4 shows the scenario of ”reaching target speed” testing. The ego vehicle starts with the speed of 90km/h and needs to reach the maximum speed 130km/h (MPACC set speed).

3.2.2 Key performance indicator

For this test we only consider whether the ego vehicle can reach the target speed. And the only KPI is the final speed of ego vehicle, KPI V_final.

3.2.3 Simulation and results

The cost function of the MPACC function is described with equation (2.25). For the co-simulation and calibration, the weights of the costs are redefined as variables, also called parameters in this thesis, see table 3.1.

Table 3.1: Definition of variables for co-simulation and calibration

Weight	c^{v1}	c^{a1}	c^{j1}	c^{h1}	c^{Tr}
Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Weight	c^{v2}	c^{a2}	c^{j2}	c^{h2}	
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	

Because the variables $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$ are directly related to the vehicle speed, in this thesis they are implemented firstly for the "reaching target speed" testing. The other variables are divided into three groups: $Cost\_Acc1$ and $Cost\_Acc2$, $Cost\_Jerk1$ and $Cost\_Jerk2$ as well as $Cost\_Headway1$ and $Cost\_Headway2$.

(1) $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$

Table 3.2 shows the definition of the input parameters $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$.

Table 3.2: Definition of parameters $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$

Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Range	-10 to 10	0	0	0	1 to 20
Step	0,5	/	/	/	0,5
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Range	0 to 1	0	0	0	0
Step	0,02	/	/	/	/

Number of simulation runs: 401(with one repetition point)

Figure 3.5 shows the distribution of the variations for input parameters $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$.

Figure 3.6 shows the simulation results. From the distribution of the reached variation points (circle points) and not reached points (asterisk points), three estimated linear boundaries are generated, see equation (3.1) to (3.3). These boundaries can be used as limits during the calibration with AVL CAMEO.

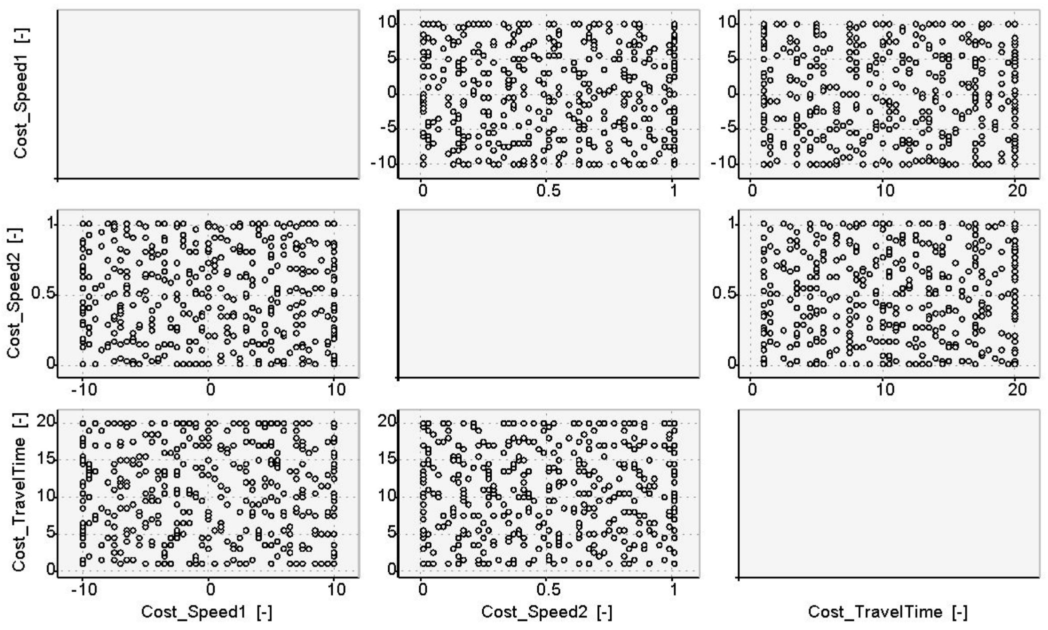


Figure 3.5: Distribution of the variations for input parameters *Cost_Speed*1, *Cost_Speed*2 and *Cost_TravelTime*

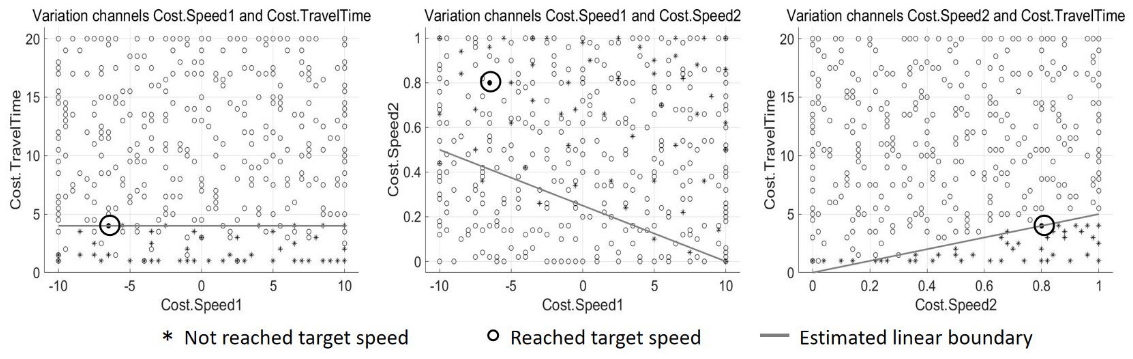


Figure 3.6: Distribution of testing results for input parameters *Cost_Speed*1, *Cost_Speed*2 and *Cost_TravelTime*

$$Cost\_Speed2 < -1/40 \cdot Cost\_Speed1 + 0,25 \tag{3.1}$$

$$Cost\_TravelTime > 4 \tag{3.2}$$

$$Cost\_TravelTime > 5 \cdot Cost\_Speed2 \tag{3.3}$$

In order to investigate the influences of other variables, two variation points for these three parameters are selected. One is $Cost\_Speed1 = -6,5$, $Cost\_Speed2 = 0,8$ and

$Cost\_TravelTime = 4$, this variation point is on the estimated linear boundary with the result of reached target speed, see figure 3.6. Another is $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$, which is used to test if the ego vehicle can always reach the target speed when $Cost\_TravelTime$ is greater than 5.

(2) $Cost\_Acc1$ and $Cost\_Acc2$

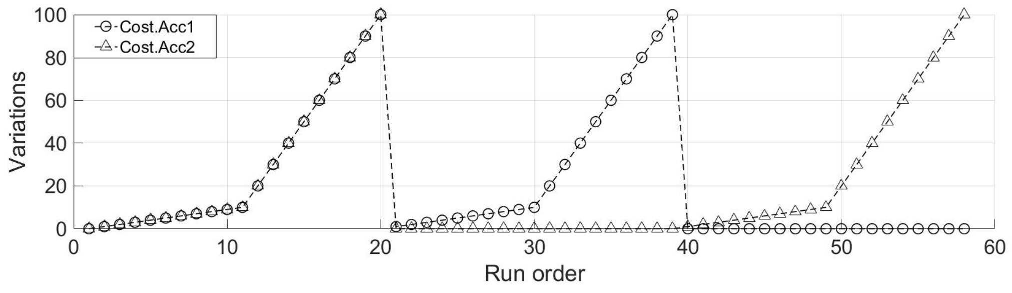
Table 3.3 shows the definition of input parameters $Cost\_Acc1$ and $Cost\_Acc2$.

Table 3.3: Definition of parameters $Cost\_Acc1$ and $Cost\_Acc2$

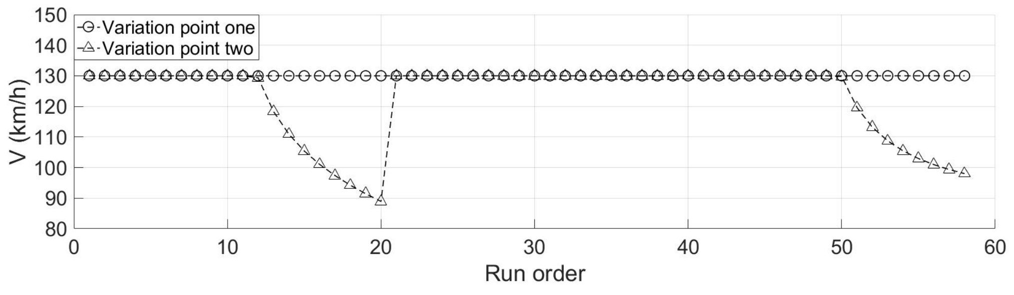
Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Range	10/-6,5	0 to 100	0	0	5 /4
Step	/	1 or 10	/	/	/
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Range	1/0,8	0 to 100	0	0	0
Step	/	1 or 10	/	/	/

Variation point one: $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$.

Variation point two: $Cost\_Speed1 = -6,5$, $Cost\_Speed2 = 0,8$ and $Cost\_TravelTime = 4$.



(a) Variation points



(b) Testing results

Figure 3.7: Variation points (a) and testing result (b) for parameters $Cost\_Acc1$ and $Cost\_Acc2$

Figure 3.7 shows the variation points (a) and corresponding simulation results (b) for parameters $Cost\_Acc1$ and $Cost\_Acc2$. For the variation point two with $Cost\_Speed1 = -6,5$, $Cost\_Speed2 = 0,8$ and $Cost\_TravelTime = 4$, the $Cost\_Acc2$ has obviously a much stronger negative effect to the maximum sped that the ego vehicle can reach. Compared to $Cost\_Acc2$, $Cost\_Acc1$ alone has no significant effect, but together with $Cost\_Acc2$ it also has negative effects. When the two variations are less than 20, the ego vehicle can always reach the target speed. For the variation point one with $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$, the ego vehicle can always reach the target speed.

(3) $Cost\_Jerk1$ and $Cost\_Jerk2$

Table 3.4 shows the definition of input parameters $Cost\_Jerk1$ and $Cost\_Jerk2$.

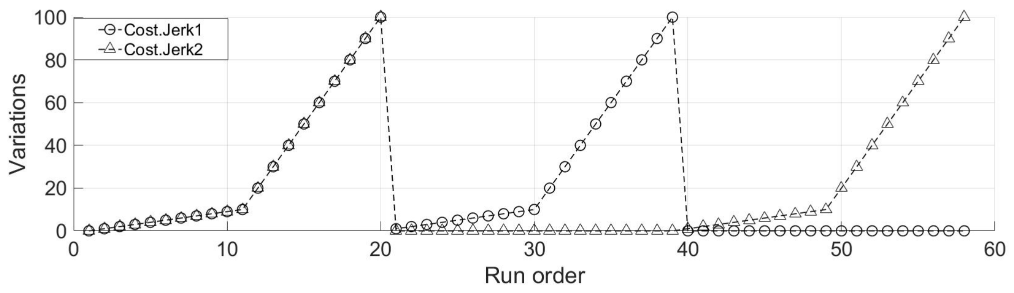
Table 3.4: Definition of parameters $Cost\_Jerk1$ and $Cost\_Jerk2$

Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Range	10/-6,5	0	0 to 100	0	5/4
Step	/	/	1 or 10	/	/
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Range	1 /0,8	0	0 to 100	0	0
Step	/	/	1 or 10	/	/

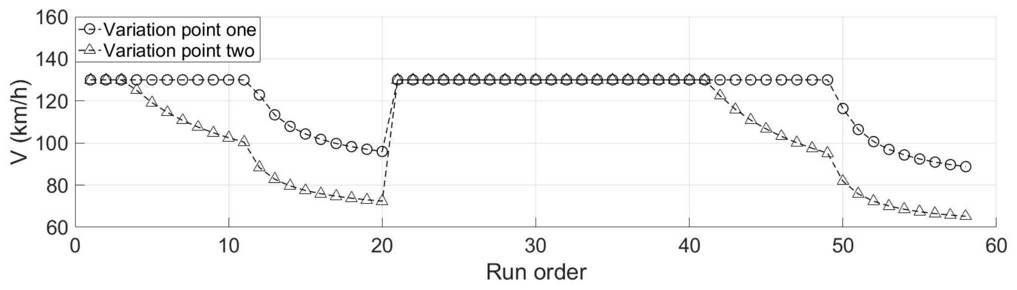
Variation point one: $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$.

Variation point two: $Cost\_Speed1 = -6,5$, $Cost\_Speed2 = 0,8$ and $Cost\_TravelTime = 4$.

Figure 3.8 shows the variation points (a) and the corresponding simulation results (b). The variation point two related to the input parameters $Cost\_Speed1 = -6,5$, $Cost\_Speed2 = 0,8$ and $Cost\_TravelTime = 4$ has obviously a much stronger negative effect to the maximum speed that the ego vehicle can reach than the variation point one related to the input parameters $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$. Compared to $Cost\_Jerk2$, $Cost\_Jerk1$ has a smaller, but positive effect. Only when the $Cost\_Jerk2$ is less than 3, the ego vehicle can reach the target speed. For the variation point one with $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$, the ego vehicle can always reach the target speed, when the two variations are less than 10.



(a) Variation points



(b) Testing results

Figure 3.8: Variation points (a) and testing result (b) for parameters $Cost\_Jerk1$ and $Cost\_Jerk2$

(4) $Cost\_Headway1$ and $Cost\_Headway2$

Table 3.5 shows the definition of input parameters $Cost\_Headway1$ and $Cost\_Headway2$.

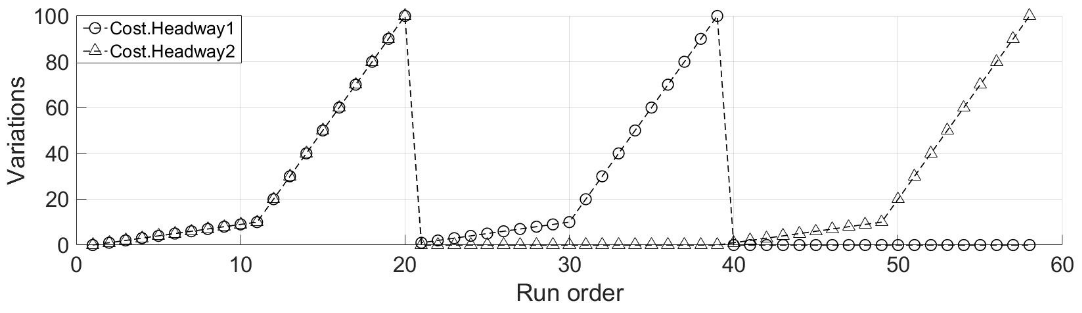
Table 3.5: Definition of parameters $Cost\_Headway1$ and $Cost\_Headway2$

Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Range	10/-6,5	0	0	0 to 100	5 /4
Step	/	/	/	1 or 10	/
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Range	1 /0,8	0	0	0 to 100	0
Step	/	/	/	1 or 10	/

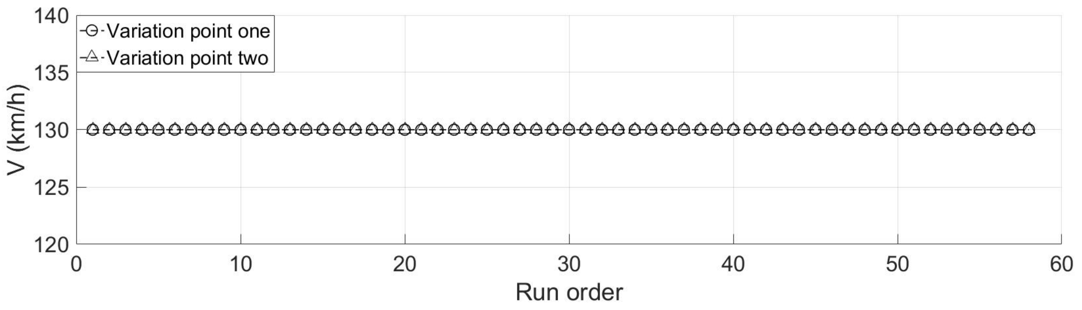
Variation point one: $Cost\_Speed1 = 10$, $Cost\_Speed2 = 1$ and $Cost\_TravelTime = 5$.

Variation point two: $Cost\_Speed1 = -6,5$, $Cost\_Speed2 = 0,8$ and $Cost\_TravelTime = 4$.

Figure 3.9 shows the variation list (a) and simulation results (b). It is obviously that the $Cost\_Headway1$ and $Cost\_Headway2$ have no significant influence for the maximum speed that the ego vehicle can reach.



(a) Variation points



(b) Testing results

Figure 3.9: Variation points (a) and testing results (b) for parameters $Cost\_Headway1$ and $Cost\_Headway2$

Based on these testing results, the limits are defined as below:

$$Cost\_TravelTime > 5 \tag{3.4}$$

$$Cost\_TravelTime > 5 \cdot Cost\_Speed2 \tag{3.5}$$

3.3 MPACC controller calibration

The functionality of the MPACC controller is to optimize the acceleration request for the ego vehicle based on the input signals. After the PI controller optimization and "reaching target speed" testing, the MPACC controller is calibrated with the target of fuel consumption reduction and comfort improvement.

3.3.1 Scenario definition

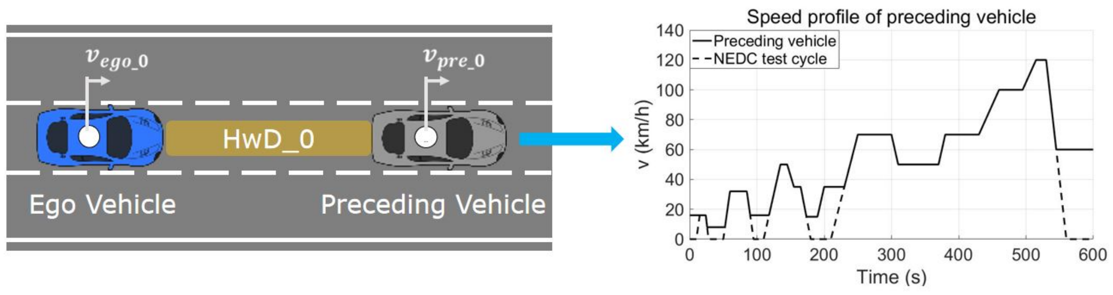


Figure 3.10: Scenario of MPACC controller calibration: left) Scenario selection, right) Reference maneuver of preceding vehicle

Figure 3.10 left) shows the scenario applied for the MPACC controller calibration. It contains of an ego vehicle following the preceding vehicle with a predefined speed profile, see figure 3.10 right). The speed profile of preceding vehicle is modified based on the NEDC (New European Driving Cycle) [36] test cycle speed profile. At time 0, the initial speed of ego vehicle and preceding vehicle are set to $v_{\text{ego\_0}} = v_{\text{pre\_0}} = 16\text{km/h}$ and initial headway distance to $HwD_0 = 20\text{m}$.

3.3.2 Key performance indicator

In order to achieve the calibration targets, fuel-efficiency, comfort and safety, the following KPIs are defined:

- Fuel efficiency - The MPACC function should control the ego vehicle to follow the preceding vehicle without heavy braking or acceleration, in order to reduce the average fuel consumption (l/100km) during the whole cruise control phase. A baseline is defined, where the ego vehicle equipped with the MPACC function under the base setting before calibration. The KPI $KpiFuelConsumpAvg$ (average fuel consumption in l/100km) is defined for evaluation of the fuel efficiency of the vehicle [37].
- Comfort of driver and passengers - The MPACC function should maximize the comfort by reducing the jerk (the derivative of acceleration), which is defined as variable j, see equation (3.6). Because the change of the force would lead to the longitudinal vibration of the vehicle, which would make the driver and passengers uncomfortable [38]. So the ego vehicle would be controlled to follow the preceding vehicle without sudden change of deceleration and acceleration, in order to achieve the smooth ride. The KPI $KpiJerkIntegral$ (integration of the absolute value of jerk) is defined [37].

$$j = \frac{\mathrm{d}a}{\mathrm{d}t} \tag{3.6}$$

where a is the acceleration of ego vehicle

- Safety - The MPACC controller should control the ego vehicle not to crash or be too close to the preceding vehicle. In section 2.5.3, we have defined hard and soft constraints for the minimum headway distance. The hard limit should not be violated whereas the soft limit can be violated with cost. In order to take the ego vehicle speed into consideration, the minimum headway time, see equation 3.7), is also defined as the KPI for safety. So the KPIs $KpiHeadwayDistanceMin$ (minimum headway time) and $KpiHeadwayTimeMin$ (minimum headway distance) are defined. Figure 3.11 illustrates the definition of headway distance [37].

$$\text{headway time} = \text{headway distance}/v_{\text{ego}} \tag{3.7}$$

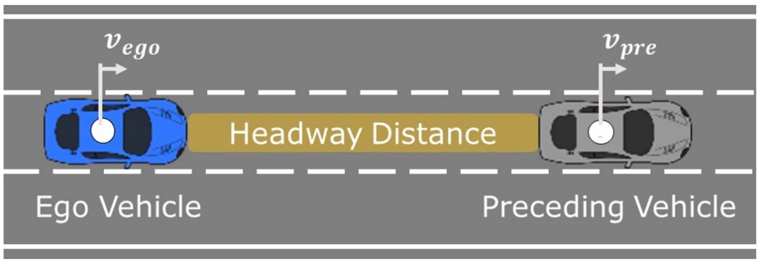


Figure 3.11: Definition of headway distance

3.3.3 Correlation analysis

We do not have a priori knowledge about the behavior of the MPACC controller. In total there are 9 interdependent cost parameters (see section 2.5.3), so called weights of cost function, that need to be calibrated. Even when applying DoE methods for defining the inputs, it is still too complicated to calibrate all the parameters of the function at the same time.

To get a quick overview to quantify the influence of the parameters affecting the behavior of the ego vehicle, a correlation analysis algorithm is applied. As described in section 2.4.4, the sensitivity analysis algorithm in this master thesis analyses the reduction of the SSE by an input parameter on a selected response, in order to get the relative significance indicator (RSI) for each parameter.

Figure 3.12 shows the results of the parameter correlation analysis. The relative significance indicator of the 9 cost parameters regarding the four predefined KPIs are evaluated for the correlation between cost parameters and responses. For the KPI $KpiFuelConsumpAvg$, the parameter $Cost\_TravelTime$, has the strongest influences.

For the KPI $KpiJerkIntegral$, the parameters $Cost\_Jerk1$, $Cost\_Speed2$ and $Cost\_Jerk2$. For the KPIs $KpiHeadwayDistanceMin$ and $KpiHeadwayTimeMin$, the parameters $Cost\_Jerk2$ and $Cost\_Speed2$. So the parameters $Cost\_TravelTime$, $Cost\_Speed1$, $Cost\_Speed2$, $Cost\_Jerk1$ and $Cost\_Jerk2$ have the priority to calibration. The other four parameters should be calibrated after the first five parameters.

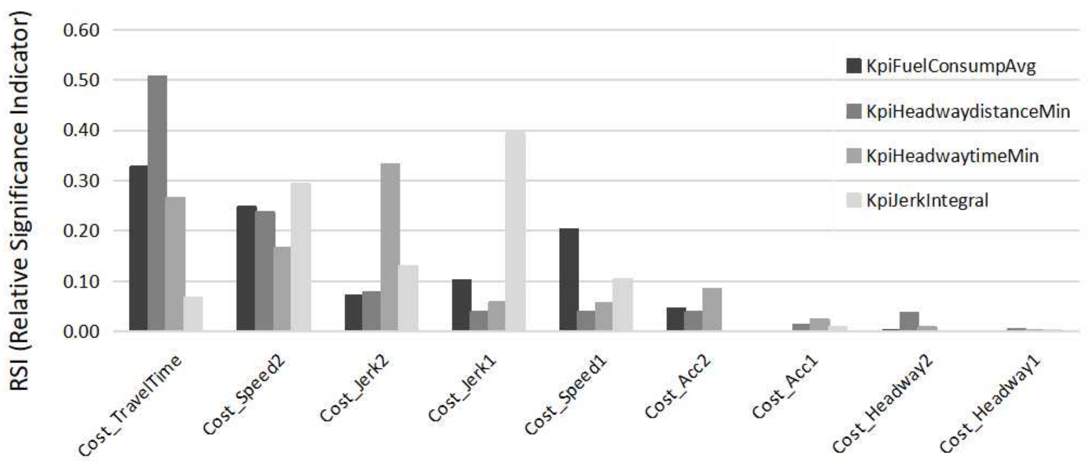


Figure 3.12: Correlation analysis

3.3.4 Simulation results with conventional DoE

In this section, the calibration process is based on conventional DoE, see section 2.3.3. At time $t = 0$, with the initial velocity of ego vehicle and preceding vehicle $v_{\text{ego\_0}} = v_{\text{pre\_0}} = 16\text{km/h}$ and initial headway distance $HwD_0 = 20\text{m}$ the simulation runs are prepared. In order to guarantee stable initial conditions, the ego vehicle should keep the speed 16km/h for 25s, see figure 3.10. Using the limit described in section 3.2.3, see equations (3.4) and (3.5), the following settings are applied for the DoE:

(1) Simulation set-up and execution

As the first step, table 3.6 shows the definition of input parameters $Cost\_Jerk1$, $Cost\_Jerk2$, $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$ for the conventional DoE.

Table 3.6: Definition of parameters $Cost\_Jerk1$, $Cost\_Jerk2$, $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$ (Conventional DoE)

Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Range	-10 to 0	0	0 to 5	0	4 to 10
Step	0,2	/	0,1	/	0,1
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Range	0 to 1	0	5 to 20	0	0
Step	0,02	/	0,3	/	/

Number of simulation runs: 206 (with one repetition point)

Figure 3.13 show the distribution of the variations for conventional DoE. The design space is filled with the variation points, especially in the corner of the design space are filled with more variation points.

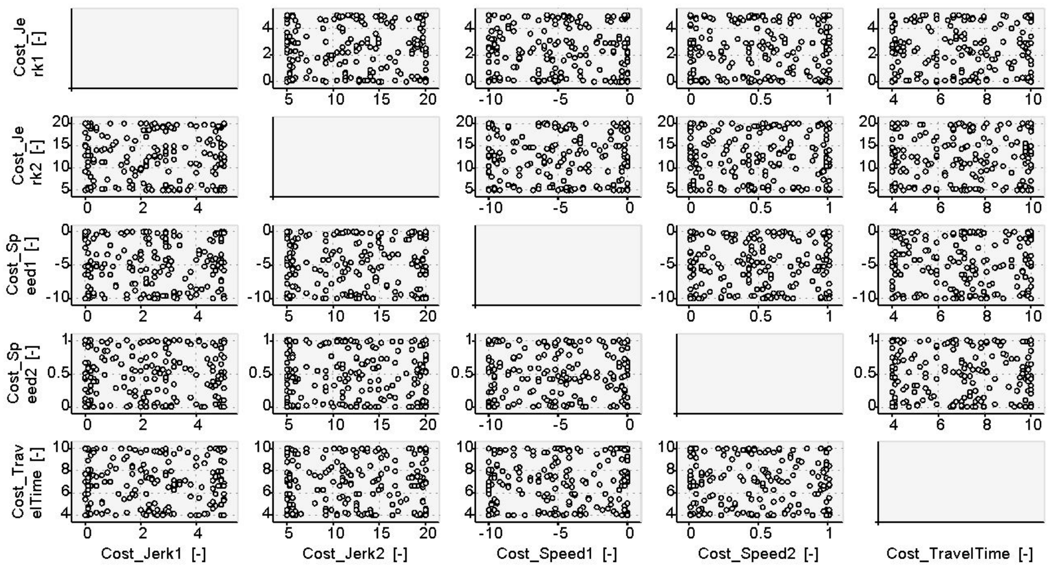


Figure 3.13: Distribution of variations (Conventional DoE)

(2) Row data processing - Outliers detection and deactivation

In the next step, outliers are detected using the method described in the section 2.4.1. Figure 3.14 shows the results of the simulations using the conventional DoE method. The outliers of the measurements are detected, as shown in Figure 3.15. Using the method described in section 2.4.1, the points, which is not in the range of $\bar{y} \pm 3s$ are recognized as outliers, see equation 2.3. The points not in the range of $\bar{y} \pm 2s$ are treated as suspected outliers. These outliers can be deactivated while modeling, in order to improve the model quality.

For the measurements *KpiHeadwayDistanceMin* and *KpiHeadwayTimeMin* there are no outliers existing. For each of the measurements *KpiFuelConsumpAvg* and *KpiJerkIntegral*, there is one outlier for each. Before modeling, the outliers are deactivated.

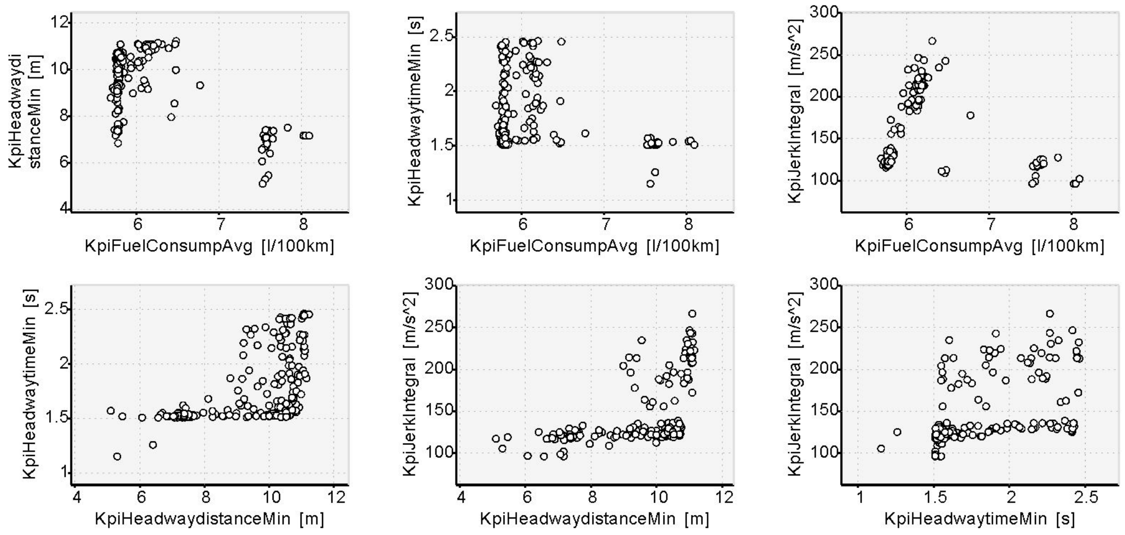


Figure 3.14: Measurements (Conventional DoE)

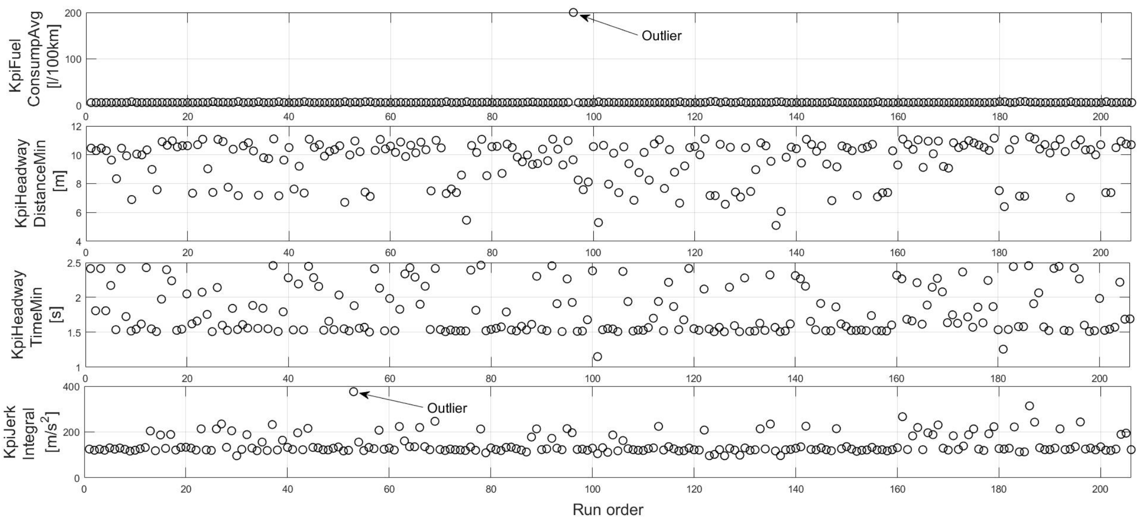
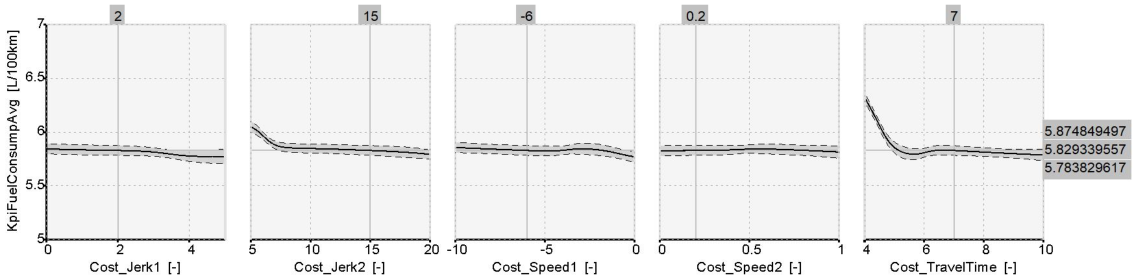


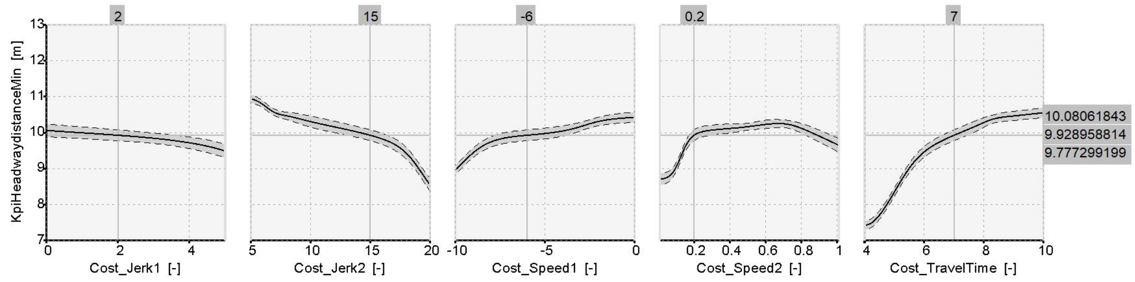
Figure 3.15: Outliers detection and deactivation (Conventional DoE)

(3) Behavior modeling

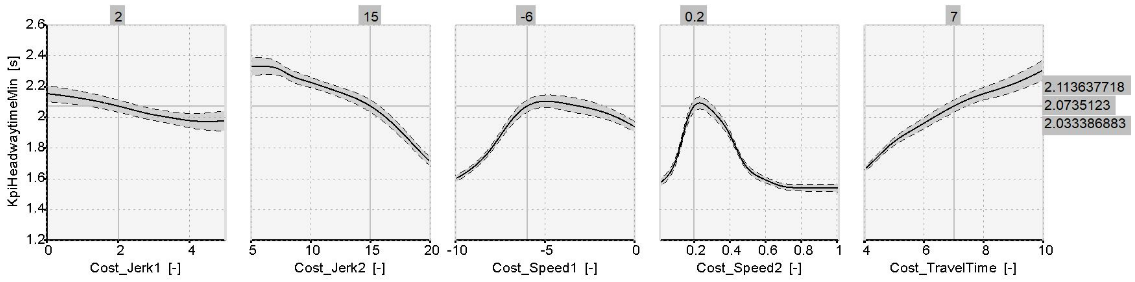
The third step describes the modeling of KPI behavior models with the calibrated parameters as inputs. Figure 3.16 shows the intersection graphics of the behavior model with 95% confidence level for the KPIs at certain variation point, see table 3.7.



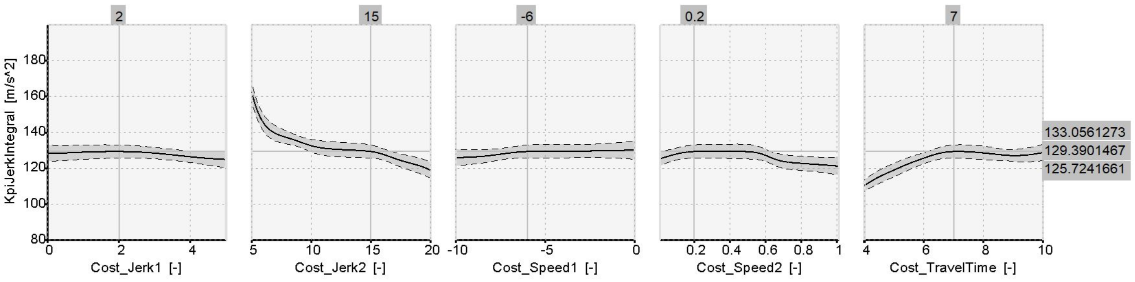
(a) KPI *KpiFuelConsumpAvg*



(b) KPI *KpiHeadwayDistanceMin*



(c) KPI *KpiHeadwayTimeMin*



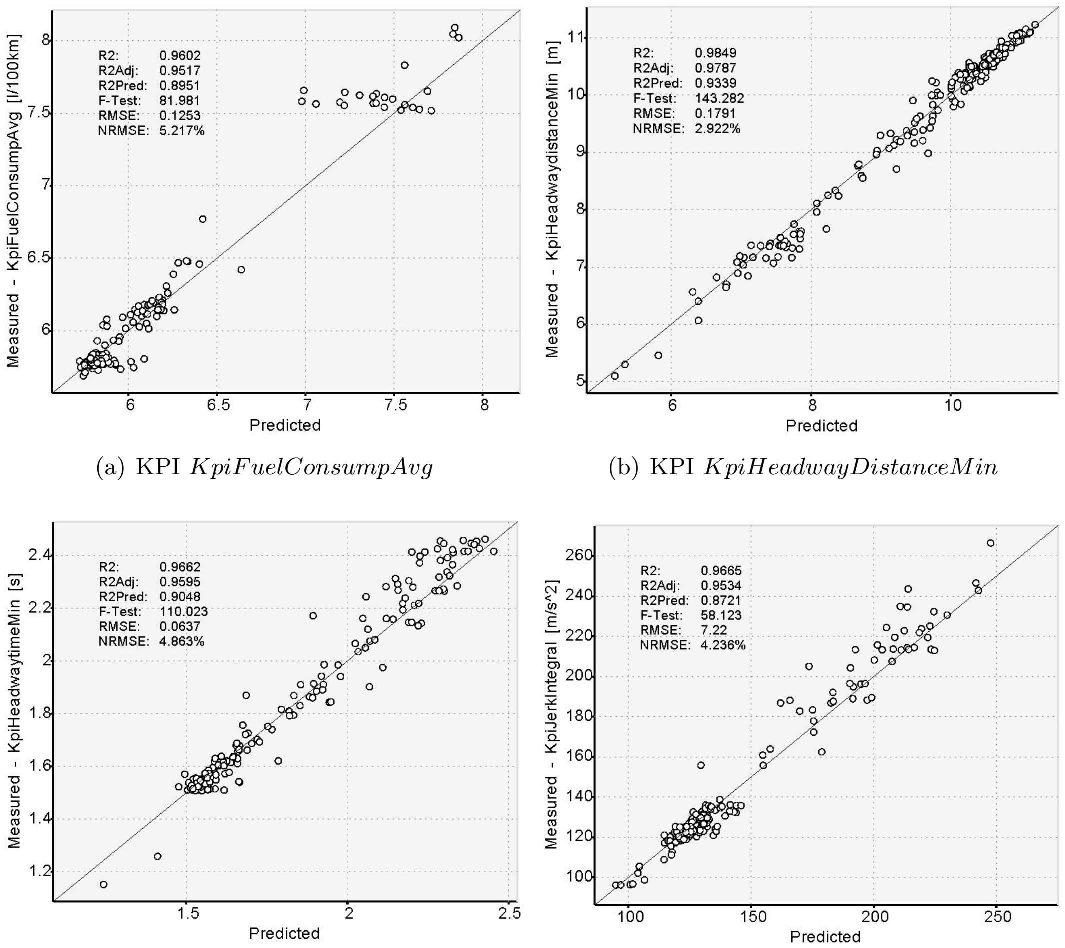
(d) KPI *KpiJerkIntegral*

Figure 3.16: Intersection graphics of behavior model for KPIs (Conventional DoE)

Table 3.7: The parameter values of the variation point for intersection graphics (Conventional DoE)

Variable	*Cost_Speed*1	*Cost_Acc*1	*Cost_Jerk*1	*Cost_Headway*1	*Cost_TravelTime*
Value	-6	0	3	0	7
Variable	*Cost_Speed*2	*Cost_Acc*2	*Cost_Jerk*2	*Cost_Headway*2	
Value	0,2	0	15	0	0

In figure 3.16, the solid lines are the behavior models and the gray zones with broken line boundaries are the confidence interval with 95% confidence level, which means the real KPI behavior lies in the confidence interval with 95% possibility.



(a) KPI *KpiFuelConsumpAvg*

(b) KPI *KpiHeadwayDistanceMin*

(c) KPI *KpiHeadwayTimeMin*

(d) KPI *KpiJerkIntegral*

Figure 3.17: Predicted value of KPIs (Conventional DoE)

Figure 3.17 shows the predicted values of the KPIs calculated with the KPI behavior models. For a perfectly correlated good model, the point with x-axis predicted value and y-axis measured value should be located on the center line. The model quality is evaluated by the statistical coefficients R^2, R^2_{adj} and R^2_{pred}, see section 2.4.3. The quality of these KPI behavior models is discussed in section 4.1.

(4) Optimization

The final step is optimization with the KPI behavior model under certain targets and constraints. The minimization of KPIs $KpiFuelConsumpAvg$ and $KpiJerkIntegral$ are defined as the optimization target functions. At the same time, the following constraints are also defined before the optimization:

- $KpiHeadwayDistanceMin$: Lower limit is 5m for safety
- $KpiHeadwayTimeMin$: Lower limit is $0,5$s for safety
- $KpiFuelConsumpAvg$: Upper limit is $6,0$l/100km for fuel efficiency
- $KpiJerkIntegral$: Upper limit is 150m/s^2 for comfort

Figure 3.18 shows the optimization results with Pareto points, which are the potential optimization points. These points form the Pareto front in "the area of interest", which is shaped by the optimization targets and constraints. The random points are generated based on the behavior model, in order to fill the design space. The random points, which are out of "the area of interest", are treated as unfeasible random points. Table 3.8 shows the parameter values of the start point with base settings. Table 3.9 shows the parameter values of the chosen optimization point with conventional DoE method.

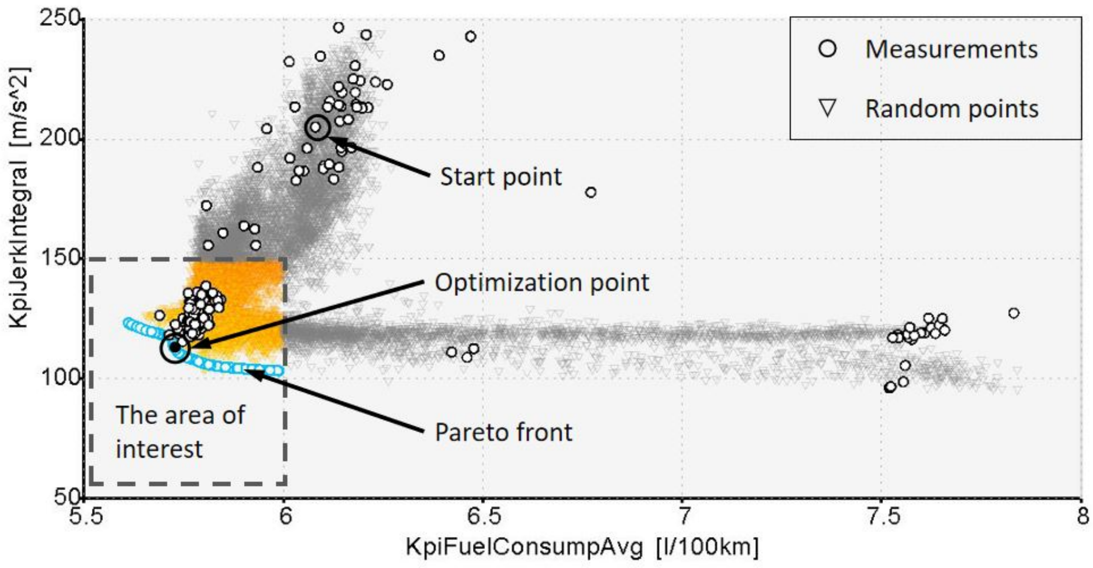


Figure 3.18: Results of the optimization (Conventional DoE)

Table 3.8: The values of parameters for the start point

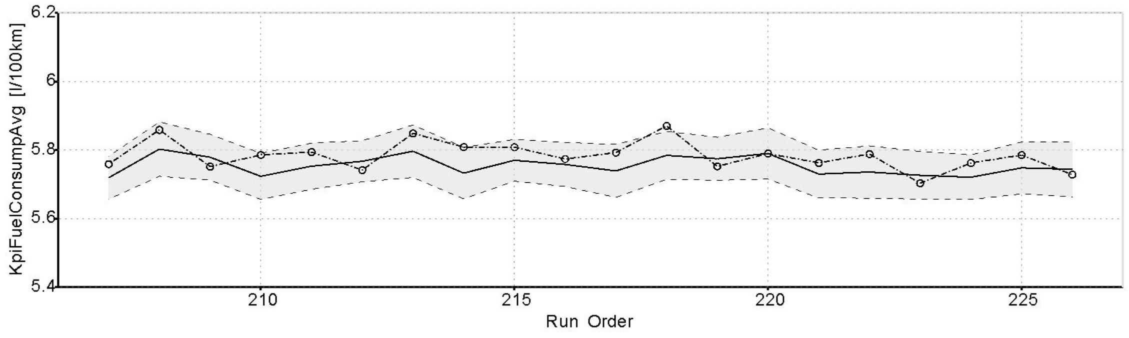
Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Value	-5	5	5	5	5
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Value	0,5	5	5	5	

Table 3.9: The values of parameters for the optimization point (Conventional DoE)

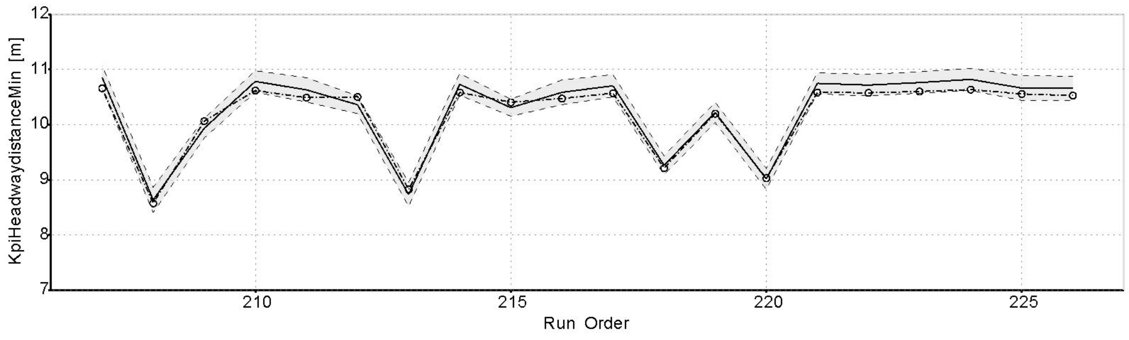
Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Value	-6,26	0	3,72	0	7,42
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Value	0,25	0	20	0	

(5) Verification

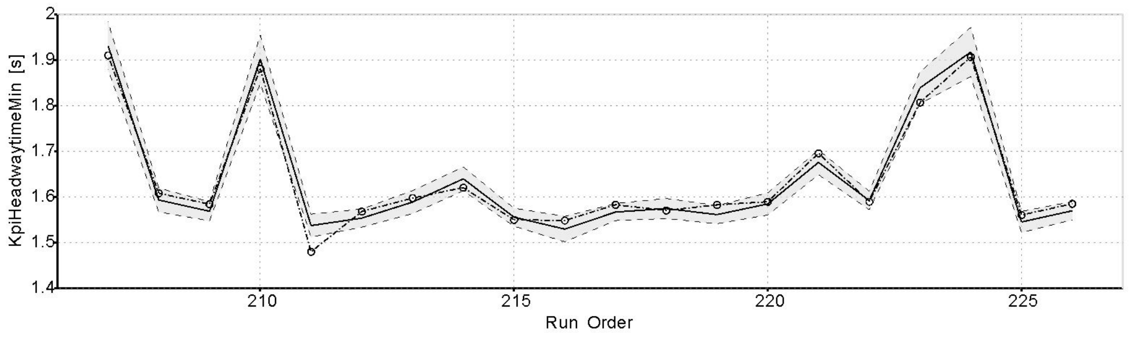
In order to the verify the KPI behavior models, there are 20 Pareto points selected and simulated again using the same scenario. The measurements are then compared with the predicted values. Figure 3.19 illustrates the results of the verification for conventional DoE method. In section 4.1, the verification results are discussed.



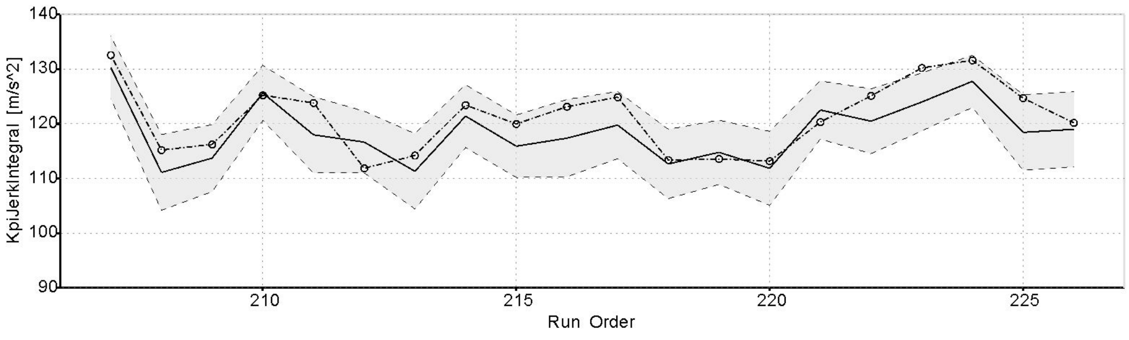
(a) KPI "KpiFuelConsumpAvg"



(b) KPI "KpiHeadwayDistanceMin"



(c) KPI "KpiHeadwayTimeMin"



(d) KPI "KpiJerkIntegral"

Figure 3.19: Verification with selected Pareto points (Conventional DoE)

3.3.5 Simulation results with active DoE

In this section, the calibration process is based on active DoE, see section 2.3.4. With the same initial settings as conventional DoE. At time $t = 0$, $v_{\text{ego\_0}} = v_{\text{pre\_0}} = 16\text{km/h}$ and $HwD_0 = 20\text{m}$ the simulation is prepared, see figure 3.10.

(1) Simulation set-up and execution

As the first step, for the active DoE method, the definition of input parameters $Cost\_Jerk1$, $Cost\_Jerk2$, $Cost\_Speed1$, $Cost\_Speed2$ and $Cost\_TravelTime$ is same as conventional DoE, see table 3.6 and figure 3.13. But following is the differences:

Maximum number of iterations is 99, which is used to optimize the behavior model online after the conventional DoE points are executed, see section 2.3.4.

The targets for the optimization of the KPIs $KpiFuelConsumpAvg$, $KpiJerkIntegral$ and $KpiTargetLost$ using the active DoE method are listed in table 3.10. The output ranges of these three KPIs form ”the area of interest”. The active DoE type ”Minimize” means that the predicted value of KPI for the next iteration point is smaller than last iteration point. The active DoE type ”ModelLimit” means that the predicted value of KPI for the next iteration point just need to be in the range of ”Minimum Output” and ”Maximum Output”

Table 3.10: Active DoE targets

KPI	Minimum Output	Maximum Output	Active DoE Type
KpiFuelConsumpAvg	0	6	Minimize
KpiJerkIntegral	0	150	Minimize
KpiTargetLost	-0,1	0,4	ModelLimit

(2) Row data processing - Outliers detection and deactivation

In the next step outliers are detected using the method described in the section 2.4.1. Figure 3.20 shows the measurements and figure 3.21 shows the results of the outliers detection using the active DoE method.As we can see in figure 3.21, for the measurements $KpiHeadwayDistanceMin$ and $KpiHeadwayTimeMin$ there are no outliers existing. For each of the measurements $KpiFuelConsumpAvg$ and $KpiJerkIntegral$, there is one outlier for each. Before modeling, the outliers are deactivated.

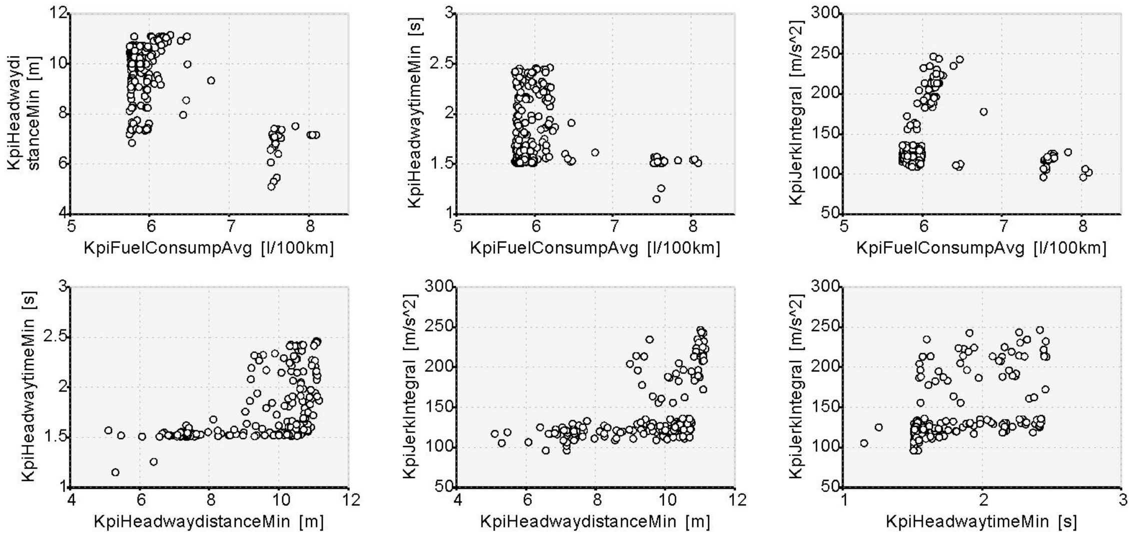


Figure 3.20: Measurements (Active DoE)

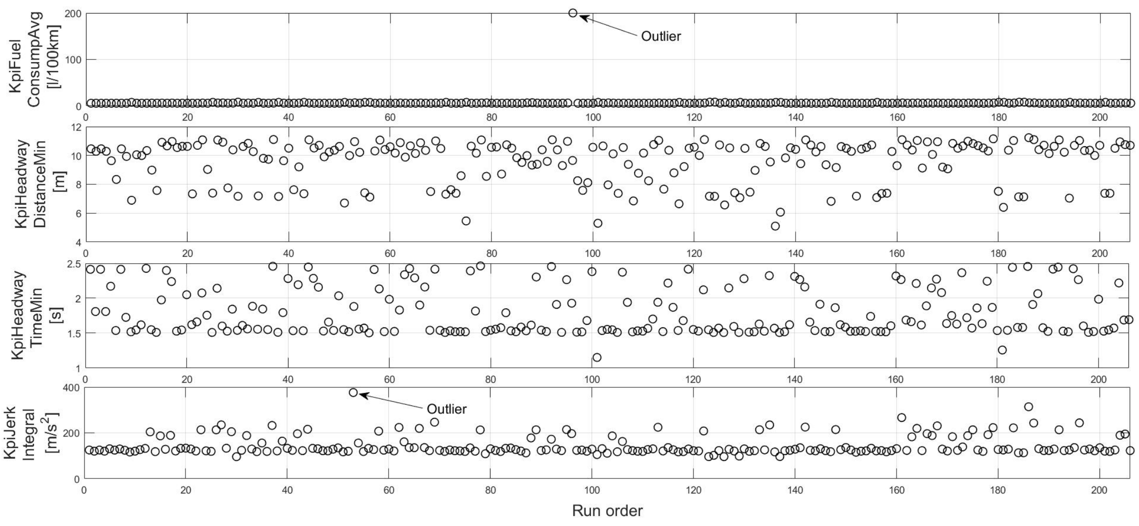
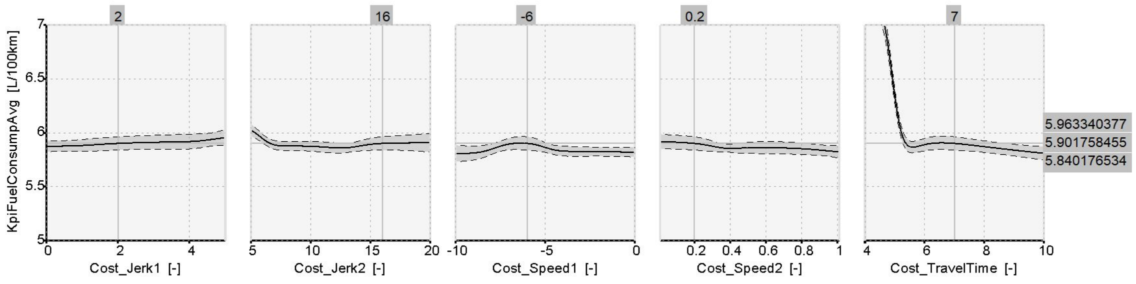


Figure 3.21: Outliers detection and deactivation (Active DoE)

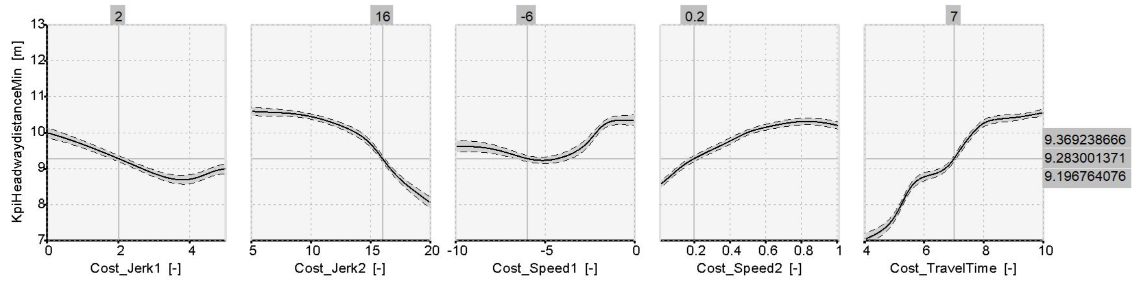
(3) Behavior modeling

The third step describes the modeling of KPI behavior models with the calibrated parameters as inputs. Figure 3.22 shows the intersection graphics of the behavior model with 95% confidence level for the KPIs with Active DoE method.

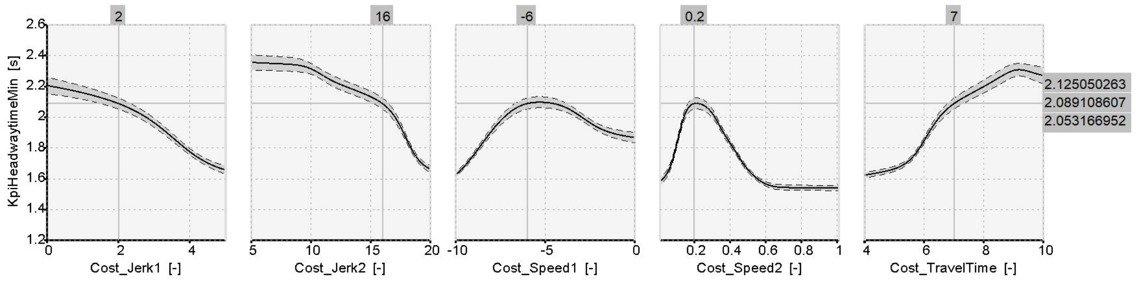
Figure 3.23 shows the predicted values of the measurements based on the behavior models with active DoE method. The quality of these KPI behavior models is discussed in section 4.1.



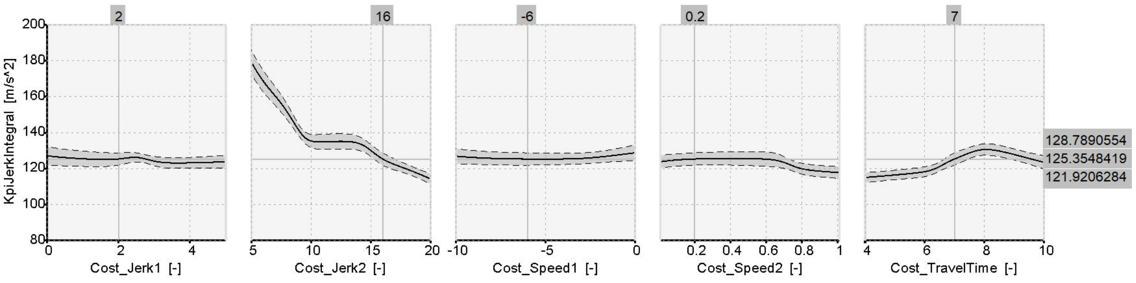
(a) KPI $KpiFuelConsumpAvg$



(b) KPI $KpiHeadwayDistanceMin$

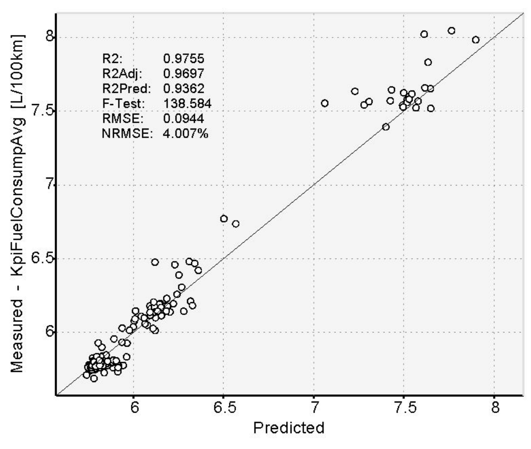


(c) KPI $KpiHeadwayTimeMin$

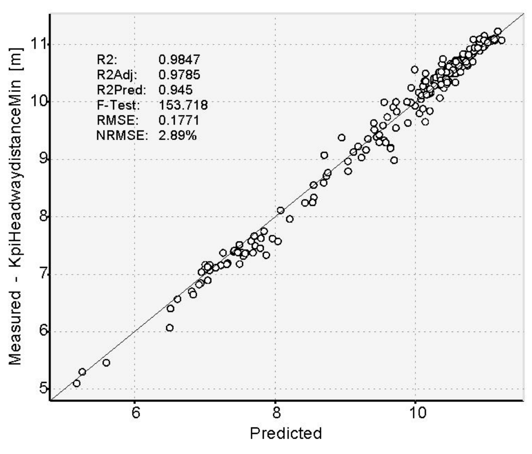


(d) KPI $KpiJerkIntegral$

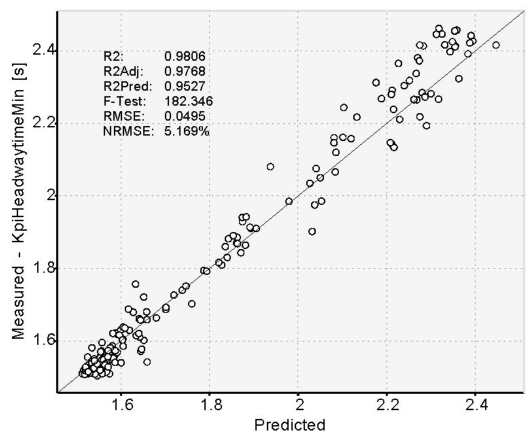
Figure 3.22: Intersection graphics of behavior model for KPIs (Active DoE)



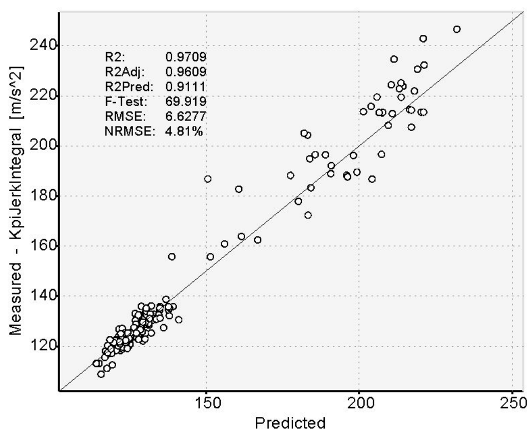
(a) KPI *KpiFuelConsumpAvg*



(b) KPI *KpiHeadwayDistanceMin*



(c) KPI *KpiHeadwayTimeMin*



(d) KPI *KpiJerkIntegral*

Figure 3.23: Predicted value of KPIs (Active DoE)

(4) Optimization

The final step is optimization with the KPI behavior model under certain targets and constraints. The optimization target functions and constraints are the same as conventional DoE, see section 3.3.4. The *KpiFuelConsumpAvg* and *KpiJerkIntegral* need to be minimized under certain the following constraints.

- *KpiHeadwayDistanceMin*: Lower limit is 5m for safety
- *KpiHeadwayTimeMin*: Lower limit is 0, 5s for safety
- *KpiFuelConsumpAvg*: Upper limit is 6, 0l/100km for fuel efficiency
- *KpiJerkIntegral*: Upper limit is 150m/s^2 for comfort

Figure 3.24 shows the optimization results with active DoE. Compared to the conventional DoE method, there are more measurements in ”the area of interest” to optimize the behavior model. Table 3.11 shows the parameter values of the chosen optimization point with conventional DoE method.

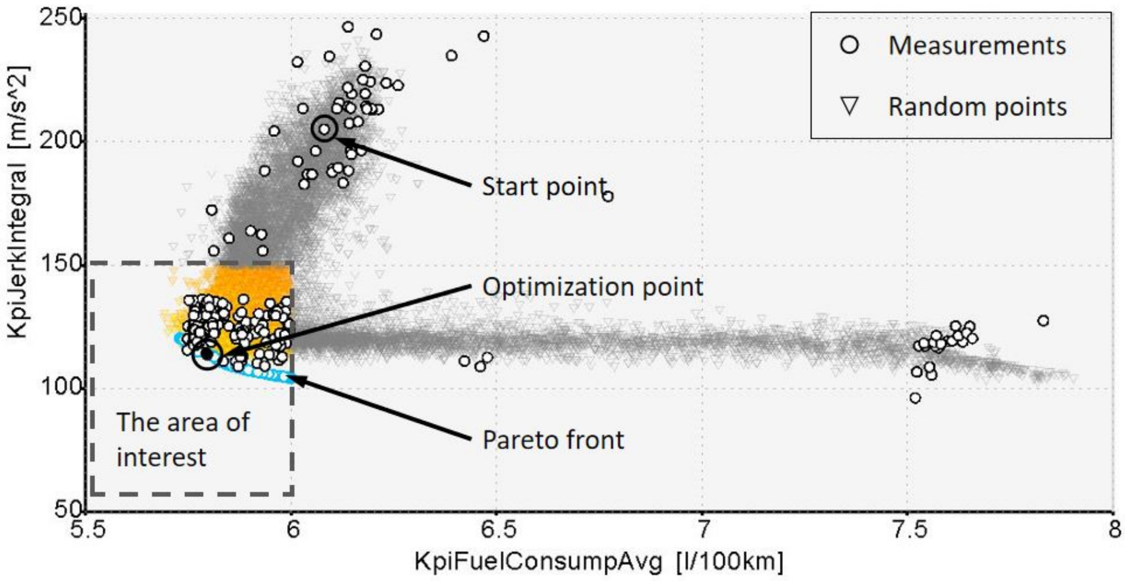


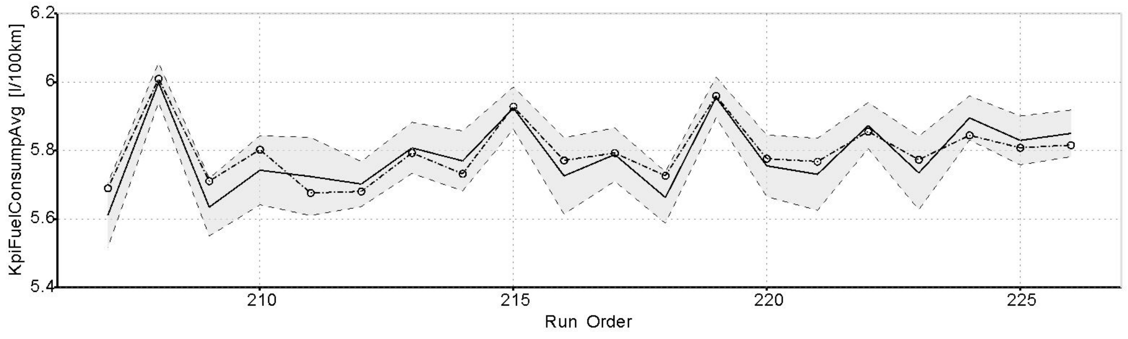
Figure 3.24: Results of the optimization (Active DoE)

Table 3.11: The values of parameters for the optimization point (Active DoE)

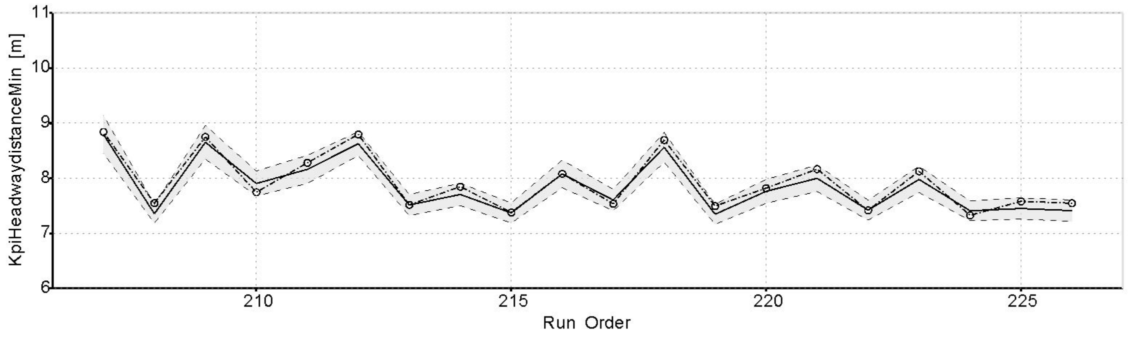
Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Value	-7,28	0	4,69	0	8,22
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Value	0,21	0	20	0	

(5) Verification

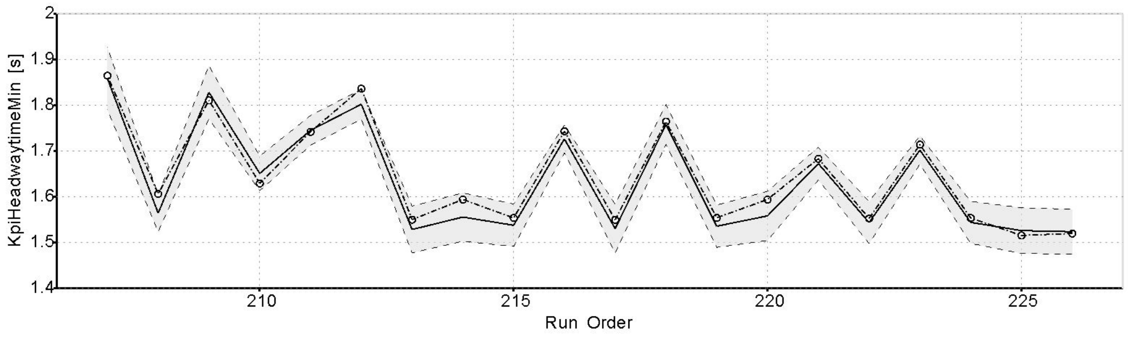
20 Pareto points are selected for the model verification. Figure 3.25 illustrates the results of the verification for active DoE method. In section 4.1, the verification results are discussed. Final, for the five most significant parameters, the values in table 3.11 are taken as the calibration result.



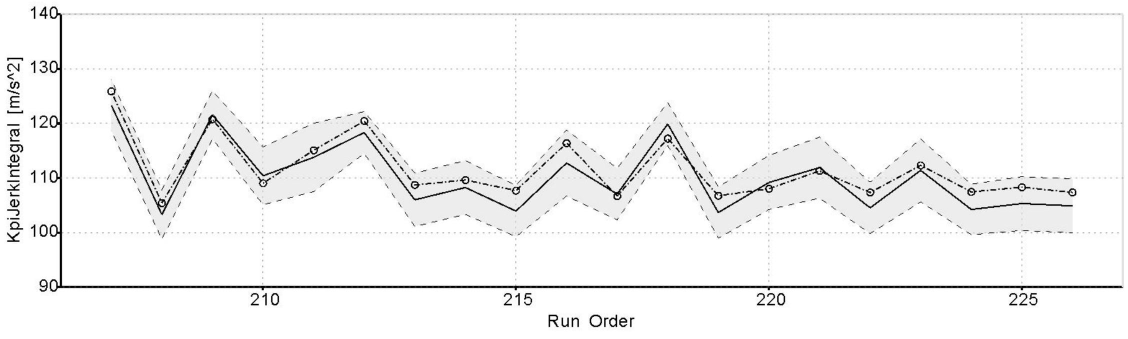
(a) KPI *KpiFuelConsumpAvg*



(b) KPI *KpiHeadwayDistanceMin*



(c) KPI *KpiHeadwayTimeMin*



(d) KPI *KpiJerkIntegral*

Figure 3.25: Verification with selected Pareto points (Active DoE)

3.3.6 Calibration with the less significant parameters

Based on the optimization result of the first five most significant variations, the rest four variations ($Cost\_Acc1$, $Cost\_Acc2$, $Cost\_Headway1$ and $Cost\_Headway2$) are calibrated in another step.

(1)Simulation set-up and execution

As the first step, table shows the definition of input parameters $Cost\_Acc1$, $Cost\_Acc2$, $Cost\_Headway1$ and $Cost\_Headway2$.

Table 3.12: Definition of parameters $Cost\_Acc1$, $Cost\_Acc2$, $Cost\_Headway1$ and $Cost\_Headway2$

Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Range	-7,28	0 to 10	4,69	0 to 10	8,22
Step	/	0,5	/	0,5	/
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Range	0,21	0 to 10	20	0 to 10	
Step	/	0,5	/	0,5	

Number of simulations: 131 (with one repetition point)

Figure 3.26 show the distribution of the variations.

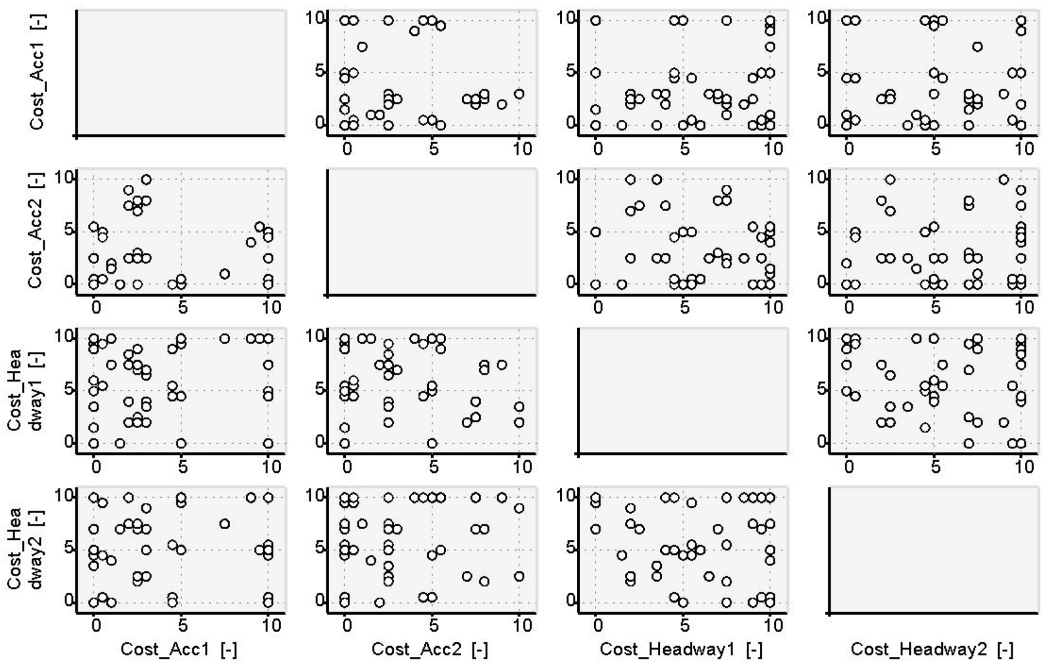


Figure 3.26: Distribution of variations (less significant parameters)

(2) Measurements

Figure 3.27 shows the measurements of the simulation runs.

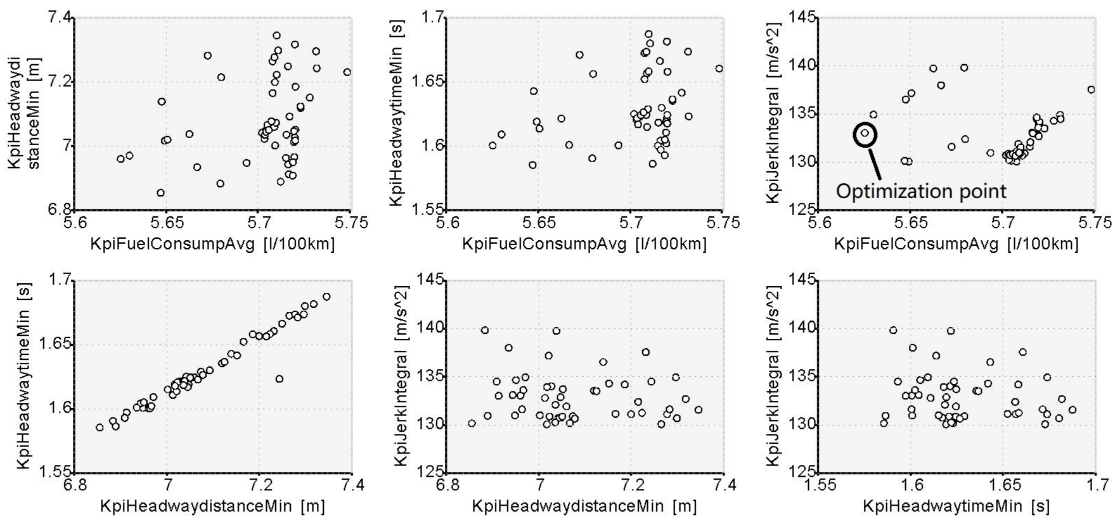


Figure 3.27: Measurements (less significant parameters)

From the measurements, these four KPIs just focus in a narrow range, so we just manually select the optimization point as the result, see figure 3.27. And combining with the calibration result, see table 3.13 :

Table 3.13: The values of parameters for the optimization point

Variable	$Cost\_Speed1$	$Cost\_Acc1$	$Cost\_Jerk1$	$Cost\_Headway1$	$Cost\_TravelTime$
Value	-7,28	10	4,69	10	8,22
Variable	$Cost\_Speed2$	$Cost\_Acc2$	$Cost\_Jerk2$	$Cost\_Headway2$	
Value	0,21	0	20	10	

3.4 Validation

After the virtual calibration with KPI-model-based method, the MPACC function is validated with the optimization point, see table 3.13.

3.4.1 Scenario definition

With the same scenario, that ego vehicle following the preceding vehicle with certain speed profile, see figure 3.10. But the speed profile of preceding vehicle is modified based on the WLTP (Worldwide harmonized Light vehicles Test Procedure) [39] test cycle.

Figure 3.28 illustrates the speed profile of preceding vehicle, if the speed in WLTP test cycle is lower than 20km/h, it would be raised to 20km/h for preceding vehicle. At time $t = 0$, the initial velocity of ego vehicle and preceding vehicle $v_{\text{ego\_0}} = v_{\text{pre\_0}} = 20\text{km/h}$ and initial headway distance $HwD_0 = 20\text{m}$.

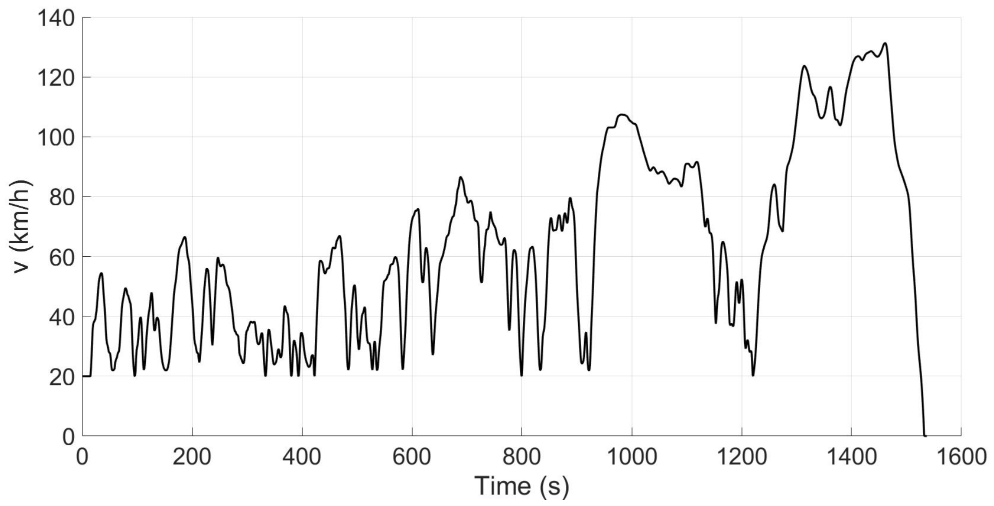


Figure 3.28: Speed profile of preceding vehicle for validation

3.4.2 Key performance indicator

For the validation, the KPIs are the same as calibration, see section 3.3.2.

3.4.3 Validation results

The simulation results of the MPACC function with the optimization point are compared with the MPACC function with base settings before the calibration. The results are discussed in section 4.2.1 to section 4.2.3.

4 Results and discussion

The objective of this master thesis is to preform and demonstrate a method for the virtual calibration of the MPACC function regarding fuel efficiency, comfort and safety. In the previous chapters, the set-up of the project, the method as well as co-simulation environment for the calibration and the process of the virtual calibration are explained.

In this chapter, the results of the virtual calibration are discussed. The performances of the calibrated MPACC function with respect to the fuel efficiency, comfort and safety are compared with the MPACC function before the calibration with base settings.

4.1 DoE methods

Figure 3.17 and figure 3.23 illustrate the predicted values of the measurements based on the behavior models for conventional DoE and active DoE methods. The goodness of fit is evaluated by means of the statistical coefficients R^2, R^2_{adj} and R^2_{adj} (see section 2.4.3, table 2.1), the goodness of the behavior models is shown in table 4.1.

Table 4.1: Goodness of fit for behavior models with normal DoE and active DoE

DoE type	KPI	R^2	R^2_{adj}	R^2_{adj}	Goodness
Conventional DoE	$KpiFuelConsumpAvg$	0,96	0,95	0,89	Good
	$KpiHeadwayDistanceMin$	0,98	0,97	0,93	Very good
	$KpiHeadwayTimeMin$	0,96	0,95	0,90	Very good
	$KpiJerkIntegral$	0,96	0,96	0,87	Good
Active DoE	$KpiFuelConsumpAvg$	0,98	0,97	0,94	Very good
	$KpiHeadwayDistanceMin$	0,98	0,98	0,95	Very good
	$KpiHeadwayTimeMin$	0,98	0,98	0,95	Very good
	$KpiJerkIntegral$	0,97	0,96	0,91	Very good

The behavior models of KPIs with conventional DoE are at least good, especially for KPIs $KpiHeadwayDistanceMin$ and $KpiHeadwayTimeMin$, they are very good. Compared to conventional DoE the behavior models with active DoE are even better, all the four KPI behavior models are very good. Figure 4.1 shows the distribution of the measurements in "the area of interest", for KPI $KpiFuelConsumpAvg$ under $6,0$l/100km and for KPI $KpiJerkIntegral$ under 150m/s^2. For the active DoE method, the more variation points distributed in "the area of interest", which would have a better fit of the behavior model, especially for the KPI $KpiFuelConsumpAvg$ from $5,8 - 6,0$l/100km.

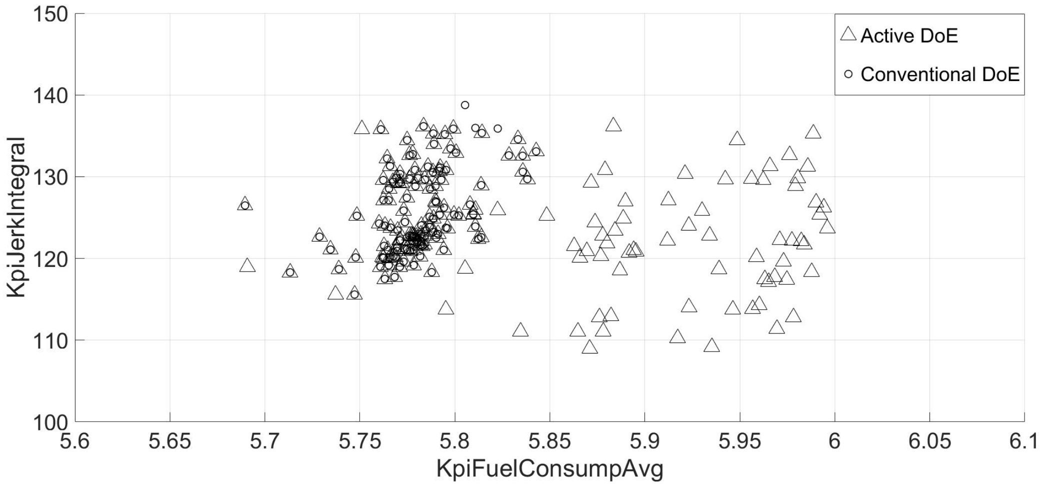


Figure 4.1: Distribution of the measurements in "the area of interest"

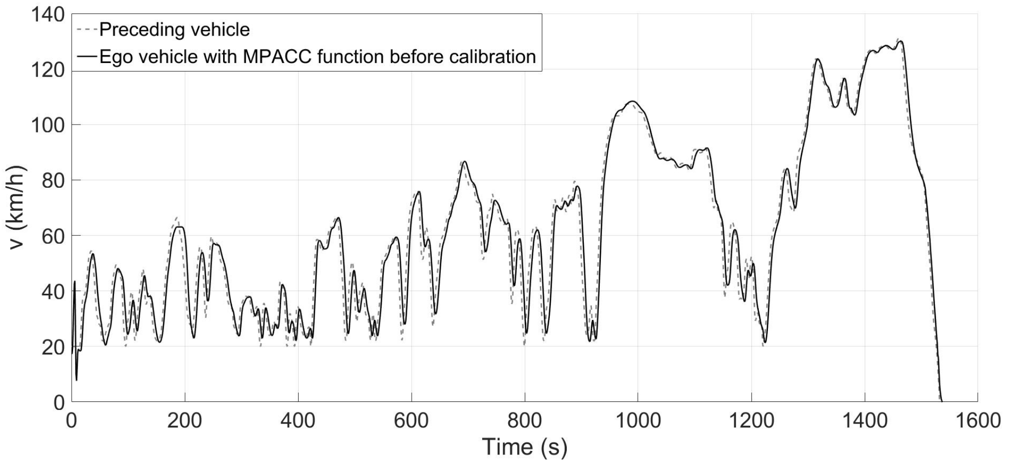
Table 4.2 shows the verification results with the selected Pareto points under conventional DoE and active DoE methods based on the figure 3.19 and figure 3.25. For the conventional DoE method, at least 95% of the verification points are in the behavior model confidence interval with 95% confidence level, which means the behavior models are trusted under this master thesis. For the active DoE, all of the verification points are in the confidence interval. It is obviously that with active DoE, the behavior of the KPIs can be better modeled.

Table 4.2: Verification of behavior models with normal DoE and active DoE

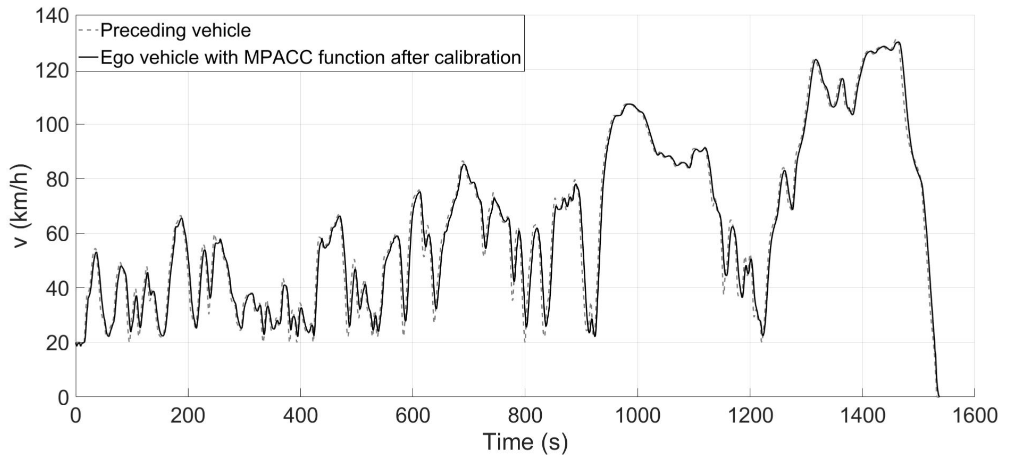
DoE type	KPI	Number of verification points	Points in confidence interval	Percentage
Conventional DoE	$KpiFuelConsumpAvg$	20	19	95%
	$KpiHeadwayDistanceMin$	20	20	100%
	$KpiHeadwayTimeMin$	20	19	95%
	$KpiJerkIntegral$	20	19	95%
Active DoE	$KpiFuelConsumpAvg$	20	20	100%
	$KpiHeadwayDistanceMin$	20	20	100%
	$KpiHeadwayTimeMin$	20	20	100%
	$KpiJerkIntegral$	20	20	100%

It took about two weeks for the developer of the MPACC function to manually calibrate the MPACC function with the trial and error method. Compared to the trial and error method, the KPI-model based method with DoE application takes about three days for the calibration of the MPACC funktion. One day for the co-simulation and calibration preparation, one day and two nights for the co-simulation execution, one day for the data precessing.

4.2 MPACC function



(a) Ego vehicle with MPACC function before calibration

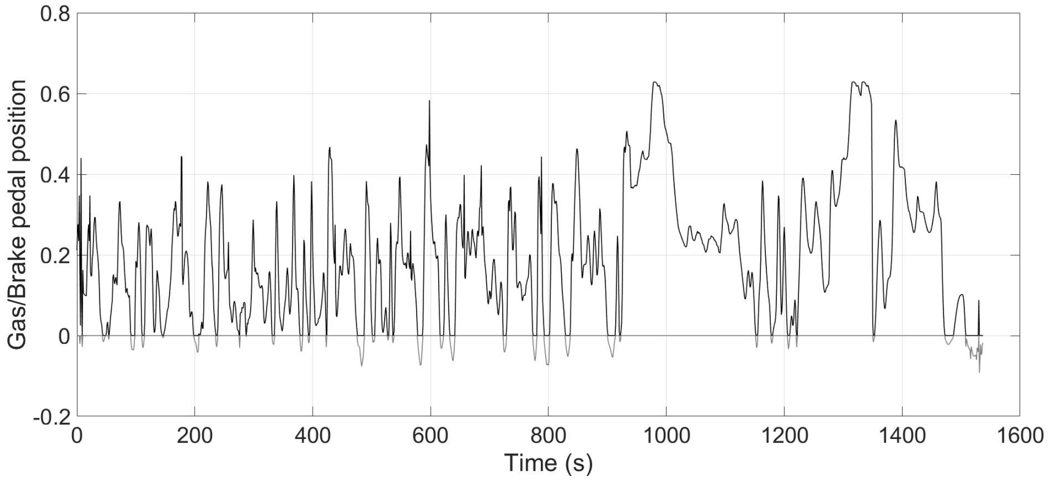


(b) Ego vehicle with MPACC function after calibration

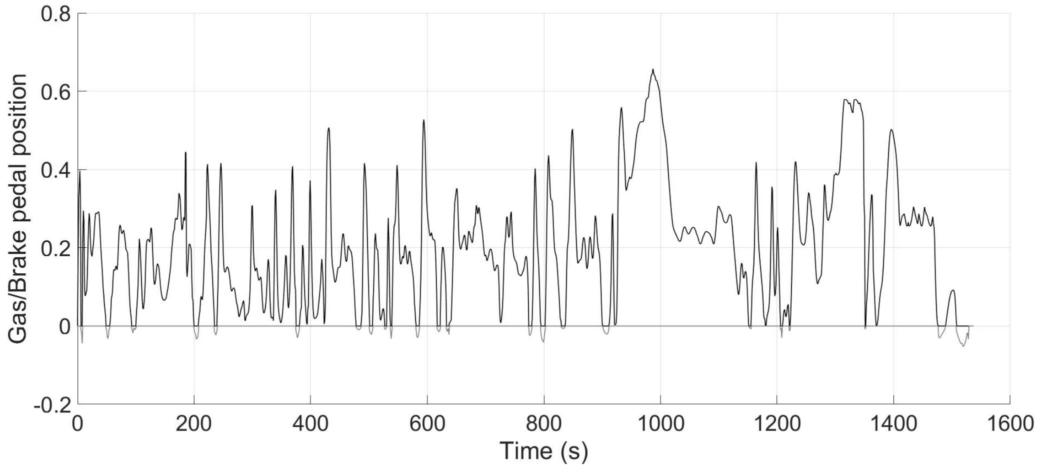
Figure 4.2: Speed profile of the ego vehicle and the preceding vehicle

In order to evaluate the virtual KPI-model-based calibration method, the performances (fuel efficiency, comfort and safety) of the calibrated MPACC function are compared to the MPACC function with base settings. Figure 4.2 shows the speed of ego vehicle with MPACC function before calibration (a) and ego vehicle with MPACC function after calibration (b). From the speed profiles of the ego vehicle and the preceding vehicle, we can find that the MPACC function can complete the task of speed tracking, which is the basic function of the MPACC system. The fuel efficiency, comfort and safety performance of the MPACC function will be discussed in the following sections.

4.2.1 Fuel efficiency



(a) Ego vehicle with MPACC function before calibration



(b) Ego vehicle with MPACC function after calibration

Figure 4.3: Gas and brake pedal position of the ego vehicle (minus value means braking)

Figure 4.3 shows the gas and brake pedal position of the ego vehicle with the MPACC function before calibration and with the MPACC function after calibration. The ego vehicle with the MPACC function after calibration will avoid heavy acceleration and too much braking, in order to reduce the fuel consumption. But for the ego vehicle with the MPACC function before calibration, the acceleration and braking are heavier, especially there are more braking applied to avoid collision. For these reason, the average fuel consumption of the ego vehicle with the MPACC function after calibration is $6,42$l/100km, which is 9% lower than the average fuel consumption of the ego vehicle with the MPACC function before calibration, $7,06$l/100km.

4.2.2 Comfort

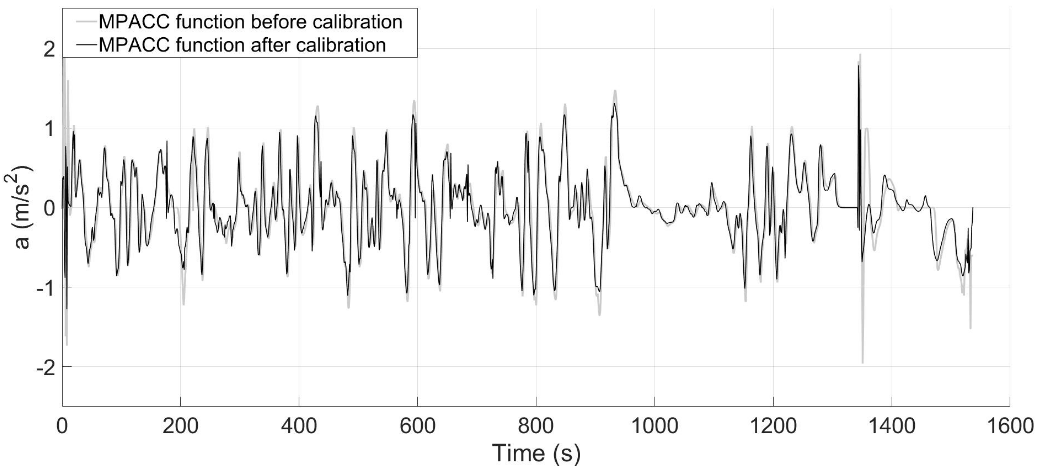


Figure 4.4: Acceleration and deceleration of the ego vehicle

Figure 4.4 shows the acceleration and deceleration of the ego vehicle with the MPACC function before calibration and with the MPACC function after calibration. For the ego vehicle with the MPACC function before calibration the acceleration and deceleration are higher than ego vehicle with the MPACC function after calibration. At the same time, the change of acceleration and deceleration of ego vehicle with the MPACC function before calibration is also faster. Figure 4.5 shows the jerk of the ego vehicle with the MPACC function before calibration and with the MPACC function after calibration. The integral of the absolute jerk for the ego vehicle with MPACC function after calibration is 263m/s^2, which is 13,5% lower much more lower than the ego vehicle with the MPACC function before calibration, 304m/s^2. So the ego vehicle with the MPACC function after calibration has a better comfort performance than with the MPACC function before calibration.

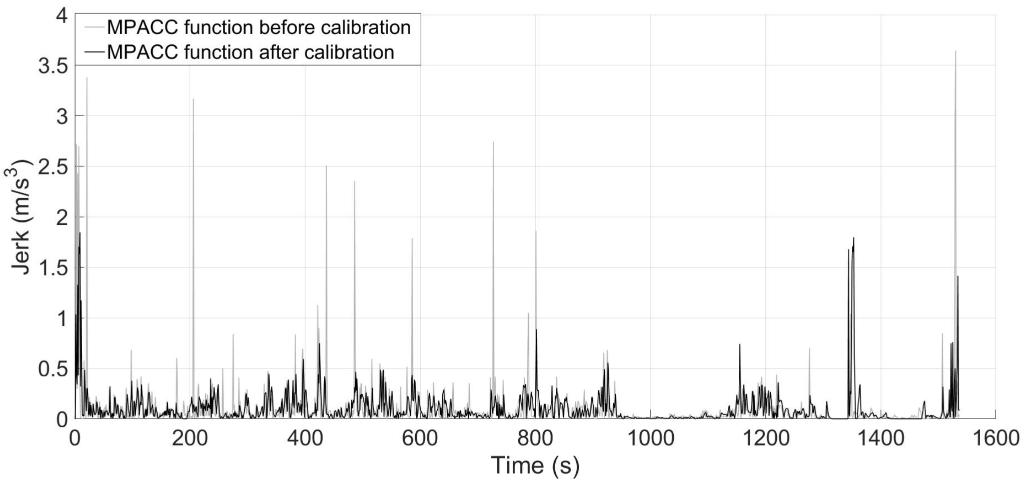


Figure 4.5: Absolute jerk of the ego vehicle

4.2.3 Safety

In order to keep the safe distance with the preceding vehicle, on the one hand, the ego vehicle should have a sufficient speed-based distance (headway time) to the preceding vehicle, and on the other hand, the ego vehicle should also have a sufficient absolute distance to the preceding vehicle. However, the distance should not be too long, because of the cut in possibility by the other traffic vehicles, which could lead to an accident.

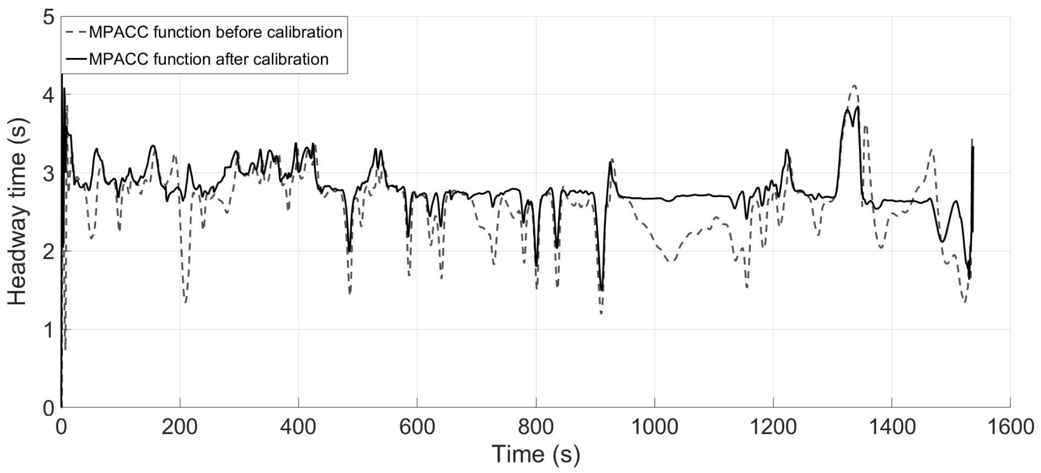


Figure 4.6: Headway time between the ego vehicle and the preceding vehicle

Figure 4.6 illustrates the headway time between the ego vehicle and the preceding vehicle. The headway time of the ego vehicle with the MPACC function before calibration vibrates a lot, almost from 1, 2s (at t = 910s) to 4, 1s (at t = 1337s). Compared to the MPACC function before calibration, the headway time of the ego vehicle with the MPACC function after calibration varies from 1, 5s (at t = 910s) to 3, 9s (at t = 1348s). With a shorter minimum headway time, the ego vehicle with the MPACC function after calibration is safer as with the MPACC function before calibration.

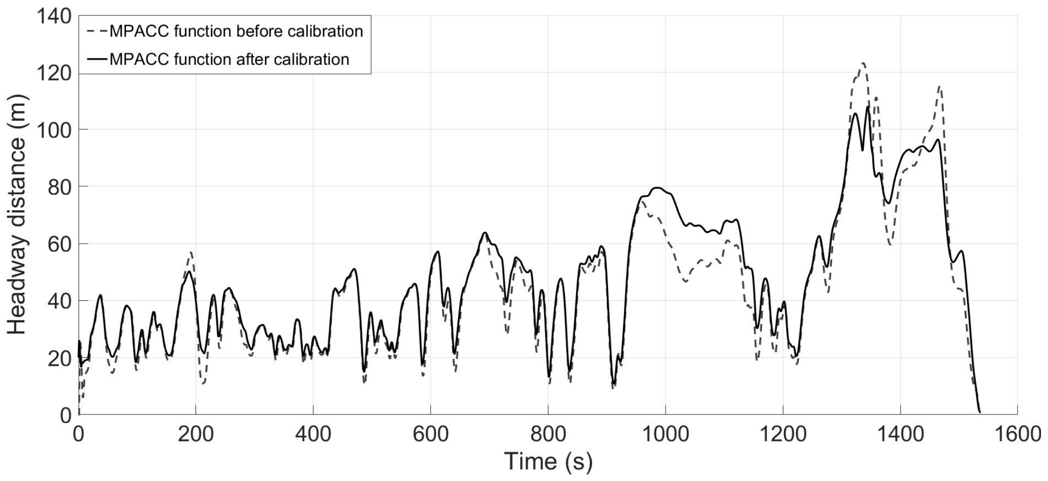


Figure 4.7: Headway distance between the ego vehicle and the preceding vehicle

Figure 4.7 illustrates the headway distance between the ego vehicle and the preceding vehicle. The results is the same as the headway time. For the ego vehicle with the MPACC function before calibration, the headway distance varies from 8, 6m (at t = 912s) to 123, 2m (at t = 1337s). For the ego vehicle with the MPACC function after calibration is from 10, 8m (at t = 912s) to 108, 1m (at t = 1348s). With a shorter minimum headway distance and longer maximum headway distance, the ego vehicle with the MPACC function after calibration is safer as with the MPACC function before calibration.

5 Summary

The object of this master thesis is to perform and demonstrate a virtual calibration method for an ACC function. With the performed method, the MPACC (Model Predictive Adaptive Cruise Control) function is calibrated regarding the fuel efficiency, comfort performance and safety. In this chapter, the most important contents of all the previous chapters are summarized.

Chapter 1 Introduction: In the USA alone, there were over 1,6 million people killed in motor vehicle traffic crashes over the past 40 years. The purpose for the development of ADAS function and AD (Autonomous Driving) vehicle is to reduce the traffic accidents, which is mostly (94%) caused by human errors. There are already some ADAS functions available in the production vehicles, such as ACC, AEB, LKA and so on. Based on the SAE levels of driving automation, the driving automation are divided into 6 levels and 5 stages. In these levels, the tasks distribution between human driver and ADAS/AD system are different. In level 0, the human driver takes the complete vehicle control responsibility and in level 5 the vehicle would take over the complete vehicle control responsibility. In order to develop the ADAS/AD function, the V-Model development process is used. With the increase of vehicle complexity and decrease of vehicle development time, new challenging appears, how to develop an ADAS/AD function with lower costs as well as shorter development time but with higher reliability? This is the motivation of this master thesis to perform and demonstrate an KPI-model-based calibration method with the MPACC function as example.

Chapter 2 Methodology: The simulation of the MPACC function is realized with the co-simulation platform AVL Mode.CONNECT. With this co-simulation platform, it is possible to connect different simulation software together and take the advantage of each. In this master thesis, the co-simulation model is built using vehicle model as well as simulation environment form IPG CarMaker and the MPACC function from MATLAB/Simulink. The coordinate system for the co-simulation is from IPG CarMaker coordinate system. The simulation is visualized with IPG CarMaker. The calibration is executed with AVL CAMEO, which can automatically start the co-simulation with AVL Model.CONNECT and retrieve the responses after every single co-simulation run. With online and/or offline data processing, the MPACC function can be calibrated with predefined KPIs. For the virtual calibration, the conventional DoE and active DoE methods are applied to reduce the numbers of co-simulation runs, in order to reduce the calibration efforts. The behavior models of KPIs are built in AVL CAMEO with the results of the co-simulation runs. The outliers of the raw data are firstly detected with statistical method and then deactivated. For the modeling there are two types of model available, the RNN (Recurrent Neural Network) model and the free polynomic model, which can be automatically generated with AVL CAMEO. In order to evaluate the goodness of fit and quality of the behavior models, the statistical coefficients R^2, R^2_{adj} and R^2_{pred} are introduced. They indicate how well the behavior model can model the measurements, and how good is the quality of the model calculation and the goodness of prediction. Then the confidence interval of the model is generated based on the residual analysis, which analyses the residual of the measurements and predicted values, in order to find prediction outliers and present the confidence interval based on certain confidence levels. The correlation analysis is used to find the correlation between variation channels and KPIs, so the most significant parameters can be selected and calibrated firstly, in order to reduce the complexity of the calibration. It is based on the reduction of SSE (Sum of Square of Error) for each variation channel. The to be calibrated MPACC function will predict the motion of preceding vehicle and output vehicle control to the ego vehicle, in order to achieve low fuel consumption and high driving comfort. Finally the co-simulation models as well as the inputs and outputs connection between different components in this master thesis are described.

Chapter 3 Simulation and calibration: In this chapter, the presented co-simulation model is simulated and the MPACC function is calibrated with certain scenarios, in which the ego vehicle follows the preceding vehicle with the specified speed profile that is based on the NEDC test cycle. The PI controller in the MPACC function is an acceleration controller, which controls the ego vehicle to reach the desired acceleration as fast and precisely as possible. It is optimized firstly with step signal input for acceleration. The corn algorithm is applied in the MPACC controller, which is then calibrated with the predefined scenario. The KPIs regarding fuel efficiency, driving comfort and safety are defined right after the definition of the scenarios. Before the calibration, the correlation analysis is applied to find the most significant parameters. Then the conventional DoE and active DoE methods are applied to calibrate the MPACC controller with the data processing methods presented in chapter 2. After that, the behavior models are verified with the selected Pareto points and the same scenario. Finally the other less significant parameters are calibrated with the same process. After the calibration, the validation process is applied regarding the fuel efficiency, driving comfort and safety with a different speed profile of the preceding vehicle, modified from the speed profile as given by the WLTP test cycle.

Chapter 4 Results and discussion: The results of the calibration and validation are described and discussed in this chapter. With the conventional DoE and active DoE methods, the efforts of the virtual calibration can be reduced significantly. The performance of active DoE is better than conventional DoE. The ego vehicle with the MPACC function after calibration has a 9% reduction of average fuel consumption, 13,5% of jerk and longer minimum headway time and headway distance compared to the ego vehicle with the MPACC function before calibration.

5.1 Conclusion

The performed KPI-model-based virtual calibration method for ADAS function was successfully executed with AVL CAMEO and AVL Model.CONNCET. The interface between AVL Model.CONNCET and IPG CarMaker as well as MATLAB/Simulink could be implemented easily. With the performed KPI-model-based virtual calibration method, we successfully calibrated the MPACC function with less application efforts. The active DoE method has a better performance than conventional DoE method. Finally the calibrated MPACC function could significant reduce fuel consumption, improve comfort and safety performance of the ego vehicle.

5.2 Outlook

In the future, the application of the virtual KPI-model-based calibration method for complex ADAS and AD function will be a necessary process, in order to reduce the development time and calibration efforts. In order to apply the introduced virtual calibration method to a realistic vehicle development process, there are still some necessary processes to be implemented. The calibration could be based on different scenarios and have different parameters for the ADAS function when the vehicle runs under different driving conditions. With the DoE method, it is also possible to implement virtual validation and testing, in order to find the corner cases, also called critical cases, which need to be validated and tested in on-road tests.

List of Figures

List of Tables

Bibliography

[1] U.S. Department of Transportation. 2018 Fatal Motor Vehicle Crashes: Overview. Available at: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812826, 2019. Accessed on 05. December 2019.

[2] U.S. Department of Transportation. Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey. Available at: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812506, 2019. Accessed on 05. December 2019.

[3] T. M. Gasser, C. Arzt, M. Ayoubi, A. Bartels, L. Buerkle, J. Eier, F. Flemisch, D. Haecker, T. Hesse, W. Huber, C. Lotz, M.Maurer, S.Ruth-Schumacher, J. Schwarz, W. Vogt. Rechtsfolgen zunehmender Fahrzeugautomatisierung. Bundesanstalt fr̈ Straßenwesen, 2012.

[4] The national academies of sciences. Rechtsfolgen zunehmender Fahrzeugautomatisierung. Available at: https://trid.trb.org/view/1217277, 2019. Accessed on 05. December 2019.

[5] R. Kastner. Modellbuildung und Simulation eines koorperativen Stauassistenten ueber ein Co-simulationframework. Master thesis, Graz university of technical, Graz, 2017.

[6] J. Pell. Virtuelle Validierung von automatisierten Fahrfunktionen im Nutzfahrzeug. Master thesis, Graz university of technical, Graz, 2018.

[7] H. Winner, S. Hakuli, F. Lotz, and C. Singer, editors. Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort. Wiesbaden: Springer Vieweg, 2015.

[8] Verband der Atomobilindustrie. Lane Keeping Assist Systems. Available at: https://www.vda.de/de/themen/sicherheit-und-standards/spurhalteassistenzsysteme/lane-keeping-assist-systems-lkas.html, 2019. Accessed on 05. December 2019.

[9] Stanford Law School. SAE levels of driving automation. Available at: http://cyberlaw.stanford.edu/blog/2013/12/sae-levels-driving-automation, 2019. Accessed on 05. December 2019.

[10] P. Royger. Autonomes Fahren Wie viel Autonomie das Autofahren vertraegt. ATZelektronik, 10(2):S.50-53, 2015.

[11] Verband der Automobilindustrie. Automatisierung - Von Fahrerassistenzsystemen zum automatisierten Fahren. VDA Magazin-Automatisierung, 2015.

[12] Audi MediaCenter. The new Audi A8 - conditional automated at level 3. Available at: https://www.audi-mediacenter.com/en/on-autopilot-into-the-future-the-audi-vision-of-autonomous-driving-9305/the-new-audi-a8-conditional-automated-at-level-3-9307, 2019. Accessed on 05. December 2019.

[13] The Verge. Riding in Waymo one, the Google Spinoff's first self-driving taxi service. Available at: https://www.theverge.com/2018/12/5/18126103/waymo-one-self-driving-taxi-service-ride-safety-alphabet-cost-app, 2019. Accessed on 05. December 2019.

[14] Waymo. Waymo one self driving taxi service. Available at: https://waymo.com/, 2019. Accessed on 05. December 2019.

[15] O. Gietelink. Design and Validation of Advanced Driver Assistance Systems. Trail Research School, 2007.

[16] R. Rakesh, S. Miroslaw, H. Joergen, N. Martin, T. Fredrik. Increasing Efficiency of ISO 26262 Verification and Validation by Combining Fault Injection and Mutation Testing with Model Based Development, 2013.

[17] AVL List GmbH. Driveability Engineering for Commercial Vehicles. Available at: https://www.avl.com/de/vehicle-benchmarking/, 2019. Accessed on 05. December 2019.

[18] The Verge. Autonomous miles driven in California from 12. 2016 to 11. 2017. Available at: https://www.theverge.com/2018/1/31/16956902/california-dmv-self-driving-car-disengagement-2017, 2019. Accessed on 05. December2019.

[19] The Verge. 2018 autonomous driving takeover report. Available at: https://www.theverge.com/2019/2/13/18223356/california-dmv-self-driving-car-disengagement-report-2018, 2019. Accessed on 05. December 2019.

[20] General Motors. 2018 autonomous driving takeover report. Available at: https://www.gm.com/content/dam/company/docs/us/en/gmcom/gmsafetyreport.pdf, 2019. Accessed on 05. December 2019.

[21] C. Laurgeau. Intelligent Vehicle Potential and Benefits, pages 1537-1551. Springer London, London, 2012.

[22] AVL List GmbH. Model.CONNECT. Available at https://www.avl.com/-/model-connect-, 2019. Accessed on 04. December 2019.

[23] IPG Automotive. CarMaker. Available at https://ipg-automotive.com/products-services/simulation-software/carmaker/, 2019. Accessed on 04. December 2019.

[24] IPG Automotive. CarMaker. User's Guide version 7.1.2 : pp. 54, 242. 2019.

[25] International Standard for Standardization. ISO 8855:2011: Road vehicles - Vehicle dynamics and road-holding ability - Vocabulary, 2011.

[26] M. Kissai, B. Monsuez, X. Mouton, D. Martinez, and A. Tapus. Adaptive robust vehicle motion control for future over-actuated vehicles. In Proceedings of the 6th International Conference on Control, Mechatronics and Automation (ICCMA 2018), Tokyo, Japan, 12-14 October 2018; pp. 97-104.

[27] AVL List GmbH. AVL CAMEO. Available at https://www.avl.com/web/guest/-/avl-cameo-4-, 2019. Accessed on 04. December 2019.

[28] H. M. Koegeler. Lecture material: Statistical design of experiment : pp. 6-7. 2018.

[29] B. Durakovic. Design of Experiments Application, Concepts, Examples: State of the Art. Periodicals of Engineering and Natural Sciences, 5(3):421-439, 2017.

[30] AVL List GmbH. AVL CAMEO Online Help version 4.2.835.0. 2019.

[31] Wikipedia. Student's t distribution. Available at: https://en.wikipedia.org/wiki/Student Accessed on 04. December 2019.

[32] K. Benjamin. AVL CAMEO User's Guide version 1.0.3: AVL sensitivity prototype for CAMEO applications, 2018.

[33] Wikipedia. PID controller. Available at: https://en.wikipedia.org/wiki/PID_controller, 2019. Accessed on 04. December 2019.

[34] Wikipedia. Model predictive control. Available at: https://en.wikipedia.org/wiki/Model_predictive_control, 2019. Accessed on 04. December 2019.

[35] Robert H. Cannon. Dynamics of Physical Systems: Chapter 5 pp.151, 1967.

[36] Wikipedia. New European Driving Cycle. Available at: https://en.wikipedia.org/wiki/New_European_Driving_Cycle, 2019. Accessed on 04. December 2019.

[37] H. Beglerovic, A. Ravi, N. Wikstroem, H. M. Koegeler, A. Leitner, J. Holzinger. Model-based safety validation of the automated driving function Highway Pilot. Wiesbaden: Springer Vieweg, 2017.

[38] F. Huang, C. Zhao, Y. Huang, P. Dai, D. Hao, Y. Yue. Study on the Evaluation Model of Vehicle Comfort Based on the Neural Network. IFAC PapersOnLine, 51(31):553-558, 2018.

[39] Wikipedia. Worldwide Harmonized Light Vehicles Test Procedure. Available at: https://en.wikipedia.org/wiki/Worldwid_Harmonised_Light_Vehicles_Test_Procedure, 2019. Accessed on 04. December 2019.