Safety of automated vehicles

Citation for published version (APA):
Hoogeboom, F. N. (2020). *Safety of automated vehicles: design, implementation, and analysis*. [Phd Thesis 1 (Research TU/e / Graduation TU/e), Mechanical Engineering]. Technische Universiteit Eindhoven.

Document status and date:
Published: 27/10/2020

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

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

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

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 04. Oct. 2023

Safety of automated vehicles: design, implementation, and analysis

disc
The author has succesfully completed the educational program of the Graduate School of the Dutch Institute of Systems and Control (DISC).

A catalogue record is available from the Eindhoven University of Technology Library.
ISBN: 978-90-386-5141-5

Typeset by the author using LaTeX 2_ε.

Cover design: Manon van Dijk.
Reproduction: ProefschriftMaken.

© 2020 F.N. Hoogeboom. All rights reserved.

Safety of automated vehicles: design, implementation, and analysis

PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Technische Universiteit Eindhoven, op gezag van de
rector magnificus prof.dr.ir. F.P.T. Baaijens, voor een
commissie aangewezen door het College voor
Promoties, in het openbaar te verdedigen op
dinsdag 27 oktober 2020 om 16.00 uur

door

Francisciscus Nicolaas Hoogeboom

geboren te Harenkarspel

Dit proefschrift is goedgekeurd door de promotoren en de samenstelling van de promotiecommissie is als volgt:

voorzitter:	prof.dr. L.P.H. de Goey
promotor:	prof.dr. H. Nijmeijer
copromotor:	dr.ir. T.P.J. van der Sande
leden:	prof. A. Khajepour PhD PEng (University of Waterloo)
	prof.dr. M.G.J. van den Brand
	dr.ir. G.M. Bonnema PDEng (Universiteit Twente)
	dr.ir. I.J.M. Besselink

Het onderzoek dat in dit proefschrift wordt beschreven is uitgevoerd in overeenstemming met de TU/e Gedragscode Wetenschapsbeoefening.

Summary

Safety of automated vehicles: design, implementation, and analysis

Road transport demand is ever increasing while road capacity is limited. Additionally, the societal quest for safer and environmentally friendly transportation is growing. Intelligent transportation systems can contribute to improve both road throughput and safety. The primary reason for this is that driver-related error is the most critical factor leading to accidents. Currently, several commercially available driver assistance systems already help to improve road safety by assisting the driver. Automated and cooperative vehicles can reduce human driving tasks even further and are therefore considered a promising way to improve road safety and traffic throughput.

However, to develop automated and cooperative vehicles, a research and development platform is required. The vehicle has to be able to perform the driving tasks normally executed by the driver. In this research, an automated and cooperative vehicle is developed by converting a simple electrical vehicle—a Renault Twizy. This vehicle performs the basic automotive functions such as drive, brake, and steering but has no additional on-board intelligence or actuators. Additional sensors and actuators have to be installed to achieve full vehicle automation. These add-on features have to be designed with safety in mind to guarantee safe vehicle operation. To address this safety need, a hardware design is proposed that integrates safety features, such as sensor monitoring and a reliable communication structure.

As hardware safety features alone are not sufficient to guarantee safe automated vehicle operation, the real-time application that processes sensor signals, plans control actions, and executes these actions also integrates safety measures. A functional software architecture has been developed in this research that includes sensor monitoring, actuator monitoring, and automated driving mode selection based on the status of the sensors and actuators as well as user inputs. The designed architecture enables modular automated driving system development while ensuring safe vehicle operation. In case of any automated system fault, all control outputs are disabled and the safety driver regains control of the vehicle.

Besides actual automated driving system faults, it is also possible that the safety driver tries to intervene by, for example, grabbing the steering wheel or depressing the brake pedal. In these cases, the control system should also be able to detect the interruptions and react accordingly. This thesis shows both in simulation and with experiments that user interventions can be detected by applying a combinations of techniques, such as model-based fault diagnosis.

To validate the designed hardware and software systems before commencing road tests, a common practice in automotive industry is to perform hardware-in-the-loop tests. To this end, a hardware-in-the-loop test setup is developed, consisting of two connected computers. One computer executes the real-time control model as in the automated Renault Twizy and the other computer executes a virtual representation of the automated Renault Twizy. The virtual representation consists of a multibody model to mimic the vehicle dynamical behavior as well as sensor and actuator blocks to model their characteristics. To simulate and evaluate the system as a whole, the real-time computers are coupled with the exact same physical communication interfaces as in the real automated Renault Twizy. Several identification experiments are conducted to obtain the model parameters, such as tyre and suspension properties. The dynamical behavior of the whole model is validated using data obtained from full-scale driving test experiments. These tests show that the model is capable of accurately predicting the dynamical behavior of the real automated Renault Twizy. In contrast to the real automated Renault Twizy, the sensors and actuators as well as vehicle properties can be easily manipulated in the virtual representation. Therefore, the hardware-in-the-loop setup is used to test the behavior of the designed control system safety features in a controlled environment. The setup can also be used for performance evaluation and optimization of the automated controllers.

In addition to fault diagnosis of the control system, correctness of the environmental perception system is also important to verify. Fusing observations from different sources leads to one coherent image of the surroundings, but it can also be used to detect sensor faults with cooperative driving. By combining radar, vehicle-to-vehicle communication, and vehicle sensor data, it is possible to detect when incorrect information is received from the preceding vehicle or when a wrong object is tracked. Simulations and experiments show that the proposed method is able to detect faults, thereby avoiding unsafe cooperative driving situations.

The main contributions of this work are on the safe operation of an automated vehicle. It covers the steps from hardware and software design to validation and testing. The result is an available set of tools to further develop automated and cooperative vehicles, consisting of a research and development platform on which automated controllers can be tested in a safe and controlled manner. In addition, a hardware-in-the-loop setup is available that can be used to validate and improve the automated and cooperative algorithms before commencing road tests.

Contents

Summary v

Nomenclature xi

1. Introduction 1
- 1.1. Driving Automation System . . . 1
  - 1.1.1. Driving automation levels . . . 1
  - 1.1.2. Road vehicles safety . . . 3
  - 1.1.3. Development process . . . 6
- 1.2. Challenges in Automated Driving Systems . . . 8
- 1.3. Research Objectives and Contributions . . . 9
- 1.4. Outline . . . 10

2. Development Vehicle Instrumentation 13
- 2.1. Introduction . . . 13
- 2.2. Renault Twizy . . . 15
  - 2.2.1. Control units . . . 15
  - 2.2.2. Decoding CAN messages . . . 18
- 2.3. Automated Driving Platform Hardware Requirements . . . 19
- 2.4. Additional Vehicle Instrumentation . . . 20
- 2.5. Discussion . . . 29

3. Safety Framework for Automated Driving 31
- 3.1. Introduction . . . 31
- 3.2. Automated Driving Platform Software Requirements . . . 33
- 3.3. Functional Software Architecture . . . 33
  - 3.3.1. Interface layer . . . 34
  - 3.3.2. Sensor layer . . . 35
  - 3.3.3. Actuator layer . . . 36
  - 3.3.4. System and safety management layer . . . 37
  - 3.3.5. Other layers . . . 37
- 3.4. Software Design Verification . . . 38
  - 3.4.1. Camshaft angle sensor fault . . . 38
  - 3.4.2. Real-time application fault . . . 39

3.5. Discussion 39

4. Automated Driving Safety Verification Framework 41
4.1. Introduction 41
4.2. Simulation Framework 43
    4.2.1. Vehicle dynamics simulation 44
    4.2.2. Actuators simulation 46
    4.2.3. Sensor simulation 54
4.3. Simulation Hardware Setup 55
4.4. Simulation Model Validation 55
4.5. Conclusion 60

5. Model-Based Fault Diagnosis of an Automated Steering System 61
5.1. Introduction 61
5.2. Model-Based Fault Diagnosis 63
    5.2.1. Linear system model with additive fault representation . . . 64
    5.2.2. Residual generation 64
    5.2.3. Fault detectability 65
    5.2.4. Fault isolability 65
    5.2.5. Model detection 66
    5.2.6. Residual evaluation 67
    5.2.7. Decision making 68
5.3. Steering System Modeling 68
    5.3.1. Steering system model 70
    5.3.2. Driver's arm model 71
    5.3.3. Single track model 72
5.4. Gray-Box System Identification 73
    5.4.1. Experiment design 73
    5.4.2. Estimation model 74
    5.4.3. Model estimation 74
    5.4.4. Model validation 75
5.5. Fault Diagnosis 79
    5.5.1. Actuator faults 79
    5.5.2. Driver state detection 82
5.6. Simulation Results 84
    5.6.1. Actuator faults 87
    5.6.2. Driver state detection 87
5.7. Experimental Results 89
    5.7.1. Actuator faults 90
    5.7.2. Driver state detection 90
5.8. Conclusion 92

6. Fault Diagnosis for Connected Vehicles 95
6.1. Introduction 95
6.2. CACC System Implementation 97
    6.2.1. Host-tracking 98
    6.2.2. Course prediction 105
    6.2.3. Wireless communication 106
    6.2.4. Target selection 107
    6.2.5. Longitudinal vehicle dynamics model 111
    6.2.6. CACC controller 111
6.3. CACC Fault Diagnosis 113
    6.3.1. System modeling 115
    6.3.2. Fault modeling 115
    6.3.3. Model-based fault diagnosis 116
6.4. Simulation Results 118
6.5. Experimental Results 122
6.6. Conclusion 124

7. Conclusions and Recommendations 125
7.1. Conclusions 126
7.2. Recommendations 127

A. Multibody Model 129

Bibliography 133

Dankwoord 141

Curriculum Vitae 143

Nomenclature

Acronyms

ACC	Adaptive Cruise Control
ADAS	Advanced Driver-Assistance System
ADS	Automated Driving System
AMR	Anisotropic Magneto Resistance
ASIL	Automotive Safety Integrity Level
BCB	Battery Charge Box
BCM	Body Control Module
BMS	Battery Management System
CACC	Cooperative Adaptive Cruise Control
CAM	Cooperative Awareness Message
CAN	Controller Area Network
COTS	Commercial Off-The-Shelf
DENM	Decentralized Environmental Notification Messages
EGNOS	European Geostationary Navigation Overlay Service
EHB	Electronic Hydraulic Brake
ESC	Electronic Stability Control
ESKF	error-state Kalman filter
GNSS	Global Navigation Satellite System
HIL	Hardware-In-the-Loop
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
ITS	Intelligent Transport System
MIO	Most Important Object
NRMSE	Normalized Root Mean Squared Error
ODD	Operational Design Domain
OEM	Original Equipment Manufacturer
PBRS	Pseudo-Random Binary Sequence
PEB	Power Electronics Block
SAE	Society of Automotive Engineers
SOTIF	Safety Of The Intended Functionality

TFM	Transfer-Function Matrix
V2X	Vehicle-to-Everything

1.

Introduction

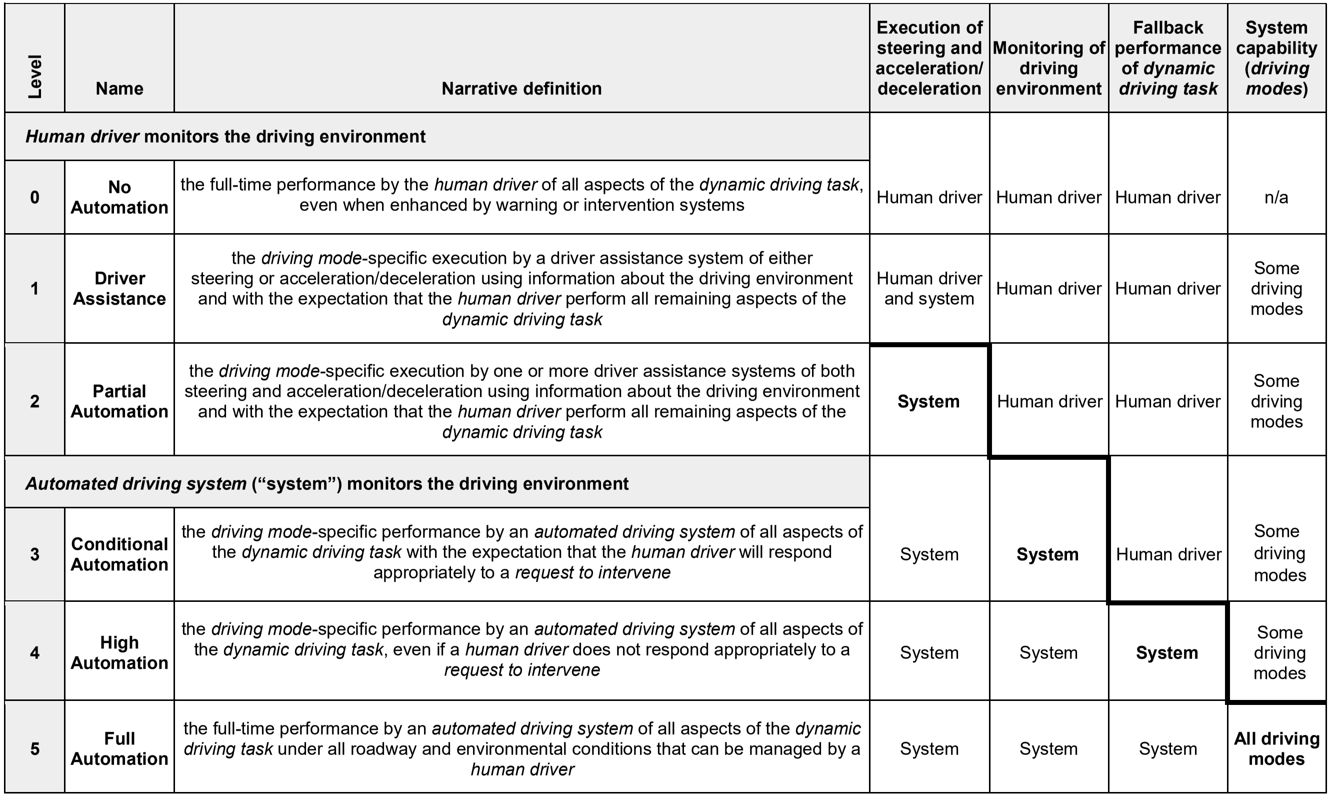
The major factor (94 percent) leading to fatal crashes in road transportation is human driving error (Singh 2015). Out of the human driving error 41 percent is classified as recognition error and 33 percent as decision error. Automated Driving Systems (ADSs) have the potential to significantly reduce the recognition and decision error as sophisticated systems can be more vigilant, competent, and reliable than the average human driver. In addition to safety, ADSs can also increase driver comfort, road throughput, and efficiency by taking over driving tasks (Van Arem et al. 2006). To help ADSs in becoming a reality, this thesis aims to contribute to the safe operation of automated vehicles. This chapter provides an introduction to the field of ADSs and is organized as follows. Section 1.1 briefly introduces ADSs and explains the development process involved with ADSs. Section 1.2 identifies a set of current unsolved technological challenges within the field of ADSs. Section 1.3 addresses these challenges by formulating and listing the thesis objectives and contributions. Finally, Section 1.4 presents the thesis outline.

1.1. Driving Automation System

The main goal of driving automation is to increase safety and comfort of the passengers and other traffic participants. Unlike human drivers, driving automation systems are never tired or distracted from driving as they are based on predefined rules. The performance of driving automation systems may also be better than a human driver as they have multi-sensor environmental perception systems, faster decision making and responses, and more precise execution of commands.

1.1.1. Driving automation levels

The level of driving automation is classified by the Society of Automotive Engineers (SAE) in their J3016 standard (SAE International 2018). The standard defines 6 levels ranging from level 0: no driving automation to level 5: full driving automation. A schematic overview of all these levels is depicted in Figure 1.1.



Level	Name	Narrative definition	Execution of steering and acceleration/ deceleration	Monitoring of driving environment	Fallback performance of *dynamic driving task*	System capability (*driving modes*)
***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
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
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
***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
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
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 1.1.: SAE driving automation levels (retrieved from (Van der Sande and Nijmeijer 2017))

Levels 1 and 2 are considered driver assistance systems and belong to the domain of Advanced Driver-Assistance Systems (ADASs). An example of such a system is Adaptive Cruise Control (ACC) which utilizes range sensors such as camera and radar to measure relative distance and velocity to a preceding vehicle. Based on this sensor info, the system tries to maintain a desired inter-vehicle distance by control of the driveline and brakes.

A system belongs to the domain of ADSs when it is classified as level 3 or higher. An example of such a system is an autonomous or self-driving vehicle, which perform all aspects of the driving task, such as accelerating/decelerating, steering, and monitoring of the driving environment. The main difference between ADASs (levels 1 and 2) and ADSs (levels 3 through 5) is the location of responsibility for driving. With ADASs, the human driver must monitor the driving environment, whereas with ADSs, the ADS has to monitor the driving environment and alert the driver in case of a situation that may lead to discontinued ADS operation. On level 3, the driver must respond to the take-over request by regaining control of the vehicle. On levels 4 and 5, the ADS must also be capable of performing the fallback and bring the vehicle to a safe stop in case the driver does not regain control of the vehicle.

1.1.2. Road vehicles safety

The Intelligent Transport Systems (ITSs) designed to make traffic safer and more comfortable may also introduce new safety hazards. A safety hazard can, for example, be a malfunctioning component or a software bug, but it can also be a driver neglecting to monitor the driving environment while it was supposed to do so. With malfunction, a temporary interruption of the system's ability to perform a desired function is defined. The malfunction may lead to a failure in which a permanent interruption of the ability of the system to perform the desired function is achieved. Both malfunction and failure result from one or more faults which is defined as the deviation of a system property from the standard condition. In safety critical systems, faults must be detected as early as possible to prevent undesired and possibly unsafe behavior.

An approach to detect sensor faults is by implementing redundant sensors. Sensor faults can be detected by comparing the sensor value with the redundant sensor value. The hardware redundancy design approach makes the design more expensive and complex. Next to hardware redundancy, analytical redundancy can be employed to detect faults. With analytical redundancy, a mathematical model of the system is used to predict sensor measurements, and faults can be detected by comparing the sensor output with the predicted output.

Random hardware failures and systematic design failures are covered by the ISO 26262 standard (ISO 26262 2011). This functional safety standard is designed for electrical and/or electronic systems in series production vehicles and covers aspects from functional safety concept to design. Functional safety is defined as how the system should detect and react to failures and errors. The standard defines requirements and provides guidance to avoid and control random hardware and systematic faults that could violate the safety goal.

Figure 1.2 depicts the main steps of the ISO 26262 design process, known as the safety life cycle. The design process starts with the item definition that describes the function at the vehicle level from a functional non-technical perspective. Based on the item definition, the list of possible malfunctions leading to hazards are identified. The hazards are categorized based on exposure, severity, and controllability. Each hazardous event is assigned to a certain Automotive Safety Integrity Level (ASIL) ranging from A to D. The strongest safety requirements are on ASIL D. For each hazardous event, safety goals must be formulated that serve as a basis for the functional safety concept. The safety goals are addressed by defining appropriate safety measures in the functional safety requirements. The functional safety requirements are refined during the development phase into technical safety requirements, and all technical safety requirements are allocated to architectural components, such as hardware and software components. The safety requirements are tested and validated during the final stages of the design process.

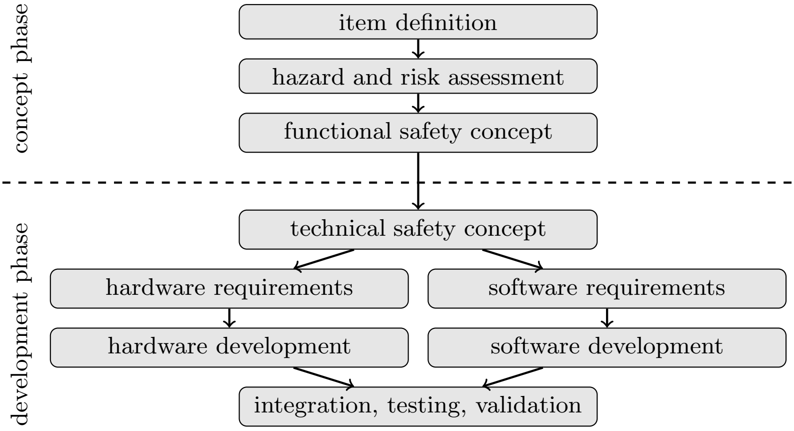


Figure 1.2.: Main steps of the ISO 26262 development process (adapted from (Schildbach 2018))

Even in the absence of hardware and software faults in the system as covered by the ISO 26262 standard, potential hazardous events can happen caused by functional insufficiencies or by misuse by persons. For example, one can think of a system designed for highway driving being employed on urban roads. The implemented system designed for the highway might inadequately react to situations encountered on urban roads as the driving conditions between these situations can significantly differ. For instance, on highways the traffic flow is one direction with the same class of vehicles while on urban roads oncoming traffic and other type of traffic participants can be encountered that do not match the expected behavior model of the system.

To address these and other events, the Safety Of The Intended Functionality (SOTIF) standard (ISO/PAS 21448 2019) has been defined. The SOTIF is complementary to the ISO 26262 standard. The SOTIF classifies all possible scenarios into four areas: known safe, known unsafe, unknown unsafe, and unknown safe. Figure 1.3 depicts the four categories. The purpose is to make all unsafe (known and unknown) scenarios safe, which can be achieved by, for example, limiting the Operational Design Domain (ODD). The ODD defines the conditions under which the system is designed to operate, such as weather conditions, geographic domain, and dynamic scene properties. Known scenarios can be evaluated using requirement-based testing. To identify and evaluate unknown unsafe scenarios, the standard recommends various testing approaches, such as test cases with high coverage of relevant scenarios, fault injection tests, and in the loop testing of relevant scenarios. Where the SOTIF is designed for ADASs, it can also be considered for ADSs. Additional steps, however, might be required in the future as ADSs are currently still in development phase (Khabbaz Saberi 2020).

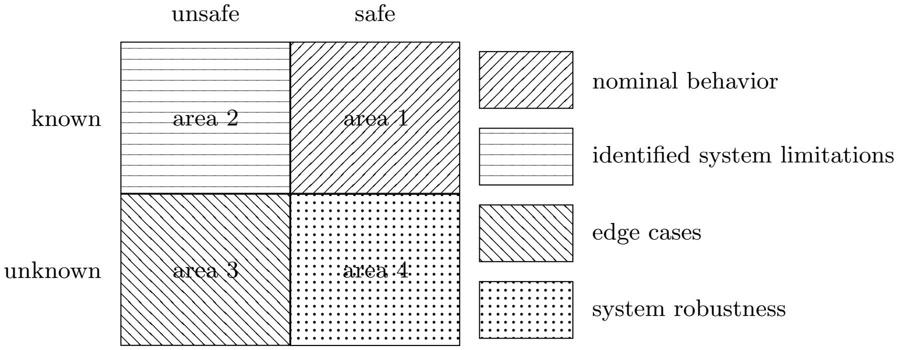


Figure 1.3.: Visualization of the known/unknown and safe/unsafe scenario categories (adapted from (ISO/PAS 21448 2019))

The highest level on current commercially available vehicles is level 2 (Bigelow 2019). These systems support the driver in longitudinal and lateral control tasks, but the driver should always remain alert and monitor the driving environment. Driver monitoring is important when systems are released to the public that require supervision because drivers might get overconfident and rely too much on the systems. In these cases, severe accidents can happen. Consider for example the Tesla Model S that was involved in a fatal accident with an automated vehicle (NTSB 2017) by crashing into a truck that was turning in front of it. The vehicle was operating under Tesla's Autopilot software which is an adaptive cruise control system with automated lane keeping. The camera system of the Tesla was unable to distinguish the white trailer against a bright lit sky and the radar sensor falsely assumed the trailer for a construction sign.

Another fatal example with a Tesla operating under Autopilot was with a Tesla Model X (NTSB 2018a) driving on a highway. The vehicle was following a lead vehicle on the left lane until it steered out of the lane and entered the gore area seven seconds prior to the crash. The lead vehicle was lost four seconds prior to the crash, resulting into the vehicle accelerating onto a crash attenuator. The crash attenuator was mounted on a concrete median barrier which divides the main travel lanes from the exit lane. No pre-crash braking or evasive steering event was detected prior to the crash, indicating that the crash attenuator was not recognized by the autopilot system as an object.

The first fatal accident with an ADS was with an Uber self-driving test vehicle (NTSB 2018b). The Volvo XC90 was driving 70 km/h on an urban road when a pedestrian was crossing the road while walking a bicycle. While the Volvo was equipped with an auto emergency brake system and a driver alertness monitoring system from factory, these system were disabled by Uber when the car was driving with their self-developed ADS. The radar and lidar sensors detected the pedestrian 6 s prior to the crash with different classifications as unknown object, vehicle, and pedestrian. At 1.3 s before impact, the ADS determined that a braking action was needed to mitigate a collision. However, auto emergency braking was not enabled by Uber when the system was under their computer control to avoid unpredictable vehicle behavior. The safety driver that was monitoring the ADS was also not informed. Instead, they completely relied on the safety driver to intervene when the system fails to react appropriately. The safety driver, however, was monitoring the ADS instead of monitoring the driving environment. Less than a second before impact, the safety driver intervened by using the steering wheel. However, too late to avoid the collision with the pedestrian.

1.1.3. Development process

While certain Original Equipment Manufacturers (OEMs) currently test their automated vehicles on public roads (Herger 2019), there are many unsolved technical challenges to overcome before ADSs becomes reality. Therefore, several academic institutions and research institutes also develop solutions and methods for testing ADSs. To validate and enhance their ADSs algorithms, a research and development platform is required. As most commercial vehicles do not provide complete automated functionality, vehicles have to be modified, which requires knowledge of the installed sensors, actuators, communication interfaces, and software architecture. Hence, support of an OEM would be convenient when building a research and development platform. However, most OEMs do not share any information as this is valuable data for competitors. In addition, the information might pose security risks to the integrity of the in-car systems. Therefore, original systems will have to be explored and decoded before it is possible to make the required changes.

As there currently are no state of the art and development standards on how to develop ADSs, road authorities and certain OEMs publish their vision on safety. For example, NHTSA (2017) published their vision on safety for ADSs which covers a design best practices for testing and safe deployment of ADSs. Similarly, Waymo (2019), General Motors (2018), and Aptiv et al. (2019) published their visions on safety for ADSs, along with their experiences with developing ADSs.

The verification and validation phase starts after the vehicle is developed. Safety and validation of ADSs can be a major challenge since the systems are composed of several hardware components and complex processing algorithms. The admittance to open roads without the need for an accumulation of road distance is highly desirable. Kalra and Paddock (2016) indicate that a credible safety argument requires millions of road miles. Obtaining these road miles are costly and it can even result in unsafe situations. A safer approach is closed-course testing but this is not scalable and therefore time consuming. As quality of the data is more important than the amount of data, scenario based testing can be more valuable in requiring a safety statement than driving for million of miles. Also scenarios can be tested which are unsafe or do not exist in current traffic.

No matter what level of sophistication ADSs will achieve, accidents will always happen. When these accidents involve automated vehicles, questions of liability will arise. Therefore, a legal framework must be developed for accidents that happen during automated driving. To cope with this, the Netherlands Vehicle Authority (RDW), suggests there must be a vehicle driving license (Feddes 2016). Their view is that with software constantly being updated the vehicles are ever-changing, and therefore, an one-off admittance is not adequate. Instead, they propose a performance-based testing approval method consisting of virtual testing, behavior testing on a closed track, and admittance to open roads with in-use compliance.

As the human driver is responsible for monitoring of the driving environment on (levels 1 to 2), safety admittance of ADASs limits to safe operation of the implemented system. Cooperative vehicles with system as Cooperative Adaptive

Cruise Control (CACC) also belong to the class of ADASs. CACC systems utilize a wireless link between vehicle enabling driving with a shorter inter-vehicle distance compared to a normal driver operated vehicle, thereby increasing road throughput and driver comfort. However, at these reduced vehicle following intervals, the driver cannot be considered as a backup and the CACC system must supervise itself to avoid unsafe driving situations.

On levels 3 to 4, the driver cannot be assumed to be alert at all times and it might be harder for the driver to notice certain faults, such as a flat tire, as it is also not actively controlling the vehicle. Therefore, additional safety checks must be performed by the ADS. A common approach is to use an extensive sensor set with hardware redundancy similar to those used in aviation (Zolghadri et al. 2014). While this is a workable solution for a research and development platform in which costs are not most important, it limits full scale deployment. The additional hardware costs of the ADS must be limited otherwise no one can afford an automated vehicle. To reduce hardware costs while ensuring safety, smart fault diagnosis approaches must be developed. Also on levels 3 and 4, it is possible that the driver tries to intervene the ADSs. These takeover situations must be reliable detected and the control must be handed over in a smooth transition.

1.2. Challenges in Automated Driving Systems

Several road authorities and OEMs published their vision on safety (General Motors 2018; Aptiv et al. 2019; NHTSA 2017; Waymo 2019). These documents provide recommendations and guidelines on how to design safe automated vehicles with high levels of redundancy. However, high levels of redundancy increases cost and complexity and might not be the optimal solution for a research and development platform. The actual challenge in designing a research and development platform is designing a system that achieves a sufficient reliability level and includes self-monitoring mechanisms.

All the examples in Section 1.1.2 show why it has the utmost importance to design a safe and robust ADSs. Succes of a new technology, such as ADSs, can be prohibited by one example of neglecting safety. Higher levels of safety and reliability of environmental perception systems may be achieved by including information from other sources, such as wireless communication. The challenge that yet has to be solved is the design of safety mechanisms that include the additional information. The examples also show why testing must be performed in a careful and safe manner in which, for example, user interventions must be reliable and fast detected.

Especially with safety-critical systems such as ADSs, verification of the safety mechanisms is desired before starting road tests. Verification strategies of ADSs can consists of using virtual prototypes and closed-course testing. To limit the time for verification, a layered simulation approach is desired in which the right level of model fidelity can be selected based on the to be tested use case since running lower fidelity models is usually less time-consuming.

1.3. Research Objectives and Contributions

Given the aforementioned challenges, this thesis aims to contribute to the safety aspect of automated vehicles, in particular on how to develop a safe and reliable test bed for SAE level 3 and beyond driving. In addition, fault detection strategies are pursued to, for example, detect user interventions or when incorrect information is received from preceding vehicles. Consequently, the following research objectives are defined:

- develop a systematic design approach of a safe and robust research and development platform by identifying the relevant components for the design and taking into account the safety aspects required for SAE level 3 and beyond driving;
- build the research and development platform and verify the design, thereby ensuring that the automated vehicle is safe to operate;
- develop and validate a simulation framework for testing and validating automated driving controllers;
- develop fault diagnosis algorithms to detect actuator faults and user interventions, and deploy and operate them on the research and development platform; and
- develop fault diagnosis strategies for connected vehicles, including environmental perception systems.

These objectives are addressed through the following contributions:

1. **Safe research and development platform design**
   The first contribution is a systematic design approach in developing additional features required for automated driving, guaranteeing safe and reliable vehicle operation. This starts in Chapter 2 with identifying the requirements for an experimental platform and based on that proposes a set of additional sensors and actuators. Related to this, the communication architecture with diagnosis capability is discussed. A functional software architecture is developed in Chapter 3 to structure the software development. This architecture ensures safe vehicle operation and handles the automated mode based on user inputs.

2. **Design verification of automated vehicles**
   The second contribution covers the aspect of design verification strategies for automated vehicles by developing a simulation framework for safety testing and validating automated driving controllers. The framework, which is described in Chapter 4, uses a virtual prototype of the research and development platform. The developed simulation framework can be used in two different simulation modes: accelerated-time or real-time, which makes it suitable for requirement generation and refinement, design verification, and automated controller performance evaluation.

3. **Fault diagnosis of actuators**
   Next to hardware and software faults, an automated vehicle (levels 3 and 4) needs to detect when a user tries to regain control of the vehicle. Therefore, in Chapter 5, a method is proposed to detect actuator faults and user interventions, employing model-based fault diagnosis techniques. The specific use case here is the steering system. The potential of the developed methods are illustrated using computer simulations as well as experiments. The developed methods enable fast and reliable detection of user interventions.

4. **Fault diagnosis for connected vehicles**
   Finally, this thesis contributes to the safety of connected vehicles. Connected vehicles employ a wireless link to share information to other traffic participants. The wireless link allows vehicles to drive with small inter-vehicle distances, thereby increasing the road capacity. At these reduced intervals, the human driver cannot be considered as backup and the system must supervise itself to avoid unsafe driving situations. Therefore, Chapter 6 presents a method used to detect when incorrect information is received from a preceding vehicle or when a wrong object is tracked. The presented method applies model-based fault diagnosis principles. Computer simulations as well as experimental results show that the system is able to timely detect when incorrect information is received from the preceding vehicle or when a wrong object is tracked.

1.4. Outline

The organization of the thesis can be presented in a V-cycle design process as shown in Figure 1.4. Chapter 2 starts with a detailed overview of the base vehicle selected for the research and development platform to determine the required changes to achieve full vehicle automation. Based on that, additional hardware with integrated safety features have been selected and are installed in the vehicle.

Chapter 3 presents the real-time control application which contains all the algorithms for the automated driving platform, such as host tracking, target

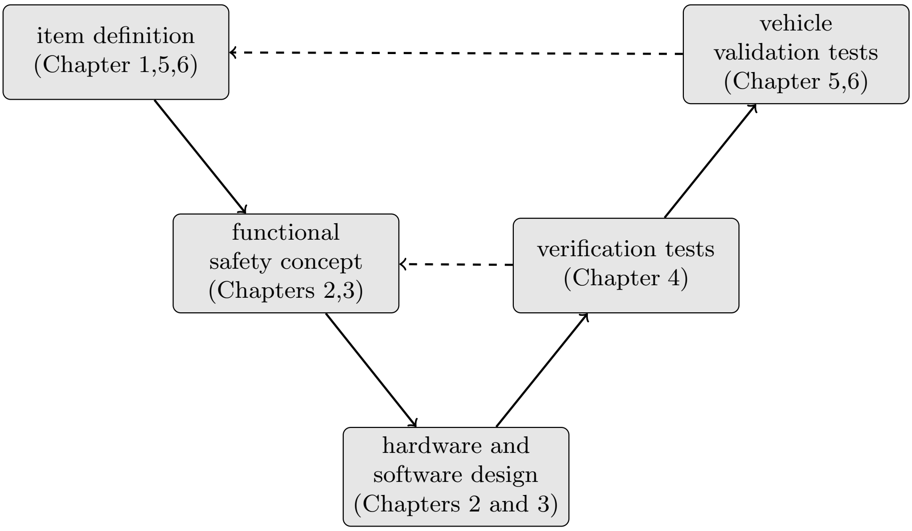


Figure 1.4.: Overview of the thesis presented as a V-cycle design process

tracking, and vehicle control. A functional software architecture is presented to structure the real-time control application. Fault injection tests are conducted to verify the implemented design. Chapters 2 and 3 have been combined and are under review for journal publication.

Chapter 4 covers the development of a simulation framework for safety testing and validating automated driving controllers. The created simulation framework can be used in two configurations: accelerated-time in which (a part of) the control system is integrated in the simulation environment and real-time in which the control system is separate from the simulation environment. The layered simulation approach enables selection of simulation fidelity based on the use case, thereby reducing simulation and development time.

Chapter 5, which is based on Hoogeboom et al. (2018), focusses on fault diagnosis of an automated vehicle. Model-based fault diagnosis is applied to detect user-interventions as well as actuator faults.

Chapter 6 focusses on fault diagnosis for connected vehicles. Connected vehicles allow driving with short inter-vehicle distances due to the wireless received information. When the wireless link fails or wrong information is received, this results in an inherently unsafe situation.

Finally, Chapter 7 summarizes the main conclusions of this thesis and provides recommendations for further research into these topics.

2.

Development Vehicle Instrumentation

Abstract - Many vehicles sold today are capable of some level of automated driving. This number is expected to grow rapidly in the near future as automated driving promises many benefits, such as improved road safety and reduced traffic congestions. A research and development platform is required to validate and demonstrate technical feasibility of the developed automated driving algorithms. This chapter presents a systematic design approach to convert a small electric vehicle into a demonstrator/test platform with as objective to comply with SAE level-3 driving. As base vehicle, a Renault Twizy is selected, which performs the basic automotive features, such as drive, brake, and steering, but has no additional on-board intelligence or actuators. Since no information was shared by the manufacturer, steps are presented to decode the original network structure, consisting of identifying the original functionality, locating the control units, and decoding the messages being communicated. The additional hardware mounted in the vehicle to obtain the required automated functionality is discussed, and their safety features are described.

2.1. Introduction

Automated driving has made a tremendous progress over the past decade as it has many potential benefits to society, such as reducing the number of traffic accidents (NHTSA 2016) and improving driver comfort. The level of vehicle automation in automated cars varies from systems that assist the driver (Advanced Driver-Assistance System (ADAS)) to systems that entirely take over control of the vehicle (Automated Driving System (ADS)).

ADSs actuate the vehicle in longitudinal and lateral direction and therefore control of the drive, brake, and steering system is required. In addition to dynamic vehicle actuation, control of auxiliary vehicle functions is desired, such as indicator signals or brake lights. Normally, these control commands are communicated over bus communication structures on vehicles with drive-by-wire technology. Hence, drive-by-wire vehicles can easily be controlled by an ADS by sending the right messages over the bus. This requires system knowledge which is proprietary information of the Original Equipment Manufacturers (OEMs).

To enable drive-by-wire on existing vehicles without support from an OEM, hacking or decoding of existing systems is required, such as self-parking or adaptive cruise control. Dataspeed (2018), for example, converts a selected set of American-made vehicles to drive-by-wire capable vehicles that can be used to develop and test automated vehicle functionality. Similarly, AutonomouStuff (2019) delivers drive-by-wire controls for a Lexus RX450h. They provide control to core driving functions as well as ancillary components.

However, when manipulating inputs, safety features of the existing system might be bypassed and safety of the overall system can not be guaranteed. In addition, the permitted control actions of the existing system can be limited. For example, a lane-keeping system designed for highway driving normally requires small steer corrections with low torque, while parking maneuvers at low speed usually require large steering angles with high torque. When the lane-keeping system is used to perform parking maneuvers, the permitted controls can be limited and the system might react inadequate. An ADS must be aware when the permitted controls are limited otherwise the implemented controllers may show unpredictable and/or unstable behavior. In addition, large time delays are not uncommon in commercial system and these delays can significantly reduce control performance or even result in instability when not taken into account.

Therefore, unlike many other automated vehicles on the market, a simple vehicle has been selected as basis for the research platform that does not have any intelligence or any of the common ADAS functionalities. This vehicle—a Renault Twizy (Renault 2015)—was selected because of its simple design, electric drivetrain, and lack of advanced safety features. The body consists of a space-frame with clip-on panels, enabling installment and access of additional hardware. The maximum speed of the vehicle is 80 km/h, and the electric range is specified as 90 km on one charge which should be suitable for a normal day of testing.

The aim of this chapter is to present a systematic design approach in developing the add-on features required for automated driving guaranteeing a safe and reliable vehicle operation. It provides insight into the required functionality for automated driving, and it describes the base vehicle plus the additional instrumentation needed to convert this vehicle into an automated vehicle. The integrated safety features in the implemented hardware are explained as well.

The outline of this chapter is as follows. Section 2.2 describes the vehicle that is used as basis for the research and development platform. Section 2.3 lists the hardware requirements for the automated driving platform. Next, Section 2.4 describes the additional instrumentation required to convert the basis vehicle into an automated vehicle. Finally, Section 2.5 summarizes the main conclusions.

2.2. Renault Twizy

The basis vehicle used as research and development platform is a Renault Twizy, which is a small two-seated electric vehicle. This vehicle has to be modified to make it suitable for automated driving. As the Renault Twizy can be considered as a generic vehicle without any intelligence, the approach to convert this vehicle to an automated vehicle can be applied to any vehicle. When making changes to a vehicle, it is required to know how the vehicle works. Therefore, as a first step identification of the control units and network structure is required. This section explains the approach. The first step is to identify the control units and their functionality. After that, the Controller Area Network (CAN) messages communicated among these control units can be decoded.

2.2.1. Control units

To identify the hardware components and their functionality, a list of the original control units can be obtained from the manufacturer together with their locations and communication interfaces. Figure 2.1 shows the hardware components of a Renault Twizy with their locations and communication interfaces. The original equipment level and communication structure of a Renault Twizy is simple compared to most modern vehicles, as some modern vehicles can have up to 100 control units (Möller and Haas 2019). The advantage of the limited equipment level is that less problems are expected with interference between modules compared to most modern vehicles. The following subsections explain the components in more detail. This detailed information is necessary to decode the CAN messages as it must be clear what information can be expected on the CAN bus.

Power Electronics Block

The Power Electronics Block (PEB) is mounted next to the electric drive motor and contains the power electronics that drive the three main coils of the 13 kW (17 hp) electric drive motor. The PEB is a Sevcon Gen4 (Sevcon 2012) controller that uses CAN bus communication to communicate with other devices in a Renault Twizy. An accelerator pedal is used to inform the PEB about the requested drive torque. The pedal sensor contains two potentiometers with different gains and offsets, enabling sensor diagnosis.

The PEB controller uses up to 330 A or 16 kW in standard configuration and recuperates up to 70 A or 3.5 kW. These and other configuration parameters can be changed since the device uses CANopen, which is higher layer protocol based on CAN. The CANopen protocol enables communication between nodes of different manufactures. The CANopen protocol defines a set of device and application profiles and every CANopen device is described by an object dictionary.

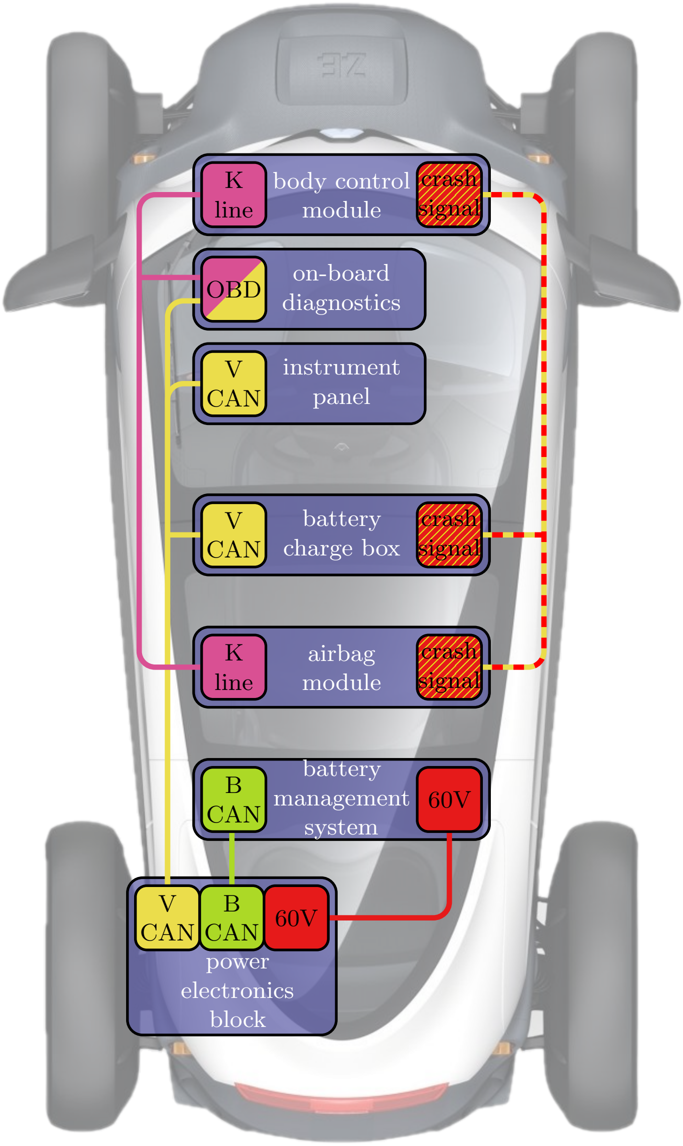


Figure 2.1.: Overview of the Renault Twizy's original hardware components with their installed locations and corresponding communication interfaces: vehicle CAN bus (🟨), battery CAN bus (🟩), crash signal (🟥🟨)

This register has the same structure for all CANopen devices and contains the configuration and communication parameters. The indices in the object dictionary can be read and, depending on access rights, also be written. The powertrain characteristics in terms of regenerative behavior have been changed by making changes to the object dictionary (OVMS 2018; Jafarnejad et al. 2015) to make them more suitable for automated driving.

Body Control Module

The Body Control Module (BCM) is mounted above the steering column under the dashboard and controls and monitors various body electronic functions, such as the exterior lights and windscreen wipers. The BCM also checks whether the right coded ignition-starter key is being used to prevent unauthorised vehicle use. The BCM is not connected to the CAN network and uses a K-line for diagnostic purposes. This means that to be able to control the indicator and brake lights this cannot be achieved by software alone. Hardware changes are necessary to be able to control the indicator and brake lights.

Battery Charge Box

The Battery Charge Box (BCB) is mounted in front of the traction battery case under the floor and can charge both the high-voltage (60 V) as well as the low-voltage (12 V) battery. The high-voltage system is used for the driveline, while the low-voltage system is used to power other electrical components, such as the heated windscreen, radio, and lighting. The BCB is an Elips 2000W (ies-synergy 2018) and can charge the high-voltage battery with a maximum of 60 V and 40 A when connected to a power outlet. The BCB can charge the low-voltage battery with a maximum of 12 V and 25 A while driving. This power limit must be taken into account when selecting the additional hardware.

Airbag computer

The airbag computer is mounted underneath the seat in front of the electric motor. The airbag computer uses K-line and is connected to the instrument panel for diagnostic purposes. The airbag computer is also connected to the BCM and BCB such that in case of severe accident, both the BCM and BCB are informed. The BCB will respond to this by opening the main relay, thereby disconnecting the power supply to the other components.

Battery Management System

The Battery Management System (BMS) is included in the traction battery case underneath the seat together with the 6.1 kWh lithium-ion 58 V traction battery.

The BMS monitors the cell voltages and temperatures of the 42 battery cells, which are arranged in seven modules of six cells each. Based on these parameters, the BMS communicates the maximum allowed drive and regeneration currents to the PEB via the battery CAN bus, and the PEB broadcasts this information on the vehicle CAN bus.

Instrument panel

An instrument panel is mounted on top of the dashboard. This panel informs the driver about vehicle states, such as velocity, battery state of charge, and estimated range. The instrument panel obtains these vehicle states via the CAN bus. The display also contains warning lights for the different nodes to notify the user in case of a fault.

2.2.2. Decoding CAN messages

With the components and their functionality identified, the remaining part is to decode the CAN messages being communicated between these devices. While CAN bus is a standardized protocol, the CAN message IDs and signal information are manufacturer proprietary information. Therefore, decoding of the CAN messages is necessary to extract the required signals, such as motor speed, selected gear, and brake status. The following five step approach can be used to decode the CAN messages (Hermans et al. 2009):

(i) research, i.e., investigating which control units are present in the CAN bus;

(ii) interface, i.e., making the physical connection to the CAN bus with a CAN tool;

(iii) linking identifiers, i.e., disconnecting control units one by one to determine which identifiers belongs to the control units;

(iv) finding essential data, i.e., identifying data that must be present from a functional point of view; and

(v) finding the meaning of unknown data.

The decoded CAN network information can be stored in a DBC file. This is a CAN database format, containing the network nodes, CAN messages and signals in these messages, signal bit count, physical units and linear conversion formulas.

2.3. Automated Driving Platform Hardware Requirements

The developed vehicle is a research platform which means that the hardware and software setup evolves over time. The followed design process is therefore not completely according to ISO 26262, but the most important tools of ISO 26262 are employed. During the concept phase of ISO 26262, an item definition should be made, which gives an overview of the function, its nominal behavior, system limits, as well as its interfaces. The developed vehicle will be used for automated driving in an urban environment with speeds up to 50 km/h. The aim of the research and development platform is to achieve Society of Automotive Engineers (SAE) level-3 driving, which means that the automated driving platform should monitor its own performance and notify the user in case of a fault, such that the fallback-ready user can take over control of the vehicle. In other words, fault detection should be included in the ADS, and the vehicle always has to be occupied with a fallback-ready user that can intervene or take over control in case of a fault.

Based on the item definition and safety analysis, a functional safety concept is formulated. This concept is refined during the development phase into the technical safety concept and the hardware requirements. The six hardware requirements for the automated driving platform are:

(i) fall back to original functionality must always be possible, which means that no invasive changes to vehicle CAN bus and/or drive, brake, and steering system are allowed;

(ii) control of longitudinal and lateral vehicle motion, which means that for the selected basis vehicle accelerator pedal override and add-on steer and brake actuator are required;

(iii) add-on hardware components must have fault diagnosis capability, which means that the components should have integrated self-diagnosis functions, such as range checks or redundancy;

(iv) all vehicle control sensors and actuators must have CAN bus communication. CAN bus communication is used for its extendibility and high data transfer reliability properties;

(v) the vehicle should always be occupied by two persons: one to act as a fallback-ready user who monitors the driving environment and which can intervene when necessary. The second person controls and monitors the automated driving system to evaluate whether everything functions as it should. As such, the fallback-ready user can never be distracted from monitoring the driving environment; and

(vi) for development purposes and maintenance, the implemented components should be accessible for updates and/or repairs.

In addition to the requirements, there is a strong preference to use Commercial Off-The-Shelf (COTS) components. By using COTS products, time and cost can be reduced involved with designing and testing custom solutions. However, some custom solutions are inevitable for complex systems such as a research and development platform for automated driving.

2.4. Additional Vehicle Instrumentation

Based on the hardware requirements, a considerable amount of additional hardware is needed to realize the research and development vehicle. Storing these components is normally not a problem for passenger cars as the trunk provides significant space. The selected basis vehicle, however, has limited space to store additional components. Borraz et al. (2018) solved the additional instrumentation placement issue by sacrificing the back seat of the Renault Twizy to place a rack with control and decision-making systems and batteries to power these systems. Similarly, New Atlas (2018) adapted the back seat of the vehicle to make space for the additional components. Sacrificing the back seat was no option for the research and development platform as it is a safety requirement to occupy the vehicle with two persons. Therefore, all additional hardware components are integrated in various places, as is schematically shown in Figure 2.2. The components are explained in more detail in the following subsections.

Real-time computer

To implement the control software, a real-time computer is required. This computer executes the real-time application that processes sensor information, performs calculations on it, and controls the actuators based on the processed information. As real-time computer, an Advantech ARK-3520P is selected (Advantech 2019). This fanless embedded PC is mounted behind the passenger seat in a waterproof box. As operating system, Simulink RealTime is selected, which provides hard real-time guarantees (Buttazzo 2011) and allows rapid prototyping in a graphical environment. The real-time application runs at 100 Hz, equaling the update rate of the fastest sensors. Two Softing CAN-AC2-PCI (Softing 2012) interface boards are installed in the real-time computer to enable CAN bus communication. Each hardware interface board supports two independent CAN channels. The other available and supported interfaces on the real-time computer are: three Ethernet ports, four serial RS-232 ports, and four configurable RS-232/422/485 ports.

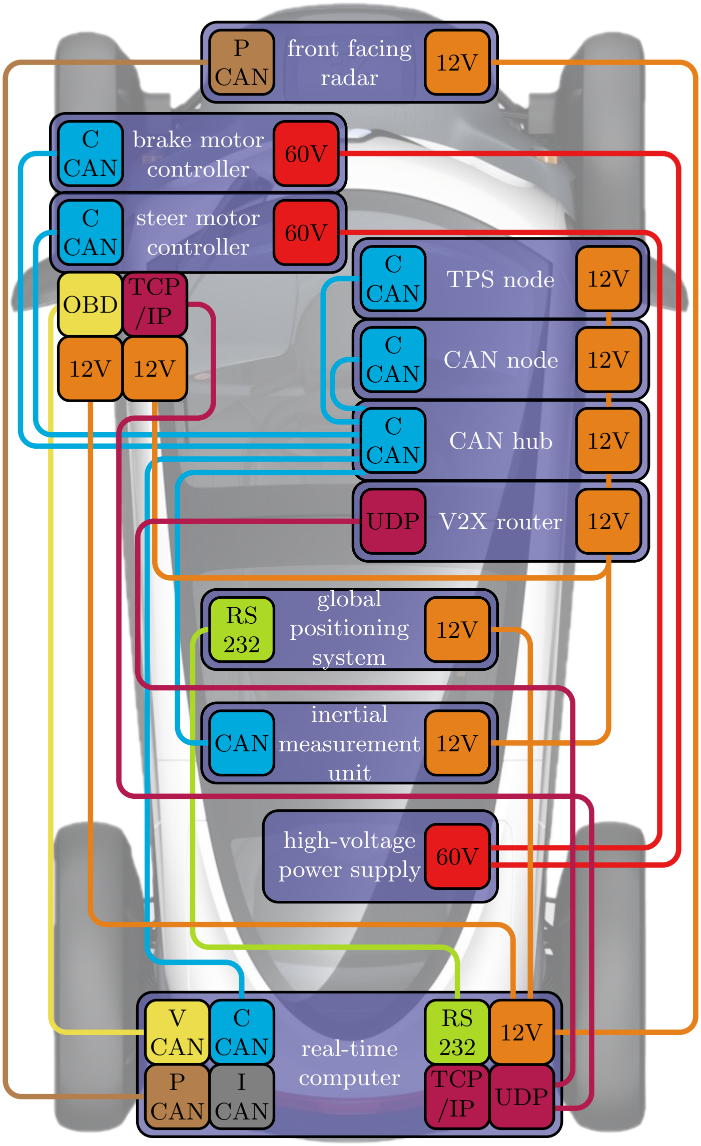


Figure 2.2.: Overview of the additional hardware components of the Renault Twizy with their installed locations, power supply, and corresponding communication interfaces: vehicle CAN bus (▬), control CAN bus (▬), perception CAN bus (▬), RS232 (▬)

The CAN busses in the vehicle are split based on functionality. The following four CAN busses are defined:

(i) vehicle CAN bus (V-CAN), which is the native vehicle CAN bus, as described in Section 2.2.2;

(ii) control CAN bus (C-CAN), which interfaces all the components used for the drive-by-wire functionality;

(iii) perception CAN bus (P-CAN), which interfaces with a front mounted radar sensor; and

(iv) instrumentation CAN bus (I-CAN), which interfaces with additional devices such as a Nvidia DrivePX2 (not considered in this thesis).

As a consequence, when, for example, a faulty component grounds a CAN bus, the other CAN busses are not influenced. The real-time computer is connected to all of them and can act as a gateway relaying information from one bus to another. The real-time computer is silent on the vehicle CAN bus meaning that it only listens and it adds no information. This is done to avoid any possible interference with other nodes on the vehicle bus which might result in erroneous or unsafe behavior.

CAN hub

All components on a CAN bus need to be physically connected to that bus and the bus must be terminated on both ends with 120 Ω resistors. For a research and development vehicle, the number of components connected to a bus and also the bus routing might change as the project develops. Therefore, to make an adaptable bus design, a CAN hub is developed that connects all C-CAN devices. The other CAN busses are point-to-point connections and do not need extra connections. The CAN hub board can connect up to 11 devices in a star connection and is mounted in the right-hand storage compartment. Each connector contains a CAN bus, 12 V power, and a reset signal. The board contains a debug/service connector on top of the board for diagnostic purposes on which a CAN logger can be connected to monitor the CAN bus.

CAN node

Besides the serial communication ports, the real-time computer does not have other I/O capabilities. Therefore, a CAN node is developed which acts as a CAN to I/O interface for the real-time computer. This CAN node is mounted behind the right headlamp next to the CAN hub. The CAN node controls the status LEDs for the fallback-ready user, high-voltage power take-off relay, brake light relay, and reads and perform diagnosis of the analog sensor signals, such as steering torque and brake pressure signal. The node checks whether the signals lie in their specified range, and if available, compares it with a redundant signal.

Drive system

To control the drivetrain, accelerator pedal take-over is required. An accelerator pedal position sensor normally consists of two independent signal lines with different values used for fault detection. Therefore, a take-over module is required which can emulate the accelerator pedal position sensor signal. The take-over module must have two switched outputs that can be connected to the control platform or to the accelerator pedal, as fall back to original functionality is a basic safety requirement (see Section 2.3 requirement (i)).

There are no COTS solutions available as the sensor signal range of the basis vehicle (0 V to 10 V) was different than normal vehicles (0 V to 5 V). Therefore, a custom solution is designed. The designed take-over module contains two take-over relays to switch between normal driving, where the accelerator pedal signal is connected to the PEB, or automated driving, where the control signal is connected to the PEB. The take-over relays defaults to the accelerator pedal so that in case of power outage or fault, the vehicle can still be normally driven. The take-over module is mounted behind the right headlamp and is connected to the real-time computer via CAN bus. Integrated self-diagnosis functions and watchdogs timers are implemented to guarantee safe behavior. The input values of the accelerator pedal as well as the relay output values are measured and broadcasted on the CAN bus to be able to detect user interventions and to perform fault diagnosis of the TPS node.

Steering system

To control the lateral motion of the vehicle, control of the steering system is required. When a vehicle is equipped with electrical power steering, this motor can be used to control the steering system (see e.g., Loof 2018). A Renault Twizy does not have power steering and it can therefore not be electrically controlled by making use of the power-steering motor. To make the steering column suitable for control, a custom solution would be to mount an electric motor with gears to the steering column, similar to Borraz et al. (2018). However, there is an off-the-shelf solution that makes use of the steering column of a Renault Clio II, which has column type electric power-steering assist. This column fits in the Renault Twizy with some minor modifications.

Figure 2.3 shows an overview of the new steering system, consisting of a steering wheel, steering column, and steering rack. The original steering wheel, steering axis, and steering rack have not been modified. The electric motor is connected to the steering column via a worm gear. To measure the torque exerted on the steering wheel, the steering column contains a torsion bar with a redundant torque sensor system. The two analog sensor outputs are processed by the CAN node which compares both signals and check whether they lie in their specified range. The steering torque and sensor status are broadcasted on the CAN bus.

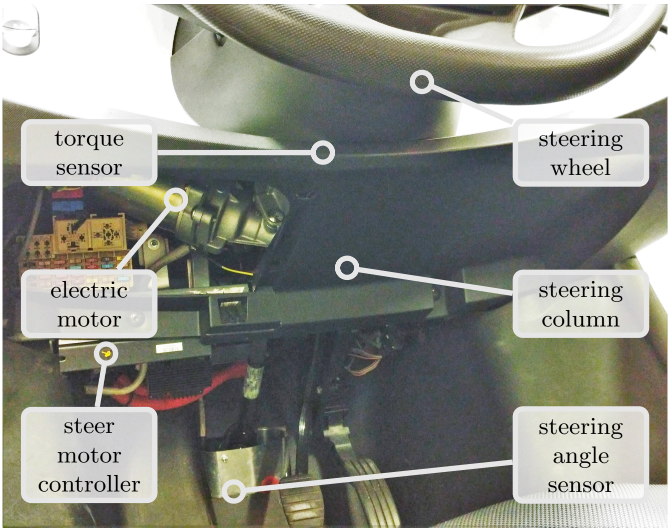


Figure 2.3.: Overview of the steering system

A steering angle sensor is mounted on the steering rack to measure the steering angle and angular velocity. The sensor is a Bosch LWS3 (Bosch 2014) which uses two Anisotropic Magneto Resistance (AMR) elements both attached to the steering column with a different gear ratio. By making use of the Nonius/Vernier principle, the steering angle can be unambiguously measured over more than four full steering wheel rotations with an accuracy of 0.1° and angular velocity resolution of 4 °/s. The sensor is a fault tolerant sensor which means that if one of the AMR elements fails the sensor is still able to provide a measurement of the steering angle. Sensor evaluation and fault detection is performed by two separate micro controllers which supervise each other. The steering angle and velocity as well as sensor status are communicated via CAN.

To drive the electric power steering motor, a motor controller is used. The controller is a Maxon Epos2 70/10 (Maxon motor 2017) which can supply 10 A continuous and up to 20 A peak. The controller is connected to the high-voltage system of the Renault Twizy as the low-voltage power supply is limited. Communication between the amplifier and the real-time control platform is performed through CANopen.

Brake system

Besides control of the driveline, actuation of the brakes is required to control the vehicle in longitudinal motion. In case a vehicle has an Electronic Stability Control (ESC) or Electronic Hydraulic Brake (EHB) system, this can be used to control the braking system. The Renault Twizy does not have an ESC or an EHB system and thus control of the brake system requires hardware modifications. Instead of fitting it with an ESC or an EHB system, a brake actuator is added to the brake master cylinder. This keeps the original system intact, while mounting an ESC or EHB system would require modifications to the brake system.

Figure 2.4 shows an overview of the brake system, consisting of a brake motor that is connected to a cam-follower mechanism via a gearbox. The advantage of this system is that it does not alter the original brake system, and it is not directly coupled to the brake pedal system. This means that the cam follower mechanism can only apply pressure on the brake master cylinder. It can not pull the brake master cylinder. As a consequence, the fallback-ready user can always operate the brakes more without additional effort compared to the original situation, even with automated mode engaged. In case the brake motor or gearbox fails during a braking operation and the camshaft is locked, the vehicle will slow down to a full stop. This is assumed to be a safe situation.

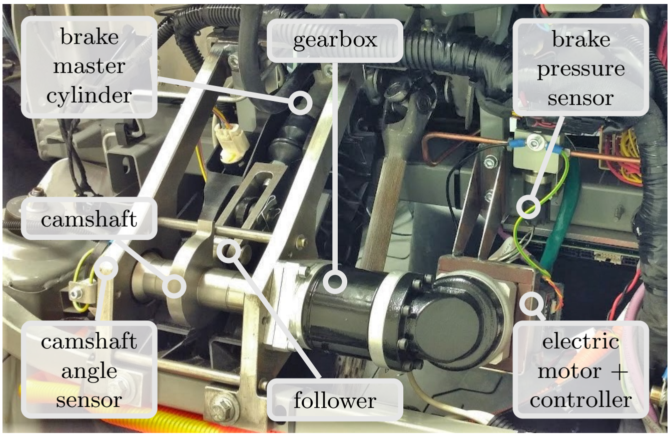


Figure 2.4.: Overview of the brake system assembly

The rotation angle of the camshaft is required to control the brake system. This angle can be measured with the brake motor encoder. This is, however, a relative sensor and it will always start at zero after a power cycle. Therefore, an absolute sensor is added similar to the one that is used for the steering system. This avoids problems with power outages during operation. Additionally, the two sensors are compared to detect mechanical failures of the brake system.

In addition to the camshaft rotation angle, the brake pressure is also used in the control loop of the brake system. The brake pressure is measured with a Bosch DS brake pressure sensor (Bosch 2018b, p. 70). The measurement range of this sensor is 0 bar to 250 bar with an accuracy of ≤ 0.7 percent in the range 0 bar to 35 bar and an accuracy of ≤ 5 percent in the range 35 bar to 250 bar. The brake pressure sensor uses two internal signal paths, enabling the detection of offset and amplification errors. In addition, several other self-diagnosis functions are included. A CAN node processes the analog sensor output and places the brake pressure and sensor status on the CAN bus.

To drive the camshaft, a Moons TSM23C (Applied Motion Products 2013) step servo motor is used. This motor combines the motor and controller in one package. The motor output is coupled to the camshaft through a gearbox, amplifying motor torque. The Moons TSM23C is connected to the real-time computer via CAN bus and utilizes the CANopen protocol.

ETSI ITS-G5 modem

To connect with other vehicles, a communication module (Severinson 2018) is mounted next to the CAN hub. The router is a PC Engines APU2 computer running the open-source software GeoNetworking (Voronov et al. 2016). The software stack facilitates IEEE 802.11p wireless communication compliant with the ETSI ITS-G5 standard (ETSI 2019). The V2X router support Cooperative Awareness Message (CAM), Decentralized Environmental Notification Messages (DENM), and custom messages. The CAM message is periodically broadcasted to other vehicles in the neighborhood and contains basic vehicle information, such as position, velocity, and acceleration. The DENM message is event-triggered and is used to alert road users in case of a hazardous event, such as a passing emergency vehicle.

GNSS receiver

Automated driving requires precise knowledge of the vehicle's pose in terms of position, velocity, and attitude to guide the vehicle along a certain path or trajectory. The required information in terms of a navigation solution can be obtained from a Global Navigation Satellite System (GNSS) receiver. As GNSS receiver, an u-blox EVK M8T (u-blox 2018a) is used and is mounted under the roof. The GNSS update frequency is set to 5 Hz and is communicated with the real-time computer via a serial RS-232 interface. The acquired precision of the module is improved from 3 m to 2.4 m (2σ) by using European Geostationary

Navigation Overlay Service (EGNOS) and an improved antenna. To increase the navigation solution accuracy further without using a costly RTK-GPS, sensor fusion of the GNSS and inertial sensors is applied. This is discussed in Chapter 6.

To synchronize a local clock on the real-time computer to the absolute time, the GNSS receiver provides a precise digital time pulse every second. The local clock can be used to determine the message age of the wireless received messages as well as the GNSS time delay resulting from data transmission and internal processing.

IMU

To measure the inertial signals, an Inertial Measurement Unit (IMU) is mounted underneath the driver seat near the center of gravity. The IMU is a Bosch MM5.10 (Bosch 2018a) and measures the following five signals: yaw rate, roll rate, longitudinal, lateral, and vertical acceleration. The signals are internally low-pass filtered at 15 Hz. The resulting signals and sensor status are communicated via CAN bus.

Front facing radar

To detect preceding vehicles and other objects in front of the vehicle, a front facing radar sensor is mounted at the center of the vehicle behind the front bumper. The radar sensor is a Bosch MMRevo14 (Bosch 2015) and measures the relative distance, velocity, and acceleration of objects. The radar has a maximum range of 160 m and uses four independent receiving channels and digital beam forming. Figure 2.5 visualizes the sensor coverage. The narrow main antennas are used to detect vehicles driving in front at long range, and the wide close-range elevation antennas are used to detect objects at close range. The height and vertical position of objects can be measured using the elevation antenna, enabling reliable tracking and classification of objects, even when the objects are stationary. The radar is able to distinguish the following objects: unknown, car, motorcycle, pedestrian, and construction element. A maximum of 32 objects can be simultaneously detected and tracked. The objects are communicated via CAN bus to the real-time platform with a refresh time of roughly 60 ms.

Power supply

All additional components need a power supply. The power supply to additional components is switched to ensure that no unexpected events can happen, for example, during maintenance or when driving the car manually. The low-voltage power supply of the Renault Twizy has limited electrical power, as explained in Section 2.2.1, and is not sufficient to power both the steer and brake actuators.

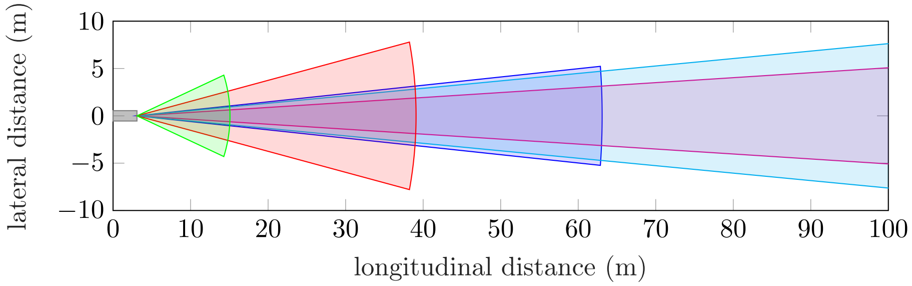


Figure 2.5.: Radar sensor coverage visualization: ego vehicle outline (⬜), main antenna (🟦, 🟪, 🟦), and elevation antenna (🟥, 🟩)

Therefore, both the steer and brake actuators are powered from the high-voltage system. To this end, a fused relay box is mounted near the PEB and connected to the PEB motor output. The relay box outputs going to the brake and steer system are mounted within small boxed sections on the right side of the vehicle to prevent interference with the communication and signal lines routed on the left side of the vehicle.

The CAN hub distributes power to the other additional components besides the real-time computer. The CAN hub itself is powered via the original accessoires supply lead, which is fused and ignition-switched. An additional switch is added in series with the supply wire and mounted on the operator panel to be able to power-off all additional hardware in case of manual driving. The real-time computer is powered by a fused constant 12 V power supply. Similarly, an extra switch is added in series with the supply wire and mounted on the operator panel to control the power to the real-time computer. The reason to use a always-on power supply for the real-time computer is to be able to save and retrieve log files in case of an accident when the Renault Twizy powers itself off, as explained in Section 2.2.1.

Operator panel

To control and monitor the status and of the automated control system, an operator panel is required for the fallback-ready user. The operator panel is mounted on top of the left-hand storage compartment in view of the fallback-ready user, as can be seen in Figure 2.6. The operator panel contains two toggle switches: one to power the real-time computer and the other one to power the additional devices via the CAN hub. In addition, a red emergency switch is mounted that can be operated in case of an emergency situation. This switch is connected in series with the CAN hub. When the emergency switch is operated, the CAN hub sends a reset signal to the TPS and CAN node putting them in emergency reset mode, which disables their output. As a consequence, the high-voltage power take off is disconnected as the CAN node controls the high-voltage power take off relay.

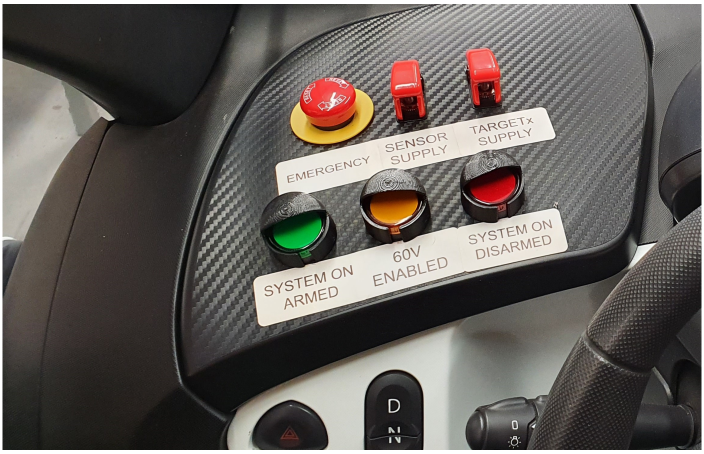


Figure 2.6.: Operator panel for the fallback-ready user with LEDs to indicate the automated control system status

To inform the fallback-ready user of the automated driving system status, the operator panel contains three LEDs. A orange LED is wired to the high-voltage power take-off relay and indicates whether the 60 V power take-off is enabled. The red LED indicates ‘system on and disarmed’ and turns automatically on when the system is booted. The green LED indicates ‘system on and armed’ and goes on when any automated mode is selected. In that case, the red LED turns off. When the green LED is on, the system is active and the fallback-ready user must be alert and intervene when necessary.

Below the operator panel, an Ethernet connector is installed to connect a laptop to the real-time computer. This connection can be used by the back-seat passenger to monitor and control the automated drive system. It can also be used to upload the real-time application to the real-time computer.

2.5. Discussion

Practical implementation of an ADS requires a vehicle equipped with drive-by-wire technology and additional hardware. Starting with a vehicle that is equipped with automated features, such lane-keep assist and/or adaptive cruise control, can save time to implement drive-by-wire functionality but this requires knowledge of system limitations and message information. Without the support of an OEM, this information must be decoded, which is a time consuming task. The presented research and development vehicle is constructed by using a simple vehicle without any intelligence or any of the common ADAS functionalities as basis and equipping it with mainly COTS components. The presented systematic design approach and vehicle implementation provides valuable insights in how to construct a research and development vehicle for automated driving and can be used for other candidate vehicles.

3.

Safety Framework for Automated Driving

Abstract - The previous chapter provides an overview of the research and development vehicle and the additional hardware with its integrated safety features. To operate the automated vehicle, a real-time application is required that processes sensor signals, plans control actions, and executes these actions. The integrated hardware features are however not enough to guarantee safe operation of the automated vehicle. The software also needs to comply to certain standards. Therefore, an automated driving software framework is developed, consisting of a layered functional architecture. The layered approach with the safety framework enables safe development of automated driving functionality. The designed safety mechanisms are verified using fault injection tests.

3.1. Introduction

Automated driving vehicles are the subject of enormous research interest as they promise many benefits to society, such as improved road safety and new mobility solutions. Automated driving requires large amounts of software development as an Automated Driving System (ADS) must sense the environment, determine its exact position on a road or map, and need to determine which control actions to take.

As the complexity of the software increases with the amount of automated driving functionality, open software platforms and frameworks are emerging, enabling collaborations and reducing complexity. For example, Apollo (2019) provides an open platform to autonomous driving, consisting of an extensive collection of modules starting from automatic predefined GPS waypoints following to geo-fenced autonomous driving. The system requires a vehicle equipped with drive-by-wire technology and a selected set of hardware that depends on the chosen architecture. The selected set of hardware can be a combination of supported sensors, such as cameras, lidars, and radars. The current implementation heavily relies on lidar sensors to sense the environment, but they are also planning to release an Apollo lite version, which relies completely on cameras to achieve Society of Automotive

Engineers (SAE) level-4 driving. Similar to Apollo, Kato et al. (2018) is an open-source project to enable self-driving. Their software includes a rich set of packages and libraries for autonomous driving, including algorithms and packages for perception, mapping, localization, and planning. Similarly to the Apollo platform, a limited set of hardware is supported and the system requires at minimum 3D map data and lidar sensors. Also, NVIDIA (2019) launched an open platform series called ‘Drive’, which mainly focusses on environmental perception using deep learning techniques. Out of the box, only basic functionality is provided and a limited sensor set is supported. A software development kit is provided to speed up the development process. Next to highly automated driving, Comma.ai (2019) enables via retrofit a highway driving pilot using existing vehicles. The highway pilot system consists of a camera module and car interface. The car interface reads vehicle states and sends the control commands. The highway pilot runs completely on dedicated hardware but the software is open-source, such that changes to the implementation can easily be made.

All the discussed platforms provide automated functionality, but they assume a drive-by-wire capable vehicle. When this is not the case, vehicle controls (drive, steer, and brake) along with fault diagnosis, such as sensor and actuator diagnosis, must be implemented. The discussed platforms have architectures suitable for environmental perception, but these systems are less suitable for safety-critical systems requiring real-time guarantees. In addition, a safety layer and cooperative vehicle control is not included in most of the discussed platforms.

Therefore, a different approach has been chosen in this research to split the functionality related to vehicle control and environmental perception. The vehicle controls (drive, steer, and brake), including a safety layer, are implemented on a real-time computer, and the environmental perception will be implemented on other systems. To handle the software development in a structured way, a decomposition of the modules as well as data flows between them can be made. Such a decomposition is called a functional software architecture which improves readability, maintainability, and reliability (Faitelson et al. 2018).

The aim of this chapter is to describe the functional software architecture of the automated driving control platform. The architecture gives an overview the functional software building blocks for automated driving. By specifying the interactions and dividing functionality, software blocks can be separately developed and tested, encouraging collaborations. The systematic design approach also helps to prevent programming errors.

The outline of this chapter is as follows. Section 3.2 lists the software requirements for the automated driving platform. Section 3.3 describes the functional software architecture used on the real-time control platform. The software application is developed in MATLAB Simulink, which is a graphical environment providing an intuitive development suite for complex systems. Section 3.4 verifies the design of the safety framework by performing fault injection tests. Finally, Section 3.5 summarizes the results and provides conclusions.

3.2. Automated Driving Platform Software Requirements

Based on the item definition and safety analysis, a functional safety concept is formulated. During the development phase, this concept is refined into a technical safety concept and software requirements. The six software requirements for the automated driving platform are:

(i) the fallback-ready user must be able to take immediate control of the vehicle by stepping on the brake, accelerator, or applying a steering correction;

(ii) the vehicle must not alter its trajectory too quickly, such that the fallback-ready user is able to intervene when necessary;

(iii) the control system must prevent for the user unexpected start-up operations;

(iv) all data elements in the application must have units according to the SI system to prevent confusion and conversion errors;

(v) the control system must disable control in case of a fault in a sensor that is critical to the control system. The fallback-ready user must be notified of this event; and

(vi) the actuators must disable their outputs in case of a control system fault.

3.3. Functional Software Architecture

The selected functional software architecture for the real-time application is based on Serban et al. (2019), EB robinos (2017), and Kochanthara et al. (2020) and is shown in Figure 3.1. As can be seen, not all layers described by Serban et al. (2019) are present in the real-time application as some of the functionality required for automated driving will be implemented on other platforms, such as the Nivdia Drive PX 2. These platforms have architectures that are more suited to perform tasks related to environmental perception, such as image processing and sensor fusion.

The data stream in the diagram is from the left to right following a sense-plan-act structure, starting with the interface layer that receives and translates the

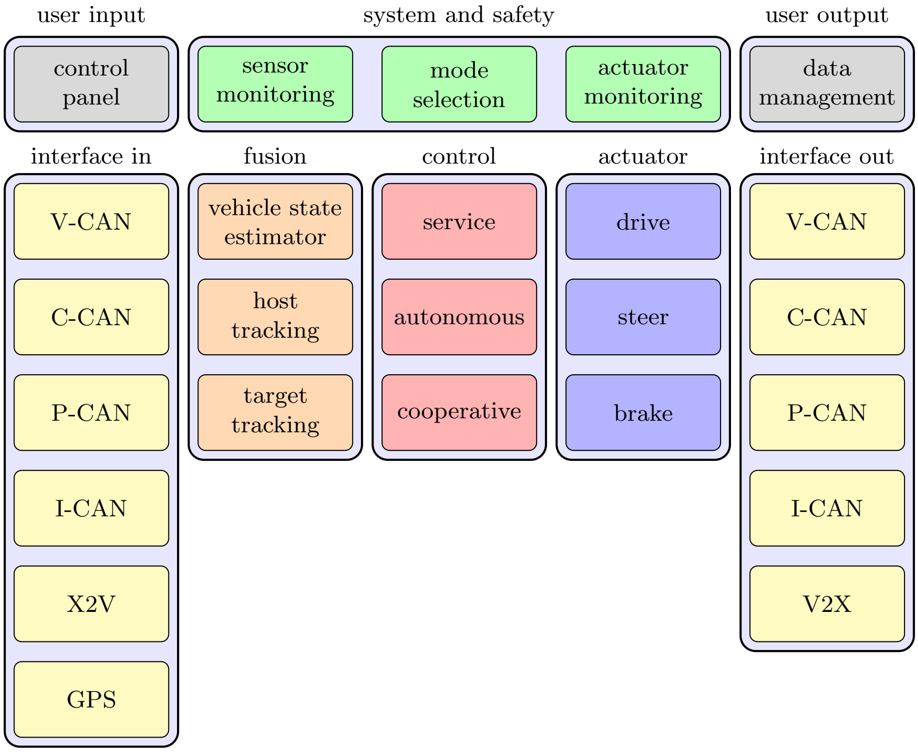


Figure 3.1.: Functional software architecture for real-time application, based on (Serban et al. 2019; EB robinos 2017)

received data into physical signals. Then, these signals are filtered and merged in the sensor fusion layer to produce consistent and accurate signals. Next, in the vehicle control layer, based on the filtered signals, a preceding vehicle is followed or a trajectory is planned and tracked (Van Hoek et al. 2018). After that, the by the vehicle control layer generated setpoints are transformed by the actuator layer into actuator setpoints. Finally, the interface layer converts the actuator setpoints into data again and transit them. The control model is available from (Hoogeboom 2020a).

3.3.1. Interface layer

The real-time application interfaces with other devices/computers by using communication channels. The interface layer converts the received data bytes into physical signals. In addition, the interface layer checks for correctness of the received data. The ISO 26262 part 5 Annex D (ISO 26262 2011) recommends a combination of information redundancy and a frame counter to detect communication bus faults.

The Controller Area Network (CAN) protocol uses the Cyclic Redundancy Check (Lawrenz 1997) to check for message correctness. In addition to this, all implemented sensors have a message counter and checksum in the data field. The message counter is used to detect whether messages are lost between the reception of two messages and the checksum is used to verify correctness of the received data bytes. The receiver re-calculates the checksum of the received message using a checksum algorithm and compares this with the received checksum. All implemented sensors have a message counter in the message to detect whether messages are lost between reception of two messages. A CAN node timeout is generated when five or more subsequent messages are lost or contain an invalid checksum. The threshold is set to five missed or incorrect messages to avoid false alarms due to, for example, bus timing issues. With the update rate of the sensors of 100 Hz, the response time of 0.05 s is fast enough in case of a real error given the timescales of the actuator dynamics.

The Global Navigation Satellite System (GNSS) receiver uses a serial RS-232 connection for data exchange. To encapsulate the message data, the UBX protocol is used (u-blox 2018b) which contains a checksum to verify whether the message is correctly received. The GNSS message does however not include a message counter. Therefore, a GNSS receiver error is triggered when no new messages are received for 0.5 s, which equals two missed or incorrect messages.

The V2X router employs a UDP connection to interface with the real-time computer. The message does not contain a checksum or message counter. Therefore, a V2X timeout is generated when no new messages are received for 0.1 s, which equals two missed messages.

Besides communication bus faults, signal conversion faults must be avoided in the interface layer. Therefore, all CAN interface blocks, which translate the received bytes to physical signals and physical signals to be transmitted bytes, are automatically generated using DBC files. Each CAN bus has its own DBC file and contains the network information, such as network nodes, CAN messages, and signals in these messages.

3.3.2. Sensor layer

The next layer is the sensor layer, which monitors the status of the sensors. Sensor fault diagnosis is required to guarantee safe behavior of the controlled outputs. As all C-CAN sensors have integrated self-diagnosis functions, only the communicated status has to be monitored.

3.3.3. Actuator layer

Besides sensor monitoring, the status of the actuators must also be monitored. Since both the steer and brake actuator use the CANopen protocol, the CANopen heartbeat protocol is utilized for this. This supervisory service requires every node to cyclically send a short heartbeat message that contains its current status to prove its communication ability. The time interval between these messages is the producer time and is set to 0.25 s for all CANopen nodes. The other nodes in the network can analyze these heartbeat messages to verify whether particular nodes relevant for them are online. The maximum time in which a heartbeat message is expected by a particular node is the consumer time. When the consumer time of a receiving node expires without receipt of the corresponding heartbeat message, the receiving node generates a timeout event and a fault is triggered. The brake and steer actuator are both configured as such that when two heartbeat messages from the real-time computer are missed, i.e., the consumer time of 0.5 s expires, they both go in fault mode. The output of both actuators is disabled in this mode.

A custom protocol is developed for the TPS and CAN node, containing a handshake initialization procedure and a time triggered message called lifeline. This message is comparable to the heartbeat protocol. The time triggered message contains the takeover command and is sent upon a takeover command change or when the timeout of 0.25 s expires. The handshake procedure prevents undesired behavior in case the real-time application hangs for some time and then eventually resumes. This can happen when, for example, the real-time computer is rebooted and the accelerator pedal override switch is still enabled on the remote control panel.

In case of hardware or software error, resulting in that the same CAN messages are retransmitted, the counter and checksum in the CAN message to the CAN and TPS node will not be updated. The CAN node will respond to this by signaling an error and opening the high-voltage relay. As a consequence, the steer and brake actuator are powered off. The TPS node will also respond to this by signaling an error and disables the accelerator pedal override, resulting in the vehicle returning to the manual mode.

To reduce the bus load of the C-CAN bus, a new setpoint for the actuators is only communicated when it differs from the previous time step. The actuators continue executing the previous setpoint when no new setpoint is communicated. In this way, the number of messages on the C-CAN bus is reduced.

The maximum torque delivered by the electric motor of the steering system is limited to make user interventions on the steering system possible. The fallback-ready user should always be able to counteract the steering wheel and therefore the maximum torque is limited to 10 Nm. This limit must be taken into account when designing the steering controller. To prevent for the user fast and unexpected steering events, the bandwidth of the steering controller is limited at 1 Hz, while a higher bandwidth would be possible.

3.3.4. System and safety management layer

On top of the model architecture, is the system and safety management layer. This layer monitors the status of the sensors and actuators as well as the user inputs. Based on these inputs, the system and safety management layer handles the mode selection. The following four modes are implemented in the real-time application:

(i) manual, in which the modified vehicle can be driven by the operator like a normal car, all ADS outputs are disabled;

(ii) service, in which the actuators can be controlled from a remote control panel to test functionality or evaluate control performances;

(iii) cooperative, in which drive, brake, and steering are automated to automatically follow a predecessor; and

(iv) autonomous, in which the drive, brake, and steering are automated to track a certain path (not considered in this thesis and the required automated functionality will have to be developed in the future).

Each mode has two sets of conditions, the first set of conditions are evaluated before the automated mode is engaged. For example, the service mode can only be engaged from standstill with the gear selector set in neutral. The second set of conditions are evaluated when an automated mode is engaged and checks whether it is still safe to be in that mode. For example, when a sensor fault or user intervention is detected the runtime conditions evaluate to true and the automated mode is disengaged. Switching between modes and unexpected operation is prevented by making sure that an automated mode can only be engaged by an explicit user action. All dynamic controller states are reset upon automated mode switch to prevent undesired effects, such as integrator windup (Astrom and Rundqwist 1989).

3.3.5. Other layers

The fusion, control, user input, and user output layers are open to automated driving system developers. With the safety framework and fallback-ready user in place, safe design and evaluation of automated driving algorithms is supported.

3.4. Software Design Verification

The software architecture has been implemented and the design has been tested by performing design reviews, unit tests, and full scale tests. Fault injection tests are performed to verify whether the designed and included safety features from the previous sections actually function as intended. As not all cases can be presented here, only two critical fault cases are considered. The first test is a faulty sensor that is critical to the control system and the second test is a real-time application crash which is similar to when it is powered-off during operation.

3.4.1. Camshaft angle sensor fault

In case a for the control system critical sensor is faulty, such as the brake camshaft angle sensor, the control system should respond by turning off the brake controller to avoid any unsafe behavior. In addition, it should inform the fallback-ready user to take over control of the brake system. To test this fault case, a checksum fault is injected at $t = 1\,\mathrm{s}$ to the brake camshaft angle sensor while the brake controller is on. Figure 3.2 shows the results of this test. As can be seen, a brake sensor fault is detected after $0.05\,\mathrm{s}$ which corresponds to five invalid messages. The safety system responds to this by disengaging the brake controller. When the injected fault is turned off, the brake sensor fault vanishes but the brake controller stays off. The brake controller must first be disabled on the remote control panel before it can be re-enabled to prevent unexpected behavior. The same principle is implemented for the drive and steering controllers.

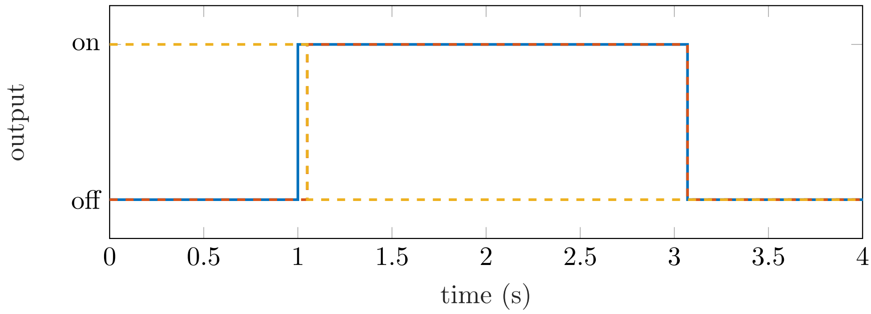


Figure 3.2.: Brake sensor fault response: fault injection (——), brake sensor error (- - -), and brake actuator output (- - -)

3.4.2. Real-time application fault

In case the real-time application crashes or is powered-off during operation, the controlled outputs must be disabled. This is achieved by the heartbeat message which is used to let other nodes know it is alive. When the real-time application crashes or is powered-off during operation, this message will also stop. The response of the actuators monitoring this signal should then be to disable their outputs, as explained in Section 3.3.3. To test this on the control platform, both heartbeat and lifeline signals are first active and then at time $t = 1\,\text{s}$ they are turned off. The last messages sent are then somewhere in the interval $t = 0.75\,\text{s}$ to $1\,\text{s}$. The results of this test are shown in Figure 3.3. As can be seen, both the steer and brake actuator respond to the vanishing heartbeat signal by disabling their output at $0.5\,\text{s}$ after the last received heartbeat message $t = 1.3\,\text{s}$, and stay in that state even when the heartbeat is enabled again at $t = 2.5\,\text{s}$. The TPS node responds to the vanishing lifeline signal disables its output $0.5\,\text{s}$ after the last received lifeline message at $t = 1.5\,\text{s}$. To enable the actuators again, they first must be disabled to prevent unexpected behavior.

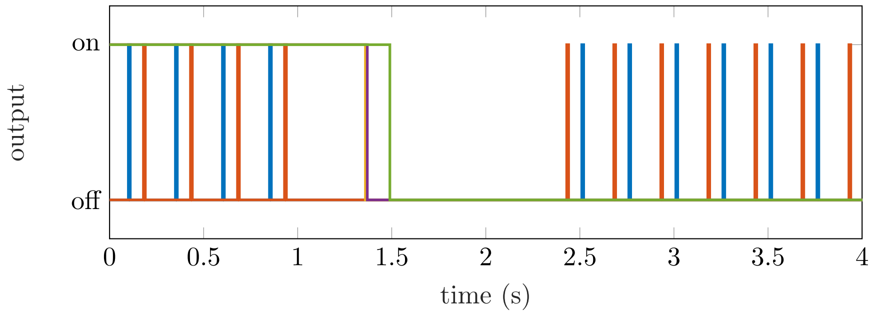


Figure 3.3.: Real-time application fault response: heartbeat signal (—), lifeline signal (—), drive actuator output (—), steer actuator output (—), and brake actuator output (—)

3.5. Discussion

A software framework is presented consisting of several layers. The layered approach specifies clear boundaries, enabling separate design and testing of software components. The developed architecture ensures safe operation of the research and development vehicle, without having to explicitly take safety into account when developing automated driving functionality. The effectiveness of the safety mechanisms has been successfully demonstrated by testing two critical fault cases.

4.

Automated Driving Safety Verification Framework

Abstract - Automated driving vehicles contain complex and sophisticated systems to perceive and interpret its surroundings, plan its driving strategy based on the received information, and execute the driving plan. A vast amount of vehicle testing is required to provide certain confidence levels. Closed course testing and validation is commonly employed as most automated vehicles are adapted and not road legal. Simulations are also used to reduce time and costs. Assumptions and gaps in lower-fidelity simulations might however result in residual risks. This chapter presents a simulation framework for testing and validating automated driving controllers, consisting of a layered simulation approach. The automated driving controllers can be validated in simulation as well as using a hardware-in-the-loop setup. The simulation framework is validated using full-scale driving experiments. The results show a good resemblance between the model and experimental output. Using the framework, automated driving algorithms can be developed and validated in a controlled simulated environment before deployment on the automated platform.

4.1. Introduction

Automated driving vehicles have the potential to increase road safety as nine out of ten fatal accidents are related to human driver error (Singh 2015). Hence, several companies and institutes around the world are currently developing and testing their automated vehicles (e.g., Waymo 2019; General Motors 2018).

To allow these automated vehicles on the road, safety assessment of the automated driving functionality is necessary to guarantee that they are safe. However, a credible safety statement requires millions of road miles (Kalra and Paddock 2016). Obtaining these road miles are costly, time consuming, and it can result in unsafe situations. A safer approach is closed-course testing, but during closed-course testing no out of the ordinary situations are encountered as in real-world traffic. Closed-course testing is also not scalable and is therefore time consuming. Real automated driving platforms can also be very expensive or not available in the required amount. Simulations, on the other hand, are scalable and may therefore be a good option for safety validation of automated driving functionality. The benefit is that testing is reproducible and safety-critical tests can also be performed without risk. At this time of writing, there is no agreed strategy to validate and approve automated vehicles.

Proving that the system does what it is designed to do can be extremely difficult, which is especially the case with opaque techniques, such as machine learning (Nguyen et al. 2015). These techniques are commonly used for environmental perception systems in automated vehicles (Koopman and Wagner 2016). Even when these systems are fault free, potential unsafe situations can happen, for instance, due to limitations of sensor performance. To cover these and other events, the Safety Of The Intended Functionality (SOTIF) standard is designed. The SOTIF is developed for Advanced Driver-Assistance Systems (ADASs) but it will be extended in the future to also cover Automated Driving Systems (ADSs) (Khabbaz Saberi 2020). The SOTIF (ISO/PAS 21448 2019) addresses hazardous events resulting from external causes, such as sensor performance limitations or driver misuse, whereas the ISO 26262 standard addresses hazardous events resulting from internal causes, such as malfunctions. The SOTIF classifies all possible scenarios into four areas: known safe, known unsafe, unknown unsafe, and unknown safe, and its purpose is to make the unsafe (known and unknown) scenarios safe. This can, for example, be achieved by limiting the operational design domain. To identify and test the unsafe scenarios, the standard recommends various activities such as test cases with high coverage of relevant scenarios, fault injection tests, and in the loop testing of relevant scenarios.

However, assumptions and gaps in lower-fidelity simulation models can result in residual risks. Therefore, for safety validation, higher-fidelity simulation models are desired. By using a layered simulation approach, which includes lower- and higher-fidelity simulation models, the right model can be selected based on the scenario to be tested (Koopman and Wagner 2018). In this way, the validation approach is more efficient compared to solely using the higher-fidelity simulations or road testing.

The aim of this chapter is to develop a simulation framework for safety testing and validating automated driving controllers. The developed automated driving controllers can be validated with the digital prototype in simulation as well as using a Hardware-In-the-Loop (HIL) setup. The HIL setup model uses the same communication interfaces as the real automated vehicle. The model must be capable of running in real-time, such that it can be coupled to the real-time control system, as presented in Chapter 3. After the automated driving controllers are tested and validated, they can be deployed on the real automated platform.

The outline of this chapter is as follows. Section 4.2 explains the model framework consisting of a highly representative dynamic model of the automated vehicle.

Section 4.3 presents the HIL setup consisting of two connected real-time computers: one running the virtual automated vehicle model and the other running the real-time control model. Section 4.4 validates the simulation framework with data obtained from real driving experiments. Finally, Section 4.5 lists the conclusions.

4.2. Simulation Framework

The simulation framework for safety testing and validating automated driving controllers can be used in two configurations: accelerated-time in which the control system is integrated in the simulation environment and real-time in which the control system is separated from the simulation environment. Figure 4.1 shows a comparison of the two options. The accelerated-time configurations runs on a single computer and can be used when many iterations or small changes to the design have to be made. The control system to be tested can just be a small part of the whole control system. The real-time configuration is more complex and needs two PCs running Simulink real-time operating system (Matlab 2020c). One to run the real-time virtual automated vehicle simulation and the other to run the real-time control system. The real-time virtual automated vehicle model runs at 1000 Hz and the real-time control system model at 100 Hz. Both real-time applications can be controlled via a control panel running on a PC with Windows. The control panel for the real-time simulation model mimics the driver inputs, such as accelerator pedal position, steering torque and ignition switch status. The simulation framework is available from (Hoogeboom 2020b).

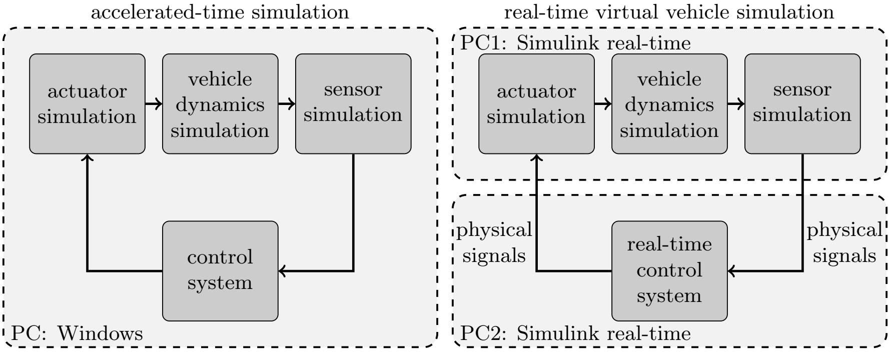


Figure 4.1.: Simulation framework options: accelerated-time simulation (left) and real-time hardware-in-the-loop simulation (right)

As the real-time applications exchange data in the same way as the real automated vehicle exchanges data with the real-time control platform, the complete real-time control system must be used, including the interface blocks responsible for signal conversions and communicating over a serial line. In this way, all the code is tested on realistic hardware, thereby increasing the simulation fidelity. After the system is tested and validated, the system can be deployed on the real automated vehicle in accordance with the V-cycle development process.

4.2.1. Vehicle dynamics simulation

The vehicle dynamics simulation block, as shown in Figure 4.1, is based on (Baaij 2019). The block is modeled in Simscape Multibody (Matlab 2020b), which provides a multibody environment to model 3D mechanical systems. The actual vehicle and its virtual representation are shown in Figure 4.2. The virtual representation consists of a body shell to which the suspension components and tyres are mounted. The vehicle dynamics model inputs are the drive torque, front and rear brake torque, and steering rack position.

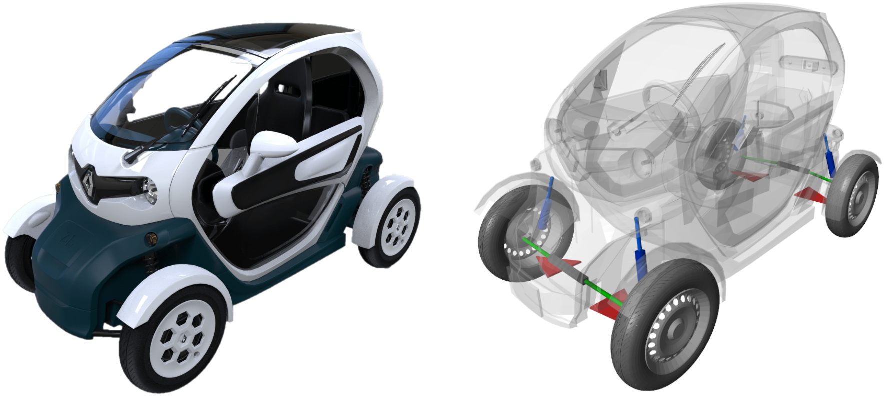


Figure 4.2.: Renault Twizy (left) and corresponding virtual representation (right)

Figure 4.3 gives an overview of the multibody model, consisting of subsystems for suspension, wheels, anti-roll bar, and steering rack. The chassis is modeled as a rigid body to which the suspension components are connected. The McPherson suspension is modeled with a lower wishbone, wheel carrier, and suspension strut. The left and right suspensions are connected via toe links and a steering rack. The coordinates of all the joints have been measured using a tactile measurement device. The detailed schematic representations of all these individual components are shown in Appendix A.

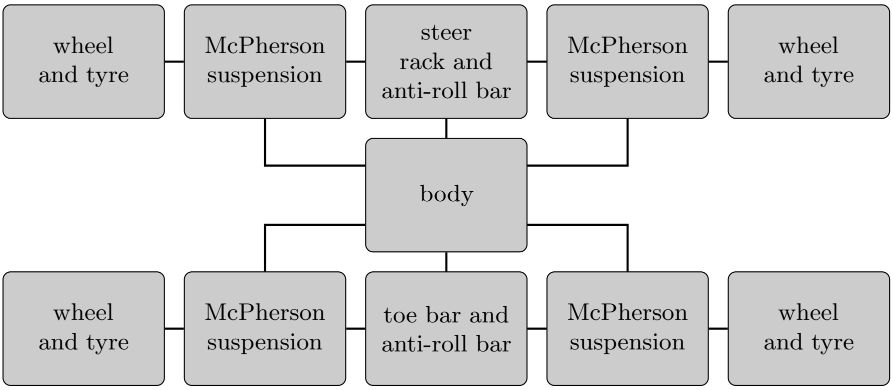


Figure 4.3.: Multibody simulation model layout (adapted from (Baaij 2019))

Suspension bushing compliance is neglected as the model must be able to simulate real-time. This would otherwise not be possible as the elements cause fast dynamics that require a small time step size of the fixed step solver to accurately describe the dynamics (Miller and Wendlandt 2010). The spring and damper characteristics are determined using a spring and damper test setup and have been included in the multibody simulation model via lookup tables. The corresponding characteristics are shown in Appendix A. Similarly, the front and rear stabilizer bars have been tested on a test setup to measure their stiffness.

The electric motor torque is actuated at the wheel side similar to the brake system. This is a simplification as the drive moment is applied to the upright instead of the vehicle body which results into suspension travel during acceleration/regenerative braking. Modeling of the drive shafts is neglected in the real-time environment as these elements cause fast dynamics that require a small time step size of the fixed step solver to accurately describe the dynamics (Miller and Wendlandt 2010). The effect might however be negligible in relation to the limited available electric motor torque.

The tyres have been modeled using MFeval (Marco Furlan 2019), which is an open-source MATLAB toolbox that implements the Magic Formula equations (Pacejka 2005). The tyre model has been integrated in the multibody environment similar to the Delft tyre model (TASS International 2020). The MFeval tyre model can be used in real-time mode without an additional license. The real-time capabilities of the tyre model have been improved by removing all for the model unnecessary elements, such as unused operating modes or outputs. The required front and rear tyre characteristics, such as the cornering stiffness, vertical stiffness, and pneumatic trail, have been measured on a flat plank tyre tester (Besselink 2019, p.99) as well as with a tyre measurement tower on a drum (Besselink 2019, p.106). More details about the tyre testing as well as identification of other vehicle dynamics parameters can be found in (Baaij 2019).

4.2.2. Actuators simulation

The inputs to the multibody vehicle model are the drive torque, brake torque front, brake torque rear, and steering rack position, while the outputs of the control system are the acceleration pedal position, brake motor current, and steer motor current. The actuator simulation block converts the outputs of the control system to the required inputs for the vehicle dynamics simulation. The brake and steering system are included as models while the driveline characteristic is included via a lookup table.

Driveline

The driveline of the automated Renault Twizy consists of an electric motor, gear reduction with differential, and drive shafts. The driveline characteristics are identified on a chassis dynamometer, as depicted in Figure 4.4. During the identification tests, the chassis dynamometer speed is kept constant while the accelerator pedal position α_p is varied with discrete steps of 10 % on the interval 0 % to 100 %. This test is repeated for all chassis dynamometer speeds on the interval 5 km/h to 100 km/h with a step size of 5 km/h. The resulting grid of electric motor torque τ_dm as function of accelerator pedal position α_p and motor speed ω_dm is included as a lookup table which is shown in Figure 4.5. Gear selection (drive, neutral, and reverse) is implemented which multiplies the output torque τ_dm with one, zero, or minus one, respectively, to allow for forward driving, coasting, and reverse driving.

Brake system

The brake system of the automated Renault Twizy consists of disc brakes front and rear. The automated vehicle does not contain an ABS system but the rear brakes contain a brake pressure limiter to prevent rear wheel lock-up. The mapping from brake pressure p_b to brake torque τ_b is determined by placing the automated vehicle on the chassis dynamometer similar to measuring the driveline. The speed of the chassis dynamometer is kept constant at 30 km/h and a constant brake pressure p_b is applied for several seconds to let the chassis dynamometer settle to steady state. This test is repeated for every brake pressure on the interval 0 bar to 70 bar with a step size of 5 bar. Figure 4.6 presents the measured brake pressure and wheel torque in which the effect of the brake pressure limiter is clearly visible. The influence of brake temperature is neglected in the mapping.

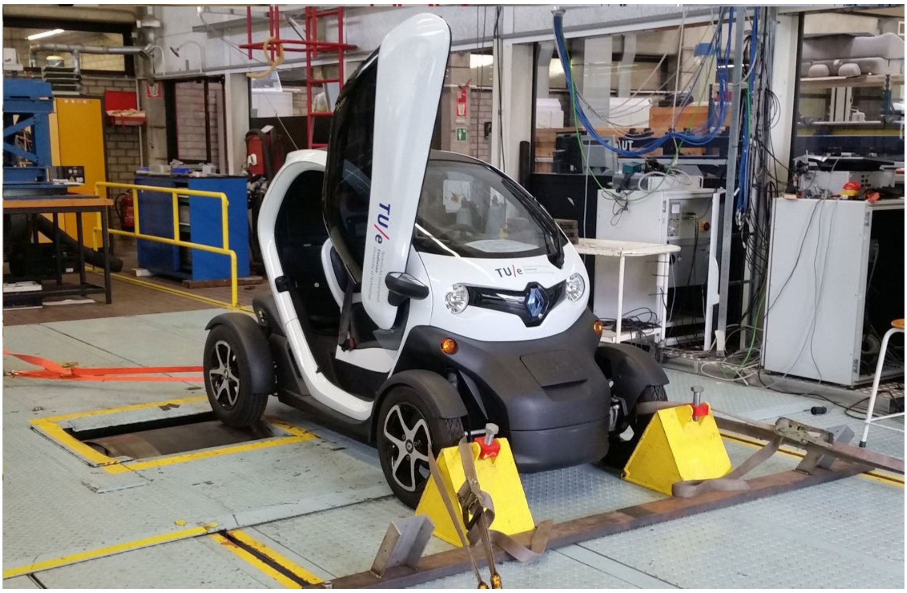


Figure 4.4.: Driveline characteristics identification of the automated Renault Twizy (courtesy: Brian Rensen)

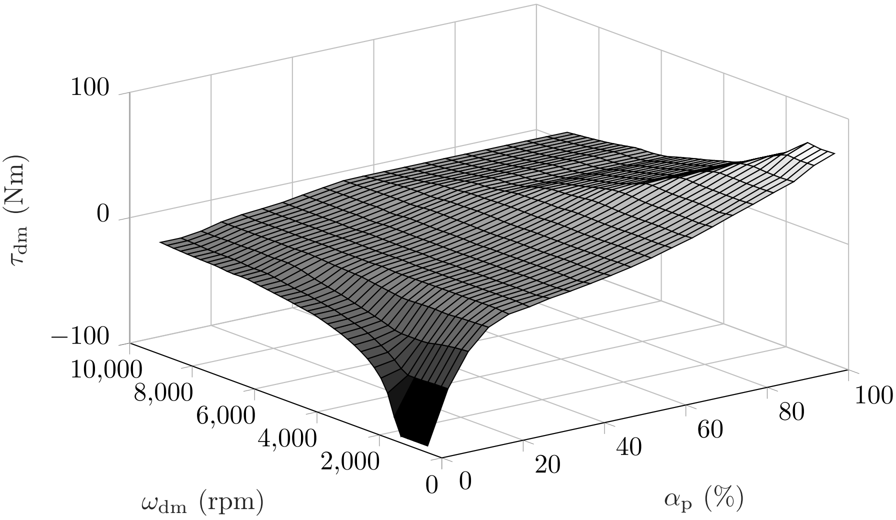


Figure 4.5.: Electric motor torque τ_{dm} as function of accelerator pedal position α_{p} and electric motor speed ω_{dm}

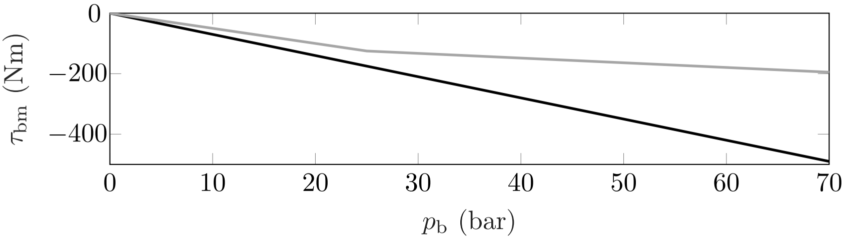


Figure 4.6.: Brake torque τ_{bm} as function of brake pressure p_{b}: front (——) and rear (——)

When the brakes are controlled by the ADS, a cam follower mechanism applies pressure on the brake master cylinder, as shown in Chapter 2. The relation of the camshaft angle to brake pressure is determined using a dynamic test in which on the cam position δ_{c} steps are provided as input and the brake pressure p_{b} is measured as output. The steady state values are extracted and shown in Figure 4.7, along with a third order polynomial model fit. As can be seen, the model fit represents the data trend well besides some hysteresis effects, which are mainly caused by friction in the brake master cylinder (Dardanelli et al. 2010).

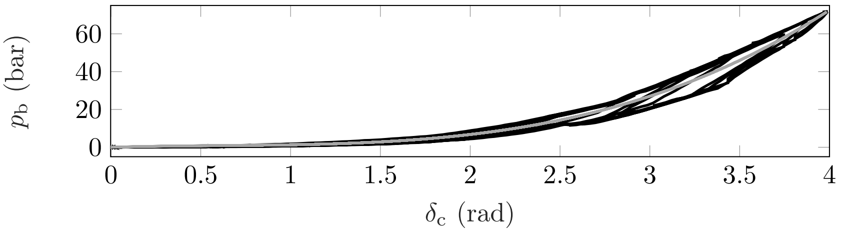


Figure 4.7.: Brake pressure p_{b} as function of camshaft angle δ_{c}: measurement (——) and third order model fit (——)

To obtain a relation of motor torque τ_{bm} to camshaft position δ_{c}, a model of the brake system is created. Here, it is assumed that the time constant of electrical dynamics is considerably smaller than the time constant of the mechanical system, so the brake motor electrical dynamics is neglected and a torque τ_{bm} is modeled on the brake motor inertia J_{bm}. The camshaft with follower and gearbox is modeled as lumped mass with inertia J_{c}, and the resistance provided by the brake hydraulics is modeled as a nonlinear spring and damper. Based on the brake pressure p_{b} to camshaft angle δ_{c} relation shown in Figure 4.7, the nonlinear spring is modeled as linear spring plus a cubic spring. Figure 4.8 depicts the brake system model in which τ_{bm} is the motor torque, δ_{c} the camshaft position, J_{c} the motor, gearbox, and camshaft inertia, k_{c} the nonlinear spring, and T_{c} the dry friction.

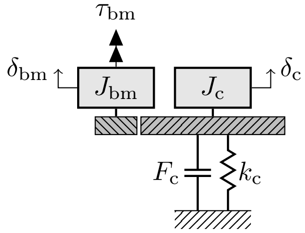


Figure 4.8.: Brake system model of the brake assembly as depicted in Figure 2.4

Given the schematic depiction of the brake system in Figure 4.8, the equation of motion for this system is given as

$$(J_{\mathrm{c}} + J_{\mathrm{bm}} i_{\mathrm{bm}}^2)\ddot{\delta}_{\mathrm{c}} = \tau_{\mathrm{bm}} i_{\mathrm{bm}} - k_l \delta_{\mathrm{c}} - k_c \delta_{\mathrm{c}}^3 - T_{\mathrm{c}} \operatorname{sign}(\dot{\delta}_{\mathrm{c}}), \tag{4.1}$$

where J_{c} is the camshaft moment of inertia, J_{bm} the motor moment of inertia, k_{l} the linear spring stiffness, k_{c} the cubic spring stiffness, $i_{\mathrm{bm}} = {}^{\delta_{\mathrm{bm}}}/_{\delta_{\mathrm{c}}}$ the gear ratio, and T_{c} the Coulomb friction.

The parameter values for the motor inertia J_{bm} and gearbox ratio i_{bm} are obtained from specsheets (Apex 2017; Applied Motion Products 2013). The Coulomb friction torque in the system has been identified by slowly moving the camshaft over its full range by controlling it with the electric motor and measuring the required brake motor torque τ_{bm}. Figure 4.9 shows the brake motor torque τ_{bm} and camshaft position δ_{c}. As can be seen, the system contains stick-slip friction as steps in torque results in interrupted motion instead of smooth motion. The stick-slip friction is modeled as Coulomb friction as the rapid changes of stick-slip friction model can be difficult for a fixed-step solver (Miller and Wendlandt 2010). The Coulomb friction torque T_{c} has been calculated from half of the average value between the forward motion and backward motion multiplied with the gear ratio i_{bm}.

The other unknown parameters are identified via a nonlinear grey-box identification procedure (Bohlin 2006). With grey-box identification, the model representation is fixed and experimental data is used to estimate the parameters. A random phase multisine signal is used as identification signal. The signal consists of 50 sine waves linearly spaced on the interval of 0.05 Hz to 50 Hz. A multisine identification signal is selected for the experiments as the base frequency sine wave keeps the system in motion to reduce dry friction influences. In addition, the energy in the input signal is concentrated at discrete frequency points, thereby

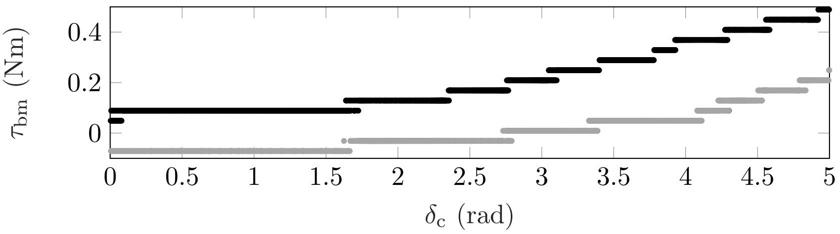


Figure 4.9.: Brake motor torque τ_{bm} and camshaft position δ_{c}: forward motion (•) and backward motion (•)

increasing the signal-to-noise ratio. The multisine signal is a segment of 2048 samples and with a sample time of 0.01 seconds, the generated signal has a period of 20.48 seconds. The signal is constantly repeated during the experiment to obtain multiple segments of data. The periodicity and continuity of the signal enables averaging of sequences to improve the signal-to-noise ratio without the need for tools, such as windowing.

While the brake system is a constrained system in terms of maximum camshaft rotation, open loop system identification is applied. This is possible with the hydraulic brake pressure that acts as a nonlinear spring and a careful selection of the identification signal gain and offset. The gain and offset are selected such that the system operates over a wide range during the identification experiments while making sure contact with the end stops is avoided since this introduces unwanted nonlinear effects.

The model parameters are estimated by utilizing MATLAB's function 'nlgreyest' which uses a least squares algorithm. The quality of the fit is assessed by the Normalized Root Mean Squared Error (NRMSE) expressed as a percentage

$$\varepsilon_{\mathrm{fit}} = \left(1 - \frac{\|\mathbf{y}_{\mathrm{m}} - \mathbf{y}_{\mathrm{p}}\|}{\|\mathbf{y}_{\mathrm{m}} - \bar{\mathbf{y}}_{\mathrm{m}}\|}\right) \cdot 100\,\%, \tag{4.2}$$

where $\|\cdot\|$ denotes the 2-norm of a vector, $\mathbf{y}_{\mathrm{m}}$ the measured outputs, $\mathbf{y}_{\mathrm{p}}$ the predicted model output, and $\bar{\mathbf{y}}_{\mathrm{m}}$ the mean of the measured output. A $\varepsilon_{\mathrm{fit}}$ value of 100 % indicates a perfect fit where the predicted outputs are identical to the measured outputs while a value of 0 % indicates the model is no better than a straight line through the mean of the data.

Figure 4.10 shows one segment length of the identification signal and a comparison between the model outputs and the measured outputs. As can be seen, the model outputs and measured outputs agree fairly well. The $\varepsilon_{\mathrm{fit}}$ percentage for camshaft angle δ_{c} is 72.60 %, which indicates that the model is suitable as a digital prototype for the brake system. Table 4.1 lists the model parameter values.

Table 4.1.: Brake system model parameter values

Description	Symbol	Value	Source
Camshaft moment of inertia	J_{c}	$0.0393\,\mathrm{kg\,m^2}$	Identification
Brake motor moment of inertia	J_{bm}	$0.26\mathrm{e}{-4}\,\mathrm{kg\,m^2}$	Specsheet
Motor gear ratio	i_{bm}	20	Specsheet
Linear spring stiffness	k_{l}	$0.1291\,\mathrm{Nm/rad}$	Identification
Cubic spring stiffness	k_{c}	$0.0604\,\mathrm{Nm/rad}$	Identification
Coulomb friction torque	T_{c}	$2.5340\,\mathrm{Nm}$	Identification

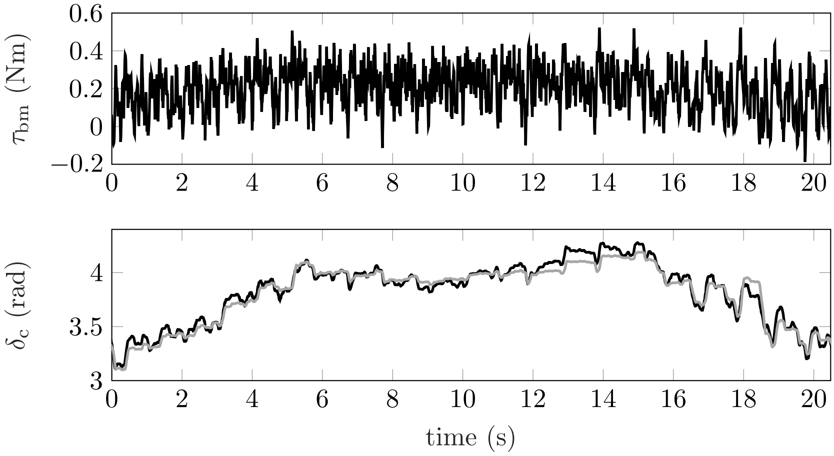


Figure 4.10.: Comparison between model output and measured camshaft angle δ_{c} with the multisine identification signal as brake motor torque τ_{bm} input: measurement data (——) and model output (——)

Steering system

As shown in Chapter 2, the steering system of the automated vehicle consists of a steering column with electric power-assist. The steering wheel is coupled to the steering column via a torsion bar. This torsion bar is used to measure the torque exerted on the steering wheel. The electric motor is coupled to the steering column via a worm gear. The measurable outputs on the real system are steering column angle δ_{sc}, steering column angular velocity $\dot{\delta}_{\text{sc}}$, and steering column torque τ_{tb}.

Figure 4.11 presents a model of the steering system where the steering wheel is modeled as inertia J_{sw}, coupled to the steering column via the torsion bar modeled as a spring k_{tb} and damper c_{tb}. The torque exerted on the steering wheel by the driver is denoted by τ_{d}. The electric motor has inertia J_{sm} and is coupled to the steering via a worm-gear with gear ratio $i_{\text{sm}} = {}^{\delta_{\text{m}}}/_{\delta_{\text{sc}}}$. The rotational motion of the steering column is converted to translational motion of the steering rack via a rack and pinion with radius r_{p}. Coulomb friction torque T_{sc} is added as rack and pinion based designs contain significant dry friction (Harrer and Pfeffer 2017). The self-centering effect as a result of suspension geometry often referred as jacking torque is modeled with a spring k_{r} and damper c_{r}. The steering motor torque is given as $\tau_{\text{sm}} = k_{\text{sm}} I_{\text{sm}}$ in which the electrical dynamics of the steering motor are neglected since the time constant of the electrical system is considerably smaller than that of the mechanical system.

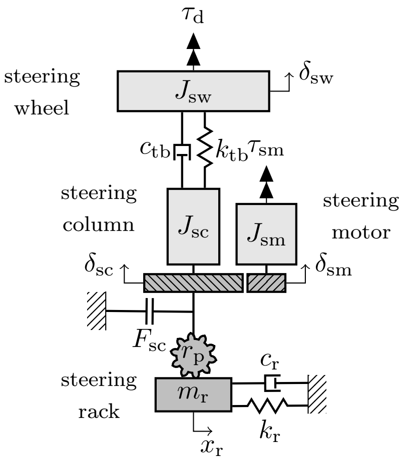


Figure 4.11.: Nonlinear steering system model of the steering system as depicted in Figure 2.3

Given the schematic representation of the brake system in Figure 4.11, the equations of motion for this system are given as (Marouf et al. 2012)

$$J_{\mathrm{sw}}\ddot{\delta}_{\mathrm{sw}} = \tau_{\mathrm{d}} - c_{\mathrm{tb}}(\dot{\delta}_{\mathrm{sw}} - \dot{\delta}_{\mathrm{sc}}) - k_{\mathrm{tb}}(\delta_{\mathrm{sw}} - \delta_{\mathrm{sc}}), \tag{4.3a}$$

$$J_{\mathrm{eq}}\ddot{\delta}_{\mathrm{sc}} = \tau_{\mathrm{sm}} i_{\mathrm{sm}} + c_{\mathrm{tb}}(\dot{\delta}_{\mathrm{sw}} - \dot{\delta}_{\mathrm{sc}}) + k_{\mathrm{tb}}(\delta_{\mathrm{sw}} - \delta_{\mathrm{sc}}) - T_{\mathrm{sc}}\,\mathrm{sign}\,(\dot{\delta}_{\mathrm{sc}}) - c_{\mathrm{r}} r_{\mathrm{p}}^2 \dot{\delta}_{\mathrm{sc}} - k_{\mathrm{r}} r_{\mathrm{p}}^2 \delta_{\mathrm{sc}}, \tag{4.3b}$$

where $J_{\mathrm{eq}} = J_{\mathrm{sc}} + J_{\mathrm{sm}} i_{\mathrm{sm}}^2 + m_{\mathrm{r}}^2 r_{\mathrm{p}}^2$. The required steering system parameters are obtained from specsheets, component identification tests (Stoffels 2019), and a nonlinear grey box identification procedure. Table 4.2 lists the estimated steering model parameter values.

Table 4.2.: Steering system model parameter values

Description	Symbol	Value	Source
Steering wheel inertia	J_{sw}	0.0114 kg m^2	Identification
Steering column inertia	J_{sc}	0.005 kg m^2	Identification
Steering motor moment of inertia	J_{sm}	3.28e−4 kg m^2	Specsheet
Steering rack mass	m_{r}	3 kg	Specsheet
Steering motor gear ratio	i_{sm}	13.67	Identification
Motor constant	k_{sm}	0.052 Nm/A	Specsheet
Torsion bar damping	c_{tb}	0.029 Nms/rad	Identification
Torsion bar stiffness	k_{tb}	80.97 Nm/rad	Identification
Jacking torque stiffness	k_{r}	16 132 N/m	Identification
Jacking torque damping	c_{r}	3645 Ns/m	Identification
Steering rack pinion radius	r_{p}	5.6e−3 m	Identification
Coulomb friction torque	T_{sc}	2.43 Nm	Identification

To evaluate the model performance, the automated vehicle has been placed turn plates to reduce static friction between the tyres and road surface, thereby aiming to represent normal driving conditions. As input signal, a random signal has been applied. This input signal is obtained by applying alternating steering actions via the remote control panel of the real-time control system. Figure 4.12 shows a comparison between the measurement and model output for the discussed input signal. As can be seen, the model represents the measurement data well. The $\varepsilon_{\mathrm{fit}}$ percentage for steering column angle δ_{sc} is 89.20 %, which indicates that the steering system is able to serve as a digital prototype for the real system.

The steering system is included in the complete vehicle model by modeling it in Simscape. This one-dimensional model is coupled to the Simscape Multibody environment by using the Simscape Multibody Multiphysics library (Miller 2020).

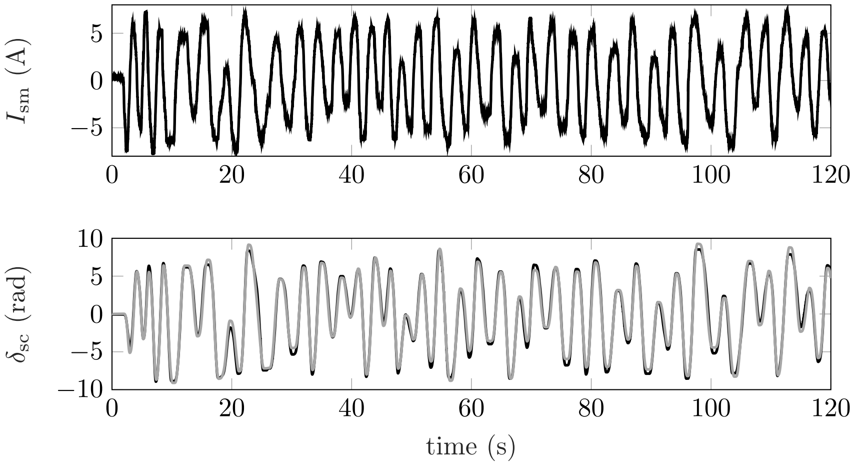


Figure 4.12.: Comparison between model output and measured steering column angle δ_{sc} with the random steering motor current I_{sm} signal as input: measurement data (—) and model output (—)

The jacking torque spring and damper are not modeled as this is already included in the suspension geometry of the vehicle model. By using damped transition regions for the nonlinear effects such as hard stops, the model is suitable for real-time simulation.

4.2.3. Sensor simulation

Next to the vehicle inputs, it is necessary to mimic the sensor outputs as well. This means that the physical sensors need to be modeled with their corresponding properties such as update rate, signal filtering, and sensor noise. The implementation depends on the measurement model. The sensors that are mimicked are the native automated vehicle sensors and the additional sensors as described in Chapter 2. The interface blocks are automatically selected depending on the simulation mode. In the accelerated-time simulation mode, the signals are directly coupled to the outputs whereas in the real-time mode they are transmitted to the PC via an electrical interface, such as Controller Area Network (CAN)-bus, RS-232, or Ethernet.

For a high-fidelity simulation, a representive CAN-bus load is required as this might introduce issues with message latencies. Therefore, all the information present on the native automated vehicle CAN bus is simulated, even when the information is not used in the real-time control system. The required information of update rate, identifiers, message content of the native automated vehicle CAN bus is decoded in Chapter 2. The automated control CAN-bus contains all the additional sensors and actuators needed for the automated functionality. All the checksums are added to the CAN messages as the real-time control system verifies the checksums. For the custom hardware this is a simple addition, while the implemented Bosch sensors use a more complicated algorithm (Miller and Valasek 2015, p.82).

Additional sensors can be included in the simulation environment to determine the influence on the CAN bus load or to measure quantities which is not possible on the real automated vehicle, such as ground truth data. Also, fault injection tests can easily be performed with the simulated sensors.

4.3. Simulation Hardware Setup

The real-time simulation setup consist of two Dell OptiPlex 7010 PCs, as depicted in Figure 4.13. Softing CAN-AC2-PCI (Softing 2012) interfaces have been installed to enable CAN bus communication. Each interface board provides two CAN busses. The CAN bus interfaces are used to simulate the vehicle (V-CAN) and control (C-CAN) CAN bus. A RS-232 interface is used to transmit the simulated Global Navigation Satellite System (GNSS) receiver data. An Ethernet connection can be used to include data from the environment, such as radar, camera, or V2X communication.

Driving scenarios will be designed with the help of the automated driving toolbox (Matlab 2020a). With the toolbox, radar and vision sensors can be configured and simulated before implementing them on the automated driving platform. The simulated object data can be included in the control model. The driving scenario is advanced using the updated state information of the simulation model, such as position and velocity.

4.4. Simulation Model Validation

Full-scale driving tests have been conducted on a proving ground to identify the vehicle parameters that cannot be easily measured and to validate the complete simulation model. These test have been performed on a dry day at Ford Lommel Proving grounds in Lommel, Belgium. The automated vehicle was manually driven during the tests since that was a safety constraint from Ford. To identify vehicle dynamic parameters, several tests were executed, such as a coast down and steady-state cornering. The results and model fits are shown in Appendix A.

Test data from a handling track test is used to validate the simulation model including the actuator and sensor simulation. The handling track is a relatively long test and includes acceleration, deceleration, and cornering. Hence, most of

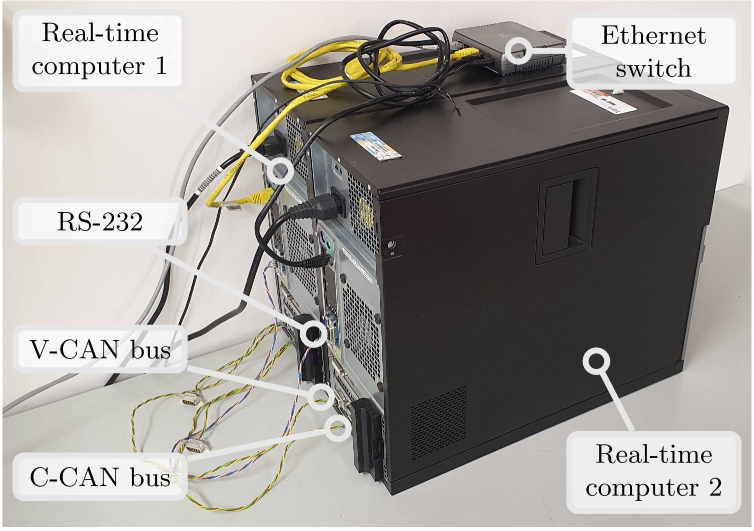


Figure 4.13.: Real-time simulation setup consisting of two Dell OptiPlex 7010 PCs running Simulink real-time (Matlab 2020c)

the various model aspects can be validated at once. Since the vehicle was not automated during the test day, not all relevant data of the steer and brake system is available to validate the actuator simulation. These parts have been separately validated. Figure 4.14 depicts the driven path during the test in which the start and end of the driven path are indicated by an circle and square, respectively. The track contains a section with cobble stones that is indicated by the white line in the figure.

The accelerator pedal position α_p, steer wheel torque τ_d, and brake pressure p_b are used as model inputs. These signals are converted in the actuator simulation block into the inputs of the multibody block which are drive torque, front and rear brake torque, and steering rack displacement. The outputs of the multibody block are fed through the sensor abstraction block to include sensor characteristics, such as low pass filtering, signal noise, and sampling. The simulated sensor signals are compared with the measured outputs.

Figure 4.15 shows the simulation results. The top three figures show the measured inputs to the actuator simulation block which are the accelerator pedal position α_p, brake pressure p_b, and driver torque τ_d, respectively. The following five figures show the measured and simulated sensor signals, namely steering column angle δ_sc, forward velocity v_x, longitudinal acceleration a_x, lateral acceleration a_y, and yawrate w_z. As can be seen, the simulated and measurement data

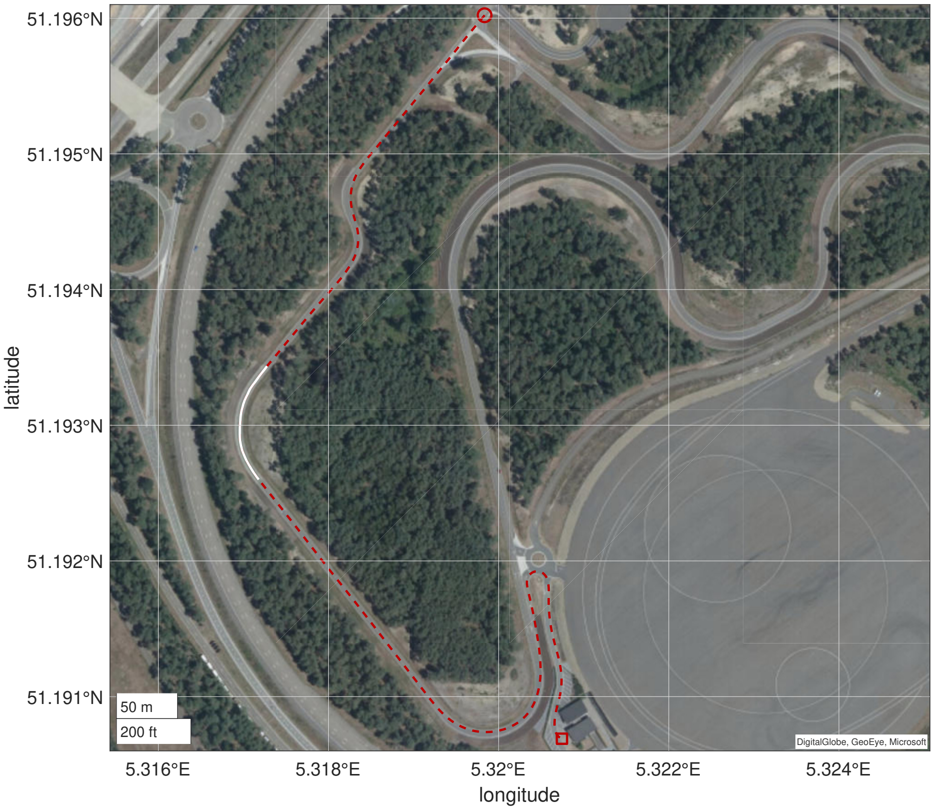


Figure 4.14.: Handling track test at the Ford Lommel proving grounds. The driven path starts at the circle and ends at the box. The white line indicate a section with cobble stones

show a good resemblance. During the interval 27 s to 50 s, the automated vehicle drives over the cobble stones, while in the simulation a flat road is used. Therefore, some differences are noticeable between the measured and the simulated sensor signals.

To provide an objective measure of fit, the Pearson product-moment correlation coefficient ρ is used, which is defined as (Freedman et al. 2007)

$$\rho_{x,y} = \frac{\operatorname{cov}(x,y)}{\sigma(x)\sigma(y)}, \tag{4.4}$$

where $\operatorname{cov}(x,y)$ denotes the covariance of x and y and $\sigma(x)$ denotes the standard deviation of x. The Pearson correlation coefficient is a number between -1 and 1 in which 0 means no correlation, 1 perfect correlation, and -1 perfect negative correlation. Since this measure does not include signal offsets, an additional measure is used to identify signal offsets. The following measure is used (Loof 2018)

$$\mu = \frac{\bar{x}-\bar{y}}{|\max(x)-\min(x)|}\cdot 100\,\%, \tag{4.5}$$

where $\bar{x}$ denotes the mean value of x. An μ of 0 % indicates the means $\bar{x}$ and $\bar{y}$ are equal. A μ of 100 % indicates the mean of $\bar{x}$ is higher than $\bar{y}$ with the same value as the range of x is. Similarly, a μ value of $-100\,\%$ indicates the mean of $\bar{x}$ is lower than $\bar{y}$ with the same value as the range of x is. Any ρ above 0.7 and offset μ below 10 % is considered a good model fit.

Table 4.3 lists the model performance expressed in Pearson product-moment correlation coefficient ρ and offset μ. As can be seen, all signals have a ρ value above 0.9 and offset μ below 4 %, which indicates that the model provides a good fit to the measurement data and that the framework can be used a digital prototype to validate automated driving controllers.

Table 4.3.: Simulation model performance expressed in Pearson product-moment correlation coefficient and offset

Description	Symbol	ρ	μ (%)
Forward velocity	v_x	0.9826	3.5531
Steering column angle	δ_sc	0.9714	0.6264
Longitudinal acceleration	a_x	0.9180	0.4875
Lateral acceleration	a_y	0.9132	-0.1764
Yawrate	w_z	0.9407	-1.7557

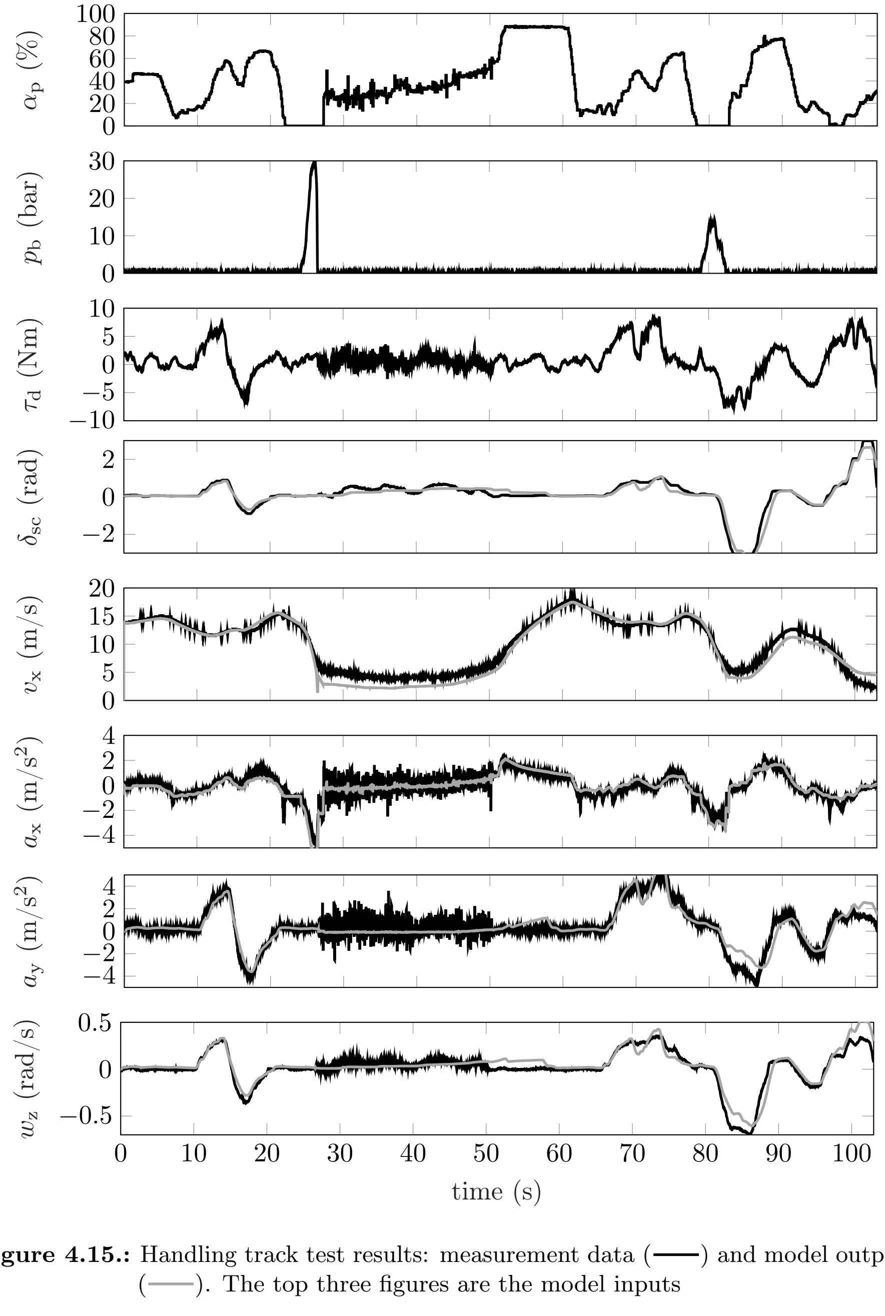


Figure 4.15.: Handling track test results: measurement data (——) and model output (——). The top three figures are the model inputs

4.5. Conclusion

A simulation framework for testing and validating automated driving controllers is developed. The created simulation framework can be used in two configurations: accelerated-time in which (a part of) the control system is integrated in the simulation environment and real-time in which the control system is separate from the simulation environment. The real-time setup can be used to verify the safety mechanisms of the real-time application developed in Chapter 3 and to test automated driving controllers in a controlled environment, for example, as a final step before deployment on the real research and development platform. The model framework has been validated with vehicle data obtained from full-scale driving tests. The data obtained from a handling track shows that the model is able to closely match the measured outputs. The framework can be adapted to other vehicle characteristics by changing the vehicle parameters.

5.

Model-Based Fault Diagnosis of an Automated Steering System

Abstract - With an automated vehicle, the driver is not actively controlling the vehicle and the driver might not notice certain faults, such as a flat tyre. An ADS must therefore supervise itself by including fault diagnosis. This chapter presents a model-based fault diagnosis method applied on the steering system of an automated vehicle. The steering system is modeled in concatenation with a driver model and a single track vehicle model. The faults of interests are a steering actuator fault and a user intervention in which the driver tries to regain control of the vehicle. The fault detection filter performance is evaluated in simulation and with experiments. The results show that it is possible to detect and isolate a steering actuator fault and a user intervention.

5.1. Introduction

Automated vehicle development aims at increasing safety and comfort of the road users by taking over driving tasks. In doing so, the driver loses the contact with the steering wheel and thereby its most important tool of vehicle feedback. As a consequence, the driver might not notice certain faults with the automated steering system, and therefore, early detection and timely handling of faults is important in safety critical systems, such as the steering system of an automated vehicle, is important to prevent any undesired and possibly unsafe behavior.

A fault is defined as the deviation of a system property from the expected behavior, and determining the occurrence of a fault is called fault detection, whereas localization of the fault corresponds to fault isolation. To detect and isolate faults in a system, model-based fault diagnosis can be applied which uses the available measurements and a mathematical model of the system (Chen and Patton 1999). The difference between the actual measurements and the on the mathematical model based estimated measurements are defined as the residuals. Faults can be detected by comparing these residuals with a static or dynamic threshold, and a fault is triggered when this residual surpasses the threshold. By making certain residuals sensitive for a specific fault, the faults can be isolated by evaluating the pattern of the residuals exceeding their threshold.

Fault detection and isolation for automated vehicles has been studied by various researchers. For instance, Jeong et al. (2015) used a bank of Kalman filters to detect sensor and actuator faults for automated vehicles. Similarly, Li et al. (2017) used Extended Kalman filters to detect and isolate a yawrate fault, lateral acceleration fault, and/or steering angle sensor fault. Fault-detection filter design for a steer-by-wire vehicle has been addressed by Gadda (2009), which uses a variety of observers to generate the residuals. The drawback of these filter methods is that normally unnecessary technical assumptions are required (Chen and Patton 1999) and/or full order state observers are employed, requiring high complexity and computational load. Therefore, Ho and Ossmann (2014) investigated fault detection and isolation of vehicle dynamic sensors and actuators for a constant longitudinal velocity by applying numerical nullspace computations to synthesize the residual generators, resulting in minimal order filters. They show via computer simulations that it is possible to detect and isolate various faults within a x-by-wire vehicle.

Besides sensor and actuator faults, user intervention detections are also of interest when considering fault detection for an automated vehicle. A user intervention is when the Automated Driving System (ADS) is performing the dynamic driving task and a user tries to take over control of the vehicle, i.e., the system response is changed due to the user intervening. In that case, the automated control system should not fight the user but instead hand over the control.

To detect these user interventions, torque sensor signal based approaches can be applied, but this requires excessive low-pass filtering to eliminate the disturbances introduced by the automated steering system when steering (Kruiswijk et al. 2012). Also, additional sensors can be applied (Maguire and Bennett 2016; IEE 2020). However, the extra sensors do increase the costs of the vehicle, and therefore, a better solution might be to apply model-based fault diagnosis techniques to detect user-interventions. Low order filtering approaches are desired to limit execution time on the real automated driving platform.

The aim of this chapter is to address the issues of fault detection and user intervention by answering how to design fault detection and isolation filters for a steering system of an automated vehicle. While most existing literature focusses on computer simulations or experiments with small laboratory setups, the in this chapter developed methods are implemented on a real automated platform, thereby providing insight in the fault diagnosis techniques when operating under real-life circumstances.

The outline of this chapter is as follows. Section 5.2 starts with a brief introduction of model-based fault diagnosis for linear systems. Next, Section 5.3 models the complete steering system, consisting of a driver model, steering-system model, and single track vehicle model. After Section 5.4 estimates the unknown model parameters using a system identification procedure. Then, Section 5.5 presents the fault diagnosis strategy using the complete steering system model. After that, Section 5.6 validates the fault diagnosis system by performing computer simulations followed by Section 5.7, which presents experimental results. Finally, Section 5.8 provides the main conclusions.

5.2. Model-Based Fault Diagnosis

Model-based fault diagnosis relies on a mathematical model of the system to be monitored. Figure 5.1 shows a schematic representation of a model-based fault diagnosis architecture. The fault diagnosis system uses the known inputs $\mathbf{u}(t)$, outputs $\mathbf{y}(t)$, and a model of the system to generate residuals $\mathbf{r}(t)$. These residuals provide information about the possible faults and serve as a fault indicator signal. A residual evaluator and decision maker are included to filter the residuals and decide based on the filtered residuals $\boldsymbol{\chi}(t)$ whether a fault is active or not $\boldsymbol{\iota}(t)$. The following section describes the fault diagnosis system blocks in more detail and is based on Varga (2017b).

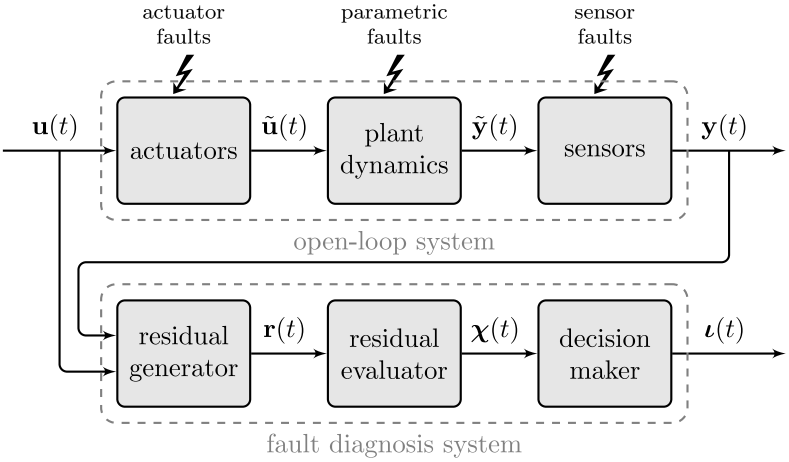


Figure 5.1.: Model-based fault diagnosis architecture

5.2.1. Linear system model with additive fault representation

The systems considered in this chapter are linear time-invariant systems with additive faults and are given by

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}_{\mathrm{u}}\mathbf{u}(t) + \mathbf{B}_{\mathrm{f}}\mathbf{d}(t) + \mathbf{B}_{\mathrm{f}}\mathbf{f}(t), \tag{5.1a}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}_{\mathrm{u}}\mathbf{u}(t) + \mathbf{D}_{\mathrm{f}}\mathbf{d}(t) + \mathbf{D}_{\mathrm{f}}\mathbf{f}(t), \tag{5.1b}$$

where $\mathbf{x}(t)$ is the n-dimensional state vector, $\mathbf{d}(t)$ are the unknown disturbances, and $\mathbf{f}(t)$ are the unknown faults. The system (5.1) can be formulated in Laplace domain as

$$\mathbf{y}(s) = \mathbf{G}_{\mathrm{u}}(s)\mathbf{u}(s) + \mathbf{G}_{\mathrm{d}}(s)\mathbf{d}(s) + \mathbf{G}_{\mathrm{f}}(s)\mathbf{f}(s), \tag{5.2}$$

where $\mathbf{y}(s), \mathbf{u}(s), \mathbf{d}(s), \mathbf{f}(s)$ denote the Laplace transformed vectors of known outputs, known inputs, unknown disturbances, and unknown faults, respectively. The matrices $\mathbf{G}_{\mathrm{u}}(s)$, $\mathbf{G}_{\mathrm{d}}(s)$, and $\mathbf{G}_{\mathrm{f}}(s)$ denote the Transfer-Function Matrices (TFMs) from control input, disturbance input, and fault input to the output, respectively. The corresponding TFMs are given by

$$\begin{aligned}
\mathbf{G}_{\mathrm{u}}(s) &= \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}_{\mathrm{u}} + \mathbf{D}_{\mathrm{u}}, \\
\mathbf{G}_{\mathrm{d}}(s) &= \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}_{\mathrm{d}} + \mathbf{D}_{\mathrm{d}}, \\
\mathbf{G}_{\mathrm{f}}(s) &= \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}_{\mathrm{f}} + \mathbf{D}_{\mathrm{f}}.
\end{aligned}$$

5.2.2. Residual generation

The linear residual generator uses the known signals to generate the residuals which serve as a fault indicator signal. The residual generator is described in the form

$$\mathbf{r}(s) = \mathbf{Q}(s)\begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix}, \tag{5.3}$$

which is called the implementation form. The fault detection filter $\mathbf{Q}(s)$ must be stable and proper. By substituting the input-output description (5.2) into (5.3), the internal form

$$\mathbf{r}(s) = \mathbf{Q}(s)\begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) & \mathbf{G}_{\mathrm{d}}(s) & \mathbf{G}_{\mathrm{f}}(s) \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix}\begin{bmatrix} \mathbf{u}(s) \\ \mathbf{d}(s) \\ \mathbf{f}(s) \end{bmatrix},$$

is obtained. This form shows that the residuals depend on all system inputs. The internal representation $\mathbf{R}(s)$ is defined as

$$\left[\begin{array}{c|c|c} \mathbf{R}_{\mathrm{u}}(s) & \mathbf{R}_{\mathrm{d}}(s) & \mathbf{R}_{\mathrm{f}}(s) \end{array}\right] := \mathbf{Q}(s)\left[\begin{array}{c|c|c} \mathbf{G}_{\mathrm{u}}(s) & \mathbf{G}_{\mathrm{d}}(s) & \mathbf{G}_{\mathrm{f}}(s) \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{array}\right].$$

5.2.3. Fault detectability

Fault detectability and complete fault detectability is about the sensitivity of a residual to a particular or all faults, respectively. The system (5.2) is j-th fault detectable when there exist a stable filter $\mathbf{Q}(s)$ such that

(i) $r(t) = 0$ if $\mathbf{f}(t) = 0$ for all $\mathbf{u}(t)$ and $\mathbf{d}(t)$,

(ii) $r(t) \neq 0$ if $f_j(t) \neq 0$ and $f_k(t) = 0$ for all $k \neq j$.

The system (5.2) is completely fault detectable when there exists a stable filter $\mathbf{Q}(s)$ such that

(i) $r(t) = 0$ if $\mathbf{f}(t) = 0$ for all $\mathbf{u}(t)$ and $\mathbf{d}(t)$,

(ii) $r(t) \neq 0$ if $f_j(t) \neq 0$ and $f_k(t) = 0$ for all $k \neq j$, $j = 1, \dots, m$,

where m denotes the number of faults. Condition (i) ensures that the residual is decoupled from the input and outputs which implies that

$$\mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) & \mathbf{G}_{\mathrm{d}}(s) \\ \mathbf{I} & \mathbf{0} \end{bmatrix} = 0, \tag{5.4}$$

while the condition (ii) ensures proper fault to residual response which implies that

$$\mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_{\mathrm{f}}(s) \\ \mathbf{0} \end{bmatrix} \neq 0. \tag{5.5}$$

The system (5.2) is j-th fault detectable if and only if

$$\operatorname{rank} \begin{bmatrix} \mathbf{G}_{\mathrm{d}}(s) & \mathbf{G}_{f_{\mathrm{j}}}(s) \end{bmatrix} > \operatorname{rank} \mathbf{G}_{\mathrm{d}}(s),$$

where $\mathbf{G}_{f_{\mathrm{j}}}(s)$ is the j-th column of $\mathbf{G}_f(s)$, and system (5.2) is complete fault detectable if and only if

$$\operatorname{rank} \begin{bmatrix} \mathbf{G}_{\mathrm{d}}(s) & \mathbf{G}_{f_{\mathrm{j}}}(s) \end{bmatrix} > \operatorname{rank} \mathbf{G}_{\mathrm{d}}(s),\ j = 1, \dots, m.$$

The notion of strong fault detectability ensures that a persistent residual is produced in case of persistent faults, such as a step or sinusoid.

5.2.4. Fault isolability

To isolate faults, the interactions among all fault inputs must be handled. Therefore, the residual vector $\mathbf{r}(s)$ is decomposed into p filters of the type

$$\mathbf{r}^{(i)}(s) = \mathbf{Q}^{(i)}(s) \begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix},$$

where $\mathbf{r}^{(i)}(s)$ is the i-th residual subvector and p is the number or residuals. The internal fault representation $\mathbf{R}^{(i)}_{f_{\mathrm{j}}}(s)$ describes how j-th fault influences the i-th residual and is denoted as

$$\mathbf{R}^{(i)}_{f_{\mathrm{j}}}(s) := \mathbf{Q}^{(i)}(s) \begin{bmatrix} \mathbf{G}_{f_{\mathrm{j}}}(s) \\ \mathbf{0} \end{bmatrix}. \tag{5.6}$$

Using (5.6), the block-structured $p \times m$ TFM from faults to residuals is obtained

$$\mathbf{R}_{\mathrm{f}}(s) = \begin{bmatrix} \mathbf{R}^{(1)}_{f_1}(s) & \cdots & \mathbf{R}^{(1)}_{f_{\mathrm{m}}}(s) \\ \vdots & \ddots & \vdots \\ \mathbf{R}^{(p)}_{f_1}(s) & \cdots & \mathbf{R}^{(p)}_{f_{\mathrm{m}}}(s) \end{bmatrix},$$

where m denotes the number of faults and p the number or residuals. Using the elements of $\mathbf{R}_{\mathrm{f}}(s)$, a matrix S_{R_f} is associated to $\mathbf{R}_{\mathrm{f}}(s)$ whose elements are defined as

$$S_{R_f}(i,j) = 1 \quad \text{if} \quad \mathbf{R}^{(i)}_{f_{\mathrm{j}}}(s) \neq 0,$$
$$S_{R_f}(i,j) = 0 \quad \text{if} \quad \mathbf{R}^{(i)}_{f_{\mathrm{j}}}(s) = 0.$$

The structure matrix S_{R_f} defines how the residuals are related to faults. The i-th residual is decoupled from the j-th fault when $S_{R_f}(i,j) = 0$, whereas with $S_{R_f}(i,j) = 1$ the i-th residual is sensitive to the j-th fault. For example, the structure matrix

$$S_{R_f} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \tag{5.7}$$

shows that the first residual is only sensitive to the third fault while the second residual is influenced by all faults. Based on the structure matrix, the isolability of the system can be determined. The structure matrix (5.7) shows weak fault isolability, which means that any individual fault can be isolated with one fault occurring at a time. Strong fault isolability allows to isolate any number of simultaneous faults. This is not possible with (5.7) as it is not possible to isolate the faults with two active faults. Strong fault isolability can, for example, be achieved with $S_{R_f} = I$.

5.2.5. Model detection

A different approach in fault detection is when each fault is physically modeled. In that case, a set of N models is obtained and the active fault can be determined by identifying the model that best fits to the input-output measurement data. The set of models can be given as a multiple model description

$$\mathbf{y}^{(i)}(s) = \mathbf{G}_{\mathrm{u}}^{(i)}(s)\mathbf{u}^{(i)}(s) + \mathbf{G}_{\mathrm{d}}^{(i)}(s)\mathbf{d}^{(i)}(s), \tag{5.8}$$

where $\mathbf{y}^{(i)}$ are known outputs, $\mathbf{u}^{(i)}$ are known inputs, and $\mathbf{d}^{(i)}$ are unknown disturbances. The task of the to be designed filter is to detect the model from the collection of models, which best matches the current input-output behavior. This can be achieved by designing a residual generator of the following form

$$\mathbf{r}^{(i)}(s) = \mathbf{Q}^{(i)}(s)\begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix}, \, i = 1, \dots, N, \tag{5.9}$$

such that

(i) $\mathbf{r}^{(i)} = 0$ if $\mathbf{y}(t) = \mathbf{y}^{(i)}(t)$,

(ii) $\mathbf{r}^{(i)} \neq 0$ if $\mathbf{y}(t) = \mathbf{y}^{(j)}(t)$ for $j \neq i$.

Condition (i) implies that

$$\mathbf{Q}^{(i)}(s)\begin{bmatrix} \mathbf{G}_{\mathrm{u}}^{(i)}(s) & \mathbf{G}_{\mathrm{d}}^{(i)}(s) \\ \mathbf{I} & \mathbf{0} \end{bmatrix} = 0,$$

while condition (ii) implies

$$\mathbf{Q}^{(i)}(s)\begin{bmatrix} \mathbf{G}_{\mathrm{u}}^{(j)}(s) & \mathbf{G}_{\mathrm{d}}^{(j)}(s) \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \neq 0, \, i \neq j.$$

The multiple model (5.8) is model detectable for all $i = 1, \dots, N$ and $j = 1, \dots, N$ if and only if

$$\operatorname{rank}\begin{bmatrix} \mathbf{G}_{\mathrm{d}}^{(i)}(s) & \mathbf{G}_{\mathrm{d}}^{(j)}(s) & \mathbf{G}_{\mathrm{u}}^{(i)}(s) - \mathbf{G}_{\mathrm{u}}^{(j)}(s) \end{bmatrix} > \operatorname{rank}\mathbf{G}_{\mathrm{d}}^{(i)}(s) \quad \forall j \neq i.$$

Similar as to the notion of strong fault detection, strong model detection refers to the ability to detect a particular model out of the set of N models in case of persistent input and disturbance signals.

5.2.6. Residual evaluation

The residual signals $\mathbf{r}(t)$ generated by the residual generator $\mathbf{Q}(s)$ are further processed in the residual evaluator block. This block evaluates the magnitude of the residuals by taking a suitable norm of the signals. For most applications, a Narendra signal evaluation scheme (Narendra and Balakrishnan 1997) is suitable.

This scheme provides a weighted combination of instantaneous and low pass filtered values and is given as

$$\chi^{(i)}(t) = \alpha^{(i)}|r^{(i)}(t)| + \beta^{(i)} \int_0^t e^{-\gamma^{(i)}(t-\tau)}|r^{(i)}(\tau)| \,\mathrm{d}\tau,$$

where $\alpha^{(i)} \geq 0$ is the instantaneous weight, $\beta^{(i)} > 0$ the long term weight, and $\gamma^{(i)} > 0$ the forgetting factor. With the weights, a trade-off can be made between fast detection and robustness for model imperfections and disturbances. The scheme can be implemented as a first-order filter with a direct feed through term

$$\dot{\xi}^{(i)}(t) = -\gamma^{(i)}\xi^{(i)}(t) + \beta^{(i)}|r^{(i)}(t)|, \tag{5.10}$$

$$\chi^{(i)}(t) = \xi^{(i)}(t) + \alpha^{(i)}|r^{(i)}(t)|, \tag{5.11}$$

where $\xi(t)$ is an internal filter state.

5.2.7. Decision making

The filtered residual $\chi^{(i)}(t)$ is further evaluated as

$$\iota^{(i)}(t) = \begin{cases} 1, & \text{if } \chi^{(i)}(t) \geq \zeta^{(i)}, \\ 0, & \text{otherwise,} \end{cases} \tag{5.12}$$

where $\iota^{(i)}(t) = 1$ indicates that the i-th fault has occurred and $\zeta^{(i)}$ is a suitable threshold. The threshold $\zeta^{(i)}$ must be selected large enough to avoid false alarms but small enough to avoid missed detections.

Regarding decision making for the model detection, all components $\iota^{(i)}(t)$ must be nonzero, except the one to indicate the active model. No model fault is detected in case all components are nonzero or several components are zero.

5.3. Steering System Modeling

To apply model-based fault diagnosis on the steering system, a complete model of the system is required. Modeling a system is commonly split in two groups: white-box modeling or black-box modeling. White-box modeling refers to first-principles modeling by writing down the equations that govern the system. One of the major advantages is that such a model can easily be adjusted by changing parameters, thereby providing insight in system behavior (Nelles 2013). Drawback of the approach is that the models become unwieldy when dealing with complex systems or systems with many uncertain parameters. Black-box modeling, on the other hand, only selects a model structure with usually a limited number of parameters such that a wide variety of phenomena can be captured. Drawback of black-box modeling is that physical interpretation of the parameters is not possible. A combination of white box and black-box modeling is gray-box modeling (Bohlin 2006) in which system knowledge is included in the modeling process.

Gray-box modeling is used in this chapter to model the complete steering system as not all required input-output relations can be obtained with black-box modeling and not all system parameters are known. Figure 5.2 shows an overview of the complete steering system model that is a concatenation of a driver model, steering system model, and single-track vehicle model. The inputs of the model are the driver torque τ_d and steering motor torque τ_sm. The torque around the steering axis τ_w is experienced when the vehicle is turning and tends to resist the attempted turn. The torque is a combination of the self-aligning torque of the tire and a torque resulting from the suspension geometry. The measurable sensor outputs are steering column angle δ_sc, steering column angular velocity $\dot{\delta}_\mathrm{sc}$, and steering column torque τ_tb.

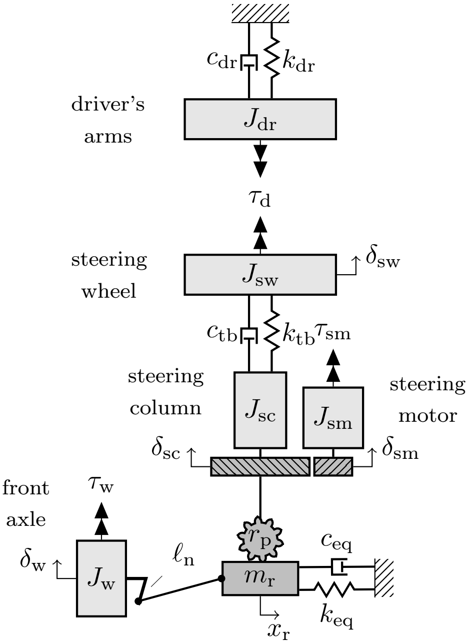


Figure 5.2.: Complete steering system model

5.3.1. Steering system model

The steering system model is a linearized variant of the model presented in Chapter 4. The jacking torque stiffness and damping is approximated by an equivalent spring and damper on the steering rack to compensate for the Coulomb friction effect. The steering system consists of a steering wheel modeled as inertia J_{sw} connected to the steering column via a torsion bar modeled as spring k_{tb} and damper c_{tb}. This torsion bar is part of the torque sensor (Yoshida 2002). The torque τ_{d} represents the torque applied by the driver, and the driver's arm are represented as inertia J_{dr} attached to fixed world with spring k_{dr} and damper c_{dr} (Pick and Cole 2007). The steering motor with inertia J_{sm} is coupled to the steering column via a worm gear with gear ratio i_{sm}. As both universal joints in the steering axis operate roughly under the same angle, their eccentricity effect is assumed to be negligible (Harrer and Pfeffer 2017). The rotational motion of the steering column is converted to a translational motion of the steering rack via a linear rack and pinion with radius r_{p}. The self-centering effects due to suspension geometry often referred as jacking torque is modeled as a spring k_{eq} and damper c_{eq} on the steering rack (Parmar and Hung 2004). The steering rack with mass m_{r} is coupled with a steering link to the knuckle arm with length ℓ_{n} (Marouf et al. 2012). Both front wheels are lumped in one axle modeled with inertia J_{w}.

Given the schematic depiction of the steering system in Figure 5.2, the equations of motion for the steering model are derived using Lagrange's equations of motion and are given as (Marouf et al. 2012)

$$J_{\mathrm{sw}}\ddot{\delta}_{\mathrm{sw}} = \tau_{\mathrm{d}} - c_{\mathrm{tb}}(\dot{\delta}_{\mathrm{sw}} - \dot{\delta}_{\mathrm{sc}}) - k_{\mathrm{tb}}(\delta_{\mathrm{sw}} - \delta_{\mathrm{sc}}), \tag{5.13a}$$

$$\begin{aligned} J_{\mathrm{eq}}\ddot{\delta}_{\mathrm{sc}} = \tau_{\mathrm{sm}} i_{\mathrm{sm}} + \tau_{\mathrm{w}} i_{\mathrm{ss}} + c_{\mathrm{tb}}(\dot{\delta}_{\mathrm{sw}} - \dot{\delta}_{\mathrm{sc}}) + k_{\mathrm{tb}}(\delta_{\mathrm{sw}} - \delta_{\mathrm{sc}}) \\ - c_{\mathrm{eq}} r_{\mathrm{p}}^2 \dot{\delta}_{\mathrm{sc}} - k_{\mathrm{eq}} r_{\mathrm{p}}^2 \delta_{\mathrm{sc}}, \end{aligned} \tag{5.13b}$$

where $J_{\mathrm{eq}} = J_{\mathrm{sc}} + J_{\mathrm{m}} i_{\mathrm{sm}}^2 + J_{\mathrm{w}} i_{\mathrm{ss}}^2 + m_{\mathrm{r}}^2 r_{\mathrm{p}}^2$ and $i_{\mathrm{ss}} = {}^{r_{\mathrm{p}}}/_{\ell_{\mathrm{n}}}$. The equivalent jacking torque damping and stiffness coefficients are given as $c_{\mathrm{eq}} = c_{\mathrm{r}} + c_{\mathrm{f}}$ and $k_{\mathrm{eq}} = k_{\mathrm{r}} + k_{\mathrm{f}}$, respectively, where c_{f} and k_{f} are additional factors to approximate the dry friction effects at certain operating conditions. The steering model inputs are the torque applied by the steering motor τ_{sm}, the torque applied by the driver τ_{d}, and the resulting torque around the steering axis τ_{w}. The steering motor torque is given by $\tau_{\mathrm{sm}} = k_{\mathrm{sm}} I_{\mathrm{sm}}$ in which the electrical dynamics of the steering motor are neglected since the time constant of the electrical system is considerably smaller than that of the mechanical system. The measured steering model outputs are steering column angle δ_{sc}, steering column angular velocity $\dot{\delta}_{\mathrm{sc}}$, and steering column torque $\tau_{\mathrm{tb}} = k_{\mathrm{tb}}(\delta_{\mathrm{sw}} - \delta_{\mathrm{sc}})$. Similar to Loof (2018), most of the steering model parameters are obtained via direct measurements, calculations, or component identification (Stoffels 2019; Baaij 2019). Table 5.1 lists these parameters. The other unknown steering model parameters are estimated in Section 5.4.

Table 5.1.: Steering model parameter values

Description	Symbol	Value	Source
Steering wheel inertia	J_{sw}	$0.0114\,\mathrm{kg\,m^2}$	Identification
Steering column inertia	J_{sc}	$5\mathrm{e}{-3}\,\mathrm{kg\,m^2}$	Identification
Axle mass moment of inertia	J_{w}	$0.3940\,\mathrm{kg\,m^2}$	Identification
Motor mass moment of inertia	J_{m}	$3.28\mathrm{e}{-4}\,\mathrm{kg\,m^2}$	Specsheet
Steering rack mass	m_{r}	3 kg	Specsheet
Motor constant	k_{sm}	0.052 Nm/A	Specsheet
Torsion bar stiffness	k_{tb}	80.97 Nm/rad	Identification
Steering knuckle length	ℓ_{n}	0.078 m	Identification
Steering rack pinion radius	r_{p}	0.0056 m	Identification
Steering motor gear ratio	i_{sm}	13.67	Identification

5.3.2. Driver's arm model

A driver touching the steering wheel can be modeled as a spring-damper system (Pick and Cole 2007). The torque exerted by the driver on the steering wheel τ_{d} is then modeled as

$$\tau_{\mathrm{d}} = -c_{\mathrm{dr}}\dot{\delta}_{\mathrm{sw}} - k_{\mathrm{dr}}\delta_{\mathrm{sw}}, \tag{5.14}$$

where c_{dr} is the driver's arm damping and k_{dr} the driver's arm stiffness. In addition, the driver's arm inertia J_{dr} must be added to the steering wheel inertia J_{sw} when the driver is holding the steering wheel.

The dynamic properties of a driver holding the steering wheel are obtained from Pick and Cole (2007) and are listed in Table 5.2. Two cases are presented: one with relaxed muscle state in which the drivers hold the steering wheel with just enough force to prevent slipping as the steering wheel moved. The second case is with contracted muscle state in which the drivers firmly hold the steering wheel to maintain it in a certain position.

Table 5.2.: Averaged driver's arm parameters (source: (Pick and Cole 2007))

Description	Symbol	Relaxed	Contracted
Driver's arms inertia	J_{dr}	$0.11\,\mathrm{kg\,m^2}$	$0.11\,\mathrm{kg\,m^2}$
Driver's arms damping	c_{dr}	0.61 Nms/rad	1.1704 Nms/rad
Driver's arms stiffness	k_{dr}	2.9 Nm/rad	56.4225 Nm/rad

5.3.3. Single track model

The operating regime of the considered automated vehicle is low speeds and accelerations. Therefore, a linear single track vehicle model can be employed to obtain the exerted torque around the steering axis τ_{w}. The steering model is concatenated with a single track vehicle model that includes tyre relaxation behavior (Genta 1997)

$$m(\dot{v}_{\mathrm{y}} + \omega_{\mathrm{z}} v_{\mathrm{x}}) = -C_{\mathrm{f}}\alpha_{\mathrm{f}} - C_{\mathrm{r}}\alpha_{\mathrm{r}}, \tag{5.15a}$$

$$J_{\mathrm{zz}}\dot{\omega}_{\mathrm{z}} = -C_{\mathrm{f}}\ell_{\mathrm{f}}\alpha_{\mathrm{f}} + C_{\mathrm{r}}\ell_{\mathrm{r}}\alpha_{\mathrm{r}}, \tag{5.15b}$$

$$\sigma_{\mathrm{f}}\dot{\alpha}_{\mathrm{f}} = -v_{\mathrm{x}}\alpha_{\mathrm{f}} + v_{\mathrm{y}} + \ell_{\mathrm{f}}\omega_{\mathrm{z}} - v_{\mathrm{x}}\delta_{\mathrm{sc}} i_{\mathrm{s}}, \tag{5.15c}$$

$$\sigma_{\mathrm{r}}\dot{\alpha}_{\mathrm{r}} = -v_{\mathrm{x}}\alpha_{\mathrm{r}} + v_{\mathrm{y}} - \ell_{\mathrm{r}}\omega_{\mathrm{z}}, \tag{5.15d}$$

where α_{f} is the sideslip angle of the front axle, α_{r} is the sideslip angle of the rear axle, C_{f} the cornering stiffness of the front axle, C_{r} the cornering stiffness of the rear axle, σ_{f} the relaxation length of the front equivalent tire, σ_{r} the relaxation length of the rear equivalent tire, ℓ_{f} the length from front axle to the center of gravity, ℓ_{r} the length from rear axle to the center of gravity, v_{x} the longitudinal velocity, v_{y} the lateral velocity, and ω_{z} the yawrate. Using the single track model (5.15), the torque around the steering axis is obtained by

$$\tau_{\mathrm{w}} = C_{\mathrm{mz}}\alpha_{\mathrm{f}},$$

where $C_{\mathrm{mz}} = (r_{\mathrm{m}} + r_{\mathrm{t}})C_{\mathrm{f}}$ is the self-centering stiffness with r_{m} the mechanical trail and r_{t} the pneumatic trail. Table 5.3 lists the parameters of the single track model which are obtained from the vehicle and tyre identification tests performed in Chapter 4. A longitudinal velocity $v_{\mathrm{x}} = 10\,\mathrm{m/s}$ is selected given the operating regime of the considered automated vehicle, see Chapters 2 and 3

Table 5.3.: Single track model parameters (source: (Baaij 2019))

Description	Symbol	Value
Cornering stiffness front axle	C_{f}	38.893 kN/rad
Cornering stiffness rear axle	C_{r}	58.054 kN/rad
Relaxation length front axle	σ_{f}	0.2667 m
Relaxation length rear axle	σ_{r}	0.5200 m
Self-centering stiffness	C_{mz}	1337.9 Nm/rad
Vehicle mass	m	708 kg
Vehicle mass moment of inertia	J_{zz}	350 kg m^2
Front axle to center of gravity	ℓ_{f}	0.9978 m
Rear axle to center of gravity	ℓ_{r}	0.6882 m

5.4. Gray-Box System Identification

A reliable model of the system is important when using model-based fault diagnosis approaches. Therefore, a gray-box system identification procedure is employed to estimate the unknown steering model parameters (Bohlin 2006), such as damping, after which the steering model is validated using experimental data.

5.4.1. Experiment design

For the identification experiment, the vehicle is placed with its front wheels on turn plates to reduce static friction between the tyres and road surface, thereby aiming to represent normal driving conditions. The measured outputs during the experiment are the steering column angle δ_{sc}, steering column angular velocity $\dot{\delta}_{\mathrm{sc}}$, and steering column torque τ_{tb}. The input current I_{sm} is applied by the steering motor.

As identification signal, a random phase multisine signal is used, consisting of 50 sine waves linearly spaced on the interval 0.05 Hz to 50 Hz. A multisine is selected for the experiments as the base frequency sine wave keeps the system in motion, thereby reducing dry friction influences. In addition, the energy in the input signal is concentrated at discrete frequency points, thereby increasing the signal-to-noise ratio. The multisine signal is a segment of 2048 samples and with a sample time of 0.01 seconds, the generated signal has a period of 20.48 seconds. During the experiment the signal is constantly repeated to obtain multiple segment of data. The periodicity and continuity of the signal enables averaging of sequences to improve the signal-to-noise ratio without the need for tools such as windowing.

The amplitude of the excitation signal is scaled as the multisine signal amplitude range lies between -1 and 1. The identification signal gain is designed such that typical driving conditions are mimicked in terms of steering column angle and steering column angular velocity. The typical maximum values obtained during normal driving are 2 radians for the steering column angle, 4 radians/s for the steering column angular velocity, and 10 Nm for the steering wheel torque. These values agree with the identified normal driving behavior values presented in (Van der Sande et al. 2012).

While the steering system is a constrained system in terms of maximum steering rack displacement, open loop system identification is applied. This is possible with a careful selection of the identification signal gain and the vehicle resting on its wheel as the jacking torque results in a self centering effect. Contact of the steering rack with its end stops and torque sensor with its end stops must be avoided during the identification experiments as this introduces nonlinear effects, which are undesired for linear system identification.

5.4.2. Estimation model

For the gray-box identification (Bohlin 2006), the estimation model must be represented in state space formulation. By defining $\mathbf{q}(t) = \begin{bmatrix} \delta_{\mathrm{sw}} & \delta_{\mathrm{sc}} \end{bmatrix}^{\mathsf{T}}$ and $\boldsymbol{\tau}(t) = \begin{bmatrix} I_{\mathrm{sm}} & \tau_{\mathrm{d}} & \tau_{\mathrm{w}} \end{bmatrix}^{\mathsf{T}}$, the steering system model (5.13) can be written as

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{D}\dot{\mathbf{q}}(t) + \mathbf{K}\mathbf{q}(t) = \mathbf{S}\boldsymbol{\tau}(t), \tag{5.16}$$

with

$$\mathbf{M} = \begin{bmatrix} J_{\mathrm{sw}} & 0 \\ 0 & J_{\mathrm{sc}} + J_{\mathrm{sm}} i_{\mathrm{sm}}^2 + J_{\mathrm{w}} i_{\mathrm{ss}}^2 + m_{\mathrm{r}} r_{\mathrm{p}}^2 \end{bmatrix}, \mathbf{D} = \begin{bmatrix} c_{\mathrm{tb}} & -c_{\mathrm{tb}} \\ -c_{\mathrm{tb}} & c_{\mathrm{tb}} + c_{\mathrm{eq}} r_{\mathrm{p}}^2 \end{bmatrix},$$
$$\mathbf{K} = \begin{bmatrix} k_{\mathrm{tb}} & -k_{\mathrm{tb}} \\ -k_{\mathrm{tb}} & k_{\mathrm{tb}} + k_{\mathrm{eq}} r_{\mathrm{p}}^2 \end{bmatrix}, \text{ and } \mathbf{S} = \begin{bmatrix} 0 & 1 & 0 \\ i_{\mathrm{sm}} k_{\mathrm{sm}} & 0 & i_{\mathrm{ss}} \end{bmatrix}.$$

By introducing the input array $\mathbf{u}(t) = \boldsymbol{\tau}^{\mathsf{T}}$, state array $\mathbf{x}(t) = \begin{bmatrix} \mathbf{q}^{\mathsf{T}} & \dot{\mathbf{q}}^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}}$, and output array $\mathbf{y}(t) = \begin{bmatrix} \delta_{\mathrm{sc}} & \dot{\delta}_{\mathrm{sc}} & \tau_{\mathrm{tb}} \end{bmatrix}^{\mathsf{T}}$, model (5.16) is transformed into standard the state-space form

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \tag{5.17a}$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t), \tag{5.17b}$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{D} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{M}^{-1}\mathbf{S} \end{bmatrix},$$
$$\mathbf{C} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ k_{\mathrm{tb}} & -k_{\mathrm{tb}} & 0 & 0 \end{bmatrix} \quad \text{and} \quad \mathbf{D} = \begin{bmatrix} \mathbf{0} \end{bmatrix}.$$

In case all unknown matrix elements of (5.17) are independent, state-space model identification with structured parameterizations can be applied (Yu et al. 2015). For the steering model, there are matrix element dependencies, and therefore, the parameters are estimated using a numerical minimization procedure.

5.4.3. Model estimation

The model parameters are estimated by minimizing the error between the measured data and the model output. The cost function is a weighted least square error defined as

$$V(\boldsymbol{\epsilon}) = \frac{1}{N} \sum_{t=1}^{N} \mathbf{e}^{\mathsf{T}}(t, \boldsymbol{\epsilon}) \mathbf{W} \mathbf{e}(t, \boldsymbol{\epsilon}), \tag{5.18}$$

where N is the number of samples, $\mathbf{e}(t, \boldsymbol{\epsilon})$ is the difference between the measured and predicted output parameterized by the parameter vector $\boldsymbol{\epsilon}$, and $\mathbf{W}$ is a weighting matrix. The weighting matrix is used to control the relative weights of the outputs and is selected as $\mathbf{W} = \mathrm{diag}(3, 1, 1)$ to give more importance to the steering column angle δ_{sc} as this signal contains less noise compared to the steering column angular velocity $\dot{\delta}_{\mathrm{sc}}$ and steering column torque τ_{tb} signal.

The cost function (5.18) is minimized by utilizing MATLAB's function 'greyest' which uses a least squares algorithm. The quality of the fit is assessed by the Normalized Root Mean Squared Error (NRMSE) expressed as a percentage

$$\varepsilon_{\mathrm{fit}} = \left(1 - \frac{\|\mathbf{y}_{\mathrm{m}} - \mathbf{y}_{\mathrm{p}}\|}{\|\mathbf{y}_{\mathrm{m}} - \bar{\mathbf{y}}_{\mathrm{m}}\|}\right) \cdot 100\,\%, \tag{5.19}$$

where $\|\cdot\|$ denotes the 2-norm of a vector, $\mathbf{y}_{\mathrm{m}}$ the vector of measured outputs, $\mathbf{y}_{\mathrm{p}}$ the vector of predicted model outputs, and $\bar{\mathbf{y}}_{\mathrm{m}}$ the vector with the means of the measured outputs. A $\varepsilon_{\mathrm{fit}}$ value of 100 % indicates a perfect fit where the predicted outputs are identical to the measured outputs while a value of 0 % indicates the model is no better than a straight line through the mean of the data.

Figure 5.3 compares the model outputs with the measured outputs for one segment length. As can be seen, the model outputs and measured outputs agree reasonably well. The fit percentage for steering column angle δ_{sc}, steering column angular velocity $\dot{\delta}_{\mathrm{sc}}$, and steering column torque τ_{tb} are 83.94 %, 33.55 %, and 31.33 %, respectively. The lower fit values for the angular velocity and steering column torque might be due to the noise levels on both signals, and the resolution of the angular velocity sensor signal is limited (0.0698 rad/s).

Table 5.4 lists the corresponding estimated steering model parameter values. The values show an increase in jacking torque damping and stiffness compared to the values presented in Chapter 4 on Page 53 as the Coulomb friction is neglected in this linear model.

Table 5.4.: Estimated steering model parameter values

Description	Symbol	Value
Torsion bar damping	c_{tb}	0.005 Nms/rad
Jacking torque damping	c_{eq}	96.02 kNs/m
Jacking torque stiffness	k_{eq}	33.825 kN/m

5.4.4. Model validation

As the model output comparison in Figure 5.3 does not guarantee that the estimated model is able to represent other dynamics of the system, the quality

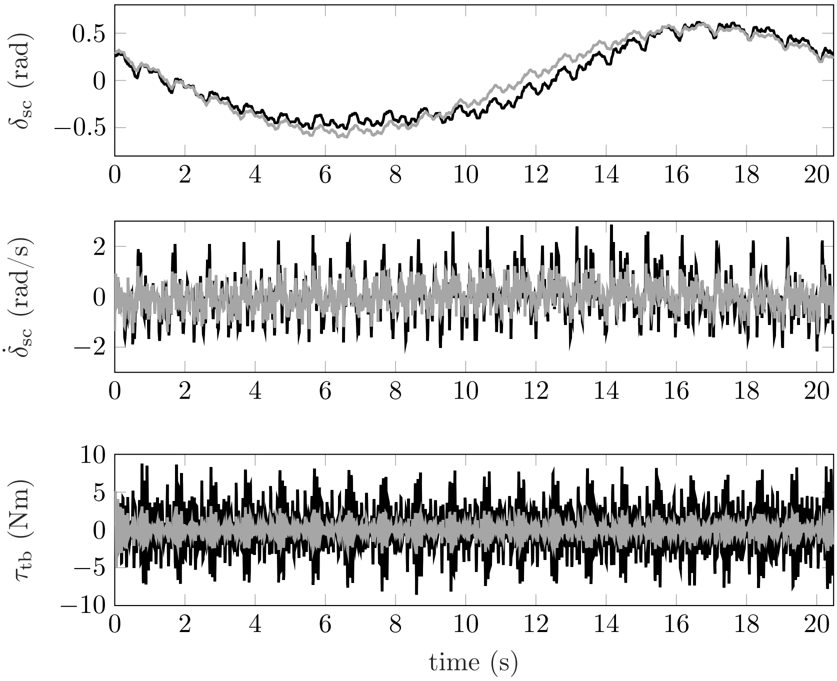


Figure 5.3.: Comparison between model output and measurement data for multisine excitation signal: measurement data (——) and model output (——)

of the model is validated using a different data set. This validation data set is obtained with the same experimental setup but with a Pseudo-Random Binary Sequence (PBRS) as excitation signal. The selected PBRS signal consists of 1024 samples that with the sample time of 0.01 s corresponds to a sequence length of 10.24 s. The signal is constantly repeated during the experiment which results in a periodic signal whose frequency properties mimic white noise. The values of the PBRS signal lie within the range -1 and 1. Therefore, a suitable gain is applied to scale the signal such that the typical driving conditions are mimicked, similar as with the multisine excitation signal.

Figure 5.4 presents the results of the validation data set in which the model output is compared with the measured data for one segment length. As can be seen, the model outputs and measured outputs agree reasonably well. The fit percentage for steering column angle δ_{sc}, steering column angular velocity $\dot{\delta}_{sc}$, and steering column torque τ_{tb} are 63.93 %, 34.16 %, and 32.07 %, respectively.

To compare the estimated gray-box model with other solutions, Figures 5.5

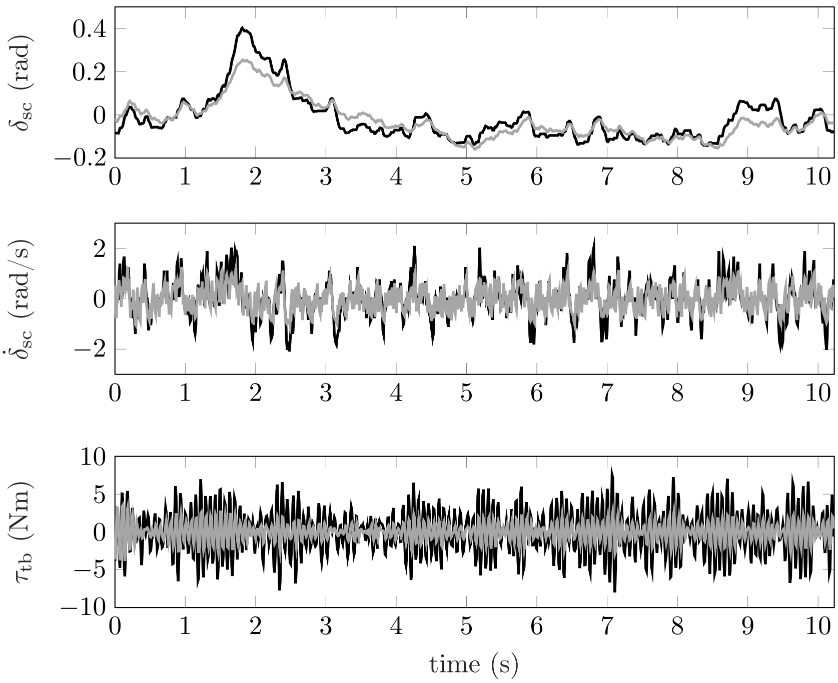


Figure 5.4.: Comparison between model output and measurement data for pseudo-random binary sequence excitation signal: measurement data (——) and model output (——)

and 5.6 depict the transfer functions from steering motor current I_{sm} to steering column angle δ_{sc} and torsion bar torque τ_{tb}, respectively. The figures show a comparison between the gray-box model estimate with a state space model estimate and the measured frequency-response function. The state space model estimate is a fourth order model generated with the MATLAB function 'n4sid'. The model order equals the order of the gray box to allow for a fair comparison. The difference in low frequent behavior in the transfer function from steering motor current I_{sm} to torsion bar torque τ_{tb} indicates a dry friction or spring effect. This can be included in the model by adding a spring from the steering wheel to the fixed world to approximate the dry friction. Besides that, the models show more or less the same behavior, indicating that the gray-box model is able to serve as a model for the steering system for normal driving conditions.

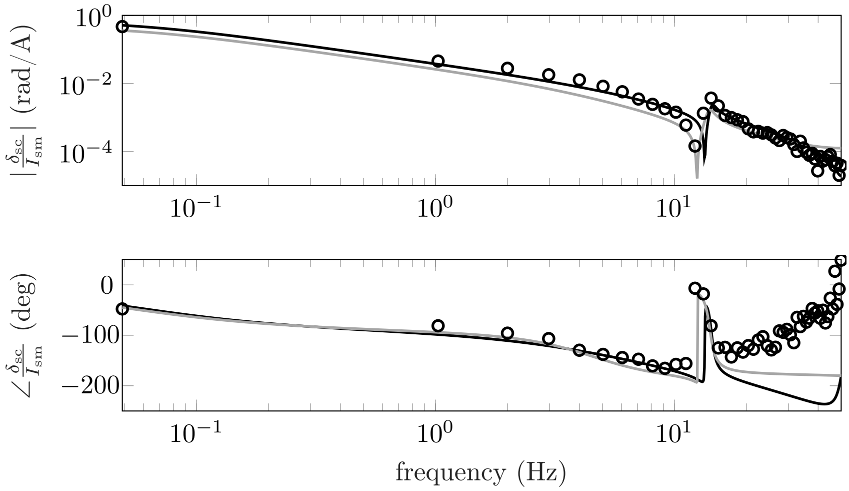


Figure 5.5.: Transfer function from steering motor current I_{sm} to steering column angle δ_{sc}: gray-box model (——), state space model (——), and frequency-response function (○)

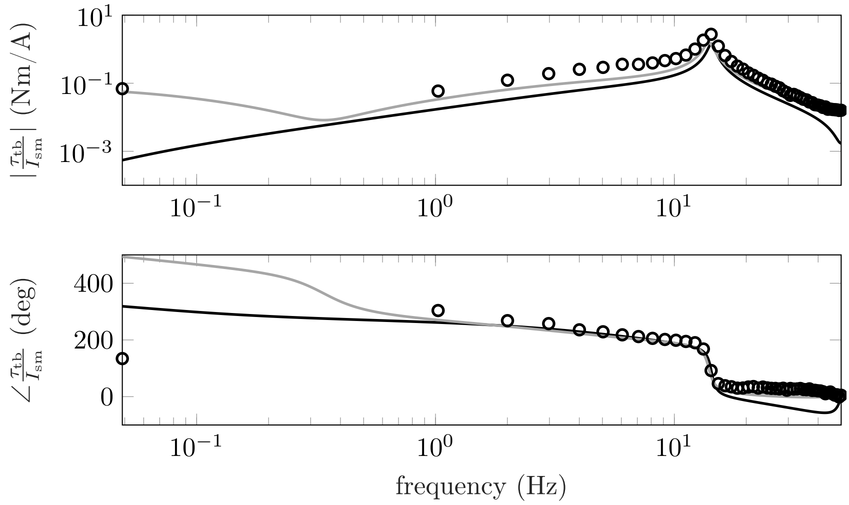


Figure 5.6.: Transfer function from steering motor current I_{sm} to torsion bar torque τ_{tb}: gray-box model (——), state space model (——), and frequency-response function (○)

5.5. Fault Diagnosis

The validated model is used to detect faults in the steering system of the automated vehicle. As the sensor faults have already been covered in the hardware and software design, as shown in Chapters 2 and 3, this section covers the steering system actuator faults.

5.5.1. Actuator faults

The automated steering system, as presented in Figure 5.2, has three inputs of which one is controlled: the steering motor torque τ_{sm}. The torque on the steering wheel τ_{d} is the results from the driver input. The torque around the steering axis τ_{w} is experienced when the vehicle is turning and tends to resist the attempted turn. The torque is a combination of the self-aligning torque of the tire and a torque resulting from the suspension geometry.

Fault modeling

The actuator faults must be modeled to be able to detect them. Most faults can be represented as an additive fault as shown by Chen and Patton (1999). Hence, the steering motor actuator fault is modeled as

$$\tau_{\mathrm{sm}} = I_{\mathrm{sm}} k_{\mathrm{sm}} + f_{\mathrm{sm}}, \tag{5.20}$$

where τ_{sm} is the actual steering motor torque, I_{sm} the commanded steering motor current, k_{sm} the steering motor constant, and f_{sm} the additive fault signal. A user intervention can also be considered as an actuator fault in which the controlled input is zero when the ADS is performing the dynamic driving task. Therefore, the user intervention fault is modeled as

$$\tau_{\mathrm{d}} = 0 + f_{\mathrm{d}}, \tag{5.21}$$

where τ_{d} is the actual torque exerted on the steering wheel, 0 the by the ADS commanded steering wheel torque, and f_{d} an additive fault signal.

Filter synthesis

By substituting the fault models (5.20) and (5.21) into the complete steering model, consisting of the steering system model (5.13) coupled with the single track model (5.15), its input-output description can be written as

$$\mathbf{y}(s) = \mathbf{G}_{\mathrm{u}}(s)\mathbf{u}(s) + \mathbf{G}_{\mathrm{f}}(s)\mathbf{f}(s), \tag{5.22}$$

where $\mathbf{y}(s)$, $\mathbf{u}(s)$, and $\mathbf{f}(s)$ are vectors transformed to the Laplace domain of the outputs $\mathbf{y}(t) = \begin{bmatrix} \delta_{\mathrm{sc}} & \tau_{\mathrm{tb}} & a_{\mathrm{y}} & \omega_{\mathrm{z}} \end{bmatrix}^{\mathsf{T}}$, inputs $\mathbf{u}(t) = \begin{bmatrix} I_{\mathrm{sm}} & 0 \end{bmatrix}^{\mathsf{T}}$, and faults $\mathbf{f}(t) = \begin{bmatrix} f_{\mathrm{sm}} & f_{\mathrm{d}} \end{bmatrix}^{\mathsf{T}}$, respectively. The 4×2 TFM $\mathbf{G}_{\mathrm{u}}(s)$ and 4×2 TFM $\mathbf{G}_{\mathrm{f}}(s)$ describe the input to output and fault to output relations, respectively. As the faults directly influence the inputs, $\mathbf{G}_{\mathrm{f}}(s)$ equals $\mathbf{G}_{\mathrm{u}}(s)$.

To produce the fault indicator signals, a general linear residual generator is employed which is given by

$$\mathbf{r}(s) = \mathbf{Q}(s) \begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix}, \tag{5.23}$$

where $\mathbf{Q}(s)$ is the to be designed filter such that the residuals are zero in the fault-free case and at least one of them distinctively nonzero when a fault is present. The filter $\mathbf{Q}(s)$ must be stable and proper to be physical realizable. By substituting the input-output description (5.22) into (5.23), the following is obtained

$$\mathbf{r}(s) = \mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) \\ \mathbf{I} \end{bmatrix} \mathbf{u}(s) + \begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) \\ \mathbf{0} \end{bmatrix} \mathbf{f}(s), \tag{5.24}$$

where $\mathbf{G}_{\mathrm{f}}(s)$ has been replaced by $\mathbf{G}_{\mathrm{u}}(s)$. To ensure the residuals $\mathbf{r}(t)$ are zero when no fault $\mathbf{f}(t)$ is present for any control input $\mathbf{u}(t)$, the residuals must be decoupled from the control input. In other words, the filter $\mathbf{Q}(s)$ must satisfy

$$\mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) \\ \mathbf{I} \end{bmatrix} = 0. \tag{5.25}$$

To solve for the fault detection filter basis, a left null-space basis of $\begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) & \mathbf{I} \end{bmatrix}^{\mathsf{T}}$ can be computed. However, a trivial filter is $\mathbf{Q}_1(s) = \begin{bmatrix} \mathbf{I} & -\mathbf{G}_{\mathrm{u}}(s) \end{bmatrix}$ which is always possible when no disturbances are included in the input-output description (5.22). In case disturbances are included in the input-output descrption, they have to be decoupled from the residuals which is not always possible, as explained in Section 5.2.

The fault to residual response can be shaped by pre-multiplying $\mathbf{Q}_1(s)$ with suitable transfer functions $\mathbf{Q}_{\mathrm{i}}(s)$ to obtain the final filter in factored form as $\mathbf{Q}(s) = \mathbf{Q}_{\mathrm{k}}(s) \cdots \mathbf{Q}_2(s)\mathbf{Q}_1(s)$, where $\mathbf{Q}_2(s)$ to $\mathbf{Q}_{\mathrm{k}}(s)$ are the filters to shape the response or to address synthesis requirements such as being physical realizable.

As the dynamical behavior of the filter can be freely chosen, the poles of the filter $\mathbf{Q}(s)$ have been set to -5 by employing the Fault Detection Toolbox (Varga 2017a) for the numerical computations of the filter. The selection of the poles allows fast detection of faults while filtering out high frequency measurement noise. A bank of three filters is obtained and each filter is of order two, so the total order of the overall filter $\mathbf{Q}(s)$ is six.

Filter assessment

To assess the obtained residual generator $\mathbf{Q}(s)$, the internal fault representation

$$\mathbf{R}_{\mathrm{f}}(s) = \mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) \\ \mathbf{0} \end{bmatrix}, \tag{5.26}$$

is used. Figure 5.7 shows the response of $\mathbf{R}_{\mathrm{f}}(s)$ to unit fault step inputs $\mathbf{f}(t) = \begin{bmatrix} f_{\mathrm{sm}} & f_{\mathrm{d}} \end{bmatrix}^{\mathsf{T}}$ with all initial conditions set to zero. As can be seen, the actuator and user intervention fault can be detected as the fault to residual response is nonzero. The corresponding fault signature matrix is given as

$$S_{R_f} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}, \tag{5.27}$$

from which it is clear that it is possible to isolate both faults as well. The filter is validated in Sections 5.6 and 5.7 in which simulation and experimental results are presented.

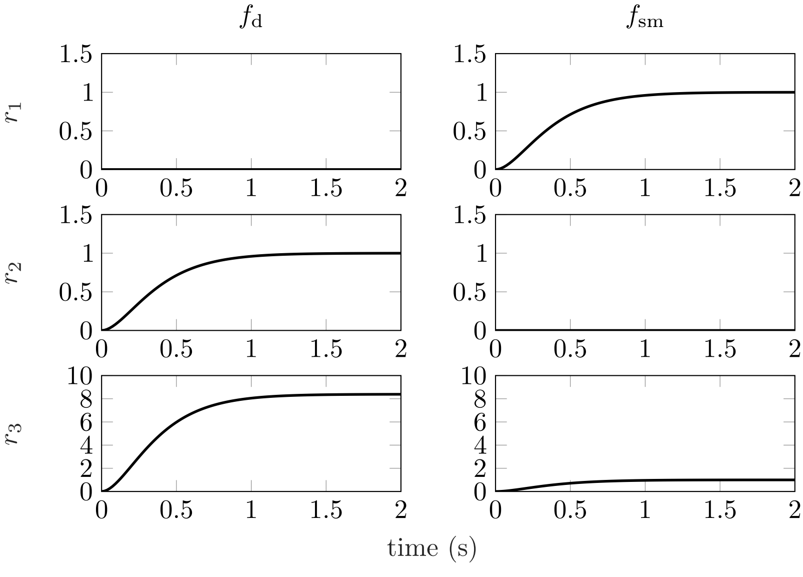


Figure 5.7.: Step response plot of $\mathbf{R}_{\mathrm{f}}(s)$ (5.26) from actuator faults f_{sm} and f_{d} to residuals $\mathbf{r}(t)$

5.5.2. Driver state detection

The previous fault diagnosis method does not differentiate between holding the steering wheel and applying a corrective torque. With automated driving, it might be undesirable to disengage the ADS each time the driver is holding the steering wheel. Disengagements must however occur when the driver is applying a corrective torque, therefore, driver state detection is required.

Multiple model modeling

The steering model (5.13) coupled with the single track model (5.15) and the driver's arm model (5.14) with the three cases hands-off, hands-on relaxed muscle state, and hands-on co-contracted muscle state can be described with multiple models as

$$\mathbf{y}^{(i)}(s) = \mathbf{G}_{\mathrm{u}}^{(i)}(s)\mathbf{u}(s), \tag{5.28}$$

where $\mathbf{y}^{(i)}(s)$ and $\mathbf{u}(s)$ are vectors transformed to the Laplace domain of the outputs $\mathbf{y}(t) = \begin{bmatrix} \delta_{\mathrm{sc}} & \tau_{\mathrm{tb}} & a_{\mathrm{y}} & \omega_{\mathrm{z}} \end{bmatrix}^{\mathsf{T}}$ and inputs $\mathbf{u}(t) = \begin{bmatrix} I_{\mathrm{sm}} & 0 \end{bmatrix}^{\mathsf{T}}$. The 4×2 TFMs $\mathbf{G}_{\mathrm{u}}^{(i)}(s)$ describe the inputs to outputs relation depending on the driver's arm state.

Figure 5.8 shows a comparison between the models $\mathbf{G}_{\mathrm{u}}^{(i)}(s)$ in which the transfer functions are presented from input to outputs for the three cases hands-off, hands-on relaxed muscle state, and hands-on co-contracted muscle state. As can be seen, the driver's arm plays an important role as the low frequent magnitude differs significantly between the responses. Also, the eigenfrequency shifts from roughly 14 Hz to 7 Hz. These observations agree with the measurement data obtained by Loof (2018) with a modified VW Lupo. This vehicle has a comparable steering geometry to a Renault Twizy.

Filter synthesis

To detect which model of (5.28) best fits the current input-output behavior, a bank of three residual generators can be constructed of the following form

$$\mathbf{r}^{(i)}(s) = \mathbf{Q}^{(i)}(s) \begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix}, \tag{5.29}$$

where $\mathbf{Q}^{(i)}(s)$ are the to be designed filters. The filters must be designed such that the residual $\mathbf{r}^{(i)}(t)$ is close to zero when the measurement $\mathbf{y}(t)$ agrees with the by model i expected measurement $\mathbf{y}^{(i)}(t)$ and distinctively nonzero when the measurement agrees with model j $\mathbf{y}(t) = \mathbf{y}^{(j)}(t)$ for $i \neq j$. The overall residual

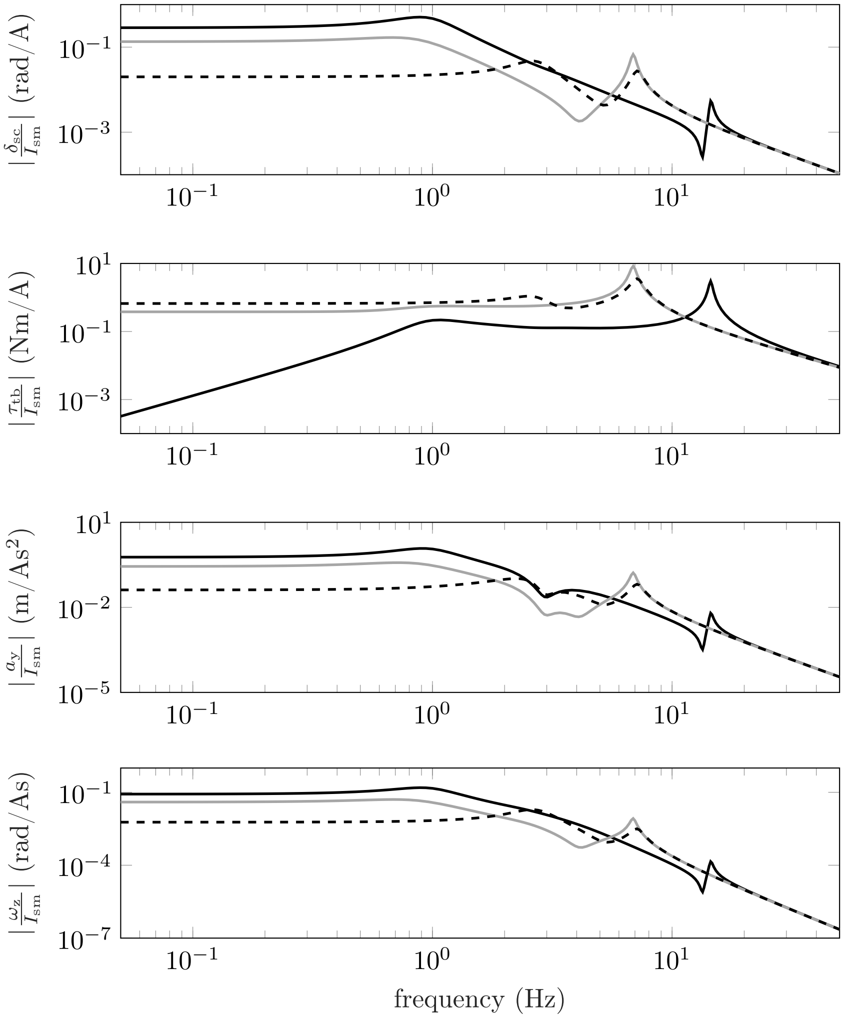


Figure 5.8.: Transfer functions from steering motor current I_{sm} to steering column angle δ_{sc}, torsion bar torque τ_{tb}, lateral acceleration a_{y}, and yawrate w_{z}: hands-off (—), hands-on relaxed muscle state (—), and hands-on contracted muscle state (- - -)

generator $\mathbf{Q}(s)$ is obtained by stacking the three filters $\mathbf{Q}^{(i)}(s)$ as

$$\mathbf{r}(s) = \begin{bmatrix} \mathbf{Q}^{(1)}(s) \\ \mathbf{Q}^{(2)}(s) \\ \mathbf{Q}^{(3)}(s) \end{bmatrix} \begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix}, \tag{5.30}$$

where the Fault Detection Toolbox (Varga 2017a) has been employed for the numerical computations of the filter.

Filter assessment

Similar to the previous filter as shown in Figure 5.7, the internal representation

$$\mathbf{R}_{\mathrm{u}}(s) = \mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) \\ \mathbf{I} \end{bmatrix}, \tag{5.31}$$

is used to assess the obtained residual generator $\mathbf{Q}(s)$. Figure 5.9 shows the response of $\mathbf{R}_{\mathrm{u}}(s)$ to an unit control step input $u(t) = I_{\mathrm{sm}}$ with all initial conditions set to zero. Each column corresponds to a specific model and the rows show the residual outputs. All diagonal entries are zero as the input-output behavior fits to that model. The off diagonal elements grow with distance from the diagonal to indicate a larger mismatch between that model and the input-output behavior.

5.6. Simulation Results

To evaluate the residual generators designed in Sections 5.5.1 and 5.5.2, a simulation model is used. The high-fidelity nonlinear simulation model that is presented in Chapter 4 is employed for the evaluation. The simulation model consists of a multibody model of the vehicle including sensors and actuators. The model is validated using data obtained from full-scale driving tests. Sensor noises are added to the model to simulate real-life conditions. The selected noise intensities and sample times are based on the specifications of the real sensors and are listed in Table 5.5.

The simulation starts with a constant longitudinal velocity of 2.5 m/s and without hands on the steering wheel. Two faults are subsequently and individual injected. First, a steering motor fault $f_{\mathrm{sm}} = 3\,\mathrm{A}$ is injected during the interval $t = 5\,\mathrm{s}$ to $10\,\mathrm{s}$. Then, the driver's arm state transitions to a relaxed muscle state during the interval $t = 14\,\mathrm{s}$ to $23\,\mathrm{s}$. Next, the driver's arm state transitions to contracted muscle state during the interval $t = 23\,\mathrm{s}$ to $32\,\mathrm{s}$. Finally, the last part of the simulation interval, $t = 32\,\mathrm{s}$ to $35\,\mathrm{s}$, consists again of a fault-free and hands-off situation.

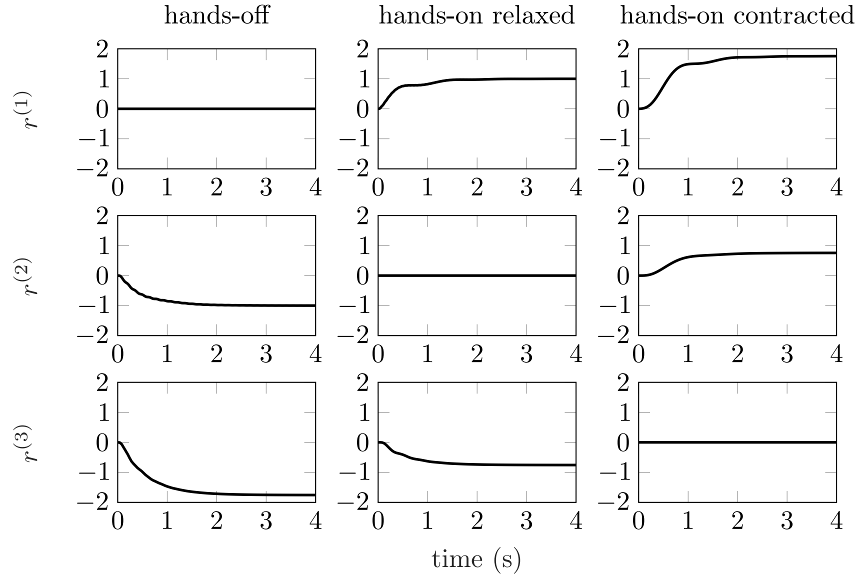


Figure 5.9.: Step response plot of $\mathbf{R}_\mathrm{u}(s)$ (5.31) to an unit input $u(t) = I_\mathrm{sm}$ with all initial conditions set to zero

As the driver's arm are modeled as a passive spring and damper, no driver state detection is possible without dynamic excitation. Therefore, a realistic steering column reference signal must be applied. The reference signal must also be safe to test with the experimental setup to be able to compare the simulation with an experiment. The reference signal

$$\delta_\mathrm{ref}(t) = 0.3\sin(0.6\pi t) + 0.2\sin(1.2\pi t), \tag{5.32}$$

is selected, which is composed of two sinusoids with a different frequency and amplitude, and where t denotes the simulation time.

Figure 5.10 shows the measured outputs and input signal during the simulation, which are the steering column angle δ_sc, steering column torque τ_tb, lateral acceleration a_y, yawrate ω_z, and steering motor current I_sm. The steering motor fault ($t = 5\,\mathrm{s}$ to $10\,\mathrm{s}$) can be noticed by a small increase in steering motor current I_sm. The driver's arm with relaxed muscle state ($t = 14\,\mathrm{s}$ to $23\,\mathrm{s}$) and user intervention ($t = 24\,\mathrm{s}$ to $32\,\mathrm{s}$) can be noticed from the increase in steering column torque τ_tb and decrease of the steering column angle δ_sc compared to the fault-free situation.

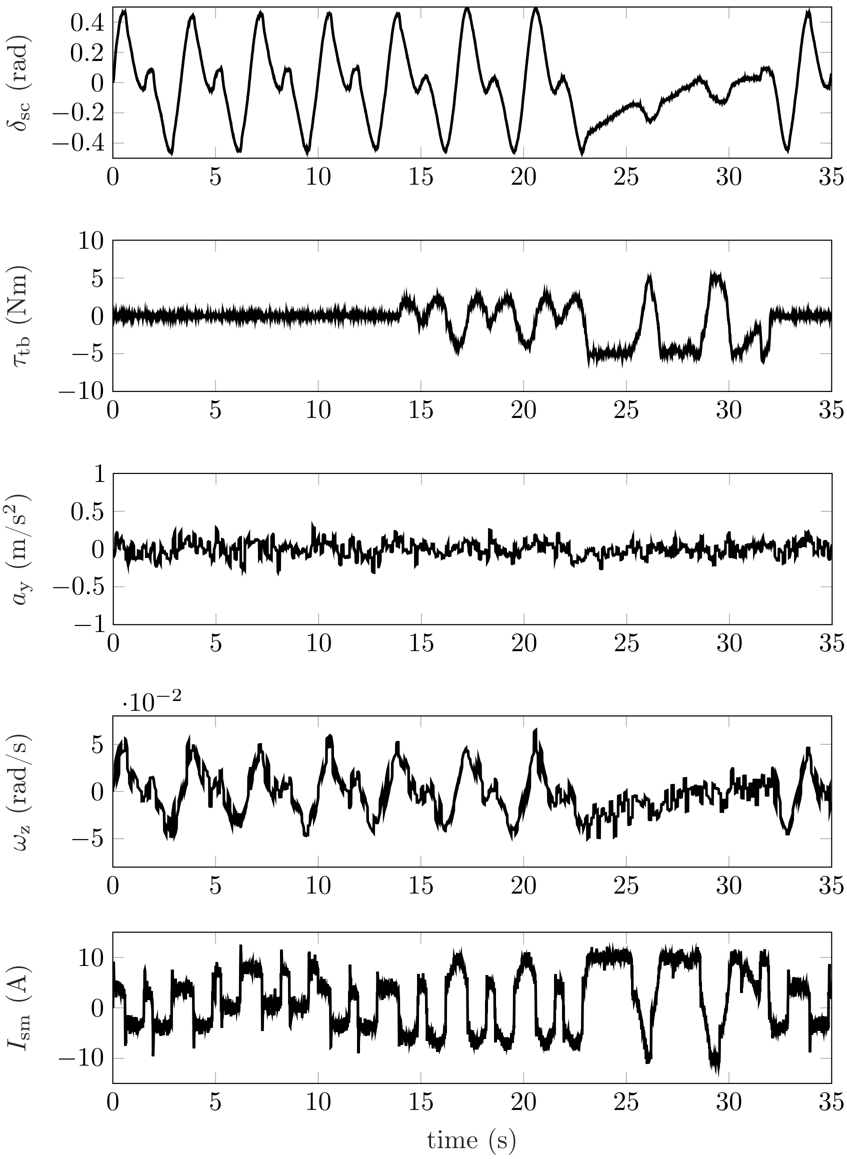


Figure 5.10.: Measured outputs $\mathbf{y}(t)$ and input $u(t)$ signals during the simulation: steering column angle δ_{sc}, steering column torque τ_{tb}, lateral acceleration a_{y}, yawrate ω_{z}, and steering motor current I_{sm}

Table 5.5.: Simulation model sensor noise standard deviations

Description	Symbol	Value
Steering column angle noise	$\sigma_{\delta_{\mathrm{sc}}}$	0.002 rad
Steering column torque	$\sigma_{\tau_{\mathrm{tb}}}$	0.1 Nm
Lateral acceleration	$\sigma_{a_{\mathrm{y}}}$	0.1 m/s^2
Yawrate	$\sigma_{\omega_{\mathrm{z}}}$	0.01 rad/s
Steering motor current	$\sigma_{I_{\mathrm{sm}}}$	0.5 A

5.6.1. Actuator faults

To detect the actuator faults, the output signals $\mathbf{y}(t)$ and input signal $u(t)$ obtained from the simulation are put through the filter $\mathbf{Q}(s)$ as constructed in Section 5.5.1. The filter outputs are the residual signals $\mathbf{r}(t)$ which are then filtered in the Narendra filter (5.10) with filter parameters $\alpha = 0.35$, $\beta = 0.65$, and $\gamma = 2$ to obtain the filtered residuals $\boldsymbol{\chi}(t)$. The faults are detected by applying suitable thresholds on the filtered residuals $\boldsymbol{\chi}(t)$ to obtain the fault index signal $\boldsymbol{\iota}(t)$. The selected thresholds are $\zeta_{\mathrm{sm}} = 2.75$ for the steering motor fault, $\zeta_{\mathrm{d}} = 2$ for the user intervention, and $\zeta_{\mathrm{f}} = 14$ for the fault signal. The threshold are ad hoc selected and fine tuning of the thresholds is needed to obtain the optimal values, which is a trade off between quick response time and low false alarm rate.

Figure 5.11 shows the residuals $\mathbf{r}(t)$ along with the filtered residuals $\boldsymbol{\chi}(t)$ and the fault index signals $\boldsymbol{\iota}(t)$. As can be seen, the steering motor fault and user intervention are both detected. The user intervention is detected by residuals $r^{(2)}$ and $r^{(3)}$ after 0.84 s. The steering motor fault is detected with $r^{(1)}$ after 1.4 s but not with $r^{(3)}$ while residual $r^{(3)}$ is sensitive to both faults, as shown in Figure 5.7. The difference in signal amplitude between both fault intervals, however, makes it unsuitable to apply a single threshold ζ. Therefore, the steering motor fault is not detected with residual $r^{(3)}$. The user intervention is also wrongly detected with $r^{(1)}$ as steering motor fault. This means that for reliable detection of the steering motor fault, residual $r^{(2)}$ should also be evaluated. A steering motor fault is only detected when the steering motor fault is detected with $r^{(1)}$ and no user intervention is detected.

5.6.2. Driver state detection

To detect the driver's arm state, the output signals $\mathbf{y}(t)$ and input signal $u(t)$ obtained from the simulation are put through the filter $\mathbf{Q}(s)$ as constructed in Section 5.5.2. The filter outputs are the residual signals $\mathbf{r}(t)$ which are normalized by dividing through their maximum values over the interval. The normalized

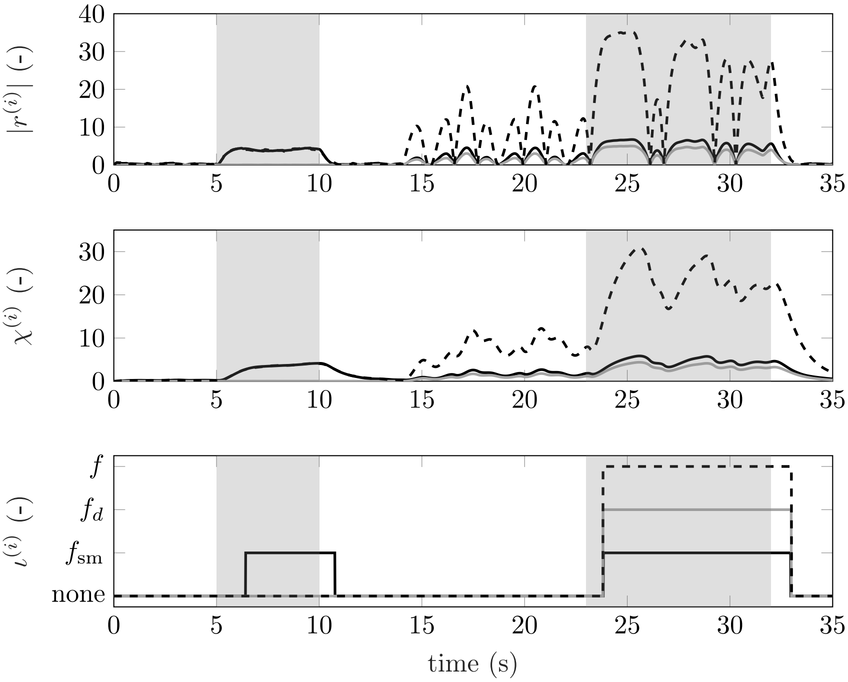


Figure 5.11.: Detected actuator faults during the simulation: f_{sm} (——), f_{d} (——), and f (- - -). The shaded areas (▭) indicate when faults are introduced in the simulation

residuals are then filtered in the Narendra filter (5.10) with filter parameters $\alpha = 0.35$, $\beta = 0.65$, and $\gamma = 2$ to obtain the filtered residuals $\boldsymbol{\chi}(t)$. The faults are detected by applying suitable thresholds on the filtered residuals $\boldsymbol{\chi}(t)$ to obtain the fault index signal $\boldsymbol{\iota}(t)$. However, since the residuals $\mathbf{r}(t)$ are normalized and always one of the models should be active, the active model can also be selected as the lowest filtered residual $\boldsymbol{\chi}(t)$. The model with the lowest filtered residual $\boldsymbol{\chi}(t)$ best fits the current input-output behavior.

Figure 5.12 shows the residuals $\mathbf{r}(t)$ along with the filtered residuals $\boldsymbol{\chi}(t)$ and the fault index signals $\boldsymbol{\iota}(t)$. As can be seen, the driver's arm states are detected but not always correctly. For instance, during the relaxed muscle interval a contracted muscle state is detected four times while during the contracted muscle state interval a hands-off state is detected once. The main reason for this is that the difference between the hands-off and relaxed muscle state residual during the simulation is close to zero.

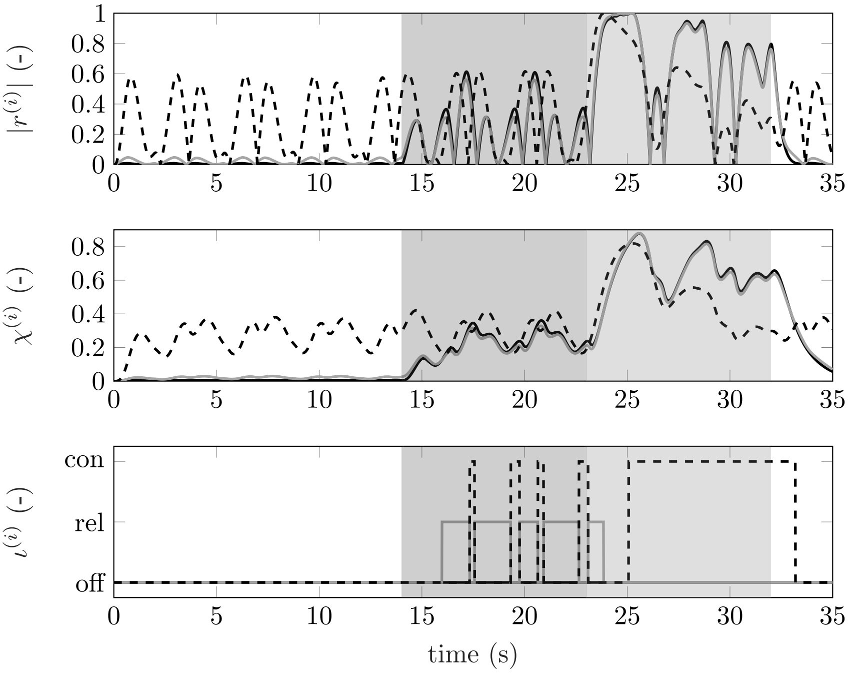


Figure 5.12.: Detected driver's arm state during the simulation: $r^{(1)}(t)$ (——), $r^{(2)}(t)$ (——), and $r^{(3)}(t)$ (- - -). The shaded area (▨) indicates relaxed arm muscle state and the area (□) indicates contracted arm muscle state

5.7. Experimental Results

The experiment is executed with the automated vehicle described in Chapters 2 and 3. The experiment is performed in the same manner as the simulation to be able to compare the results. The only difference between the simulations and experiments is the road surface, which is smooth in simulations while additional road disturbances are encountered during the experiments. Figure 5.13 shows the measured outputs and input signal during the experiment, which can be compared with the simulation results shown in Figure 5.10. The measured outputs during the experiment show similar behavior compared to the simulation but the amplitude of the steering column angle δ_{sc} and yawrate ω_{z} are slightly lower than in the simulation, which likely is caused by the fact that steering motor current I_{sm} is also slightly lower than in the simulation. Since the same steering controller is used in the simulations and experiments, the difference is probably caused by unmodeled amplifier characteristics. The user intervention is clearly visible in the interval $t \approx 24\,\mathrm{s}$ to $32\,\mathrm{s}$ but the steering motor fault ($t = 5\,\mathrm{s}$ to $10\,\mathrm{s}$) or the driver's arm with relaxed muscle state ($t \approx 14\,\mathrm{s}$ to $23\,\mathrm{s}$) are nearly unnoticeable in the unprocessed vehicle response.

5.7.1. Actuator faults

To detect the actuator faults, the output signals $\mathbf{y}(t)$ and input signal $u(t)$ obtained from the experiment are put through the filter $\mathbf{Q}(s)$ as constructed in Section 5.5.1. The filter outputs are the signals $\mathbf{r}(t)$ which are then filtered in the Narendra filter (5.10) with filter parameters $\alpha = 0.35$, $\beta = 0.65$, and $\gamma = 2$ to obtain the filtered residuals $\boldsymbol{\chi}(t)$. The faults are detected by applying suitable thresholds on the filtered residuals $\boldsymbol{\chi}(t)$ to obtain the fault index signal $\boldsymbol{\iota}(t)$. The selected thresholds are $\zeta_{\mathrm{sm}} = 1.65$ for the steering motor fault, $\zeta_{\mathrm{d}} = 4$ for the user intervention, and $\zeta_{\mathrm{f}} = 10$ for the fault signal. The threshold are ad hoc selected and fine tuning of the thresholds is needed when implementing the algorithm to obtain the optimal values, which is a trade off between quick response time and low false alarm rate.

Figure 5.14 shows the residuals $\mathbf{r}(t)$ along with the filtered residuals $\boldsymbol{\chi}(t)$ and the fault index signals $\boldsymbol{\iota}(t)$. As can be seen, the steering motor fault and user intervention are both detected. The user intervention is detected by residuals $r^{(2)}$ and $r^{(3)}$. The steering motor fault is detected with $r^{(1)}$ but not with $r^{(3)}$ while residual $r^{(3)}$ is sensitive to both faults, as shown in Figure 5.7. The difference in amplitude, however, makes it unsuitable to apply a static threshold ζ. Therefore, the steering motor fault is not detected with residual $r^{(3)}$ similar to the simulation result.

5.7.2. Driver state detection

The residual signals $\mathbf{r}(t)$ to detect the driver's arm state are obtained by feeding the output signals $\mathbf{y}(t)$ and input signal $u(t)$ obtained from the experiment through the filter $\mathbf{Q}(s)$ as constructed in Section 5.5.2. The residuals $\mathbf{r}(t)$ are then normalized and filtered in the Narendra filter (5.10) with filter parameters $\alpha = 0$, $\beta = 1$, and $\gamma = 1$ to obtain the filtered residuals $\boldsymbol{\chi}(t)$. The model that best fits the current input-output behavior is then selected as the model with the lowest filtered residual $\chi^{(i)}(t)$.

Figure 5.14 shows the residuals $\mathbf{r}(t)$ along with the filtered residuals $\boldsymbol{\chi}(t)$ and the fault index signal $\boldsymbol{\iota}(t)$. As can be seen, the driver's arm with relaxed muscle state is correctly detected for two short periods but stays most of the time undetected. The contracted muscle state is also correctly detected for a short period but most of the time it is detected as a relaxed arm muscle state. A probable cause is that the dynamic properties of the test person's arm does not match the dynamic driver's arm properties as presented by Pick and Cole (2007). Another reason might be that in the current test environment not enough excitation of all dynamics is

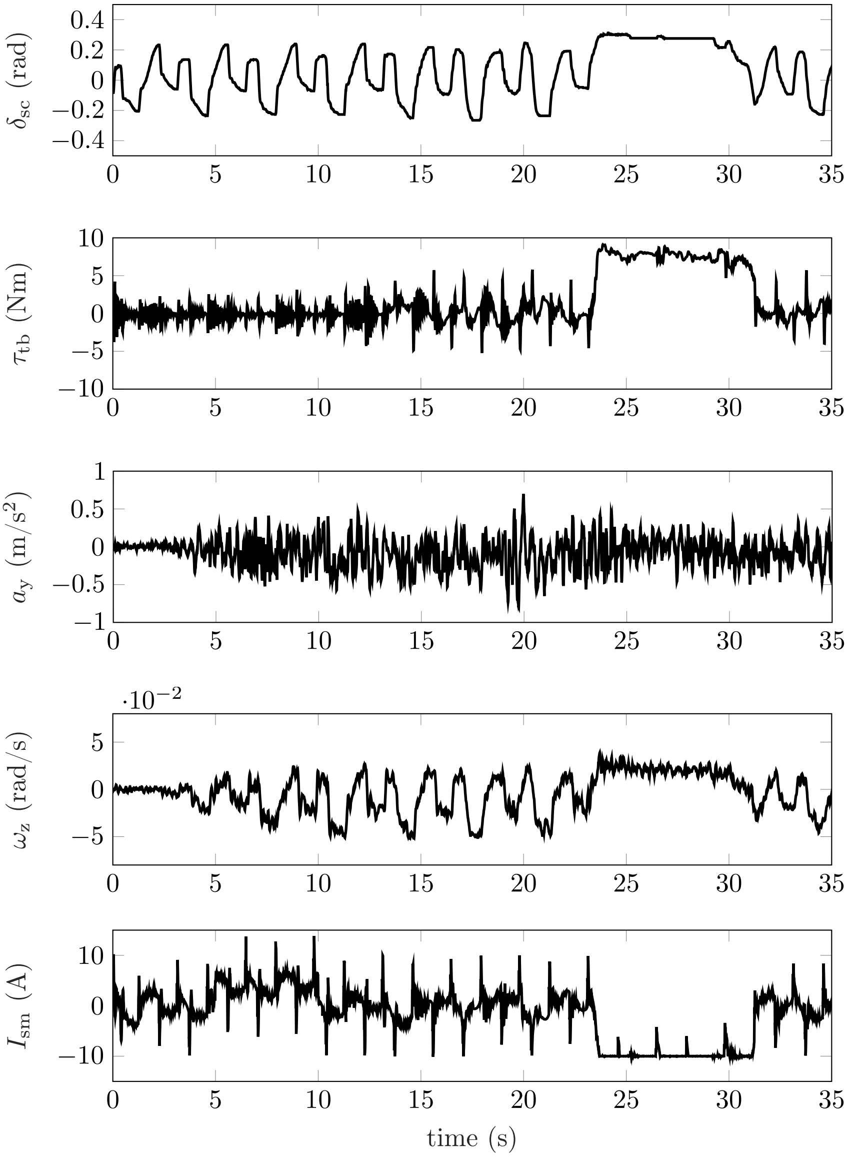


Figure 5.13.: Measured outputs $\mathbf{y}(t)$ and input $u(t)$ signals during the experiment: steering column angle δ_{sc}, steering column torque τ_{tb}, lateral acceleration a_{y}, yawrate ω_{z}, and steering motor current I_{sm}

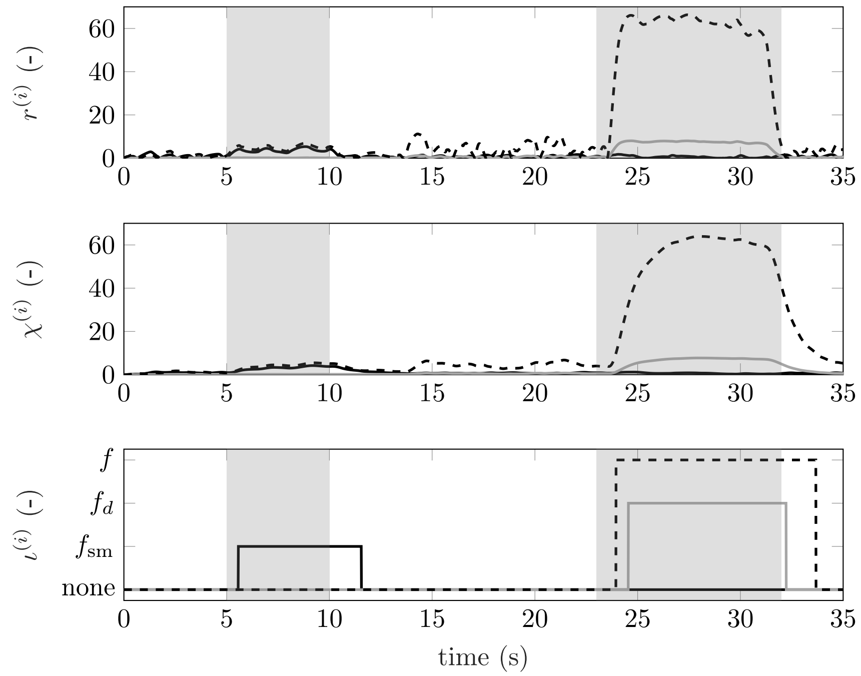


Figure 5.14.: Detected actuator faults during the experiment: f_{sm} (——), f_{d} (——), and f (- - -). The shaded areas (▭) indicate when faults are introduced in the simulation

achieved ($a_{\text{y}} \leq 0.5\,\text{m/s}^2$, $\omega_{\text{z}} \leq 0.04\,\text{rad/s}$). Higher velocities and larger steering angles can unfortunately not be achieved in a safe manner at the current available test site.

5.8. Conclusion

In this chapter, model-based fault diagnosis for an automated steering system is considered. The proposed strategy uses a linear approximated model of the complete steering system consisting of a driver's arm model, a steering system model, and a single track vehicle model. The linear steering system model captures the system well in the frequency range 1 Hz to 50 Hz. Nonlinear effects, such as dry friction in the steering system, become important below 1 Hz. The fault diagnosis method to detect a user intervention and a steering actuator fault has been successfully demonstrated in simulation and validated experimentally. Also

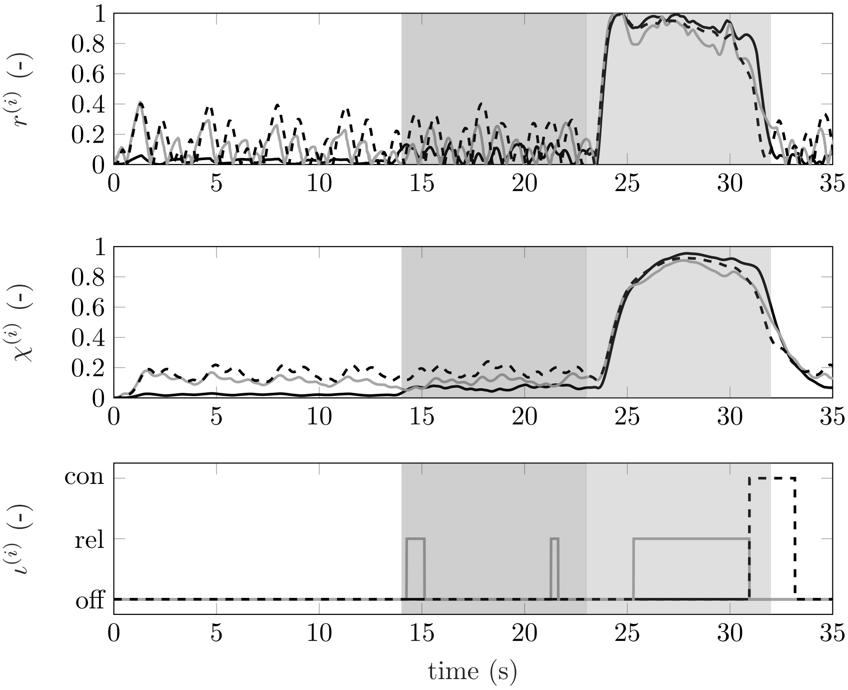


Figure 5.15.: Detected driver state during the experiment: $r^{(1)}(t)$ (——), $r^{(2)}(t)$ (——), and $r^{(3)}(t)$ (- - -). The shaded area (▨) indicates relaxed arm muscle state and the area (▭) indicates contracted arm muscle state

driver state detection is successfully shown, but the initial tests show that the method is sensitive to the size of the input signals. The fault diagnosis method is also sensitive to model inaccuracies arising from the linear approximation of the nonlinear steering system model. Wear and other long-time vehicle effects can change the system behavior and must be taken into account when designing the filters and thresholds. The model-based fault diagnosis techniques remove the need for extra sensors, thereby reducing cost and eliminating possible points of failure. The technique can also be employed to act as a redundant measure to provide higher levels of safety.

6.

Fault Diagnosis for Connected Vehicles

Abstract - Cooperative Adaptive Cruise Control (CACC) systems utilize a wireless link between vehicles which allows driving with shorter inter-vehicle distances compared to a normal driver, thereby increasing road throughput and driver comfort. At these reduced vehicle following distances, the driver cannot be considered as backup and the CACC must supervise itself to avoid unsafe driving situations. This chapter presents a fault diagnosis strategy to detect when incorrect information is received from the preceding vehicle or when a wrong object is tracked. The strategy is implemented in two prototype CACC vehicles. Simulations and experiments shows that the proposed method is able to detect faults, thereby avoiding unsafe situations.

6.1. Introduction

Cooperative driving in which real-time information of other traffic participants is shared using a wireless connection is a promising field to improve driver comfort and safety (Reichardt et al. 2002). The wireless received information complements the information received by using the onboard sensors, such as radar or vision systems. Whereas the onboard sensors only provide information in the line-of-sight of the sensors, the wireless communication receives information from all traffic participants in the vicinity, thereby extending the information horizon. Additionally, states and inputs that cannot be measured with environmental sensing systems can be communicated via this wireless link.

One example of cooperative driving is Cooperative Adaptive Cruise Control (CACC) (Shladover et al. 2012; Sheikholeslam and Desoer 1990) which is a vehicle following system that automatically maintains a desired distance to a preceding vehicle. To this end, a radar sensor is commonly used to measure the inter-vehicle distance and velocity. The wireless link is employed to communicate extra information from the preceding vehicle to the following vehicle, such as position and acceleration. The wireless information enables CACC equipped vehicles to follow each other with shorter inter-vehicle distances compared to a normal driver-operated vehicle, thereby increasing road throughput and driver comfort.

The type of CACC controller can vary between different implementations. For example, Ploeg (2014) employed CACC on a string of identical vehicles controlling the longitudinal motion of each vehicle. The lateral vehicle motion of the following vehicles is still manually controlled by applying the correct steering actions. Extending on the work of Shaw and Hedrick (2007), Naus et al. (2010), and Ploeg (2014), a method is presented by Bayuwindra (2019) to control the vehicle in longitudinal and lateral motion. The proposed controller objective is to follow exactly the same path as the preceding vehicle. Similarly, Van Hoek et al. (2018) developed a CACC algorithm that follows a predecessor by controlling the longitudinal and lateral vehicle motion with taking dynamic obstacles into account.

The discussed methods allow driving at short inter-vehicle distances with time headways smaller than what is recommended by most road authorities (SWOV 2012). At these reduced distances, the driver might not react in time in case of a fault situation, such as wireless package loss, and cannot be considered as backup (Dajsuren and Loupias 2019). Therefore, it is important for a CACC system to timely detect faults and react accordingly.

Possible faults are when the vehicle with which the wireless link is established is different to the one that is tracked with the environmental perception system. Environmental perception systems detect multiple objects and from all these detected objects a target vehicle must be selected. The target vehicle is commonly selected as the vehicle driving closest in front of the vehicle on the same predicted path. This vehicle might however be a different one compared to the one with which the wireless link is established as the wireless communication is not a line of sight sensor. This situation can happen when, for instance, an unequipped vehicle cuts-in or through a platoon or when platooning on a curved multi lane road. With a right hand corner coming ahead, the environmental perception system might incorrectly select a target vehicle in the faster driving left-hand lane. This would cause the vehicle to accelerate as the CACC tries to maintain a certain inter-vehicle distance. To prevent these unsafe situations, a fault must be quickly detected because driving with reduced inter-vehicle distances makes the system inherently unsafe in case of failure.

The aim of the research presented in this chapter is to detect when incorrect information is received from the preceding vehicle or when a wrong object is tracked without assuming a specific CACC controller structure. A fault is detected when the preceding vehicle that is tracked by the environmental perception system behaves differently compared to the data received via the wireless link. In case of a detected fault, the countermeasure is to disable CACC and switch to Adaptive Cruise Control (ACC) or switch to a degraded CACC as presented in Ploeg et al. (2014). The ACC controller will then automatically increase the following distance to a safe distance. The countermeasure in case of an environmental perception system failure is to bring the vehicle to a safe stop using a controlled deceleration.

The outline of this chapter is as follows. Section 6.2 provides an overview of the CACC implementation, consisting of localization, target tracking, and control blocks. Section 6.3 models the CACC system and provides the relevant model parameters. In addition, the fault diagnosis strategy is explained. Section 6.4 validates the fault diagnosis system by performing simulation results. Section 6.5 then presents experimental results. Finally, Section 6.6 provides the main conclusions.

6.2. CACC System Implementation

Before the system can be modeled and faults are injected, the relevant system implementation must be known. Therefore, the CACC system implementation is described together with the relevant parameters. To this end, consider Figure 6.1 that shows a top view of two connected vehicles in which $x_\mathrm{h}, y_\mathrm{h}$ are the global coordinates of the host vehicle expressed in a tangent local reference frame L, ℓ_r the distance between the rear axle and the rear bumper of the target vehicle, ℓ_f the distance between rear axle and the front bumper of the host vehicle, ψ_h the heading angle of the host vehicle, v_h the longitudinal velocity of the host vehicle, a_h the longitudinal acceleration of the host vehicle, d the distance between the rear axle of the host vehicle and rear bumper of the target vehicle, and α the angle between the rear axle of the host vehicle and rear bumper of the target vehicle.

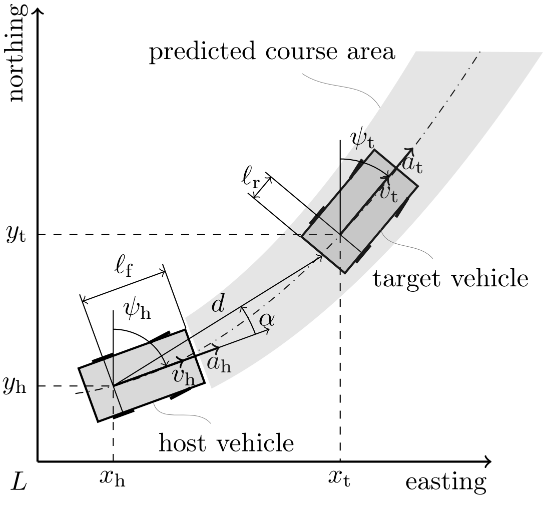


Figure 6.1.: Schematic illustration of a CACC scenario showing the kinematical states and coordinate system (adapted from (Ploeg 2014))

6.2.1. Host-tracking

Not all relevant kinematic states can be acquired directly or with sufficient accuracy, such as the heading angle ψ_{h}. Therefore, a host-tracking block is used to estimate the relevant kinematic states with their covariances. The kinematical states are determined by fusing the position $\mathbf{p}(t)$ and velocity $\mathbf{v}(t)$ data obtained from the Global Navigation Satellite System (GNSS) receiver with the output data from a Inertial Measurement Unit (IMU). The output data of the IMU consists of the angular velocity $\boldsymbol{\omega}(t)$ and specific force $\mathbf{s}(t)$, which is defined as the difference between inertial and gravitational acceleration.

The implemented scheme is shown in Figure 6.2, which is referred as a loosely-coupled closed-loop GNSS-aided Inertial Navigation System (INS) configuration (Farrell 2008). The idea behind the scheme is that the INS constantly provides the navigation solution in terms of position $\mathbf{p}(t)$, velocity $\mathbf{v}(t)$, and attitude $\mathbf{q}(t)$ by integrating information received from the motion sensors. Whenever there is an update from the GNSS receiver, the position and velocity differences are used as input to an error-state Kalman filter (ESKF). The ESKF uses these differences to estimate correction data for the navigation solution ($\delta\hat{\mathbf{p}}, \delta\hat{\mathbf{v}}, \delta\hat{\mathbf{q}}$) as well as correction data for the IMU ($\mathbf{b}_{\mathbf{s}}, \mathbf{b}_{\boldsymbol{\omega}}$). The relevant equations are discussed block by block below.

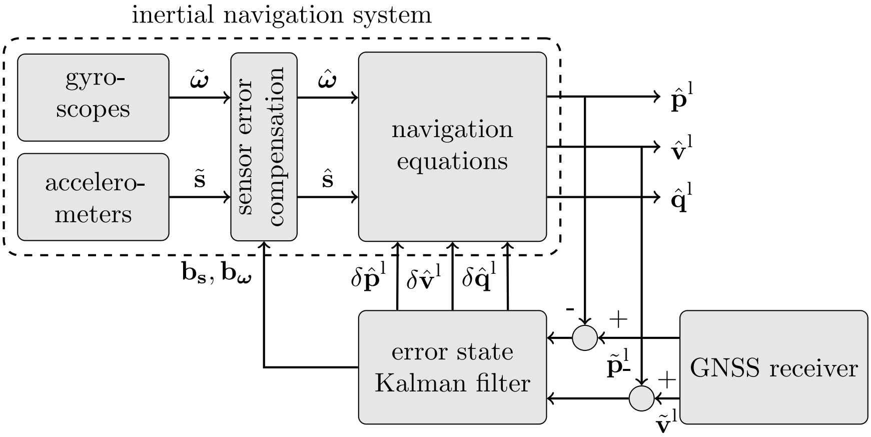

inertial navigation system
gyro-scopes
accelero-meters
$\tilde{\boldsymbol{\omega}}$
$\tilde{\mathbf{s}}$
sensor error compensation
$\hat{\boldsymbol{\omega}}$
$\hat{\mathbf{s}}$
navigation equations
$\hat{\mathbf{p}}^{\mathrm{l}}$
$\hat{\mathbf{v}}^{\mathrm{l}}$
$\hat{\mathbf{q}}^{\mathrm{l}}$
$\mathbf{b}_{\mathbf{s}}, \mathbf{b}_{\boldsymbol{\omega}}$
$\delta\hat{\mathbf{p}}^{\mathrm{l}}$
$\delta\hat{\mathbf{v}}^{\mathrm{l}}$
$\delta\hat{\mathbf{q}}^{\mathrm{l}}$
-
+
-
+
$\tilde{\mathbf{p}}^{\mathrm{l}}$
$\tilde{\mathbf{v}}^{\mathrm{l}}$
error state Kalman filter
GNSS receiver

Figure 6.2.: Loosely-coupled closed-loop GNSS-aided INS

Navigation equations

The navigation equations in the INS modeling the vehicle motion are represented in a tangent local reference frame L by using the North-East-Down (NED) coordinates. The navigation equations are given as (Farrell 2008) (omitting the time argument t for readability)

$$\dot{\mathbf{p}}^{\mathrm{l}} = \mathbf{v}^{\mathrm{l}}, \tag{6.1a}$$

$$\dot{\mathbf{v}}^{\mathrm{l}} = \mathbf{R}_{\mathrm{b}}^{\mathrm{l}}\mathbf{s}^{\mathrm{b}} + \mathbf{g}^{\mathrm{l}} - 2\boldsymbol{\Omega}_{\mathrm{ie}}^{\mathrm{l}}\mathbf{v}^{\mathrm{l}}, \tag{6.1b}$$

$$\dot{\mathbf{R}}_{\mathrm{b}}^{\mathrm{l}} = \mathbf{R}_{\mathrm{b}}^{\mathrm{l}}(\boldsymbol{\Omega}_{\mathrm{ib}}^{\mathrm{b}} - \boldsymbol{\Omega}_{\mathrm{ie}}^{\mathrm{b}}), \tag{6.1c}$$

where $\mathbf{p}^{\mathrm{l}}$ denotes the position vector expressed in frame l; $\mathbf{v}^{\mathrm{l}}$ denotes the velocity vector expressed in frame l; $\mathbf{R}_{\mathrm{b}}^{\mathrm{l}}$ represent the direction cosine matrix transforming vectors from the body frame b to the tangent local reference frame l (i.e., the attitude of the vehicle); the vector $\mathbf{g}^{\mathrm{l}}$ denotes the gravity in the l-frame which is the difference between gravitational acceleration and centrifugal acceleration due to earth rotation; $\mathbf{s}^{\mathrm{b}}$ denotes the specific force vector in the body frame b as measured by the accelerometers; and $\boldsymbol{\Omega}_{\mathrm{ib}}^{\mathrm{b}}$ is a skew-symmetric matrix of the angular velocity of the body frame b relative to the inertial frame i as resolved in the body frame b which is measured by the gyroscopes; and $\boldsymbol{\Omega}_{\mathrm{ie}}^{\mathrm{b}}$ is a skew-symmetric matrix of the angular velocity of the earth-fixed earth-centered frame e relative to the inertial frame i, which is the earth rotation as resolved in the body frame b, respectively. The quaternion representation $\mathbf{q}^{\mathrm{l}}$ is used to represent the rotational transformation $\mathbf{R}_{\mathrm{b}}^{\mathrm{l}}$ for its lack of singularities and computational efficiency.

Sensor error model

Every time step, the INS computes the a priori navigation solution in terms of position $\mathbf{p}(t)$, velocity $\mathbf{v}(t)$, and attitude $\mathbf{q}(t)$ by numerically integrating (6.1). The navigation solution would perfectly track position, velocity, and attitude in case the IMU signals would be without error and neglecting discretization and quantization errors. Disturbances are however inevitable in practice and the accelerometer and gyroscope sensor measurements include random noise terms and instrument calibration factors, such as slowly varying measurement bias, scale factors, nonlinearity, and non-orthogonality. The instrumenten calibration factors can be estimated using an ESKF by including them in the estimation problem. Since the complexity and number of augmented states increases with the number of included calibration factors, only the bias errors are considered in the following implementation. The accelerometer and gyroscope sensors measurements are modeled as

$$\tilde{\mathbf{s}}(t) = \mathbf{s}(t) + \mathbf{b}_{\mathbf{s}}(t) + \boldsymbol{\eta}_{\mathbf{s}}(t), \tag{6.2}$$

$$\tilde{\boldsymbol{\omega}}(t) = \boldsymbol{\omega}(t) + \mathbf{b}_{\boldsymbol{\omega}}(t) + \boldsymbol{\eta}_{\boldsymbol{\omega}}(t), \tag{6.3}$$

where $\tilde{\mathbf{s}}(t)$ and $\tilde{\boldsymbol{\omega}}(t)$ are the measured values, $\mathbf{s}(t)$ and $\boldsymbol{\omega}(t)$ the actual values, $\mathbf{b}_{\mathbf{s}}(t)$ and $\mathbf{b}_{\boldsymbol{\omega}}(t)$ the sensor biases, and $\boldsymbol{\eta}_{\mathbf{s}}(t)$ and $\boldsymbol{\eta}_{\boldsymbol{\omega}}(t)$ sensor noise. A random walk process (Farrell 2008) is used to model the sensor biases as

$$\dot{\mathbf{b}}_{\mathbf{s}}(t) = \boldsymbol{\eta}_{\mathbf{b}_{\mathrm{s}}}(t), \tag{6.4}$$

$$\dot{\mathbf{b}}_{\boldsymbol{\omega}}(t) = \boldsymbol{\eta}_{\mathbf{b}_{\boldsymbol{\omega}}}(t), \tag{6.5}$$

with $\boldsymbol{\eta}_{\mathbf{b}_{\mathrm{s}}}(t)$ and $\boldsymbol{\eta}_{\mathbf{b}_{\boldsymbol{\omega}}}(t)$ the bias noise. Both the sensor as well as bias noise are assumed to be zero-mean Gaussian white noise. With the biases estimated with the ESKF, the accelerometer and gyroscope measurements are corrected by applying

$$\hat{\mathbf{s}}(t) = \tilde{\mathbf{s}}(t) - \mathbf{b}_{\mathbf{s}}(t), \tag{6.6}$$

$$\hat{\boldsymbol{\omega}}(t) = \tilde{\boldsymbol{\omega}}(t) - \mathbf{b}_{\boldsymbol{\omega}}(t). \tag{6.7}$$

As only the biases are removed from the IMU signals, the filter introduces no phase lag.

Error-state Kalman filter

To account for the random noise terms and instrument calibration factors, such as slowly varying measurement bias, scale factors, nonlinearity, and non-orthogonality, a loosely-coupled closed-loop GNSS-aided INS scheme is employed that is based on a linearized Kalman filter. This filter estimates the correction data for the navigation solution $(\delta\hat{\mathbf{p}}, \delta\hat{\mathbf{v}}, \delta\hat{\mathbf{q}})$ and the correction data for the IMU $(\mathbf{b}_{\mathbf{s}}, \mathbf{b}_{\boldsymbol{\omega}})$ using the position and velocity differences between the INS and GNSS as input.

In contrast to an ordinary extended Kalman filter, only the estimation errors are determined in the filter while the actual state vector is stored outside the filter. The state vector prediction is conducted in the INS mechanization and excluded from the filter. Therefore, the filtering scheme is commonly denoted as an ESKF. The error-state always operates close to the origin as the error-state $\delta\mathbf{x}(t)$ is small compared to the real state $\mathbf{x}(t)$. The errors are collected in the error-state vector $\delta\mathbf{x}(t)$.

The dynamic error model structure of the ESKF in continuous time is given as

$$\delta\dot{\mathbf{x}}(t) = \mathbf{F}(t)\delta\mathbf{x}(t) + \mathbf{G}(t)\mathbf{w}(t), \tag{6.8}$$

where $\delta\mathbf{x}(t)$ is the error vector, $\mathbf{F}(t)$ the state transition matrix, $\mathbf{G}(t)$ the noise gain matrix, and $\mathbf{w}(t)$ the noise vector with noise covariance $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$. The error vector consists of the position error $\delta\mathbf{p}(t)$, velocity error $\delta\mathbf{v}(t)$, attitude error represented as euler angles $\delta\boldsymbol{\epsilon}(t)$ (Farrell 2008), accelerometer bias error $\delta\mathbf{b}_{\mathbf{s}}(t)$, and gyroscope bias error $\delta\mathbf{b}_{\boldsymbol{\omega}}(t)$. The dynamic error equations can be obtained by linearization of the Taylor series expansion of the actual solutions around the estimated solution and subtracting the estimated solution. For the considered system, the error-state kinematics are given as (Solà 2017) (omitting the time argument t for readability)

$$\delta\dot{\mathbf{p}}^{\mathrm{l}} = \delta\mathbf{v}^{\mathrm{l}}, \tag{6.9a}$$

$$\delta\dot{\mathbf{v}}^{\mathrm{l}} = [\mathbf{R}_{\mathrm{b}}^{\mathrm{l}}\hat{\mathbf{s}}]_{\times}\delta\mathbf{q}^{\mathrm{l}} - \mathbf{R}_{\mathrm{b}}^{\mathrm{l}}\delta\mathbf{b}_{\mathbf{s}}, \tag{6.9b}$$

$$\delta\dot{\boldsymbol{\epsilon}}^{\mathrm{l}} = \mathbf{R}_{\mathrm{b}}^{\mathrm{l}}\delta\mathbf{b}_{\boldsymbol{\omega}}, \tag{6.9c}$$

$$\delta\dot{\mathbf{b}}_{\mathbf{s}} = \boldsymbol{\eta}_{\mathbf{b}_{\mathrm{s}}}, \tag{6.9d}$$

$$\delta\dot{\mathbf{b}}_{\boldsymbol{\omega}} = \boldsymbol{\eta}_{\mathbf{b}_{\boldsymbol{\omega}}}, \tag{6.9e}$$

where $[\cdot]_{\times}$ denotes a skew-symmetric matrix of the vector $[\cdot]$. The equation can be written in the form of (6.8) as

$$\begin{bmatrix} \delta\dot{\mathbf{p}}^{\mathrm{l}} \\ \delta\dot{\mathbf{v}}^{\mathrm{l}} \\ \delta\dot{\boldsymbol{\epsilon}}^{\mathrm{l}} \\ \delta\dot{\mathbf{b}}_{\mathbf{s}} \\ \delta\dot{\mathbf{b}}_{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & [\mathbf{R}_{\mathrm{b}}^{\mathrm{l}}\mathbf{s}]_{\times} & -\mathbf{R}_{\mathrm{b}}^{\mathrm{l}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{R}_{\mathrm{b}}^{\mathrm{l}} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{p}^{\mathrm{l}} \\ \delta\mathbf{v}^{\mathrm{l}} \\ \delta\boldsymbol{\epsilon}^{\mathrm{l}} \\ \delta\mathbf{b}_{\mathbf{s}} \\ \delta\mathbf{b}_{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{R}_{\mathrm{b}}^{\mathrm{l}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{\mathrm{b}}^{\mathrm{l}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_{\mathbf{s}} \\ \boldsymbol{\eta}_{\boldsymbol{\omega}} \\ \boldsymbol{\eta}_{\mathbf{b}_{\mathrm{s}}} \\ \boldsymbol{\eta}_{\mathbf{b}_{\boldsymbol{\omega}}} \end{bmatrix}. \tag{6.10}$$

The measurement update of the considered system consists of the position $\tilde{\mathbf{p}}^{\mathrm{l}}$ and velocity $\tilde{\mathbf{v}}^{\mathrm{l}}$ expressed in the l frame. These measurements can be modeled as

$$\tilde{\mathbf{p}}^{\mathrm{l}}(t) = \mathbf{p}^{\mathrm{l}}(t) + \boldsymbol{\eta}_{\mathbf{p}}(t), \tag{6.11a}$$

$$\tilde{\mathbf{v}}^{\mathrm{l}}(t) = \mathbf{v}^{\mathrm{l}}(t) + \boldsymbol{\eta}_{\mathbf{v}}(t), \tag{6.11b}$$

where $\mathbf{p}^{\mathrm{l}}(t)$ and $\mathbf{v}^{\mathrm{l}}(t)$ are the true values and $\boldsymbol{\eta}_{\mathbf{p}}(t)$ and $\boldsymbol{\eta}_{\mathbf{v}}(t)$ are additive zero mean white noises. The measurements (6.11) have to be converted into the following measurement model structure

$$\delta\mathbf{y}(t) = \mathbf{H}(t)\delta\mathbf{y}(t) + \mathbf{v}(t), \tag{6.12}$$

with $\delta\mathbf{y}(t)$ the observation vector, $\mathbf{H}(t)$ the observation matrix, and $\mathbf{v}(t)$ the measurement noise vector with noise covariance $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$. The measurement model of (6.11) is given as

$$\begin{bmatrix} \delta\mathbf{p}^{\mathrm{l}} \\ \delta\mathbf{v}^{\mathrm{l}} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{p}^{\mathrm{l}} \\ \delta\mathbf{v}^{\mathrm{l}} \\ \delta\boldsymbol{\epsilon}^{\mathrm{l}} \\ \delta\mathbf{b}_{\mathbf{s}} \\ \delta\mathbf{b}_{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\eta}_{\mathbf{p}} \\ \boldsymbol{\eta}_{\mathbf{v}} \end{bmatrix}, \tag{6.13}$$

in which the contributions of the distance from GNSS antenna to IMU mounting position is neglected as they are roughly mounted at the same position.

To be able to implement the ESKF, the models (6.10) and (6.13) have to be converted to the discrete time domain, yielding

$$\delta\mathbf{x}_{k+1} = \mathbf{F}_k\delta\mathbf{x}_k + \mathbf{G}_{\mathrm{k}}\mathbf{w}_k, \tag{6.14}$$

$$\delta\mathbf{y}_k = \mathbf{H}_k\delta\mathbf{x}_k + \mathbf{v}_k, \tag{6.15}$$

where $\mathbf{F}_k$ is the discretized version of $\mathbf{F}(t)$, expressed as $\mathbf{F}_k = \exp(F\Delta t)$, $\mathbf{G}_{\mathrm{k}}$ is the discretized version of $\mathbf{G}(t)$, expressed as $\mathbf{G}_k = \mathbf{G}\Delta t$, and $\mathbf{H}_k$ is the discretized version of $\mathbf{H}(t)$. As the full state vector $\hat{\mathbf{x}}_k$ is stored outside the filter, the time update step of the ESKF limits to

$$\bar{\mathbf{P}}_k = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^{\mathsf{T}} + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^{\mathsf{T}},$$

where $\bar{\mathbf{P}}_k$ is the estimation error covariance matrix and $\mathbf{Q}_{k-1}$ the discrete time covariance matrix given by $\mathbf{Q}_{k-1} = \mathbf{Q}/\Delta t$ with Δt the sample time. The correction step

$$\begin{aligned} \mathbf{K}_k &= \bar{\mathbf{P}}_k\mathbf{H}_k^{\mathsf{T}}(\mathbf{H}_k\bar{\mathbf{P}}_k\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k)^{-1}, \\ \delta\hat{\mathbf{x}}_k &= \mathbf{K}_k(\tilde{\mathbf{y}}_k - \mathbf{H}_k\hat{\mathbf{x}}_k), \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\bar{\mathbf{P}}_k, \end{aligned}$$

is performed when a new GNSS measurement is received where $\mathbf{R}_k = \mathbf{R}$ denotes the measurement noise covariance matrix. After the correction step, the estimated navigation state perturbations $(\delta\mathbf{p}_k^{\mathrm{l}}, \delta\mathbf{v}_k^{\mathrm{l}}, \delta\boldsymbol{\epsilon}_k^{\mathrm{l}})$ and sensor biases $(\delta\mathbf{b}_{\mathbf{s}k}, \delta\mathbf{b}_{\boldsymbol{\omega}k})$ are injected in the INS (6.1) and (6.6) by

$$\hat{\mathbf{p}}_k^{\mathrm{l}} = \hat{\mathbf{p}}_{k-1}^{\mathrm{l}} + \delta\mathbf{p}_k^{\mathrm{l}}, \tag{6.16a}$$

$$\hat{\mathbf{v}}_k^{\mathrm{l}} = \hat{\mathbf{v}}_{k-1}^{\mathrm{l}} + \delta\mathbf{v}_k^{\mathrm{l}}, \tag{6.16b}$$

$$\hat{\mathbf{R}}_{\mathrm{b}}^{\mathrm{l}} = (\mathbf{I} - [\boldsymbol{\epsilon}]_\times)\mathbf{R}_{\mathrm{b}}^{\mathrm{l}} \tag{6.16c}$$

$$\mathbf{b}_{\mathbf{s}k} = \mathbf{b}_{\mathbf{s}k-1} + \delta\mathbf{b}_{\mathbf{s}k}, \tag{6.16d}$$

$$\mathbf{b}_{\boldsymbol{\omega}k} = \mathbf{b}_{\boldsymbol{\omega}k-1} + \delta\mathbf{b}_{\boldsymbol{\omega}k}. \tag{6.16e}$$

where $[\cdot]_\times$ denotes a skew-symmetric matrix of the vector $[\cdot]$. The error state $\delta\hat{\mathbf{x}}_k$ is reset to zero after the correction, because per definition the error is zero after applying (6.16).

Time delayed GNSS measurement

GNSS measurements are commonly affected by time delays due to internal processing and data transmission. These time delays can be significant and time varying. To be able to compensate for them, the time delays must be measured. To measure the exact time delay, the real-time computer must be synchronized to the same clock as the GNSS receiver such that the time delay can be computed by comparing the time of reception with the time indicated in the received message.

The real-time computer is synchronized to the GNSS receiver clock by using the time indicated in the received message and a pulse per second signal. This precise digital time signal is transmitted every second by the GNSS receiver and indicates the exact top of a second. By triggering on this clock pulse, the local clock on the real-time computer is resynchronized to the absolute clock each time a pulse is received.

Figure 6.3 depicts the GNSS time delay observed in an experimental evaluation. As can be seen, the GNSS time delay is significant with a mean time delay of 0.18 s. Considering a speed of 50 km/h and the mean time delay of 0.18 s, a mismatch of 2.5 m is obtained between the measured and estimated position.

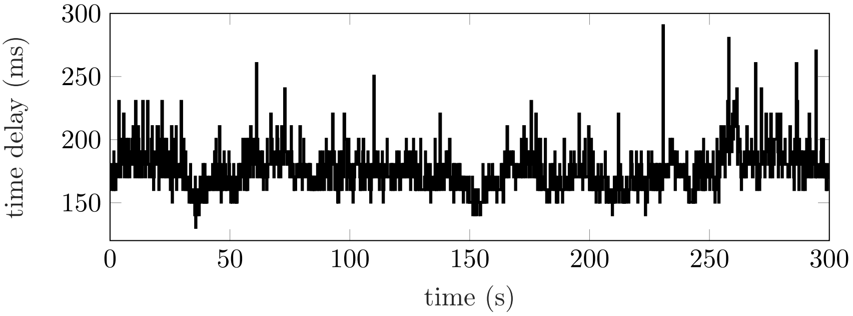


Figure 6.3.: GNSS receiver time delay during an experimental evaluation

To fully account for the time delayed measurement, reprocessing of the ESKF is required at the time when the delayed measurement is available but this is not practicable in a real-time environment. Therefore, a suboptimal solution has been implemented by storing the INS output $\hat{\mathbf{x}}_k$ for a certain amount of samples. When at time k a GNSS measurement $\tilde{\mathbf{y}}_k = [\tilde{\mathbf{p}}_k, \tilde{\mathbf{v}}_k]$ is received that was valid at time $k - \theta$ with θ the GNSS time delay in samples, the position and velocity difference is computed using $\tilde{\mathbf{y}}_k - \mathbf{h}(\hat{\mathbf{x}}_{k-\theta})$. The change in covariance matrix $\mathbf{P}$ due to the time delayed measurements is neglected.

Motion model

The navigation solution is improved by adding a motion model in the scheme (Dissanayake et al. 2001). During normal low speed driving with low lateral accelerations, a vehicle experiences almost no side slip or motion normal to the road surface. These non-holonomic vehicle constraints can be added in the scheme as virtual sensors. In addition, the vehicle speed sensor signal $v_{\mathrm{x}}(t)$ can be used as aiding sensor to improve the navigation solution between GNSS updates. Both sensors can be modeled as

$$\mathbf{y} = [v_{\mathrm{x}}, 0, 0]^{\mathsf{T}} - \mathbf{R}_{\mathrm{l}}^{\mathrm{b}} \mathbf{v}^{\mathrm{l}} + \boldsymbol{\eta}_{\mathbf{v}},$$

where $\boldsymbol{\eta}_{\mathbf{v}}$ is zero-mean Gaussian white noise used to relax the constraints.

Filter parameterization

The filter is parameterized by the variances in the process noise covariance matrix $\mathbf{Q} = \mathrm{diag}(\sigma_{\mathbf{s}}^2, \sigma_{\mathbf{b_s}}^2, \sigma_{\boldsymbol{\omega}}^2, \sigma_{\mathbf{b}_\omega}^2)$ and measurement noise covariance matrix $\mathbf{R} = \mathrm{diag}(\sigma_{\mathbf{p}}^2, \sigma_{\mathbf{v}}^2, \sigma_{v_x}^2, \sigma_{\mathbf{y}}^2)$. The selected filter parameter values are listed in Table 6.1 of which most of them are derived from sensor specifications (Bosch 2018a; u-blox 2016). Only the values for the relaxed constraint equations are determined by experimental evaluation.

Table 6.1.: Host-tracking filter noise standard deviations

Description	Symbol	Value	Source
Accelerometer noise	$\sigma_{\mathbf{s}}$	5e−2 m/s^2	IMU
Accelerometer bias noise	$\sigma_{\mathbf{b_s}}$	5e−4 m/s^2	IMU
Gyroscope noise	$\sigma_{\boldsymbol{\omega}}$	1e−1 °/s	IMU
Gyroscope bias noise	$\sigma_{\mathbf{b}_{\boldsymbol{\omega}}}$	5e−3 °/s	IMU
Horizontal position noise	$\sigma_{\mathbf{p}}$	2.5 m	GNSS
Horizontal velocity noise	$\sigma_{\mathbf{v}}$	5e−2 m/s	GNSS
Speedometer noise	σ_{v_x}	1 m/s	Experimental evaluation
Non-holonomic noise	$\sigma_{\mathbf{y}}$	2 m/s	Experimental evaluation

Host-tracking test results

Ground truth data can be used to evaluate the filter performance in terms of absolute navigation state errors. Ground truth data can for example be obtained with a high precision measurement equipment, such as RTK-GPS. As this is not readily available, the filter performance is evaluated by comparing the GNSS position data and the navigation solution of a field test with satellite map data. The results are depicted in Figure 6.4, and as can be seen, both the GNSS data and the INS agree reasonably well with the map data.

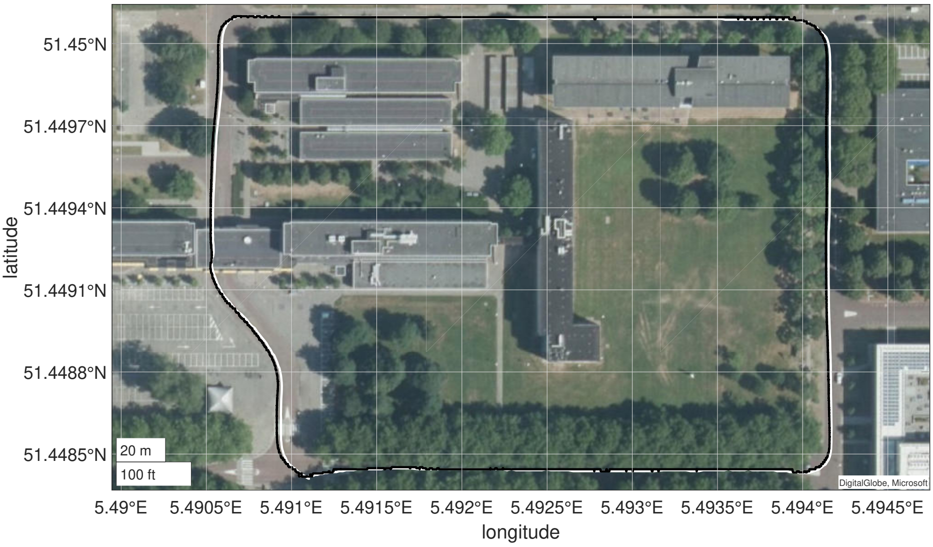


Figure 6.4.: Host-tracking field test result: GNSS measurements (white) and INS solution (black)

6.2.2. Course prediction

Course prediction plays an important role in determining the area in which the detected objects are relevant for the CACC system. The course of the host vehicle is predicted using the course-curvature that can be calculated by using a kinematic steering assumption (Bosch 2003) expressed as

$$\kappa_{\mathrm{s}} = \frac{\delta_{\mathrm{sw}}}{i_{\mathrm{sg}}\ell}, \tag{6.17}$$

where δ_{sw} is the steering wheel angle, i_{sg} is the steering-gear ratio, and ℓ the wheelbase. This method assumes kinematic driving and provides a good approximation for low vehicle speeds and low lateral accelerations. For higher velocities and accelerations, the course-curvature can be calculated by $\kappa_{\mathrm{y}} = \omega_{\mathrm{z}}/v_{\mathrm{x}}$ or $\kappa_{\mathrm{a}} = a_{\mathrm{y}}/v_{\mathrm{x}}^2$, where ω_{z} is the yawrate, v_{x} longitudinal velocity, and a_{y} the lateral acceleration. As the vehicle is designed for low speed driving (up to 50 km/h), κ_{s} is selected to calculate the course-curvature. The predicted course is then given as

$$y_{\mathrm{course}}(d) = \frac{\kappa_{\mathrm{s}}}{2} d^2, \tag{6.18}$$

where d is the longitudinal distance from the vehicle to the predicted course and y_{course} is the lateral distance from the vehicle centerline to the predicted course. The predicted course area, shown in Figure 6.1, is calculated using the predicted course and by assuming a lane width that is expanding over distance to account for the uncertainty.

6.2.3. Wireless communication

To communicate with other vehicles in the vicinity, a wireless link is employed. This link is used to receive the required kinematic states from a target vehicle as well as other relevant information such as time of message and vehicle dimensions. The Cooperative Awareness Message (CAM) message protocol (Severinson 2018) is used for this. The most important content of this message is listed in Table 6.2. The CAM requirements specify an update rate that depends on the dynamics of the vehicle and channel congestion varies between 1 Hz to 10 Hz (ETSI 2011). However, as no other vehicles are using the communication channel during the experiments, the update rate of the CAM is set to 25 Hz.

Table 6.2.: CAM message content (excerpt)

Description	Symbol	Purpose
Time of message generation (ms)	t_{cam}	Determine message age and communication delay θ
Rear-axle position (°)	G_{lat}	Bearing angle α calculation; distance d calculation
Rear-axle position (°)	G_{lon}	Bearing angle α calculation; distance d calculation
Heading (rad)	ψ_{t}	Distance d calculation
Longitudinal velocity (m/s)	v_{t}	Relative velocity calculation in host vehicle frame
Longitudinal acceleration (m/s^2)	a_{t}	Relative acceleration calculation in host vehicle frame
Desired acceleration (m/s^2)	u_{t}	Longitudinal vehicle following control

The wireless communication delay is considered an important factor in the CACC system (Xing 2019). As the time of transmission is included in the CAM message, the communication delay is computed online by comparing the time of transmission with the time of reception. The only prerequisite is that both the host and target vehicle are synchronized to the same clock. This is achieved by using the local clock of the host-tracking block, which is synchronized to the absolute time via the GNSS receiver.

Figure 6.5 shows the communication delay measured during a driving experiment. As can be seen, the communication delay is nearly constant over the experiment and relatively small with a mean of 0.02 s. The jitter effect is most likely caused by the fact that both the pulse per second required for the absolute clock resynchronization and the communication delay are sampled signals ($T_\mathrm{s} = 0.01\,\mathrm{s}$) instead of flank triggered. The communication delay is expected to grow when the number of vehicles communicating on the same network increases.

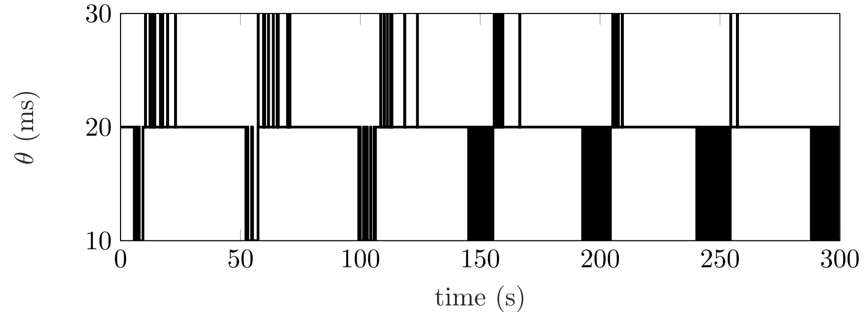


Figure 6.5.: Wireless communication delay during a driving experiment

6.2.4. Target selection

Objects in the vicinity of the host vehicle are detected using an environmental perception system and depending on the object also via Vehicle-to-Everything (V2X) communication. As environmental perception system, a radar sensor is employed that can detect and track up to 32 objects. The most relevant one must be selected out of the potentially large set of detected objects. To illustrate this, Figure 6.6 shows a bird's-eye plot of an experiment in which the radar sensor coverage, V2X detections, and tracked objects are displayed. In total ten objects are detected which are indicated by the black boxes. The line segments indicate the relative velocity of the objects towards the host vehicle. From the set of potentially relevant tracked objects S, the Most Important Object (MIO) must be selected. The MIO is the vehicle driving closest in front of the vehicle on the same predicted path.

The first step is to filter out the static objects when driving. The absolute velocity of objects is calculated by

$$v_j(t) = v_{\mathrm{h}}(t) + \Delta v_j(t), \tag{6.19}$$

where $v_j(t)$ is the absolute velocity of object $j \in S$ with S the set of detected objects, $v_{\mathrm{h}}(t)$ is the velocity of the host vehicle, and $\Delta v_j(t)$ is the relative velocity as measured by the radar sensor. Objects with an absolute velocity less than 2 m/s are considered stationary for this research and are ignored as possible target vehicle when driving ($v_{\mathrm{h}} > 2\,\mathrm{m/s}$).

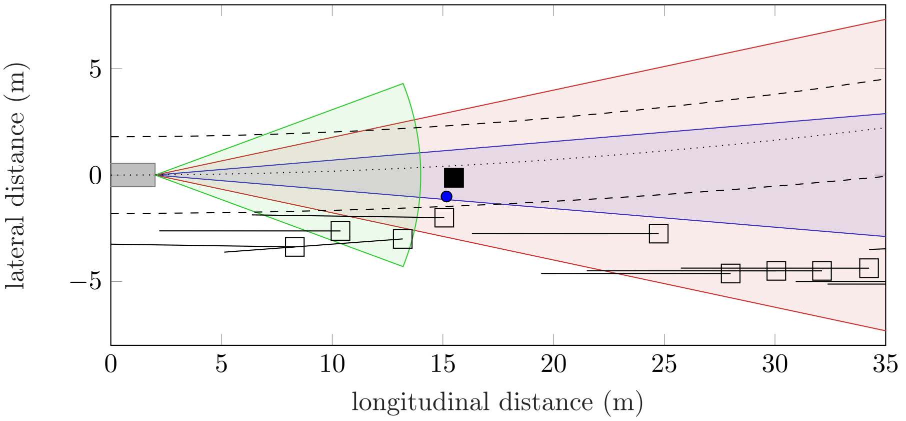


Figure 6.6.: Radar sensor coverage, V2X detections, and tracks visualization: host vehicle outline (▨), main antenna (🟦), elevation antenna (🟥), close elevation antenna (🟩), V2X relative position (●), tracked object relative position (□), tracked object relative velocity (—), MIO (■), predicted course (⋯), and predicted course area (- - -)

Traditional ACC systems are primarily designed for use on motorways with large corner radii. The objective of the research and development vehicle designed in this thesis is urban driving, which includes sharp bends. As a consequence, situations are possible in which the target is out of the field-of-view of the radar sensor. Therefore, the bearing angle $\alpha(t)$, computed from the wireless communication message, is used to check whether the target vehicle is in the field-of-view of the radar sensor. When the target vehicle is not in the field-of-view of the radar sensor, the wireless data is used to determine the relative distance, velocity, and acceleration. Otherwise the radar data is used.

The radar and wireless data can be fused when the target is in the field-of-view of the radar sensor. However, the radar sensors provides object level data, which means that the radar detections are processed inside the radar. As a consequence, the radar output data proved to be sufficiently accurate in practice. To illustrate this, the wireless and radar sensor data of a CACC experiment are compared. Figure 6.7 shows the path driven during the experiment of the target and host vehicle. To be able to compare the sensor data, the sensor data must be converted to a common frame of reference, as illustrated in Figure 6.1. The radar data is converted from the radar coordinate frame to the vehicle coordinate frame attached to the rear axle to be in agreement with the wireless received data. Figure 6.8 shows a comparison between the wireless sensor data and radar sensor data in which $d_{\mathrm{x}}(t)$ is the relative longitudinal distance, $d_{\mathrm{y}}(t)$ the relative lateral distance, $\alpha(t)$ the bearing angle between the vehicles, $\Delta v(t)$ the relative longitudinal velocity, and $\Delta a(t)$ the relative longitudinal acceleration. As can be seen, a good agreement is obtained between the radar sensor data and the data received via the wireless link. Also the missing line segments in the radar data, when no target is detected with the radar sensor, align well with predicted out field-of-view based on the wireless received data.

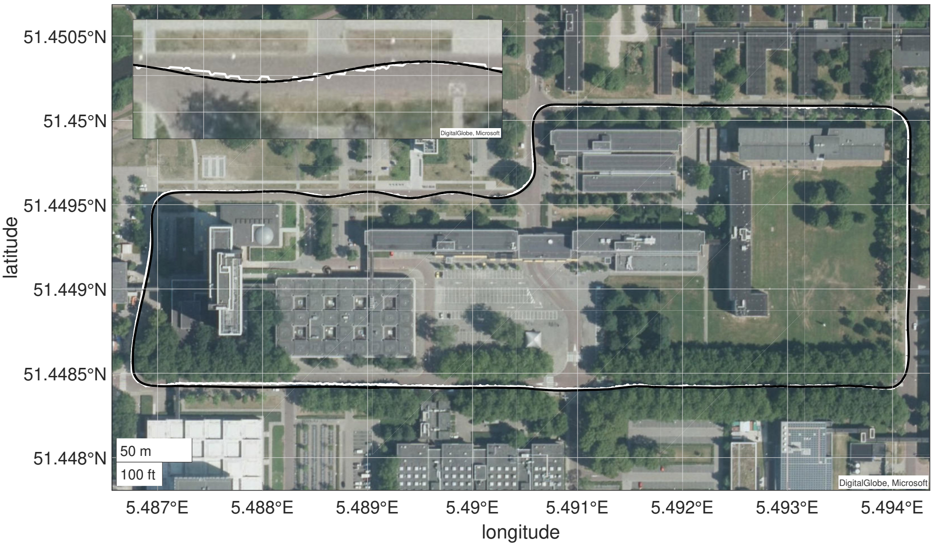


Figure 6.7.: Driven path during CACC experiment: target vehicle (white) and host vehicle (black). The inset shows a close up of the s-curved path

The current implementation uses the sensors that are normally available for CACC systems, such as radar and wireless communication. The radar sensor is able to accurate measure longitudinal position, velocity, and acceleration but has limited field-of-view and angular resolution. To complement the radar sensor

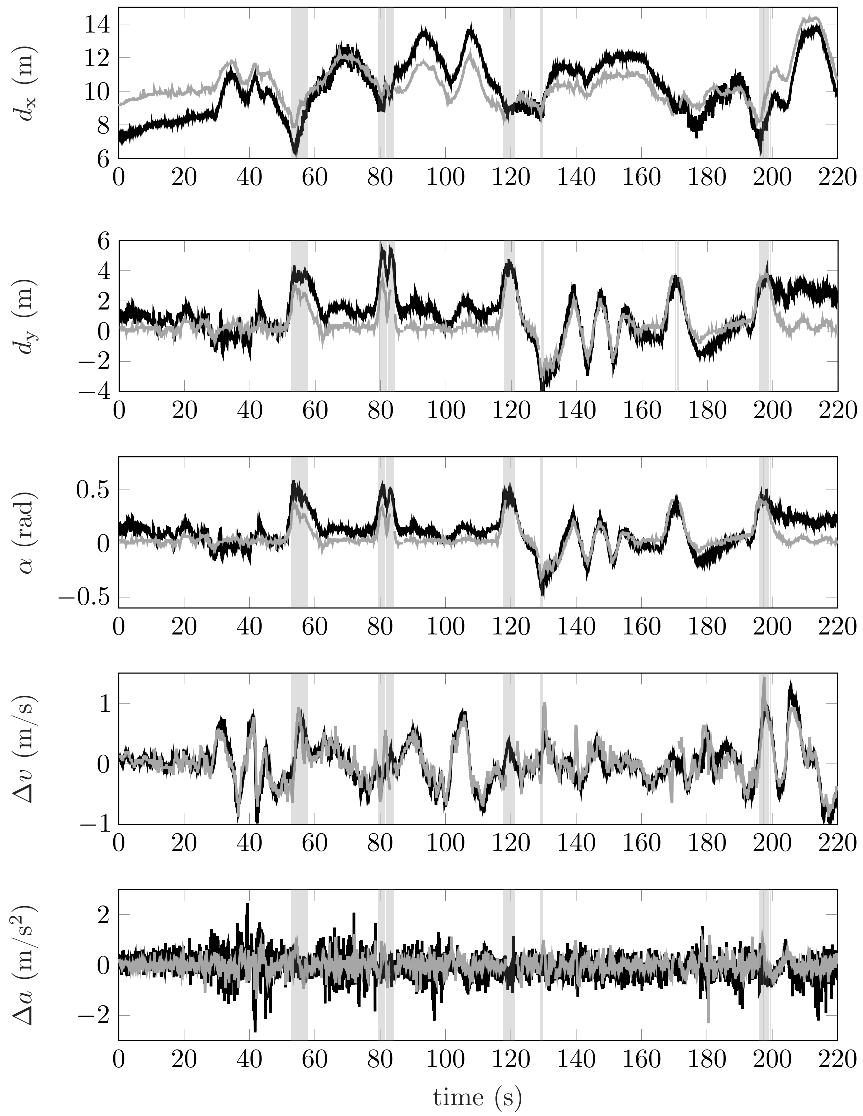


Figure 6.8.: Environmental perception sensor signals during CACC experiment: V2X (——), radar (——), and the with the V2X data predicted periods when the target vehicle is out the field-of-view of the radar sensor (▭). The missing line segments in the radar data indicate that no target vehicle is detected

and to improve the angular resolution and field-of-view, additional environmental perception sensors can be included, such as camera or lidar. As vision system have good angular resolution but limited position and velocity accuracy, fusion of these systems would result in a coherent image of the surroundings.

6.2.5. Longitudinal vehicle dynamics model

To be able to apply linear CACC controllers, input-output linearization of the longitudinal driveline dynamics is required. The longitudinal dynamics of the vehicle are linearized using an inverted map of the torquemap as presented in Chapter 4. In addition, the road load forces are compensated, such as the rolling resistance and aerodynamic losses. The following vehicle model is adopted to describe the longitudinal driveline dynamics of the vehicle (Ploeg 2014)

$$G(s) = \frac{a_{\mathrm{x}}(s)}{u(s)} = \frac{1}{\tau s + 1} e^{-\phi s}, \tag{6.20}$$

where τ is the vehicle driveline time constant and ϕ the time delay in the driveline. The driveline of the automated Renault Twizy including longitudinal acceleration controller is identified using step responses, as can be seen in Figure 6.9. The estimated linear vehicle model is only valid in a small acceleration range as the available driveline torque and power are limited.

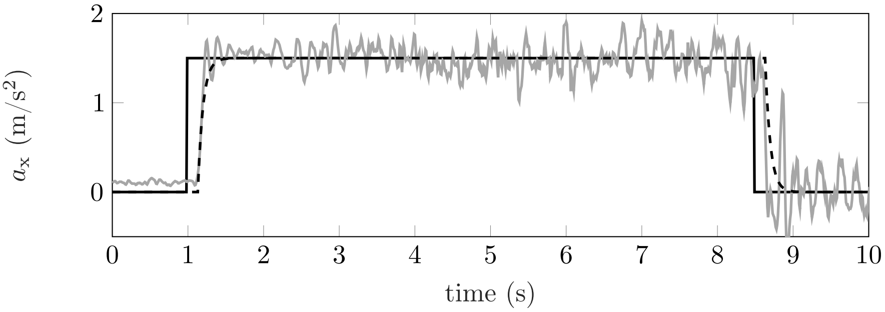


Figure 6.9.: Step response of the research and development vehicle with the inverted torquemap and road load compensation: reference acceleration (——), measured acceleration (——), and model simulated acceleration (- - -)

6.2.6. CACC controller

While the fault diagnosis method does not assume a cooperative control strategy and works for various types of cooperative controllers, a longitudinal and lateral cooperative control strategy must be adopted to be able to perform simulations and experiments. As longitudinal cooperative controller, the following CACC controller has been adopted (Ploeg 2014):

$$\dot{u}_{\mathrm{h}}(t) = -\frac{1}{h}u_{\mathrm{h}}(t) + \frac{1}{h}(k_{\mathrm{p}}e(t) + k_{\mathrm{d}}\dot{e}(t)) + \frac{1}{h}u_{\mathrm{ff}}(t), \tag{6.21}$$

where $e(t) = d(t) - (r + hv_{\mathrm{h}}(t))$ and $\dot{e}(t) = \Delta v(t) - ha_{\mathrm{h}}(t)$, with r the standstill distance, h the headway time, $v_{\mathrm{h}}(t)$ the longitudinal velocity of the host vehicle, $a_{\mathrm{h}}(t)$ the longitudinal acceleration of the host vehicle, $d(t)$ the relative distance, and $\Delta v(t)$ the relative velocity between the vehicles.

The feedforward term $u_{\mathrm{ff}}(t)$ depends on the control mode. During normal fault-free CACC operation, the communicated desired acceleration of the target vehicle $u_{\mathrm{t}}(t)$ is used similar to Ploeg (2014). In case of a wireless communication fault, the controller switches to normal ACC operation where the relative acceleration $\Delta a(t)$ of the radar sensor plus the host acceleration $a_{\mathrm{h}}(t)$ is used to estimate the longitudinal acceleration of the target vehicle $a_{\mathrm{t}}(t)$. The system is switched-off in more severe fault cases, such as a radar sensor fail.

As lateral cooperative controller, a pure pursuit controller is implemented (Campbell 2007). This control strategy is based on geometric path following and works well for low speeds and accelerations. As the vehicle in this thesis is designed for urban driving, the pure pursuit control strategy suits the objective. Figure 6.10 illustrates the pure pursuit geometry in which $R(t)$ denotes the radius of a circle that the rear axle is turning at, $d(t)$ is the distance from the rear axle to the target point, and $\alpha(t)$ the bearing angle to the target point. The required path curvature $\kappa(t)$ to reach the target point can be calculated by applying the law of sines

$$\begin{aligned}
\frac{R(t)}{\sin(\frac{\pi}{2} - \alpha(t))} &= \frac{d(t)}{\sin 2\alpha(t)},\\
\frac{R(t)}{\cos\alpha(t)} &= \frac{d(t)}{2\sin\alpha(t)\cos\alpha(t)},\\
R(t) &= \frac{d(t)}{2\sin\alpha(t)},\\
\kappa(t) &= \frac{2\sin\alpha(t)}{d(t)}.
\end{aligned}$$

Using the path curvature $\kappa(t)$ and a kinematic bicycle model, the required steering angle $\delta(t)$ is given as:

$$\delta(t) = \arctan(\ell\kappa(t)), \tag{6.22}$$

$$= \arctan\left(\frac{2\ell\sin\alpha(t)}{d(t)}\right), \tag{6.23}$$

where $\delta(t)$ is the steering angle of the front axle and ℓ is the wheelbase. The distance to the target point $d(t)$ is controlled by the longitudinal controller and increases with forward velocity, thereby decreasing the gain of the lateral controller to avoid aggressive maneuvers at high speed.

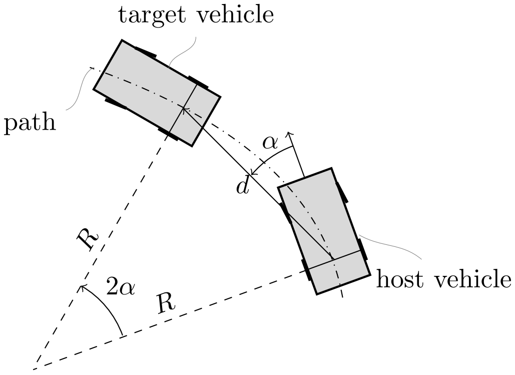


Figure 6.10.: Pure pursuit coordinate system

To verify the CACC control performance, data from the experiment shown in Figure 6.7 is used. The results are shown in Figure 6.11 in which the desired and measured inter-vehicle distance, target and host velocity, and target and host acceleration are shown. As can be seen, the host-vehicle mimics the motion of the target vehicle well as the velocity and acceleration profiles are nearly identic. The measured inter-vehicle distance follows the desired inter-vehicle distance reasonably well as the distance error remains within roughly 2 m during the experiment.

6.3. CACC Fault Diagnosis

With the CACC implementation described and validated for the developed vehicle, the next step is to apply model-based fault diagnosis to detect when an incorrect target is tracked with the environmental perception system or when incorrect information is received from the preceding vehicle. Complete wireless dropouts or environmental perception system failures are not of interest here as they easily detected when no new messages are received for a certain time period. This is discussed in Chapter 3.

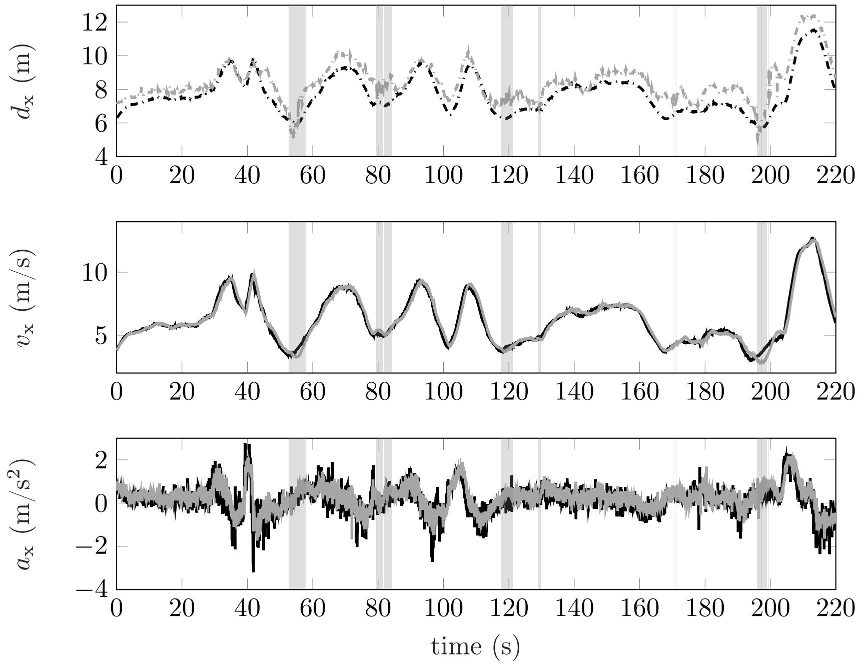


Figure 6.11.: Vehicle following control signals during CACC experiment: desired inter-vehicle distance $d_r = r + hv_h$ (-·-·-), inter-vehicle distance d (-·-·-), target vehicle (——), and host vehicle (——). The shaded areas (▭) indicate when the target vehicle was out of the field-of-view of the radar sensor and only wireless data was used in the CACC controller

6.3.1. System modeling

To apply model-based fault diagnosis, a model of the CACC system is required. To this end, consider a two vehicle platoon as depicted in Figure 6.12, where $v_\mathrm{t}(t)$, $a_\mathrm{t}(t)$, $u_\mathrm{t}(t)$, τ_t, and θ_t are the wireless communicated velocity, acceleration, external input, driveline time constant, and driveline time delay, respectively. The sensor signals consist of the relative velocity $\Delta v(t) = v_\mathrm{t}(t) - v_\mathrm{h}(t)$ and relative acceleration $\Delta a(t) = a_\mathrm{t}(t) - a_\mathrm{h}(t)$. These signals are directly measured by the radar sensor when the target is in the field-of-view of the radar sensor. Otherwise they are computed from the data received via the wireless link, as explained in Section 6.2. Using the vehicle model (6.20), the platoon model can be written as

$$\begin{bmatrix} \dot{v}_\mathrm{t}(t) \\ \dot{a}_\mathrm{t}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{\tau_\mathrm{t}} \end{bmatrix} \begin{bmatrix} v_\mathrm{t}(t) \\ a_\mathrm{t}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\tau_\mathrm{t}} \end{bmatrix} \begin{bmatrix} v_\mathrm{h}(t) \\ a_\mathrm{h}(t) \\ u_\mathrm{t}(t-\phi) \end{bmatrix}, \tag{6.24a}$$

$$\begin{bmatrix} \Delta v(t) \\ \Delta a(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_\mathrm{t}(t) \\ a_\mathrm{t}(t) \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} v_\mathrm{h}(t) \\ a_\mathrm{h}(t) \\ u_\mathrm{t}(t-\phi) \end{bmatrix}, \tag{6.24b}$$

where $v_\mathrm{h}(t)$ and $a_\mathrm{h}(t)$ are the velocity and acceleration of the host vehicle, respectively.

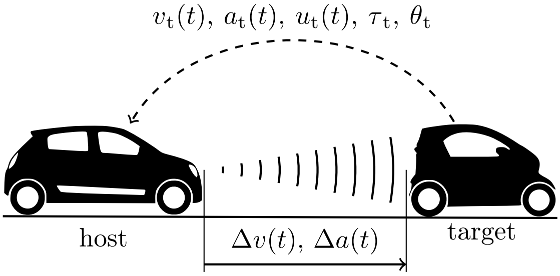

$v_\mathrm{t}(t)$, $a_\mathrm{t}(t)$, $u_\mathrm{t}(t)$, τ_t, θ_t
host
$\Delta v(t)$, $\Delta a(t)$
target

Figure 6.12.: (Hetereogenous) cooperative equipped platoon

6.3.2. Fault modeling

The objective of the CACC fault diagnosis is to detect when incorrect information is received from a preceding vehicle, for example, due to an incorrect target selection. In addition, a fault in the relative acceleration measurement is of interest. Faults are less likely to occur with the relative velocity measurement as the signal is directly measured by the radar sensor and/or with the vehicle speed sensor, while the relative acceleration is a filtered signal coming from either the radar sensor or the difference between the acceleration signals coming from the host tracking blocks.

To be able to detect these faults, they must be modeled. Since most faults can be modeled as additive faults as indicated by Chen and Patton (1999), the external input fault is modeled as

$$u_{\mathrm{t}}(t) = \tilde{u}_{\mathrm{t}}(t) + f_{u_{\mathrm{t}}}(t), \tag{6.25}$$

where $u_{\mathrm{t}}(t)$ is the wireless-communicated external input, $\tilde{u}_{\mathrm{t}}(t)$ is the real external input, and $f_{u_{\mathrm{t}}}(t)$ is an additive fault signal. Driveline saturation of the target vehicle is also considered an external input fault as the commanded vehicle response is different to the actual vehicle motion. Faults with the relative acceleration are modeled as

$$\Delta a(t) = \Delta\tilde{a}(t) + f_{\Delta a}(t), \tag{6.26}$$

where $\Delta a(t)$ is the relative acceleration obtained with the radar sensor or via the wireless link, $\Delta\tilde{a}(t)$ is the real relative acceleration, and $f_{\Delta a}(t)$ is an additive fault signal.

6.3.3. Model-based fault diagnosis

By substituting the fault models (6.25) and (6.26) into the platoon model (6.24), its input-output description can be written as

$$\mathbf{y}(s) = \mathbf{G}_{\mathrm{u}}(s)\mathbf{u}(s) + \mathbf{G}_{\mathrm{f}}(s)\mathbf{f}(s), \tag{6.27}$$

where $\mathbf{y}(s)$, $\mathbf{u}(s)$, and $\mathbf{f}(s)$ denote the Laplace transformed vectors of the outputs $\mathbf{y}(t) = \begin{bmatrix} \Delta v(t) & \Delta a(t) \end{bmatrix}^{\mathsf{T}}$, inputs $\mathbf{u}(t) = \begin{bmatrix} v_{\mathrm{h}}(t) & a_{\mathrm{h}}(t) & u_{\mathrm{t}}(t-\phi) \end{bmatrix}^{\mathsf{T}}$, and faults $\mathbf{f}(t) = \begin{bmatrix} f_{u_{\mathrm{t}}}(t) & f_{\Delta a}(t) \end{bmatrix}^{\mathsf{T}}$, respectively. To produce fault indicator signals, a linear residual generator is employed which is given by

$$\mathbf{r}(s) = \mathbf{Q}(s)\begin{bmatrix} \mathbf{y}(s) \\ \mathbf{u}(s) \end{bmatrix}, \tag{6.28}$$

where $\mathbf{Q}(s)$ is the to be designed filter such that the residual close to zero in the fault-free case and at least one of them distinctively nonzero when a fault is present. By substituting the input-output description (6.27), the following is obtained

$$\mathbf{r}(s) = \mathbf{Q}(s)\begin{bmatrix} \mathbf{G}_{\mathrm{u}}(s) \\ \mathbf{I} \end{bmatrix}\mathbf{u}(s) + \begin{bmatrix} \mathbf{G}_{\mathrm{f}}(s) \\ \mathbf{0} \end{bmatrix}\mathbf{f}(s). \tag{6.29}$$

To ensure the residuals $\mathbf{r}(t)$ are zero when no fault $\mathbf{f}(t)$ is present for any control input $\mathbf{u}(t)$ and disturbance $\mathbf{d}(t)$, the residuals must be decoupled from the control inputs and disturbance, as explained in Chapter 5. Since no disturbances are included in (6.27), a trivial fault detection filter is $\mathbf{Q}_1(s) = \begin{bmatrix} \mathbf{I} & -\mathbf{G}_\mathrm{u}(s) \end{bmatrix}$, which in essence compares the measured outputs with the predicted ones. The fault to residual response can be shaped by pre-multiplying $\mathbf{Q}_1(s)$ with suitable transfer functions $\mathbf{Q}_\mathrm{i}(s)$ to obtain the final filter in factored form as $\mathbf{Q}(s) = \mathbf{Q}_\mathrm{k}(s) \cdots \mathbf{Q}_2(s)\mathbf{Q}_1(s)$, where $\mathbf{Q}_2(s)$ to $\mathbf{Q}_\mathrm{k}(s)$ are the filters to shape the response or to address synthesis requirements, such as being physical realizable.

A filter $\mathbf{Q}(s)$ with three residuals is found after filter synthesis by employing the Fault Detection Toolbox (Varga 2017a) for the numerical computations. The first residual filter is given as

$$r_1(s) = \frac{1}{\rho_1 s + 1}\Delta a(s) + \frac{1}{\rho_1 s + 1}a_\mathrm{h}(s) - \frac{s}{\rho_1 s + 1}\Delta v(s) - \frac{s}{\rho s + 1}v_\mathrm{h}(s), \tag{6.30a}$$

$$= \frac{1}{\rho_1 s + 1}(\Delta a(s) + a_\mathrm{h}(s)) - \frac{s}{\rho_1 s + 1}(\Delta v(s) + v_\mathrm{h}(s)), \tag{6.30b}$$

$$= \frac{1}{\rho_1 s + 1}a_\mathrm{t}(s) - \frac{s}{\rho_1 s + 1}v_\mathrm{t}(s), \tag{6.30c}$$

with filter parameter ρ_1. The residual filter can be used to detect faults with the relative acceleration $f_{\Delta a}(t)$. The second residual filter is given as

$$r_2(s) = \frac{s}{\rho_2 s + 1}\Delta v(s) + \frac{s}{\rho_2 s + 1}v_\mathrm{h}(s) - \frac{1}{(\rho_2 s + 1)(\tau s + 1)}e^{-\phi s}u_\mathrm{t}(s), \tag{6.31a}$$

$$= \frac{s}{\rho_2 s + 1}(\Delta v(s) + v_\mathrm{h}(s)) - \frac{1}{\rho_2 s + 1}a_\mathrm{t}(s), \tag{6.31b}$$

$$= \frac{s}{\rho_2 s + 1}v_\mathrm{t}(s) - \frac{1}{\rho_2 s + 1}a_\mathrm{t}(s), \tag{6.31c}$$

with filter parameter ρ_2. The residual filter can be used to detect the external input faults $f_{u_\mathrm{t}}(t)$. The third and last residual filter detects both faults and can be used as an additional fault indicator signal. The third filter is given as

$$r_3(s) = \Delta a(s) + a_\mathrm{h}(s) - \frac{1}{\tau s + 1}e^{-\phi s}u_\mathrm{t}(s), \tag{6.32a}$$

$$= \Delta a(s) + a_\mathrm{h}(s) - a_\mathrm{t}(s). \tag{6.32b}$$

The last residual filter has no tunable parameters whereas the filter parameters ρ_1 and ρ_2 of the first two residual filters shape the fault to residual response.

These parameters must be selected with care, as a fast response makes it more susceptible to noise. The common approach is to tune the filter parameters based on simulation or measurement data.

To assess the obtained residual filters $r_i(s)$, the internal representation

$$\mathbf{R}_\mathrm{f} = \mathbf{Q}(s) \begin{bmatrix} \mathbf{G}_\mathrm{u}(s) \\ \mathbf{0} \end{bmatrix},$$

is used. Figure 6.13 presents the unit step response of $\mathbf{R}_\mathrm{f}$ to the faults $\mathbf{f}(t) = \begin{bmatrix} f_{u_\mathrm{t}}(t) & f_{\Delta a}(t) \end{bmatrix}^\mathsf{T}$ with all initial conditions set to zero and with the parameter values $\rho_1 = \pi/2$ and $\rho_2 = \pi/2$. The columns indicate the faults $f_{u_\mathrm{t}}(t)$ and $f_{\Delta a}(t)$ and the rows indicate the residuals $r_i(t)$ corresponding to the filters. As can be seen, both faults can be detected as for each fault at least one residual is nonzero. In addition, the fault can be isolated as well as each of the first two residuals $r_1(t)$ and $r_2(t)$ are only sensitive to a particular fault.

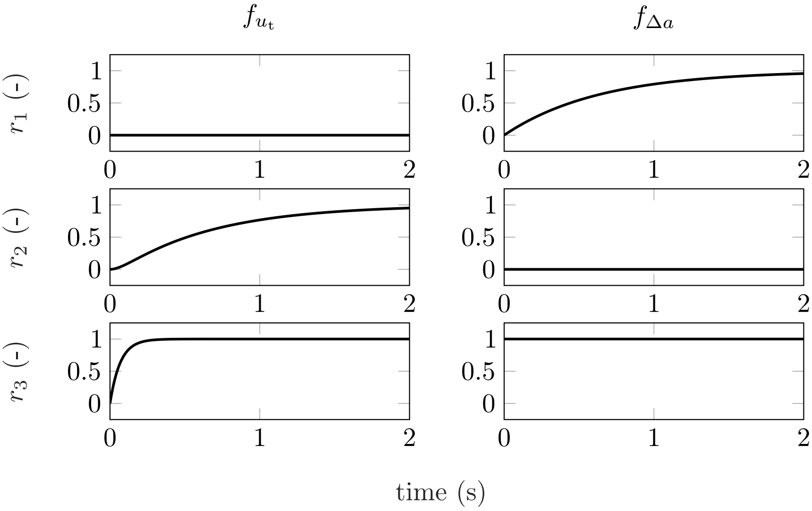


Figure 6.13.: Unit step response plot from faults $\mathbf{f}(t)$ to residuals $\mathbf{r}(t)$

6.4. Simulation Results

To evaluate the filter performance, a simulation model is constructed, consisting of a platoon to two vehicles. One target (lead) vehicle and a host (follower) vehicle. The CACC controller that is explained in Section 6.2 is implemented to control the inter-vehicle distance. The CACC controller gains have been obtained from Ploeg (2014) that guarantees string stable behavior for the identified vehicle parameters. Sensor noises are added to the model to simulate real-life driving conditions. The selected noise intensities and sample time are based on the real sensors (u-blox 2016; Bosch 2015) and are listed in Table 6.3, along with the vehicle model and CACC controller parameters.

Table 6.3.: Simulation model parameters

Description	Symbol	Value
Wireless communication delay	θ	$0.02\,\mathrm{s}$
Driveline time constant	τ	$0.0687\,\mathrm{s}$
Driveline time delay	ϕ	$0.15\,\mathrm{s}$
Maximum acceleration	$a_{\max}$	$2\,\mathrm{m/s^2}$
Minimum acceleration	$a_{\min}$	$-1\,\mathrm{m/s^2}$
Proportional CACC gain	k_p	0.2
Derivative CACC gain	k_d	0.7
Standstill distance	r	$4\,\mathrm{m}$
Headway time	h	$0.6\,\mathrm{s}$
Variance of measured relative distance	$\sigma^2_{\Delta d}$	$0.0144\,\mathrm{m^2}$
Variance of measured relative velocity	$\sigma^2_{\Delta v}$	$0.0121\,\mathrm{m^2/s^2}$
Variance of measured relative acceleration	$\sigma^2_{\Delta a}$	$0.0025\,\mathrm{m^2/s^4}$
Variance of measured host vehicle velocity	$\sigma^2_{v_\mathrm{h}}$	$0.0100\,\mathrm{m/s^2}$
Variance of measured host vehicle acceleration	$\sigma^2_{v_\mathrm{a}}$	$0.0100\,\mathrm{m^2/s^4}$
Noise sample time	Δt_n	$0.1\,\mathrm{s}$

During the simulation, the platoon drives in a straight line. The simulation starts with an initial velocity of $2.5\,\mathrm{m/s}$ and with a desired acceleration profile as depicted in Figure 6.14. Two faults are injected during the simulation. The reference and fault signals are selected such that the inter-vehicle distance is automatically increased when injecting the faults. This makes it safer to execute the experiments when the same reference and fault signals are applied. The first fault $f_{u_\mathrm{t}}(t) = 1\,\mathrm{m/s^2}$ is active during the interval $t = 4\,\mathrm{s}$ to $7\,\mathrm{s}$, and the second fault $f_{\Delta a}(t) = 1\,\mathrm{m/s^2}$ is active during the interval $t = 10\,\mathrm{s}$ to $13\,\mathrm{s}$. As can be seen in Figure 6.14, during the first fault interval $t = 4\,\mathrm{s}$ to $7\,\mathrm{s}$ an increase in relative acceleration is noticeable. The feedback part of the CACC controller compensates for the increased relative distance and velocity by increasing the host vehicle acceleration.

The external input and relative acceleration fault can be detected by feeding the output and input signal shown in Figure 6.14 through the filter $\mathbf{Q}(s)$ (6.30), (6.31), and (6.32) to obtain the residuals $\mathbf{r}(s)$. The residuals $\mathbf{r}(t)$ are then filtered

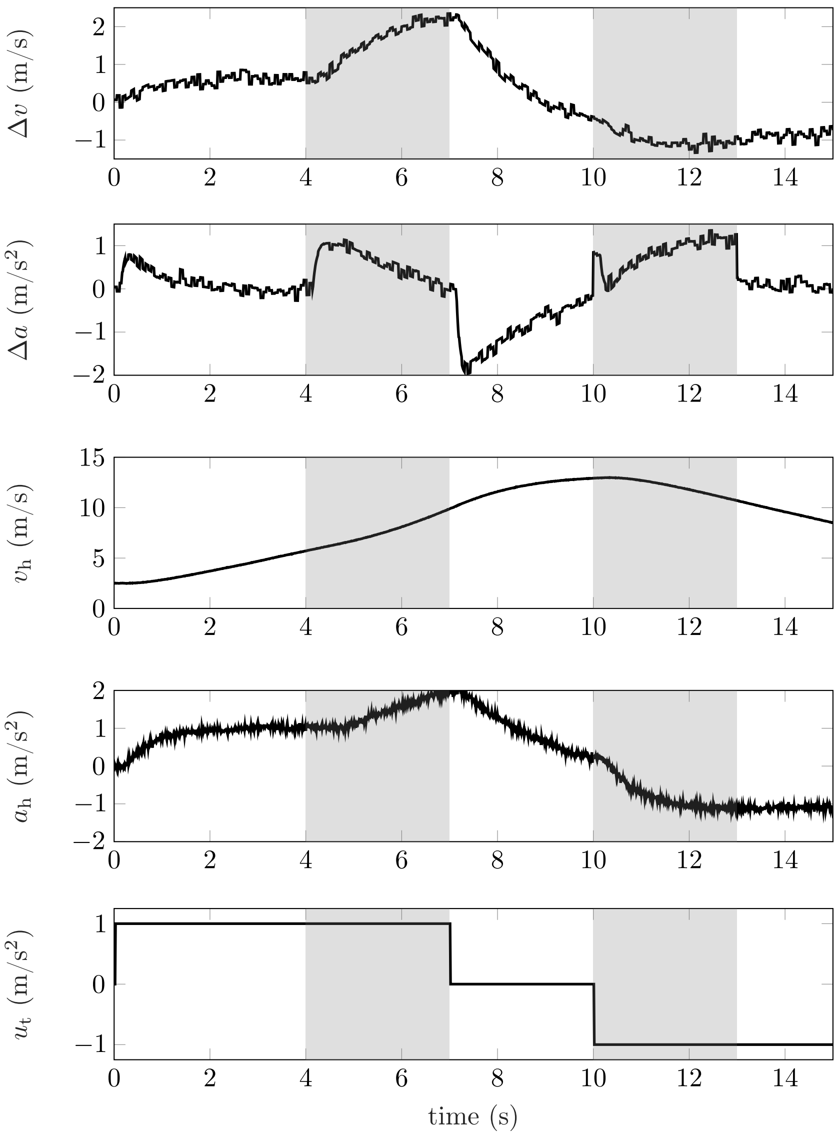


Figure 6.14.: Measured output $\mathbf{y}(t)$ and input $\mathbf{u}(t)$ signals during the simulation: relative velocity $\Delta v(t)$, relative acceleration $\Delta a(t)$, host vehicle velocity a_h, host vehicle acceleration a_h, and desired acceleration u_t. The shaded areas (▭) indicate when faults are introduced in the simulation

in the Narendra filter as explained in Chapter 5 with filter parameters $\alpha = 0.01$, $\beta = 0.99$, and $\gamma = 2$ to obtain the filtered residuals $\boldsymbol{\chi}(t)$. The faults are detected by applying suitable thresholds on the filtered residuals $\boldsymbol{\chi}(t)$ to obtain the fault index signal $\boldsymbol{\iota}(t)$. The selected threshold is $\zeta = 0.15$ for all faults.

Figure 6.15 shows the residuals along with the filtered residuals $\boldsymbol{\chi}(t)$ and the fault index signals $\boldsymbol{\iota}(t)$. As can be seen, the external input and relative acceleration faults are both detected. The external input fault $f_{u_\mathrm{t}}(t)$ is detected after 0.49 s with $\iota_1(t)$ and after 0.33 s with residual $\iota_3(t)$. The relative acceleration fault $f_{\Delta a}(t)$ is detected after 0.52 s with $\iota_2(t)$ and after 0.1 s with residual $\iota_3(t)$. After the faults are removed, the faults are still detected for a time interval due to the filter parameters in the residual generator and in the Narendra scheme.

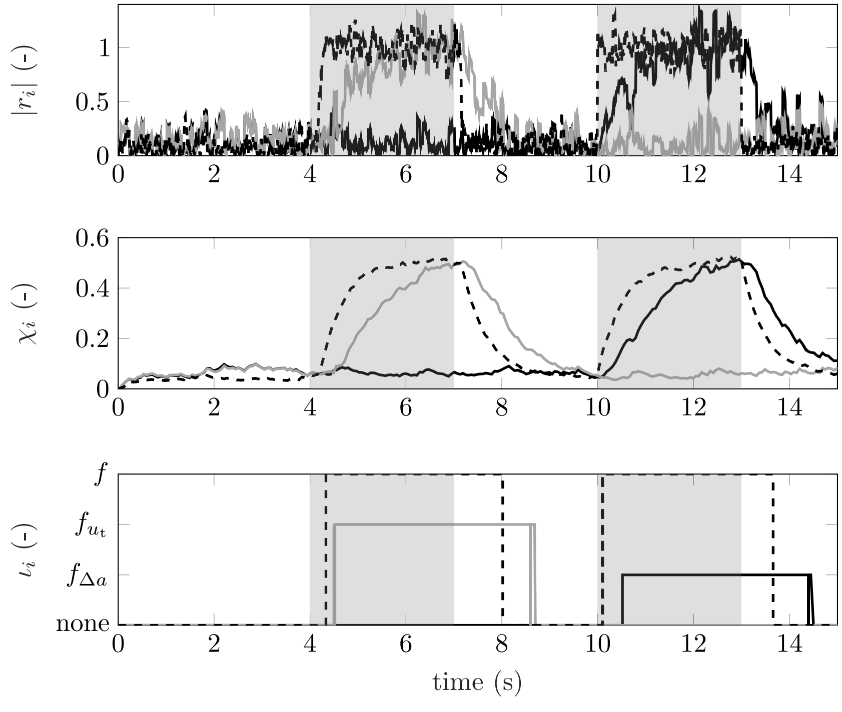


Figure 6.15.: Detected faults during the simulation: $f_{u_\mathrm{t}}(t)$ (——), $f_{\Delta a}(t)$ (——), and f (- - -). The shaded areas (▭) indicate when faults are introduced in the simulation

6.5. Experimental Results

To validate the designed method, experiments have been performed using the vehicle and implementation as described in Section 6.2. The experiment is performed in the same manner as the simulations to be able to compare the results. Figure 6.16 shows the measured output $\mathbf{y}(t)$ and input $\mathbf{u}(t)$ signals during the experiment, which can be directly compared with the simulation results, as depicted in Figure 6.14. The relative velocity, host vehicle velocity, and host vehicle acceleration clearly show similar behavior to those shown in the simulation. The offset of the host vehicle velocity is caused by the fact that the initial velocity of the vehicles during the experiment is lower than in the simulation and the achieved acceleration of the vehicles in the experiment is slightly lower than in the simulation. Even though the longitudinal dynamics are linearized using an inverted torquemap, the system is sensitive to disturbances, such as road surface or battery state of charge. The difference in relative acceleration signal is probably caused by unmodeled sensor characteristics of the radar sensor.

The external input fault and relative acceleration fault can be detected by feeding the output and input signals, as shown in Figure 6.14, through the filter $\mathbf{Q}(s)$ (6.30), (6.31), and (6.32) to obtain the residuals $\mathbf{r}(s)$. The residuals $\mathbf{r}(t)$ are then filtered in the Narendra filter as explained in Chapter 5 with filter parameters $\alpha = 0.15$, $\beta = 0.85$, and $\gamma = 1.25$ to obtain the filtered residuals $\boldsymbol{\chi}(t)$. The selected values are different than the ones that are used in the simulation as the residuals contain more noise than in the simulations. The faults are detected by applying suitable thresholds on the filtered residuals $\boldsymbol{\chi}(t)$ to obtain the fault index signal $\boldsymbol{\iota}(t)$. The selected thresholds are $\zeta_{f_{u_\mathrm{t}}} = 0.34$ for the external input fault $f_{u_\mathrm{t}}(t)$, $\zeta_{f_{\Delta a}} = 0.32$ for the relative acceleration fault $f_{\Delta a}(t)$, and $\zeta_f = 0.6$ the fault signal f.

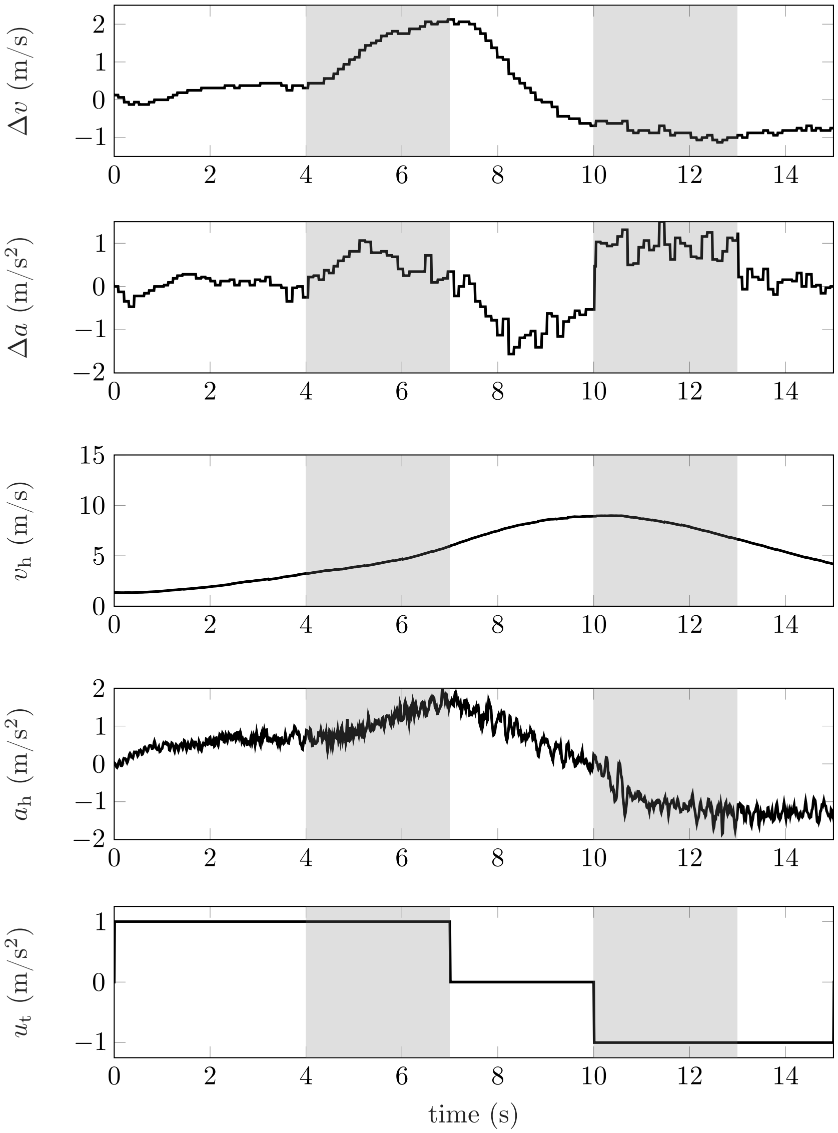


Figure 6.16.: Measured output $\mathbf{y}(t)$ and input $\mathbf{u}(t)$ signals during the experiment: relative velocity $\Delta v(t)$, relative acceleration $\Delta a(t)$, host vehicle velocity a_{h}, host vehicle acceleration a_{h}, and desired acceleration u_{t}. The shaded areas (▭) indicate when faults are introduced in the experiment

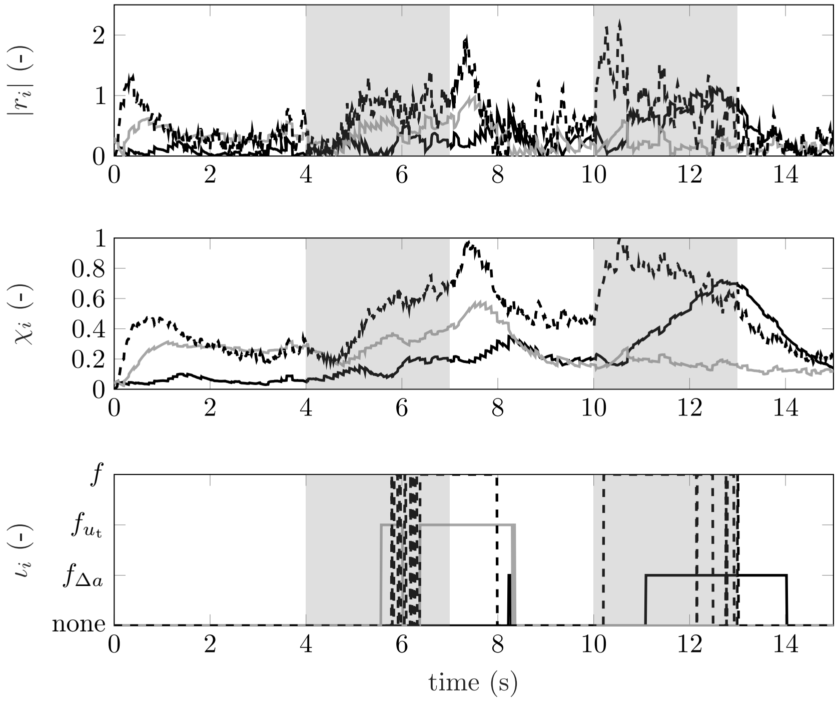


Figure 6.17.: Detected actuator faults during the experiment: $f_{u_\mathrm{t}}(t)$ (——), $f_{\Delta a}(t)$ (——), and f (- - -). The shaded areas (▭) indicate when faults are introduced in the experiment

6.6. Conclusion

A fault detection method for CACC is presented to detect when incorrect information is received from the preceding vehicle or when a wrong object is tracked without assuming a specific CACC controller structure. The proposed strategy utilizes a model of the CACC system to timely detect these faults. The method can be used with the current equipment level of CACC systems. To demonstrate the technical feasibility, the fault detection method is successfully implemented on a research and development vehicle along with mitigating measures to increase the inter-vehicle distance to a safe distance in case of a fault situation.

7.

Conclusions and Recommendations

There is a growing public demand for safer and more efficient road transportation. Automated Driving Systems (ADSs) have the potential to comply with this demand as sophisticated systems can be more vigilant, competent, and reliable than the average human driver. In addition to safety, the ADSs can improve driver comfort, road throughput, and efficiency by taking over driving tasks.

This thesis aims to contribute to the safety aspect of automated vehicles, in particular on how to develop a safe and reliable research and development for Society of Automotive Engineers (SAE) level 3 and beyond driving. Consequently, the following objectives are defined:

- develop a systematic design approach for a safe and robust research and development platform by identifying the relevant components for the design, while taking into account the safety aspects required for SAE level 3 and beyond driving;

- build the research and development platform and verify the design, thereby ensuring that the automated vehicle is safe to operate;

- develop and validate a simulation framework for testing and validating automated driving controllers;

- develop fault diagnosis algorithms to detect actuator faults and user interventions, and deploy and operate them on the research and development platform; and

- develop fault diagnosis strategies for connected vehicles, including environmental perception systems.

The main conclusions from these objectives are provided in Section 7.1, and the recommended topics for further research identified from this thesis are given in Section 7.2.

7.1. Conclusions

Practical implementation of an ADS requires a vehicle equipped with drive-by-wire technology and additional hardware. As most commercially available vehicles do not have full drive-by-wire capability, these vehicles have to be modified to be used as a research and development platform. In Chapter 2 a systematic design approach is presented to convert a small electric vehicle into a research and development platform, with the objective to comply with SAE level-3 driving. This chapter shows that a safe and reliable automated driving platform can be constructed by using a simple vehicle as a base and equipping it with mainly Commercial Off-The-Shelf (COTS) components. The approach followed in this work is to split the automated vehicle design in building blocks and integrates safety in every step. This approach leads to safety by design instead of safety through testing, i.e., statistical safety. By designing the automated driving platform from scratch, an open system is obtained that allows for safe and predictable behavior of the implemented controllers. The presented systematic design approach and vehicle implementation provides valuable insights in how to construct a research and development vehicle for automated driving and can be used for other simple candidate vehicles.

Next to the hardware design of the automated vehicle, a real-time application is required that processes sensor signals, plans control actions, and executes these actions. In Chapter 3, a well-defined automated driving software framework is developed, consisting of a layered functional architecture. The clear boundaries of the layered approach enables separate design and evaluation of software components, thereby managing the complexity. The developed architecture together with the hardware design of Chapter 2, allowing safe operation of the research and development vehicle, without having to explicitly take safety into account when developing automated driving functionality.

Testing safety-critical systems, such as an automated vehicle, requires virtual testing before starting road tests. Therefore, in Chapter 4 a simulation framework is developed for testing and validating automated driving controllers. The created simulation framework can be used in two configurations: accelerated-time in which (a part of) the control system is integrated in the simulation environment and real-time in which the control system is separated from the simulation environment. The model framework is able to run in real-time and the results quite closely matches with the data obtained from a full-scale driving test. The framework can be adapted to other vehicle characteristics by changing the vehicle parameters.

With an automated vehicle, a driver might not notice certain faults when an ADS performs the dynamic driving tasks. Therefore, an ADS must supervise itself by including fault diagnosis. Model-based fault diagnosis for an automated steering system is discussed in Chapter 5. The fault detection filters can detect a steering actuator fault and an user intervention. Also driver's arm state detection is possible, but the method heavily relies on an accurate system description. The model-based fault diagnosis techniques remove the need for extra sensors, thereby reducing cost and eliminating possible points of failure. This technique can also be employed to act as a redundant measure to provide higher levels of safety.

Next to a single automated vehicle, connected vehicles can improve road safety and efficiency even more by employing a wireless link to share information with other traffic participants. Cooperative Adaptive Cruise Control (CACC) is a system that uses a wireless link to allow vehicles to follow each other with shorter inter-vehicle distances compared to a normal driver operated vehicle, thereby increasing road throughput and driver comfort. At these reduced intervals, the human driver cannot be considered as backup and the system must supervise itself to avoid possible unsafe driving situations. In Chapter 6 a method is presented that can be used to detect when incorrect information is received from a preceding vehicle or when a wrong object is tracked. The presented method applies model-based fault diagnosis principles and is evaluated in simulation and in practice. This method can be used with the current equipment level of CACC systems. The results show that it is possible to detect the fault within 0.5 s.

7.2. Recommendations

This work touches upon various aspects of safety in automated vehicles, but it does not guarantee safe automated driving in all circumstances. The research and development vehicle as considered in this thesis is based on a small electric vehicle. While most added components are COTS, the developed brake actuator for example is a custom part as no suitable off-the-shelf solutions were available. With the ongoing developments in recent years, there might be a COTS available for the numerous custom solutions.

Next to reliable hardware, robust software is required. The vehicle-to-vehicle communication as used on the research and development platform consists of a router that receives and transmits the wireless messages. The router also checks for every signal that is being communicated whether it lies in their specified range. That does, however, not guarantee reliable data transmission as incorrect data might still fall in the specified range. In addition, the communication channel from the router to the real-time application is not included in this check. Therefore, it would be advisable to adapt the protocol and add an information redundancy check to the message, such as a checksum.

Additionally, a redundant communication channel can be included based on a different protocol, such as cellular communication. The redundant communication channel can be used to communicate with other traffic participants in case the current main wireless link is compromised. The bandwidth requirements of the redundant channel can be smaller compared to main channel as only the most vital information is required in case of a safety critical situation, such as whether the vehicle in front is applying the brakes.

As shown in this thesis, model-based fault diagnosis heavily relies on an accurate system description. Therefore, an interesting direction for future research is to investigate online model learning techniques to reduce model inaccuracies. To be able to distinguish model inaccuracies from faults, the model adaptations should be relatively slow compared to the fault dynamics.

Another possibility is to extend the fault detection method for connected vehicles is by including other high resolution sources, such as camera. By combining high-resolution sensors with extended object tracking methodologies, additional vehicle states of the target can be observed. These states can be compared with the wireless obtained data to increase the detection reliability.

Machine learning techniques are commonly used for environmental perception systems. The machine learning approaches are aimed at mimicking human driving behavior by gathering real-life driving data. This approach works well for certain traffic situations. The drawback however is that these systems behave as a black box, giving out control commands based on sensor data, making it difficult or even impossible to understand certain failures and/or to guarantee safe behavior. To cope with these situations from a fault detection perspective, two or more techniques can be implemented and their outputs compared, similar to what is common in aviation. A fault can be triggered when their outputs significantly differ.

The fault detection strategies examined in this thesis range from sensor and actuator level, in which the functioning of an individual component is considered, to a system level in which safety of connected vehicles is considered. The fault detection strategies can be extended and applied on the whole vehicle.

A.

Multibody Model

This appendix contains detailed schematic representations of the multibody model. Figure A.1 gives a schematic representation of the McPherson suspension, indicating the joints and bodies. The design of the front and rear suspension is almost similar besides the implementation of the driveshaft. Figure A.2 provides a schematic representation of the steering rack, showing the connections to the chassis and toe links. Figure A.3 gives a schematic representation of the stabilizer bar, indicating all the joints and bodies.

The spring and damper characteristics are depicted in Figures A.4 and A.5, respectively. Figure A.6 shows the result of two coastdown tests and compares it with the simulation model. Figure A.7 the results of a steady-state cornering test on a radius of 120 m. During the test, the longitudinal velocity is slowly increased till a maximum lateral acceleration is achieved of $7\,\mathrm{m/s^2}$.

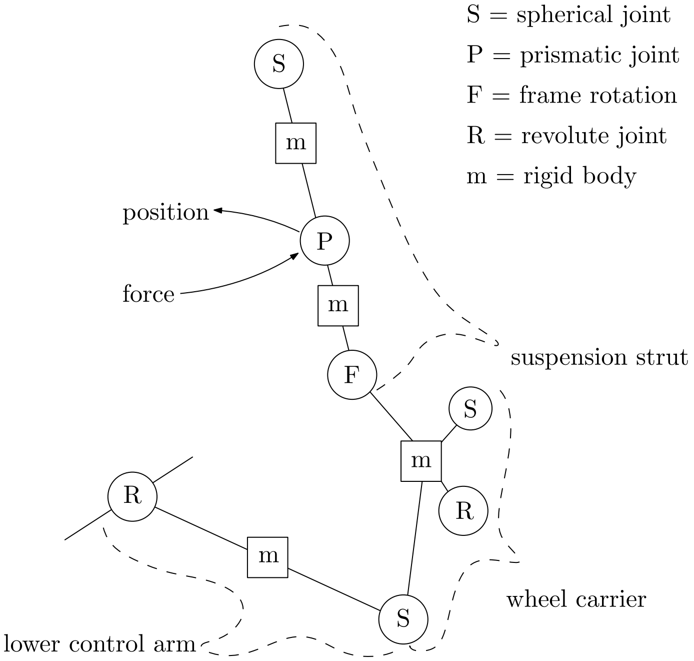


Figure A.1.: Schematic representation of a McPherson suspension (courtesy: (Baaij 2019))

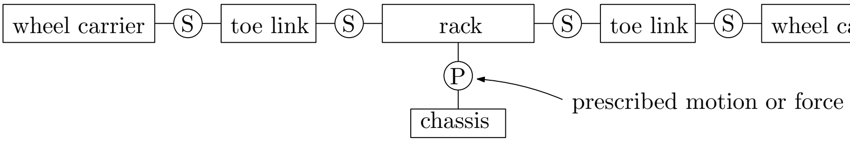


Figure A.2.: Schematic representation of the steering rack (courtesy: (Baaij 2019))

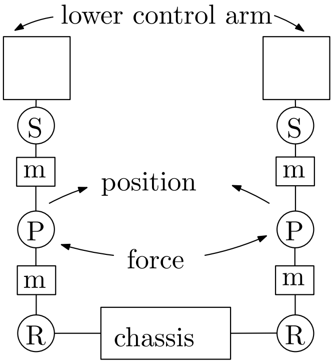


Figure A.3.: Schematic representation of the stabilizer bar (courtesy: (Baaij 2019))

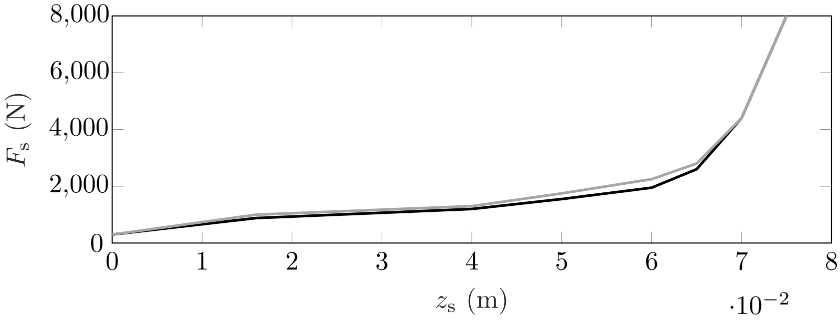


Figure A.4.: Spring force F_s as function of spring displacement z_s: front (——) and rear (——) (source: (Baaij 2019))

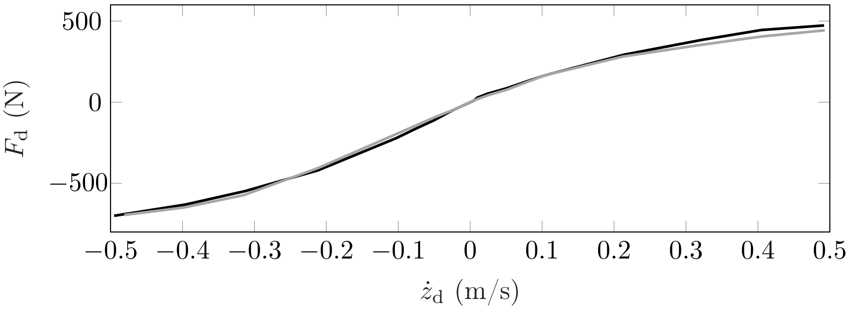


Figure A.5.: Damper force F_d as function of damper velocity $\dot{z}_\mathrm{d}$: front (——) and rear (——) (source: (Baaij 2019))

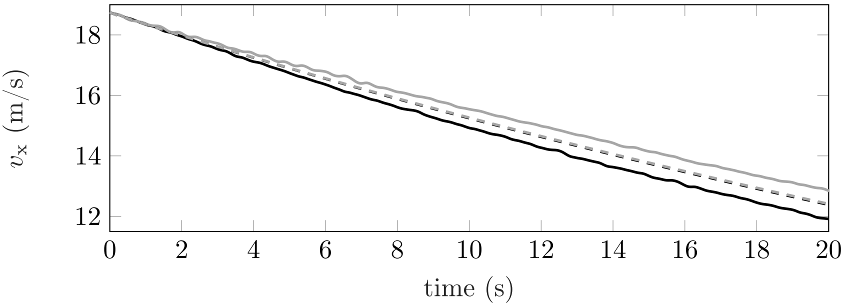


Figure A.6.: Comparison of coastdown test between model and measurement for two directions: measurement1 (——), measurement2 (——), model1 (- - -), and model2 (- - -) (source: (Baaij 2019))

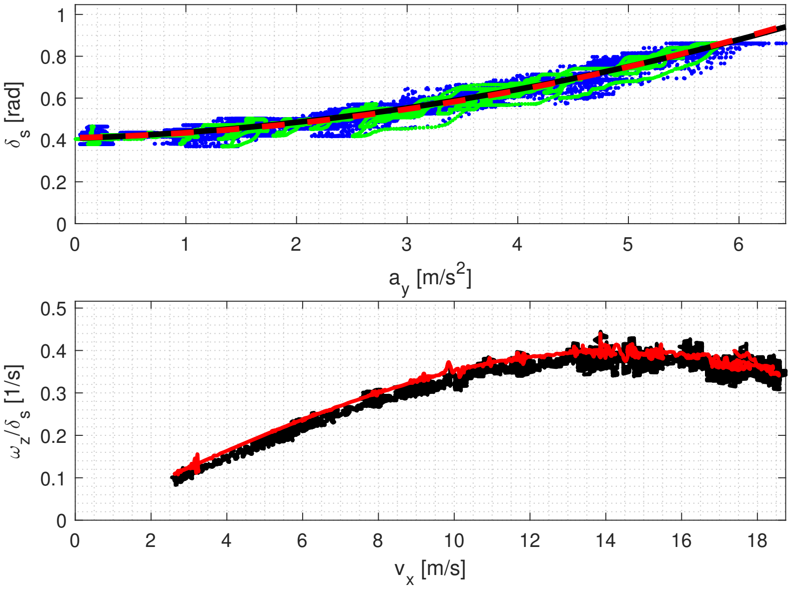


Figure A.7.: Steady state cornering test with a radius of 120 m: steering angle (top) and yaw velocity gain (bottom) (courtesy: (Baaij 2019))

Bibliography

Advantech (2019). *ARK-3520P*. Advantech.

Apex (2017). *Planetary gearbox new generation P-series*. Apex dynamics, inc.

Apollo (2019). *An open autonomous driving platform*. `http://apollo.auto` (visited on 09/06/2019).

Applied Motion Products (2013). *TSM23C Integrated Step-Servo Motor*. Rev. A. Applied Motion Products.

Aptiv et al. (2019). *Safety First for Automated Driving*. Technical report.

Astrom, K. J. and L. Rundqwist (1989). “Integrator windup and how to avoid it”. In: *1989 American Control Conference*. IEEE, pages 1693–1698.

AutonomouStuff (2019). *Lexus RX450h PACMod 3.0 System*. Technical report. AutonomouStuff.

Baaij, S. (2019). “Development and validation of a multibody model of a Renault Twizy”. Master’s thesis. Eindhoven University of Technology.

Bayuwindra, A. (2019). “Look-ahead tracking controllers for integrated longitudinal and lateral control of vehicle platoons”. PhD thesis.

Besselink, I. (2019). “Tire characteristics and modeling”. In: *CISM International Centre for Mechanical Sciences, Courses and Lectures*. CISM International Centre for Mechanical Sciences, Courses and Lectures. Springer, pages 47–108.

Bigelow, P. (2019). *Why Level 3 automated technology has failed to take hold*. `https://www.autonews.com/shift/why-level-3-automated-technology-has-failed-take-hold` (visited on 07/21/2020).

Bohlin, T. P. (2006). *Practical grey-box process identification: theory and applications*. Springer Science & Business Media.

Borraz, R. et al. (2018). “Cloud Incubator Car: A Reliable Platform for Autonomous Driving”. In: *Applied Sciences* 8.2, page 303.

Bosch (2014). *LWS3 Steering-angle sensor*. Bosch.

— (2015). *Mid-range radar sensor (MRR) for front and rear applications*. Bosch.

— (2018a). *Inertial sensor MM5.10*. Bosch.

— (2018b). *Sensors*. `https://www.bosch-ibusiness.com/media/downloads/sensors_i-business.pdf` (visited on 12/11/2018).

Bosch, R. (2003). *ACC Adaptive Cruise Control*. Bentley.

Buttazzo, G. C. (2011). *Hard real-time computing systems: predictable scheduling algorithms and applications.* Volume 24. Springer Science & Business Media.

Campbell, S. F. (2007). "Steering control of an autonomous ground vehicle with application to the DARPA urban challenge". PhD thesis. Massachusetts Institute of Technology.

Chen, J. and R. J. Patton (1999). *Robust model-based fault diagnosis for dynamic systems.* Kluwer Academic Publishers.

Comma.ai (2019). *Open source driving agent.* `https://github.com/commaai/openpilot` (visited on 06/09/2019).

Dajsuren, Y. and G. Loupias (2019). "Safety analysis method for cooperative driving systems". In: *2019 IEEE International Conference on Software Architecture (ICSA).* IEEE, pages 181–190.

Dardanelli, A., G. Alli, and S. M. Savaresi (2010). "Modeling and control of an electro-mechanical brake-by-wire actuator for a sport motorbike". In: *IFAC Proceedings Volumes* 43.18, pages 524–531.

Dataspeed (2018). *ADAS kit DRIVE-BY-WIRE TECHNOLOGY.* Technical report. Dataspeed inc.

Dissanayake, G. et al. (2001). "The aiding of a low-cost strapdown inertial measurement unit using vehicle model constraints for land vehicle applications". In: *IEEE transactions on robotics and automation* 17.5, pages 731–747.

EB robinos (2017). *open robinos specification. architecture, mechanisms, interfaces, and testing.* Technical report. Elektrobit.

ETSI (2011). "Intelligent transport systems (its); vehicular communications; basic set of applications; part 2: Specification of cooperative awareness basic service". In: *Draft ETSI TS* 20, pages 448–451.

— (2019). *302 663: Intelligent Transport Systems (ITS); ITS-G5 Access Layer Specification for Intelligent Transport Systems Operating in the 5 GHz Frequency Band.*

Faitelson, D., R. Heinrich, and S. Tyszberowicz (2018). "Functional Decomposition for Software Architecture Evolution". In: *Model-Driven Engineering and Software Development.* Edited by L. F. Pires, S. Hammoudi, and B. Selic. Cham: Springer International Publishing, pages 377–400.

Farrell, J. (2008). *Aided navigation: GPS with high rate sensors.* McGraw-Hill, Inc.

Feddes, G. (2016). "Legalisation for automation: mind the gap. Why the autonomous city taxi also need to pass (future) legislation". In: *11th ITS European Congress, Glasgow, Scotland.*

Freedman, D., R. Pisani, and R. Purves (2007). "Statistics (international student edition)". In: *Pisani, R. Purves, 4th edn. WW Norton & Company, New York.*

Gadda, C. D. (2009). *Optimal fault-detection filter design for steer-by-wire vehicles.* Stanford University.

General Motors (2018). *2018 Self-Driving Safety Report.* Technical report. General Motors.

Genta, G. (1997). *Motor vehicle dynamics: modeling and simulation.* Volume 43. World Scientific.

Harrer, M. and P. Pfeffer (2017). *Steering handbook.* Springer.

Herger, M. (2019). *Disengagement Reports 2018 – Final Results.* `https://thelastdriverlicenseholder.com/2019/02/13/update-disengagement-reports-2018-final-results/` (visited on 01/06/2020).

Hermans, T. et al. (2009). "Decoding of data on a CAN powertrain network". In: *16th Annual Symposium on Communication and Vehicular Technology in the Benelux (1).* Volume 6.

Ho, L. M. and D. Ossmann (2014). "Fault detection and isolation of vehicle dynamics sensors and actuators for an overactuated X-by-wire vehicle". In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on.* IEEE, pages 6560–6566.

Hoogeboom, F. (2020a). *Real-time control system.* `https://bitbucket.org/FHoogeboom/rt_control_system2/src/master/` (visited on 01/06/2020).

— (2020b). *Real-time simulation model.* `https://bitbucket.org/FHoogeboom/rt_simulation_model/src/master/` (visited on 01/06/2020).

Hoogeboom, F., T. van der Sande, and H. Nijmeijer (2018). "Fault Detection and Isolation of Steering System Actuators for an Automated Vehicle". In: 14th International Symposium on Advanced Vehicle Control, AVEC2018.

IEE (2020). *Hands Off Detection (HOD) Sensing System.* `https://www.iee-sensing.com/en/products/automotive/hands-off-detection` (visited on 08/02/2020).

ies-synergy (2018). *ELIPS 2000W with embedded DC/DC 360W converter, devoted to small electric vehicles.* `http://www.ies-synergy.com/en/products/onboard-sealed-chargers/elips-2000w` (visited on 11/12/2018).

ISO 26262 (2011). *Road vehicles-Functional safety.* Technical report. International Standard ISO/FDIS.

ISO/PAS 21448 (2019). *Road vehicles-Safety of the intended functionality.* Technical report. International Standard ISO/FDIS.

Jafarnejad, S. et al. (2015). "A car hacking experiment: When connectivity meets vulnerability". In: *Globecom Workshops (GC Wkshps), 2015 IEEE.* IEEE, pages 1–6.

Jeong, Y. et al. (2015). "Vehicle sensor and actuator fault detection algorithm for automated vehicles". In: *Intelligent Vehicles Symposium (IV), 2015 IEEE.* IEEE, pages 927–932.

Kalra, N. and S. M. Paddock (2016). “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?” In: *Transportation Research Part A: Policy and Practice* 94, pages 182–193.

Kato, S. et al. (2018). “Autoware on board: Enabling autonomous vehicles with embedded systems”. In: *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, pages 287–296.

Khabbaz Saberi, A. (2020). “Functional Safety: A New Architectural Perspective: Model-Based Safety Engineering for Automated Driving Systems”. PhD thesis. Department of Mathematics and Computer Science.

Kochanthara, S. et al. (2020). “Semi-automatic Architectural Suggestions for the Functional Safety of Cooperative Driving Systems”. In: *2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE, pages 55–58.

Koopman, P. and M. Wagner (2016). “Challenges in autonomous vehicle testing and validation”. In: *SAE International Journal of Transportation Safety* 4.1, pages 15–24.

— (2018). *Toward a framework for highly automated vehicle safety validation.* Technical report. SAE Technical Paper.

Kruiswijk, A., I. Besselink, and H. Nijmeijer (2012). “Towards automated steering for vehicle platooning”. In: *Proceedings of the International Symposium on Advanced Vehicle Control (AVEC), 9-12 September 2012, Seoul.*

Lawrenz, W. (1997). “CAN system engineering”. In: *From theory to practical applications, New York.*

Li, C. et al. (2017). “Model-based sensor fault detection and isolation method for a vehicle dynamics control system”. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 231.2, pages 147–160.

Loof, J. (2018). “Modeling and control of a truck steering-system for active driver support”. PhD thesis. Department of Mechanical Engineering.

Maguire, P. B. and J. Bennett (2016). *Hand sensing on steering wheel using heater element.* US Patent 9,346,480.

Marco Furlan (2019). *MFeval.* `https://mfeval.wordpress.com/` (visited on 09/10/2019).

Marouf, A. et al. (2012). “A new control strategy of an electric-power-assisted steering system”. In: *IEEE Transactions on Vehicular Technology* 61.8, pages 3574–3589.

Matlab (2020a). *Automated Driving Toolbox: Design, simulate, and test ADAS and autonomous driving systems.* `https://nl.mathworks.com/products/automated-driving.html` (visited on 02/12/2020).

— (2020b). *MATLAB Simscape: model and simulate multidomain physical systems.* `https://nl.mathworks.com/products/simscape.html`.

— (2020c). *Simulink Real-Time: Build, run, and test real-time applications.* `https://nl.mathworks.com/products/simulink-real-time.html` (visited on 07/02/2020).

Maxon motor (2017). *Epos2 Positioning Controllers, Firmware Specification.* November 2017. Maxon motor ag.

Miller, C. and C. Valasek (2015). "Remote exploitation of an unaltered passenger vehicle". In: *Black Hat USA* 2015, page 91.

Miller, S. (2020). *Simscape Multibody Multiphysics Library.* `https://nl.mathworks.com/matlabcentral/fileexchange/37636-simscape-multibody-multiphysics-library` (visited on 01/23/2020).

Miller, S. and J. Wendlandt (2010). "Real-time simulation of physical systems using simscape". In: *MATLAB News and Notes*, pages 1–13.

Möller, D. P. and R. E. Haas (2019). *Guide to Automotive Connectivity and Cybersecurity.* Springer.

Narendra, K. S. and J. Balakrishnan (1997). "Adaptive control using multiple models". In: *IEEE transactions on automatic control* 42.2, pages 171–187.

Naus, G. J. et al. (2010). "String-stable CACC design and experimental validation: A frequency-domain approach". In: *IEEE Transactions on vehicular technology* 59.9, pages 4268–4279.

Nelles, O. (2013). *Nonlinear system identification: from classical approaches to neural networks and fuzzy models.* Springer Science & Business Media.

New Atlas (2018). *StreetDrone announces open-source autonomous car platform.* `https://newatlas.com/streetdrone-open-source-autonomous-car/55343/` (visited on 11/12/2018).

Nguyen, A., J. Yosinski, and J. Clune (2015). "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436.

NHTSA (2016). "Federal Automated Vehicles Policy: Accelerating the Next Revolution in Roadway Safety". In.

— (2017). *Automated Driving Systems: A Vision for Safety.* Technical report DOT HS 812 442. U.S. Department of Transportation.

NTSB (2017). *Collision Between a Car Operating With Automated Vehicle Control Systems and a Tractor-Semitrailer Truck Near Williston, Florida, May 7, 2016.*

— (2018a). *Preliminary report highway HWY18FH011.*

— (2018b). *Preliminary report highway HWY18MH010.*

NVIDIA, D. P. (2019). *Scalable AI Supercomputer for Autonomous Driving.*

OVMS (2018). *Open Vehicle Monitoring System.* `https://dexters-web.de/` (visited on 10/06/2018).

Pacejka, H. (2005). *Tire and vehicle dynamics.* Elsevier.

Parmar, M. and J. Y. Hung (2004). “A sensorless optimal control system for an automotive electric power assist steering system”. In: *IEEE Transactions on industrial electronics* 51.2, pages 290–298.

Pick, A. and D. J. Cole (2007). “Dynamic properties of a driver’s arms holding a steering wheel”. In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 221.12, pages 1475–1486.

Ploeg, J. (2014). “Analysis and design of controllers for cooperative and automated driving”. PhD thesis.

Ploeg, J. et al. (2014). “Graceful degradation of cooperative adaptive cruise control”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.1, pages 488–497.

Reichardt, D. et al. (2002). “CarTALK 2000: Safe and comfortable driving based upon inter-vehicle-communication”. In: *Intelligent Vehicle Symposium, 2002. IEEE.* Volume 2. IEEE, pages 545–550.

Renault (2015). *Renault TWIZY Plug into the positive energy.*

SAE International (2018). *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles.* Technical report.

Schildbach, G. (2018). “On the application of iso 26262 in control design for automated vehicles”. In: *arXiv preprint arXiv:1804.04349.*

Serban, A. C., E. Poll, and J. Visser (2019). “A Standard Driven Software Architecture for Fully Autonomous Vehicles”. In: *Journal of Automotive Software Engineering.*

Sevcon (2012). *Renault Selects Sevcon To Provide Controllers For Twizy.* `http://www.sevcon.com/news/articles/26032012-elects-sevcon-to-provide-controllers-for-twizy` (visited on 11/12/2018).

Severinson, A. (2018). “An Open Platform for Research and Development in Intelligent Transportation Systems”. In.

Shaw, E. and J. K. Hedrick (2007). “Controller design for string stable heterogeneous vehicle strings”. In: *2007 46th IEEE Conference on Decision and Control.* IEEE, pages 2868–2875.

Sheikholeslam, S. and C. A. Desoer (1990). “Longitudinal control of a platoon of vehicles”. In: *1990 American control conference.* IEEE, pages 291–296.

Shladover, S. E., D. Su, and X.-Y. Lu (2012). “Impacts of cooperative adaptive cruise control on freeway traffic flow”. In: *Transportation Research Record* 2324.1, pages 63–70.

Singh, S. (2015). *Critical reasons for crashes investigated in the national motor vehicle crash causation survey.* Technical report.

Softing (2012). *CAN-AC2-PCI CAN Bus PCI Interface for Vehicle Electronics.*

Solà, J. (2017). “Quaternion kinematics for the error-state Kalman filter”. In: *CoRR* abs/1711.02508.

Stoffels, W. (2019). *Modelling a steering system of a Renault Twizy.* Technical report. Internship report. Eindhoven University of Technology.

SWOV (2012). “Headway times and road safety”. In: *SWOV Fact sheet.*

TASS International (2020). *Delft-Tyre: All-in-one solution for tyre modelling.*

u-blox (2016). *NEO/LEA-M8T u-blox M8 concurrent GNSS timing modules.* UBX-15025193. Bosch.

— (2018a). *EVK-M8T User guide.* u-blox.

— (2018b). *u-blox 8 / u-blox M8 Receiver Description, Including Protocol Specification.* R15. u-blox.

Van Arem, B., C. J. Van Driel, and R. Visser (2006). “The impact of cooperative adaptive cruise control on traffic-flow characteristics”. In: *IEEE Transactions on intelligent transportation systems* 7.4, pages 429–436.

Van der Sande, T. and H. Nijmeijer (2017). “From cooperative to autonomous vehicles”. In: *Sensing and Control for Autonomous Vehicles.* Springer, pages 435–452.

Van der Sande, T., I. Besselink, and H. Nijmeijer (2012). “Steer-by-wire: a study into the bandwidth and force requirements”. In.

Van Hoek, R., J. Ploeg, and H. Nijmeijer (2018). “Motion planning for automated connected vehicles”. In: 14th International Symposium on Advanced Vehicle Control, AVEC2018.

Varga, A. (2017a). *FDITOOLS - The Fault Detection and Isolation Tools for MATLAB.* `https://sites.google.com/site/andreasvargacontact/home/software/fditools` (visited on 06/03/2018).

— (2017b). *Solving fault diagnosis problems.* Edited by J. Kacprzyk. Volume 84. Studies in Systems, Decision and Control. Springer.

Voronov, A. et al. (2016). *Implementation of ETSI ITS G5 GeoNetworking stack, in Java: CAM-DENM / ASN.1 PER / BTP / GeoNetworking.* DOI: `10.5281/zenodo.55650`.

Waymo (2019). *Waymo Safety Report. On the Road to Fully Self-Driving.* Technical report. Waymo.

Xing, H. (2019). “Delay-aware analysis and design of cooperative adaptive cruise controllers”. English. PhD thesis. Department of Mechanical Engineering.

Yoshida, K. (2002). “Development of custom IC for EPS torque sensor”. In: *KOYO Engineering Journal English Edition* 160E, pages 48–51.

Yu, C. et al. (2015). “Identification of structured lti mimo state-space models”. In: *2015 54th IEEE Conference on Decision and Control (CDC).* IEEE, pages 2737–2742.

Zolghadri, A. et al. (2014). *Fault diagnosis and fault-tolerant control and guidance for aerospace vehicles.* Springer.

Dankwoord

Na ruim vier jaar hard werken is het dan eindelijk zover en mag ik over vier weken mijn proefschrift verdedigen. Terugkijkend op die periode was het een leuke tijd waarin ik veel geleerd heb, maar het was ook een hele uitdaging. Het eindresultaat was dan ook niet haalbaar geweest zonder de juiste mensen om mij heen.

Op de eerste plaats wil ik mijn promotoren Henk en Tom bedanken. Henk, ontzettend bedankt voor de mogelijkheid en het vertrouwen om een promotie onderzoek te mogen doorlopen binnen de Dynamics & Control groep. De afgelopen periode heeft mij veel gebracht op academisch en persoonlijk vlak. Tom, bedankt voor alle input en gezelligheid tijdens onze formele besprekingen alsmede tijdens onze actieve besprekingen. Ook waardeer ik het ten zeerste dat je bereid was om feedback te leveren op meestal onmogelijke tijden en dat je aanbood om te helpen met testen in het weekend. De afgelopen periode heb ik veel van je geleerd op wetenschappelijk en sportief vlak.

I would like to thank professor Amir Khajepour, professor Mark van den Brand, Maarten Bonnema, and Igo Besselink for taking part in my PhD committee and for providing me with constructive feedback on my thesis.

Ook wil ik graag de medewerkers bedanken waar ik veel mee heb samengewerkt bij het opbouwen van de geautomatiseerde Twizy's. Gerard, bedankt voor alle mechanische zaken en voor je begrip wanneer er toch nog snel even iets moest worden geregeld voor bijvoorbeeld een demo. Wietse, bedankt voor alle elektrische en elektronische zaken en ook voor je begrip wanneer ik al had gezegd dat het niet meer zou wijzigen, maar het toch nog anders moest. Erwin, bedankt voor alle ondersteuning met het regelen van de lab uitrusting. In het bijzonder de keren dat je weer stad en land moest bewegen om net dat ene speciale onderdeel te kunnen bemachtigen dat ik had uitgezocht. Ook heb ik bijzonder genoten van de 'reparatie evaluaties'.

Geertje, bedankt voor alle keren dat ik bij je terecht kon voor het regelen van alle formaliteiten die ook horen bij het promotietraject en voor het organiseren van de jaarlijkse groepuitjes, daar hebben wij altijd erg van genoten.

Ik wil mijn (voormalige) kantoorgenoten bedanken voor de leuke tijd, Erik, Carlos, Veronika, Ruben, and in particular Isaac. What started with some small talk and jokes in the office, resulted in doing various activities together, such as barbecues, cooking Mexican and Dutch food, and going on wintersport twice. I still remember you being ‘the snowman’.

Naast alle (oud-)collega’s op -1, wil ik ook graag onze buren bedanken, Alejandro, Bayu, Haito en in het bijzonder ‘collega’ Robbin. Ik heb de samenwerking met je als zeer prettig ervaren. Dit ging dan ook vaak gepaard met de nodige humor. Zonder jou was deze periode een stuk minder leuk geweest en ik weet niet bij wie ik anders had moeten zijn als er iets niet lukte of ik weer eens wat te klagen had.

Ook buiten het kantoor was het vaak gezellig, zoals bij de koffiepauzes, lunchpauzes, vrijmibo, sportdagen of bij het sportcentrum. Daarvan staan mij in het bijzonder de bodypump lessen op donderdagavond bij, waar na afloop van de les de verbrande koolhydraten weer rijkelijk werden aangevuld in de sportkantine.

Naast de collega’s van de TU/e wil ik uiteraard ook al mijn vrienden bedanken voor de nodige afleiding. Hetzelfde geldt voor mijn familie waar ik altijd weer met open armen ontvangen wordt. Dank voor jullie steun en betrokkenheid bij alles wat ik doe.

Tenslotte wil ik graag degene bedanken zonder wie ik dit promotietraject niet had kunnen volbrengen. Adelien, bedankt voor je onvoorwaardelijke liefde en steun tijdens deze periode met pieken en dalen. Het was niet altijd gemakkelijk, maar je wist mij altijd weer op te vrolijken als het even tegenzat.

Frans Hoogeboom
Eindhoven, september 2020

Curriculum Vitae

Frans Hoogeboom was born on May 21, 1987, in Harenkarspel, the Netherlands. He received his Bachelor of Engineering degree (cum laude) in Automotive Engineering in 2012 at HAN Arnhem. In 2015, he received his Master of Science degree (with great appreciation) in Mechanical Engineering at Eindhoven University of Technology, The Netherlands. As a part of this education, he carried out an internship at Miami University in Oxford Ohio, USA where he worked on human balance and posture control during quiet stance. His master thesis research concerned synchronization of multidimensional coupled metronome systems. In december 2015, he started a Ph.D. project in the Dynamics and Control group at the department of Mechanical Engineering of Eindhoven University of Technology, of which the results are presented in this dissertation.