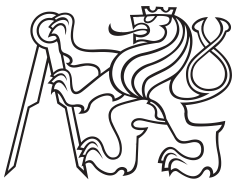
Bachelor Project



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Verification platform development for vehicle control system validation

Vývoj verifikační platformy pro účely validace řídicích algoritmů dynamiky vozu

Tomáš Twardzik

Supervisor: Ing. Tomáš Haniš Ph.D.
Field of study: Control Engineering
May 2020

Acknowledgements

I would like to express sincere gratitude to everyone, who helped me along my academic path, I would not get this far without you. Nevertheless, my greatest thanks belongs to people who worked with me on this very project, namely my supervisor Ing. Tomáš Haniš Ph.D., my two friends and colleagues Adam Konopiský and Adam Škuta and a LFS game developer Scawen Roberts.

Thank you.

Declaration

I declare that I wrote the presented thesis on my own and that I cited all the used information sources in compliance with the Methodical instructions about the ethical principles for writing an academic thesis.

In Prague, 6. May 2020

Abstract

This bachelor thesis is dedicated to a Driver-in-the-Loop vehicle dynamics simulation, mainly focused towards a realistic driver experience. Theoretical part lists various state of the art hardware as well as software simulation alternatives, describing their advantages and liabilities. Consequently, practical part mainly focuses on development of verification platform for control law validation, which will be used in automotive industry. For the purpose of development, I have modified 6 Degrees of Freedom flight simulator using driving input peripherals. The choice of simulation software fell upon the LFS game, due to its supreme vehicle handling qualities and physical model fidelity. Moreover, I have developed an interface interconnecting the Live for Speed (LFS), the modified flight simulator and two development environments, Python programming language and MATLAB/Simulink environment. In order to verify the implemented interface, I have designed two control laws, a cruise control and a yaw damper. Furthermore, the yaw damper control law underwent validation testing in three test scenarios: side wind blow, moose test and measured racing lap. The test results point towards higher vehicle stability and predictability, while having yaw damper system implemented, which suggests that the proposed design is valid.

Keywords: simulation, motion platform, LFS, control system, control law, interface

Abstrakt

Tato bakalářská práce se věnuje tématu Driver-in-the-Loop simulací dynamiky vozu. V teoretické části je přehled různých alternativních hardwarových i softwarových simulátorů s výčtem jejich výhod a nevýhod. V praktické části jsem se věnoval samotnému vývoji verifikační platformy pro validaci řídicích algoritmů v automobilovém průmyslu. Pro tento účel byl modifikován 6-osý letecký simulátor na simulátor dynamiky vozu. Jakožto simulační software byla zvolena hra LFS, která vyniká realističností ovládání vozidel. Dále bylo navrženo propojovací rozhraní mezi LFS, modifikovaným simulátorem, programovacím jazykem Python a prostředím MATLAB/Simulink, který je v oboru řízení standardem. Pro ověření funkčnosti rozhraní byly implementovány dva řídící zákony, tempomat a yaw damper, který byl následně otestován třemi experimentálními scénáři: postranní nárazový vítr, losí test a závodní kola na čas. Výsledky všech experimentů poukazují na lepší předvídatelnost chování a stabilitu vozu, tedy lze předpokládat, že implementovaný řídicí systém je validní.

Klíčová slova: simulace, pohyblivý simulátor, LFS, řídicí systém, řídicí zákon, rozhraní

Contents

Figures

Tables

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Twardzik Tomáš** Personal ID number: **474711**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Verification platform development for vehicle control system validation

Bachelor's thesis title in Czech:

Vývoj verifikační platformy pro účely validace řídicích algoritmů dynamiky vozu

Guidelines:

The virtual verification platform of vehicle dynamics will be develop in the thesis. The verification platform will be based on 6DoF aerospace simulation platform interconnected with selected vehicle dynamics simulation software. The thesis will consist of following points:
1) Review of vehicle dynamics simulation software and platforms
2) Design of simulation platform to vehicle dynamics simulation software interconnection
3) Vehicle dynamics control system development
4) Verification of developed systems

Bibliography / sources:

[1] Dieter Schramm, Manfred Hiller, Roberto Bardini – Vehicle Dynamics – Duisburg 2014
[2] Hans B. Pacejka - Tire and Vehicle Dynamics – The Netherlands 2012
[3] Franklin, Powell, Emami-Naeini: Feedback Control of Dynamics Systems. Prentice Hall, USA
[4] Robert Bosch GmbH - Bosch automotive handbook - Plochingen, Germany : Robet Bosch GmbH ; Cambridge, Mass. :Bentley Publishers

Name and workplace of bachelor's thesis supervisor:

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

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

Date of bachelor's thesis assignment: **10.02.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

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

III. Assignment receipt

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

.	
Date of assignment receipt	Student's signature

Acronyms

MIL	Model-in-the-Loop
HIL	Hardware-in-the-Loop
SIL	Software-in-the-Loop
DIL	Driver-in-the-Loop
HMI	Human-Machine Interface
ADAS	Advanced Driver Assistence Systems
AD	Autonomous Driving
EPS	Electronical Power Steering
ESC	Electronical Stability Control
FFB	Force Feedback
DoF	Degree of Freedom
NVB	Noise, Vibration and Harness
OSP	OutSim Packet
MP	Motion Packet
DI	Driver Inputs
SCD	Static Car Data
I/O	Input/Output interface

Chapter 1

Introduction

The automotive industry is one of the most competitive fields nowadays. Car manufacturers and their suppliers battle among each other and regulative bodies, while earnings are decreasing year after year. One of the costliest action companies must do is hasty development of new and improved products (picture 1.1). Repetitive prototyping and physical testing combined with design changes between each attempt consume extreme amounts of time and resources. Therefore, companies included extensive computer simulations into their development processes, focusing on every possible field of design: control systems, drivetrain technologies, combustion efficiency and emissions, chassis design, steering control etc. A key motivation for using mathematical models and simulation instead of prototyping and physical testing is cost reduction and faster time to market. Mathematical models are built to reflect the real physical system in test scenarios; therefore, their accuracy is of the utmost importance. Computer simulation allows engineers to test various designs with little to no expense and prototype only the most promising iterations, which leads to aforementioned cost reduction. It is important to note that simulations are only approximations of real physical systems and some behavioural patterns and dynamics are neglected or even overlooked by simulation designer, leading to differences between simulated system and real system dynamics. Hence, simulations are never meant to completely replace physical measurements and benchmarking, which validates designed system performance.

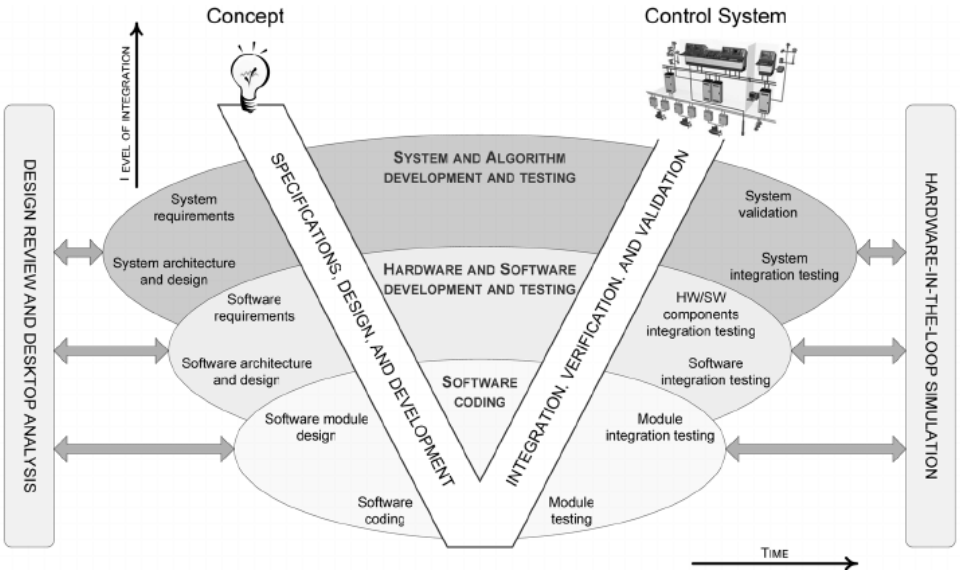


Figure 1.1: V-model describing regular development process of control system [1]

One of the key components of development is vehicle control system. It determines final dynamics, efficiency and overall "polished" feeling of the car and consequently, has an immense impact on product success. Companies developing simulators have aimed to reproduce vehicle dynamics in the most realistic manner however, driver experience was somewhat left behind during this process. This led to a unrealistic driving experience during DIL (Driver-in-Loop) simulations, some of which were caused by unpolished HMI (Human-Machine Interface, such as input controllers, steering wheels etc.), visual lag, eye to the center of coordination misalignment (leading to headache or stomachache). Therefore, the focus of my work is to develop verification platform for vehicle control system validation, which brings similar driver experience such as in a real car.

Part I

Theoretical Background

Chapter 2

Simulators

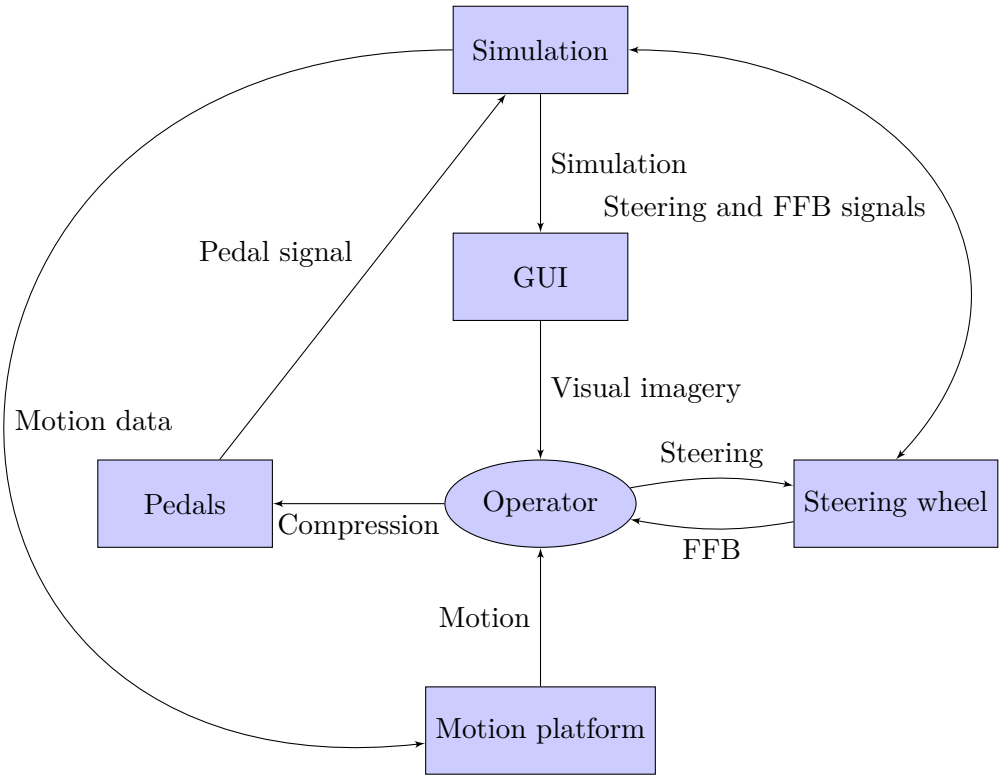


Figure 2.1: Graphical demonstration of DIL systems with motion platform and driving peripherals

Advanced simulator platforms generally consist of two interconnected parts: software simulator and hardware simulator (motion platform or any other haptic feedback devices). Software simulation uses models to study real-world system behaviour and reaction to specific conditions. Models are usually represented by a set of differential and algebraic equations, which determine system dynamics, in this matter, car dynamics. The software can also provide a graphical interface for humans to interact with, which is necessary for DIL simulators, where a human operator directly influences a course of the simulation. For more realistic operator's experience (necessary for DIL simulation), HMI (Human-Machine Interface) is included. The HMI usually consists of feedback devices, such as motorized steering wheel and motion platform (Graph 2.1). The purpose of a motion platform is to artificially reproduce similar forces and accelerations on the operator to further improve simulation fidelity.

There are multiple different options for both software and hardware simulators, each providing different specializations, advantages and drawbacks. In the next section I will describe various commonly utilized simulators for car dynamics.

2.1 Motion Simulators

To begin with, it is important to state that physical simulators are not intended to simulate the dynamics of the car itself. Its primary purpose is to imitate forces, which would impact driver in real-life driving scenarios. Driving on its own is multi-sensory action; we do not just drive with our eyes, but hands, legs and especially inner ear with its vestibular system matter to a great extend. This brings a need for specific machinery in order to create stimuli for each of our receptors, providing simulation as realistic as possible. However, this also poses a new potential issue. All stimuli must be in sync with one another; otherwise, it may result in incohesive feelings for an operator at best, motion sickness or headaches at worst.

Driver Input

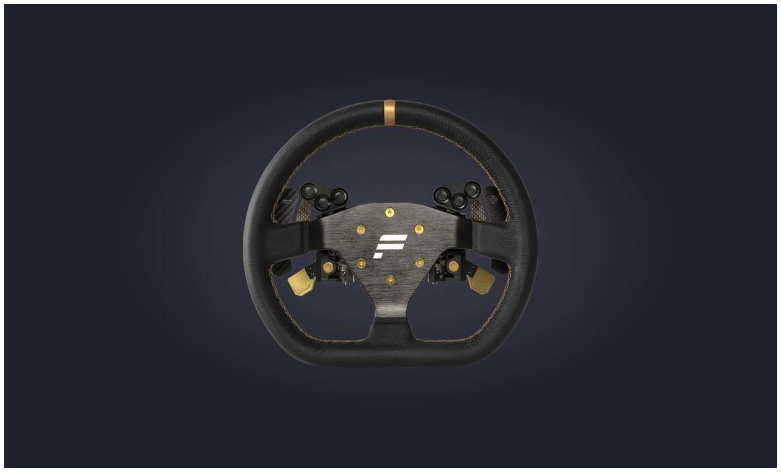
When speaking about driving a car, an immediate association is to driver input devices. Standard car HMI (Figure 2.2) consists of the steering wheel, pedals (clutch, brake, throttle), gear shift and handbrake. For the vast majority of simulation purposes it is not essential to have handbrake and gear shift, especially given the fact some racing wheels are provided with built-in shifting paddles. Therefore, I will not be discussing those unnecessary driver input devices.

Figure 2.2: Skoda Fabia interior with all driver input devices: steering wheel, pedals, handbrake, gear shift [2]

2.1.1 Steering Wheel

The most important input device for drivers is a steering wheel. Simulation steering wheels consist of a steering wheel (Figure 2.3b) and a wheel base (Figure 2.3a) mounted to simulator skeleton. Unlike in a real car, where motion of steering wheel is carried through systems of shafts and power gears, electro-hydraulic system or EPS, resulting in final wheel alignment, here is an electronic position sensor, which determines displacement of steering wheel. There can be first important difference among competitors and that is precision and range of this sensor, giving more fluent and smooth steering input for a driver. Another factor is the range of motion. Especially cheaper racing wheels do not allow to exceed $\pm$ 135 degrees of angular displacement,

(a) : Fanatec high end racing simulation wheel base [3]



(b) : Fanatec high-end racing simulator steering wheel [4]

Figure 2.3: Example of driver inputs

whereas more sophisticated wheels allow all the way up to ±540 degrees of steering. Third and last crucial feature is Force Feedback (FFB). Force feedback emulates forces applied to a steering wheel during a ride. From FFB can the driver determine, whether the car is about to slip or if he hit a bump or wheel slid off the track. Quality of this feature is measured by maximal torque FFB motor can output, response times and customization options.

Pricing options for steering wheels vastly vary. Cheapest options include basic reading sensors, low range of motion (does not accurately represent real car steering range and sensitivity) and no FBB functionality. For realistic simulation it is necessary to have advanced racing wheel, with accurate position sensors, extensive range of motion and FFB, which enables for an accurate imitation of forces. Notable brands are following: Logitech, Thrustmaster, Fanatec, Simucube.

2.1.2 Pedals

Second essential input device is pedals assembly (Figure 2.4a). In real cars, these signals (clutch, brake, throttle) are carried in mechanical, hydraulic or electric domain respectively. Similarly to steering wheels, pedal displacements are read by electronic sensors. The rebound of pedals is usually carried out by a spring, advanced racing pedals allow for specific press force options. Most sophisticated pedals utilize hydraulic systems (Figure 2.4b), that should feel much like pedals in real cars however, they do not bring meaningful advantage for simulation purposes. Commonly, pedals are included in package with racing steering wheels.

(a) : Thrustmaster pedals with customizable press force and actuation point [5]

(b) : Tilton 600 hydraulic pedals assembly [6]

Figure 2.4: Example of pedals used for simulation purposes

Motion Platforms

The main purpose of a motion platform is to increase authenticity of simulation and improve operator's experience. For this reason they are built to resemble real car interior, along with matching peripherals, field of view and last but not least imitating forces and accelerations real driving would induce.

2.1.3 Static Platform

The most fundamental physical driving simulator is a static platform (Figure 2.5). It usually consists of a driving seat and a mounting skeleton. Skeleton is set with pedals and steering wheels just like in real car. There are multiple options for visual interface of simulation, mostly utilizing TVs or PC monitors; nevertheless, there are more advanced options such curved projectors, which give panoramic image alike human field of view. This setup does not provide any advantages in terms of realistic feeling of the simulation. However, it is clearly more cost-effective in comparison with the following options.

Figure 2.5: Static car dynamics simulator at CTU FEE

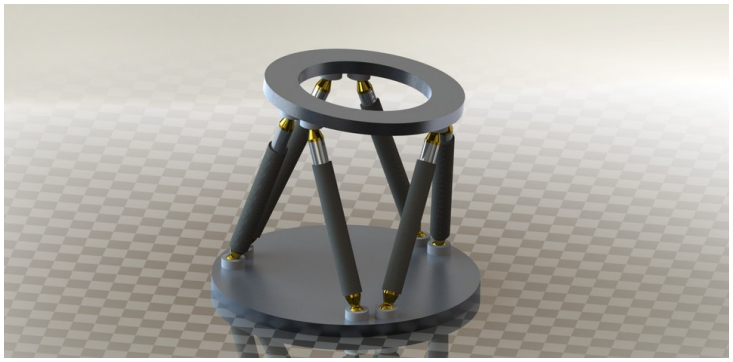
2.1.4 Dynamic Seats

Alternative option available on the market is a racing seat simulator. Its construction strongly resembles static platforms, comprising of a seat, a skeleton of some kind, with mounts and optional visual and haptic peripherals. The difference lies in the presence of the motorized skeleton. The most trivial applications use three prismatic actuators (usually electric linear motors) to produce three vehicle principal axes (3 DoF); yaw, pitch, roll. Just with those three angles, it can roughly mimic real car acceleration, deceleration and centrifugal forces, utilizing gravitational force to stimulate operator's vestibular system by tilting directions (backwards for acceleration, forward for braking, sideways for turning). With the addition of the fourth actuator, simulator is able to perform heave movement, making it 4 DoF robotic device (Figure 2.6) [7].

Figure 2.6: Dynamic seat 4 DoF simulator from Motion - Sim [7]

2.1.5 Steward-Gough Platform

Steward-Gough platform, also known as a hexapod, was developed in early 1950s. Platform uses six prismatic actuators mounted in alternating pairs on both ends, the base and the platform (Figure 2.7), imparting this construction to total of 6 degrees of freedom. Hence, objects placed on top of the platform are able to move along x, y, z axes as well as perform yaw, pitch and roll operations. A great advantage of parallel manipulators is their precision and sturdiness; however, it lacks in speed and operational space. For realistic reproduction of forces in a car simulation it is necessary to have a big operational space, especially in ship movements, sway and surge, while maintaining sufficient operational space for vehicle principal angles. Nevertheless, Steward-Gough platform ship movement limits angular displacements and vice versa ;in this regard is this architecture not optimal. As a result, some of the high-end simulators use moving base with a hexapod design simulator mounted on the top, in order to enlarge simulator's motion envelope and improve fidelity of the simulation (Figure 2.7b) [8].



(a) : Steward-Gough platform CAD schematic [8]

(b) : Toyota hexapod architecture 6 DoF simulator with moving base for motion envelope extension [9]

Figure 2.7: Steward-Gough platform with alternating pair mounting

2.1.6 Advanced Linear Actuator Simulators

Simulators falling into this category are engineered specifically to perform car dynamics simulations. Design requirements are forced to maximal authenticity and realistic driver perception. To achieve these specifications, simulators use linear actuators in configuration that allows maximum acceleration, speeds and displacements in all possible directions. This is especially important for sway and surge movements, where it does not rely just on gravity and tilt to stimulate the vertebral system of an operator, but the platform itself can considerably move, hence produce acceleration forces. This is allowed by vastly bigger motion envelope in comparison with Steward-Gough platform. State of the art simulators can move up to ± 1.5 meters in sway and surge directions, 30 degrees in yaw, whereas heave, roll and pitch motion envelopes are less important, therefore restricted by design. Another advantage beyond responsiveness is non-parallel construction of actuators. This results in an independence of particular degrees of freedom, which does not limit motion envelope as drastically as with the hexapod design. On the other hand, construction like this usually requires a substantially greater number of actuators than hexapod's six, some manufacturers use up to 13 actuators for their top of the line simulators. Prize tags for simulators belonging to this category are in millions of Euros; however, an investment like this is justified since more than 25 major carmakers have their own (2.8) [10] [11].

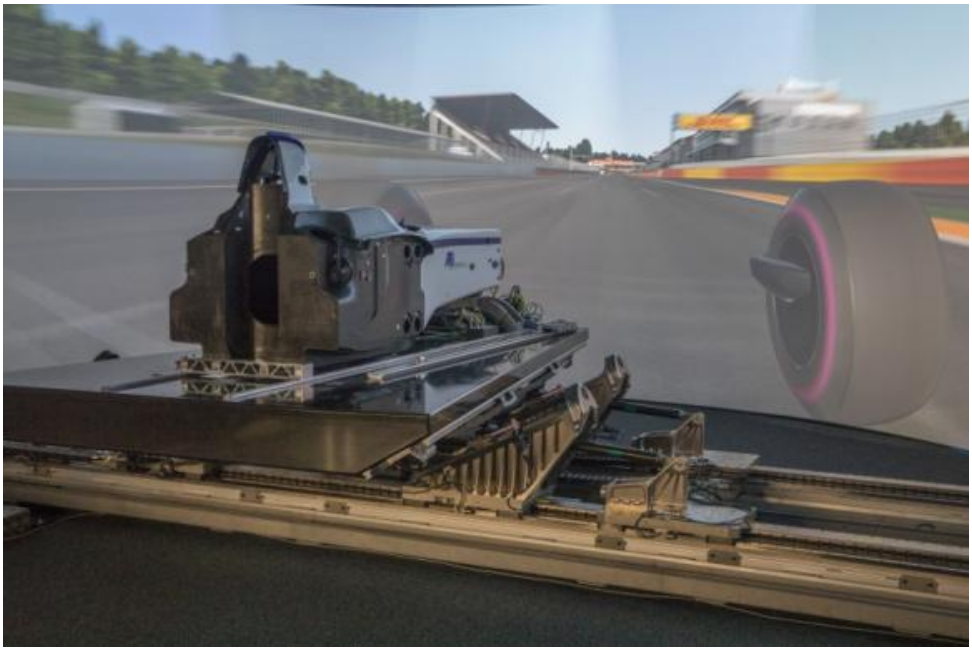


Figure 2.8: aVDS vehicle dynamics simulator from AB Dynamics developed for motorsport customers. Simulator uses 13 actuators, granting it 6 DoF [11]

2.2 Software Simulators

No motion or static platform can ever be useful without input signals to its peripherals and actuators. To provide particular signals, it is necessary to run computer simulation of car dynamics. This simulation runs in discrete-time with requirement to have iterations faster than real-time, giving us ability to calculate an additional control law for the system. The system in a computer simulation is represented by a mathematical model, which is built from differential and algebraic equations. In order to obtain an authentic simulation, it is necessary to have a high fidelity model with well identified parameters. Simulations with such model can currently get substantially close to a real-life behavior of a system, providing engineers a powerful tool for prototyping and validating the design.

There are various computer simulators available on the market, each of which generally allows user to simulate car driving simulation. Vast majority of simulators have broader applications, giving tools to design and test specific car systems such as chassis, suspension, engine, gearbox, exhaust system etc.

2.2.1 IPG CarMaker

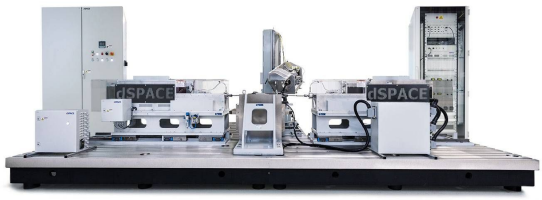
One of the world's most profound software simulators is IPG CarMaker (Figure 2.9). CarMaker focuses on all car design aspects, including vehicle dynamics, powertrain, ADAS (Advanced driver assistance systems) and AD (Autonomous Driving). In order to thoroughly examine system behavior, there are broad sets of fully automated scenarios, evenets and maneuvers included for each of these branches, including extreme condition tests. Powertrain simulations also include hybrid and electrical drives, braking simulations, real driving emission and fuel consumption tests as well as drivetrain efficiency. ADAS are currently one of the most growing segments of a car development and it grows in importance year after year. Some of ADAS are even obligatory (by EU regulations), such as ESC (Electric Stability Control), which helps to control car in dangerous states and conditions (slippery road, high speed maneuvers). CarMaker has licensed EuroNCAP automated test, which covers various scenarios and technologies, including AEB, FCW, LDW, LKA, SLIF, MSA, ISA. The IPG's software also includes tools to create vehicle dynamics control law, supports real data import, such as road maps. It also enables to use sensor simulations, namely LIDAR, radar, speed, acceleration sensors etc. AD developers can use any of foregoing ADAS or sensory inputs; moreover, CarMaker also packs semantic segmentation data for neural network training and deployment [12].

Figure 2.9: iPG CarMaker marketing picture diplaying various usage options of CarMaker software portfolio [13]

2.2.2 dSpace

dSpace company focus is to provide compact and complex simulation platform, covering every possible requirement. They not only provide simulation software, but also mechanical testbench units and a hardware for HIL (Hardware-in-the-Loop) simulations (Figure 2.10), including customizable I/O for sensors and processing units. Their software portfolio supports drivetrain and transmission optimizations, chassis design and testing, engine simulations. Alike CarMaker, dSpace solutions include automated testing in various conditions and scenarios, EuroNCAP virtual proving grounds, sensor simulations (LIDAR, radar, cameras), smart car development tools (semantic segmentation, ADAS, AI model training). Also, the powertrain designer includes options for fully electrical or hybrid driven vehicles, vehicle dynamics include multi-body systems such as trucks and special models for a motorsport car development. dSpace simulation hardware uses parallel computing, graphic cards and FPGAs in order to provide real-time simulations; however, it is not required to use proprietary dSpace simulation software with it [14].

(a) : Full Scale dSpace solusion for HIL simulations



(b) : dSpace testbench solutions cover steering, braking, motion platforms as well as radar and sensoric testing

Figure 2.10: Examples of dSpace hardware products [15]

2.2.3 CarSim

Mechanical Simulation develops a commercial simulation software VehicleSim. One of its branches focuses on car simulation, hence the name CarSim. CarSim supports MIL, SIL, HIL, DIL real-time simulations and can work as a standalone software, thus has its own computation engine. This can be particularly useful for low budget projects. Nevertheless, CarSim does not include library of automated tests for significant time savings, but developers can build their own scenarios such as U-turn maneuver. The simulator also provides vehicle sensor simulations and interactive traffic, chassis and suspension designer with analysis. Car dynamics is customizable in various points, namely tyres, engine, suspension, ESC, powertrain controllers all the way to drive train controllers (Figure 2.11) [16].

Figure 2.11: CarSim graphical interface with speed, trajectory and vertical tyre forces graphs [17]

2.2.4 Adams

Adams is a simulation software aimed to represent high fidelity physics in computational models. Its main focus is towards multi-body dynamic systems (Figure 2.12), distribution of loads and forces affecting each of the system's parts. Unlike any other simulators mentioned in this chapter, Adams performs calculations concerning material distress and durability, NVH (Noise Vibration Harness) and flexible body integration. On the other hand, Adams also includes a Control systems designer, allowing it to simulate vehicle dynamics and design control laws. Adams is usually paired with FEA (Finite Element Analysis) computations; however, in comparison with standard CAD material durability test, Adams can display change of material distress during the full range of motion of the system. Furthermore, NVH analysis can be crucial, especially if there is a risk of resonance within the system therefore, Adams provides a tool set to examine its model's frequency responses (including high frequencies) without need for prototyping [18].

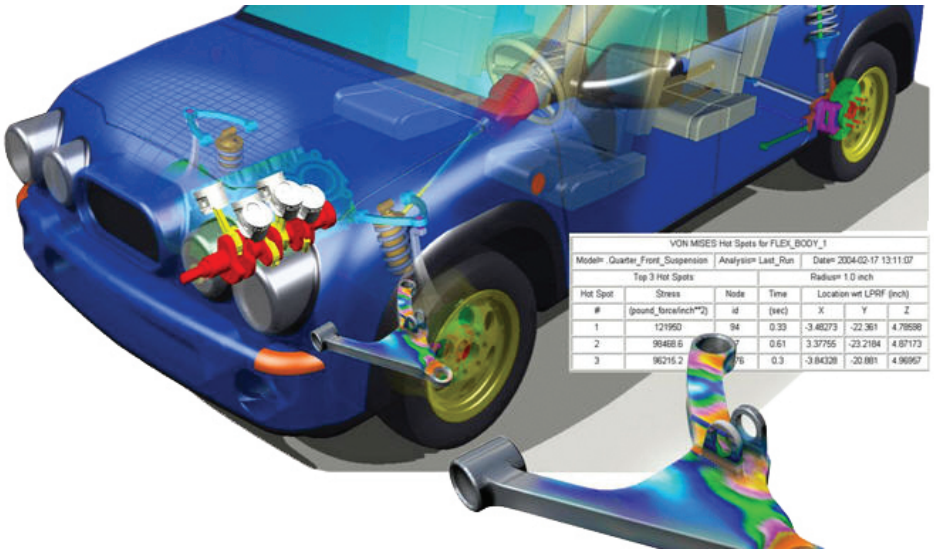
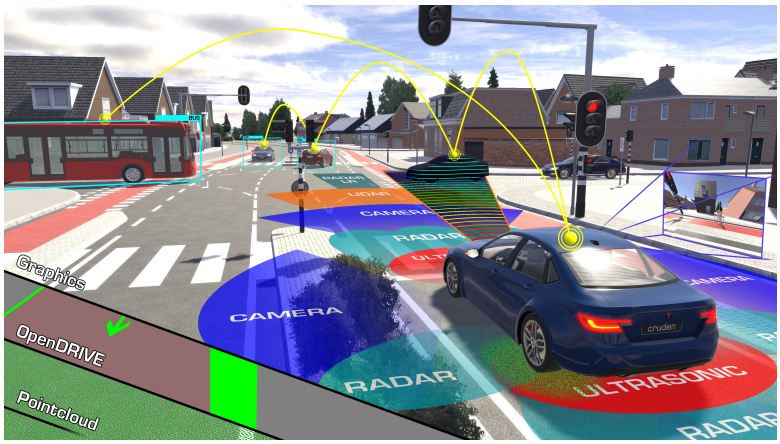


Figure 2.12: Adams multi-body simulation of car engine, steering and suspension systems [18]

2.2.5 Panthera

Panthera is software real-time simulator with SIL, HIL and DIL integration. Developers can utilize its very own physics engine or use Panthera as visualization interface for any of the previously mentioned simulators. Furthermore, Panthera supports sensor simulations for AD and ADAS development (Figure 2.13a) and is capable of interconnection with arbitrary motion platform simulators and force feedback devices. Cruden (mother company) also provides motion platforms (Figure 2.13b), including their own peripherals all very focused on realistic HMI. Hence Cruden simulation solutions were previously used for subjective and objective vehicle assessments, driver behavior and perception research as well as research for autonomous driving handover moments [19].



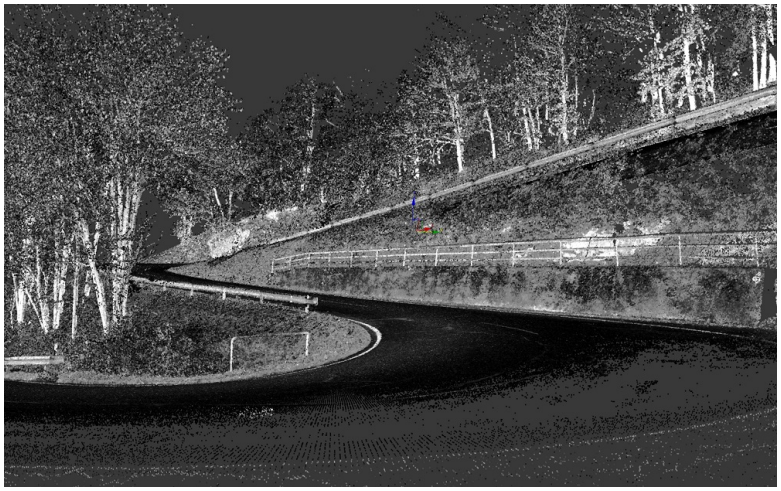
(a) : Panthera sensor simulation graphical display [20]

(b) : Panthera Steward-Gough motion platform simulator [21]

Figure 2.13: Panthera software and hardware simulation products

2.2.6 rFpro

rFpro aims specifically for vehicle dynamics, ASAS and AD control systems development. It enables engineers to use supervised learning for AD neural networks (Figure 2.14b); however, its solution brings various specific conditions into play, such as weather, noised input due to car realistic tilting, shadows and light reflections. On top of it, rFpro provides real digital public road twins from all over the world (Figure 2.14a), including racetracks, which are remodeled yearly; therefore, proving grounds are always up to date. rFpro is capable of traffic and pedestrian simulations, sensory simulation and chassis design, all in real-time execution [22].



(a) : rFpro lidar real road scan taken in Germany

(b) : rFpro tool kit includes semantic segmentation for netral network training purposes

Figure 2.14: rFpro ADAS development tools [23]

2.2.7 Live for Speed (LFS)

LFS on its own is a racing game than anything else. For this reason, one of its core attributes is a gameplay, which makes driving cars more realistic in comparison with any previously mentioned simulation platforms. However, it also brings disadvantages, such as there are no automated test or any tools available. There is no option to remodel in-game cars and their respective physics, apart from minor car setup tuning (Figure 2.15). For the purpose of ADAS development the game is lacking direct torque inputs, there are no sensory simulations or AD tools to speak of. Nevertheless, driving experience is sufficient for a objective handling quality testing and live telemetry data allows for implementation of basic control laws on this platform.

Figure 2.15: LFS in-game car setup, allowing to customize braking, suspension, tyre and steering profiles

Part II

Practical Work

Chapter 3

Concept, HW and SW Tools

For the purpose of creating a verification platform for vehicle control system validation, I needed to build interconnection interface between each individual component of the simulator (Figure 3.1), namely driver input devices (steering wheel and pedals), motion platform, software simulation, audiovisual output as well as MATLAB/Simulink environment. For a successful interface implementation it is necessary to understand simulation hardware and software specifications as well as the features, both of which will be discussed in the following chapters.

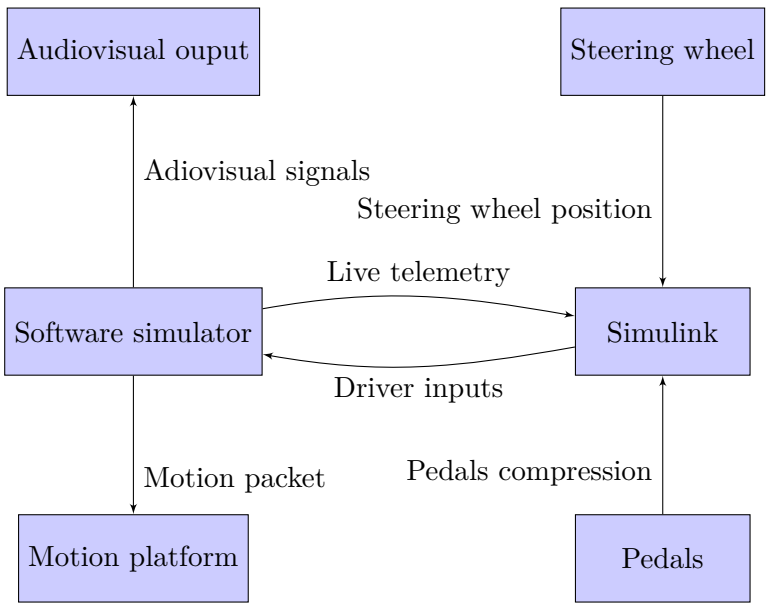


Figure 3.1: Simulator interconnection

3.1 Simulator Hardware and Software

CTU FEE has a static vehicle dynamics simulator and a 6 DoF Steward-Gough platform, which is customized to work as flight simulator (Figure 3.2). My goal was to modify Steward-Gough platform to work as a vehicle dynamics simulator, while using the LFS game as a software simulator. Furthermore, I selected a steering wheel and a pedals as driver input devices, excluding any other real car counterparts.

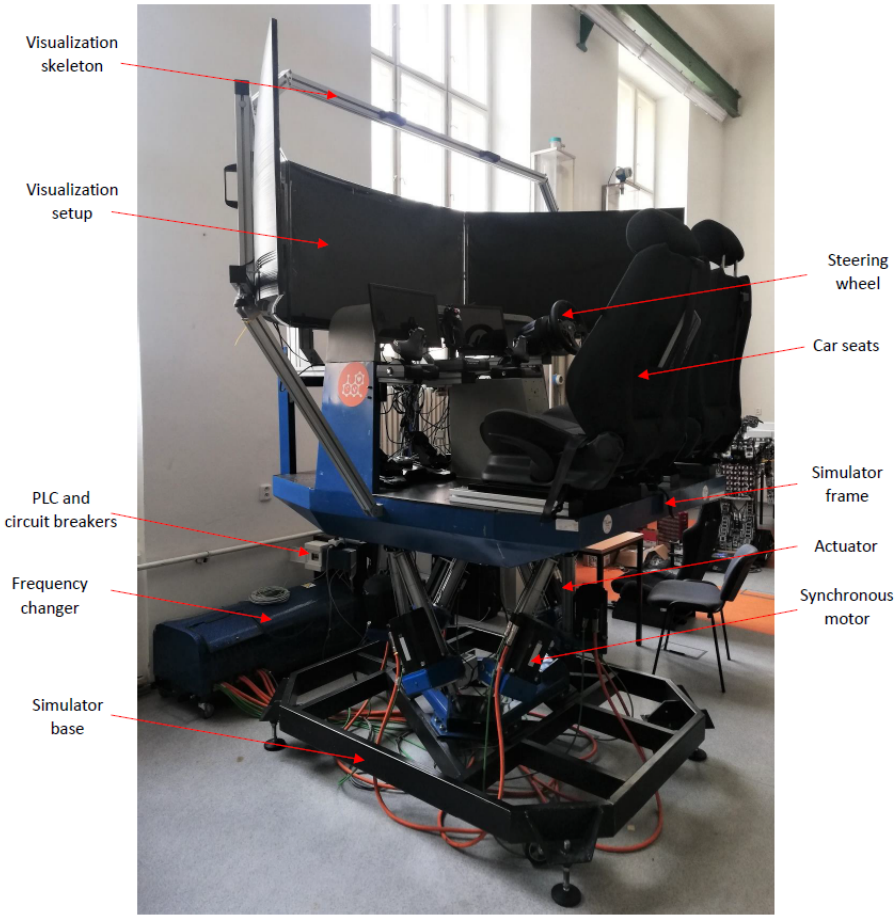


Figure 3.2: Construction of the motion platform used for development

3.1.1 Motion Platform

The only available motion platform at CTU FEE is 6 DoF Steward-Gough flight simulator. Flight simulator physical construction is outlined in the Figure 3.2. The entire construction weights approximately 700 kg and a maximum load reaches 2 000 kg. The motion simulator power supply demands 400V/32A 5-pin outlet, peek input power reaches 6kW. A hexapod architecture (described in subsection 2.1.5) operates with six linear actuators, all of which are regulated by a Programmable Logic Controller (PLC) circuit (Figure 3.3). The PLC regulates motors actions using their immediate position, nine control signals from simulation PC and a washout filter. Control signals from PC are the following: Cartesian acceleration, aircraft principal axes, angular velocities with respect to aircraft principal axes hereinafter, these signals will be referred to as a Motion Packet (MP for short). Communication between the simulation PC and the PLC is utilized via User Datagram Protocol (UDP), since there is no need for safe, lossless communication. On the other hand, the great emphasis is placed on a speed and responsiveness of the motion platform, hence a minimization of transfer overhead is important.

A washout filter performs high pass filtration of Motion packet signals; therefore, output is only sensitive to substantial input changes. Consequently, it provides self-aligning utility of the simulator, where simulator slowly returns to home position, thus preparing it for next fast actuation. Motion envelope of Steward-Gough platform is limited to the following constrictions: yaw – ±45 degrees, pitch and roll – ± 30 degrees, and Cartesian movement in all axes ± 30 cm.

Figure 3.3: PLC unit on the right, circuit breakers on the left

3.1.2 Steering Wheel and Pedals

As previously discussed in subsubsection 2.1, it is very important to choose fitting driving wheel and pedals to accompany motion platform. Main consideration points in decision-making process were: FFB feature, sufficient motion range (car-like motion range), build quality, responsiveness and price. As a result, I chose Thrustmaster T300 GT RS, which includes a steering wheel, a wheel base as well as driving pedals (Figure 3.4). The wheel base supports Force Feedback, which is induced by a industry grade BLDC motor. Position measurements are provided by a Hall effect sensor. Its motion envelope ranges ±540 degrees. Included Thrustmaster T3PA pedals provide customizable press force; however, actuation is only linear, nevertheless this does not pose any significant problem for simulation authenticity. Lastly, the steering wheel is planted with 17 buttons and two paddles for sequential shifting nevertheless, during simulation, only automatic gear shift will be used.

(a) : Steering wheel

(b) : Pedals

Figure 3.4: Driver input peripherals Thrustmaster T300 GT RS set [24]

3.1.3 Audiovisual Appliances

The CTU FEE flight simulator uses setup of 3 Samsung 55" TVs (Figure 3.5), providing high resolution (3820x2160 px) and refresh rate of 60 Hz. Furthermore, this setup provides spacious, 160 degrees wide FOV (Field of View), which helps to isolate operator and increase simulation fidelity. On the other hand, simulator does not provide surround sound system to improve its immersiveness. Sound is provided through TV's loudspeakers, which is sufficient for vehicle dynamics verification platform [25].

Figure 3.5: 3 TV visualisation setup

3.1.4 Software Simulator - LFS/Simulink

As a software simulator I chose Live for Speed (LFS). LFS in one of the most popular game racing simulators within amateur race-sim community, especially for its very accurate driving experience as well as the fact, that LFS is available as freeware, unlike many other commercial software solutions. Unfortunately, game does not allow to create or to modify car models; therefore, it is our job to provide vehicle dynamics model to work alongside with it (in Simulink enviroment), whereas in any other commercial software simulators, there is model designer or even pre-built car model twins. Models which will be used in the future are SingleTrack an TwinTrack models made by Ing. Denis Efremov, at CTU FEE.

Chapter 4

LFS Customization and Interconnection

4.1 LFS Customization with Developers

As indicated earlier, the LFS is a racing simulator game. Game has its in-built simulator features called OutSim and InSim [26]; however, their interface is vastly limited. I cannot modify or even inspect a mathematical model; there are only a few output signals informing about the state of the vehicle and input signals are limited just to driver inputs, which unfortunately is less than optimal for control purposes. Hence, I contacted game's developers to work some of these issues out.

Developers did not want to share their mathematical model of vehicles with us; however, they were willing to broaden output signal packets. I will be further referring to these packets as OutSimPackets, OSP for short. Due to a game graphics development, they also refused to improve input signals situation, but they have shown interest in working on this issue afterwards.

Currently, these signals can be obtained through live-telemetry (content of OutSimPackets):

- Time[1]
- Angular velocities[1]
- Aircraft principal axle angles[1]
- Acceleration in global coordinates[1]
- Velocity in global coordinates[1]
- Position in global coordinates[1]
- Driver input signals
- Motor and drivetrain signals
- Distance measurement
- Wheel signals

Driver input signals consist of Throttle, Brake, Steering, Clutch and Handbrake signals, however, as previously mentioned, my simulator will not be utilizing the clutch nor the handbrake input options. Motor and drivetrain signals include: Current gear, Engine angular velocity and Maximal torque at instant velocity (fully compressed throttle pedal). Wheel signals are obtained for all four wheels in the following order: left rear, right rear, left front and right front wheel. For each particular wheel, available signals are the following: Suspension deflection, Steer, X Force, Y Force, Vertical Load, Angular Velocity, Lean relative to road, Air Temperature, Slip Fraction, Touching, Slip Ratio and Tangent of slip angle.

[1]Signals available in unmodified OutSimPacket

4.2 Interconnection Layout

A complete LFS platform is built as a DIL control system validator. System's model in our case is of two kinds, one is an in-game model and the second one is built inside a Simulink environment. The goal in this case is to create a duplex bridge between those two models. LFS model is inaccessible; therefore, we are trying to fit our Simulink model as closely as possible to the LFS one. Simulink model uses operator's actions and LFS's signals as an input and creates an output, which a represents reshaped driver input. Consequently, I need to build a signal bridge from LFS to Simulink and afterwards create DI device emulator. As a result, I have created a Drive-by-wire system. LFS signals also need to control motion platform described in subsection 3.1.1. Furthermore, it is necessary to read driver input (DI) signals, possibly while preserving FFB functionality; however, interface I have created is not capable of doing so. Audiovisual interconnection proposed in Figure 2.1 is handled by LFS itself; hence, this part is not discussed entirely.

First iteration of LFS interface:

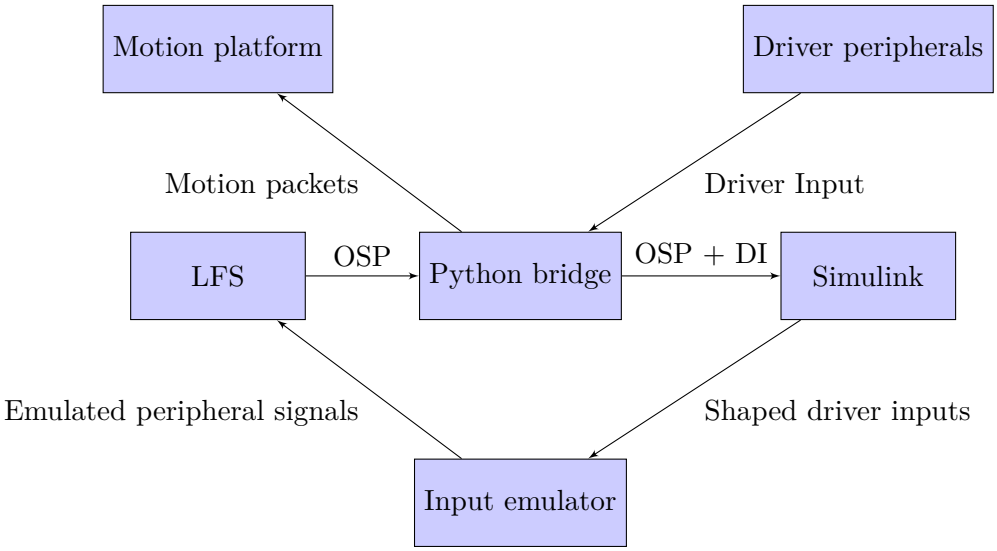


Figure 4.1: Iterconnection of LFS to Simulink

This interconnection layout (Figure 4.1) served mainly as a proof of concept. Python block acts as a bridge between every part of the interface. It utilizes three UDP connections, one as an input from LFS, two as an output for motion platform and Simulink. Driver input signals are sent along with the OSP to a Simulink envirnoment via UDP.

4.3 Connection of LFS to Python and Simulink

All of the OSP signals can be sampled once every game frame with a minimal delay between consecutive packets of 10 ms. Game itself is currently running at around 110-120 Frames per Second (FPS). As a result, we can expect worst case scenario of 20 ms offset between packets during uncompromised operation of the simulator. Nevertheless, it is important to point out, that this software does not provide real-time features; therefore, any in game lag will delay new data reads. A complete OutSimPack contains 82 variables and is 272 bytes long. LFS game allows access to this data via UDP protocol, giving fast, nonetheless unreliable interconnection. Speed and reliability are further improved by utilization of LAN (Local Area Network). Consequently, I measured an approximate delay introduced by an UDP connection, which was found out as insignificant (2 ms).

In order to read the data in Python, I used two other libraries, *socket* and *struct*. The *socket* library facilitates transmission via UDP and TCP protocols. Usage of the library is straightforward. I created a socket object with specified IP address, port and standard parameters. Afterwards, I can read data in the loop. Read itself is in a blocking mode, meaning that we read data whenever it is available; however, the program is stuck on that instruction up until the data is provided. This is not an issue of any kind, since the Python bridge has almost only one purpose, resulting in a simple single-threaded application. The data read from LFS are serialized, hence the need for *struct* library. The *struct* library provides formatted data parsing, separating all the values by their data types.

Pseudo code of the reading loop:

```
socket_in = socket.init(IP_address1,port1)
socket_out = socket.init(IP_address2,port2)
repeat until(socket.timeout){

    outsim_data = socket.read()
    data = unpack(outsim_data)

    data = handle_data()
    socket_out.sendto(data)
}
```

Reading data in Simulink (Figure 4.2) is similar to the same process in Python. To do so, I used the UDP-receiver block provided by the Simulink Real-Time package. Just as with Python, data received are serialized; therefore, I used the byte-unpack block from the same package. Furthermore, byte-unpack output is a user specified formatted vector. As a result, all live telemetry data are accessible in both Python and Simulink environments, giving a user higher flexibility in logging and testing.

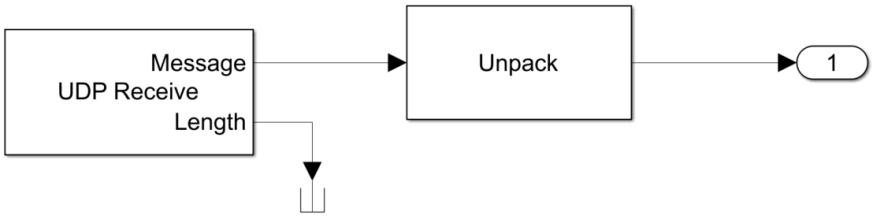


Figure 4.2: Simulink simplified schematic for data receiving and parsing

4.4 Connection of the Motion Platform

In subsection 3.1.1 I shortly described the operation of the motion platform, including a communication interface. As previously mentioned, to control the motion platform it is necessary to have an access to motion packet signals, namely vehicle principal axes angles, angular velocities with respect to the same axes and accelerations relatively a car body system. LFS OSP however, includes only accelerations in global coordinate system, therefore I needed to perform coordinate system transformation by a rotation [27]. The transform is applied to every incoming packet and it uses signals of acceleration and aircraft principal axes. Mathematical representation is as follows:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \mathbf{R}_\mathbf{g}^\mathbf{c} \cdot \begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix} \tag{4.1}$$

where x_g, y_g, z_g refer to xyz coordinates of global coordinate system, x_c, y_c, z_c refer to xyz coordinates of the car body coordinate system and angles α, β, γ refer to yaw, roll and pitch respectively.

Rotation matrix is unable to fit in one page, due to it's size and therefore it is listed in an appendix section under term Mathematical appendix

Motion platform, as any other robotic device, carries a risk of harm to its operator. Because of this, it was strictly necessary to properly handle Motion packets to avoid any accidents. MP are handled in steps: units conversion, factoring, limiting and rounding. The unit conversion changes angular signals from a radian scale to a degree scale. Factoring multiplies acceleration values by a factor of 0.4 and angular velocities by a factor of 0.1. These steps were undertaken to slow down motion platform's movement and therefore avoid

shaky and unstable bahavior. Limiting suppresses signals under user defined thresholds within platform's motion envelope. Lastly, rounding is performed in order to minimize a "floating" movement in static position, caused mostly by a floating point operation precision.

4.5 Static Binary Data

Although live data telemetry provides up to 82 signals, some key variables about vehicle are still missing, for example the mass, type of drive train etc. I will call these variables static car data (SCD) and they can be parsed out of binary file created by game in garage menu. Static car data, as name suggests, does not change during the course of the simulation. Static car data is valuable, because it provides key components for a car's model identification, hence it can vastly improve model fidelity [28].

Static data includes details regarding:

- Car construction
- Aerodynamics
- Motor and drivetrain
- Brakes
- Wheels

Car construction variables comprise of body matrix and center of gravity vectors, moment of inertia matrix, weight, wheelbase and weight distribution. Aerodynamics data provide information about drag and lift of body, undertray, front and rear spoilers. Motor and drivetrain variables include maximal torque and power at particular RPM (Rotation per Minute), number of gears, torque split and efficiency, drive type (fwd, rwd, all-wheel) and final drive ratio. Brake info consists of brake strength and brake balance variables. Wheel data contains description for each particular wheel alone. It consists of a tyre type, width and height, unsprung mass, spring constant, damping coefficients for rebound and compression, anti-roll, cam, caster, inclination, toe in, scrub radius, moment of inertia and suspension deflection.

SCD can be obtained with a dedicated Python script or in a Matlab m-file. Python script creates a *.mat* file with parsed SCD as well as a text file with all variables and their names. Matlab m-file parser read entire SCD, whereas python parser reads only data requested by my co-workers.

4.6 Custom Mounting of Driver Input Devices and Driver Input Reading

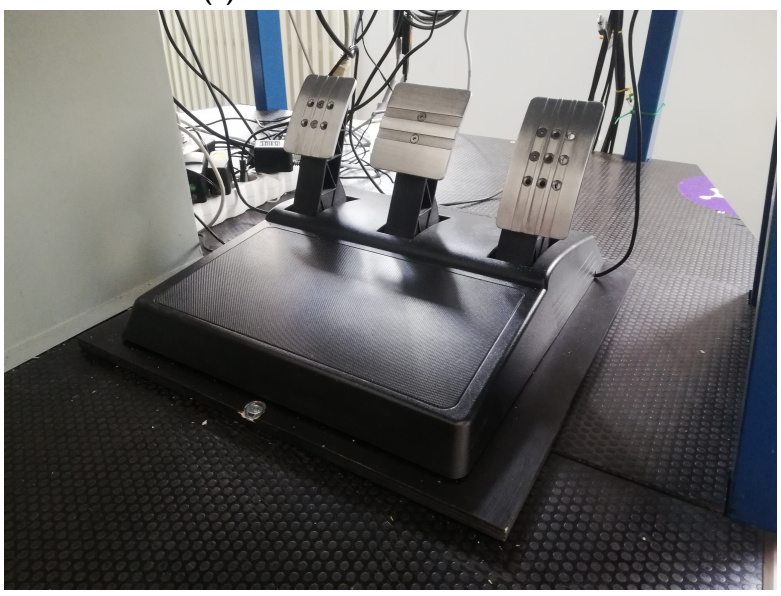
In previous chapters I have discussed importance of driver input devices and thoughts behind my choice of Thrustmaster T300 series. Thrustmaster racing wheel base is usually mounted with included table clamp however, motion platform does not provide any surface to utilize this feature. Hence I proposed cooperation with two CTU FME emplyees, Ing. Tomáš Sommer Ph.D. and Ing. Martin Helmich. Key requirements for the design was minimal conversion (between plane and vehicle peripherals) time and effort. As a result, we came up with modular design custom mounts for both pedals and racing wheel base(Figure 4.3). Material used for production was chipboard.

Driver input signals are read by two different methods, depending on environment you want to capture the data with. For python DI signals I used pygame library API, which locates all input devices plug into computer and is capable of reading operations. Read data can be afterwards transmitted to Simulink, if necessary. On the other hand, if we want to access DI signals from Simulink directly, we can make use of Joystick block from Real Time Simulink library. Joystick block is capable of reading all four DI axes as well as buttons, just like its python implemented counterpart.

4.7 DI Peripheral Emulator

Closing DIL loop is executed by a DI peripheral emulator. Current LFS interface limits signals, which we can modify, namely steering, throttle, brake, handbrake and clutch (DI signals). This severally limits our options in terms of a control law design however, LFS developer with whom I have worked with, was interested in further expansion of a LFS game towards simulation utilization. As a result of these limitations, I have created peripheral emulator using vJoy (Virtual Joystick) API written by Shaul Eizikovich [29]. Emulator is necessary, because we need to read DI signals from real peripherals, shape them in desired manner by control law and then impose them on the LFS game. Connection between Simulink and emulator is done by UDP protocol. Current version of an emulator does not provide FFB functionality, therefore it hinders simulation fidelity. This is definitely place for an improvement.

(a) : Custom mount for wheelbase



(b) : Custom mount for pedals

Figure 4.3: Built custom mounts for driver input devices

Chapter 5

Control Design

5.1 Yaw Damper

The application of simulator and its interface is mainly for control system validation. Thus, I have created simple control design, in order to showcase interconnection capabilities. Yaw damper originated in aircraft industry and it was used to mitigate roll and yaw oscillations during Dutch roll operation. In my case, yaw damper is designed to damp sudden vehicle cornering motion, which would generally lead to uncontrollable vehicle behaviour [30] [31].

Yaw damper utilizes yaw rate OutSim signal to reshape steering driver input for simulation. This leads to feedback control law as shown in Figure 5.1.

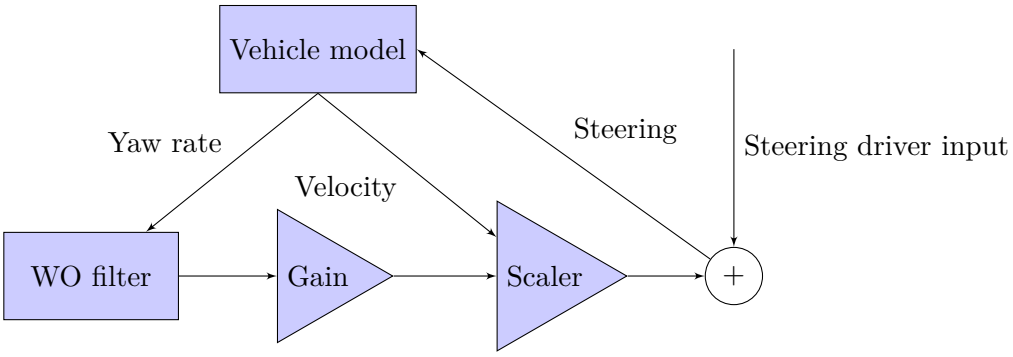


Figure 5.1: Yaw damper control law schematic

Washout filter (WO filter) is a high pass filter, whose function is to detect

quick changes in a signal. Yaw rate signal is filtered with WO filter and thereby loses its low, desirable, driver induced frequencies. Consequently, filtered signal is reshaped by two gain blocks, one of which ensures negative feedback loop and the second creates progressive yaw damping strength based on vehicle velocity, hence faster the car drives, stronger the damping effect gets.

5.2 Control system Experiments

Yaw rate damper underwent three different testing scenarios: simulation of unexpected input error (for example strong sideways wind), moose test and racing laps. Test driver during all tests was myself, car used was FZ 50 GTR. All tests were conducted using motion platform and previously described interface.

5.2.1 Auxiliary Features

Firstly, for a more convenient usage, I have created toggle switch button assigned to one of the steering wheel buttons. It allows to change between directly read or reshaped input signals (by control system) during simulation runtime. Toggle button is based on finite state automaton design.

Secondly, in order to standardize the conditions, I have also created simple cruise control system, which controls throttle pedal to keep constant speed during experiments. Cruise control is regulated with PI controller and it provides zero steady state error, however during certain maneuvers, speed may differ from set point by up to 3 km/h.

5.2.2 Unexpected Input Error Experiment

This experiment is designed as analogy to strong sideways wind blow. In real world scenario, wind would pull the steering towards one side and driver's natural reaction is to counter its movement and optimally carry on in previous driving lane. Steering input is visualized in Figure 5.2. Because of similarity of responses, I have spread both experiments one second apart from one another, so results do not overlap with each other. To properly validate yaw damper effectiveness, steering input error is introduced at same speed for

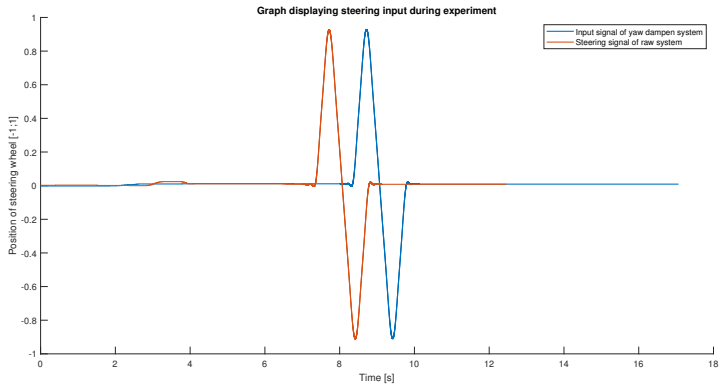


Figure 5.2: Graph of input steering signal in time

both raw and controlled system. Testing speed during this experiment was 150 km/h.

Results show (Figure 5.3), that yaw damper controlled system is capable of containing input error, thus vehicle does not lose its stability. Unlike controlled system, raw system loses its stability around 8.5 sec in the test. Due to crash of the vehicle, raw system test was also ended sooner, explaining the discrepancy in test duration. Furthermore, raw system does not accomplish test scenario even at lower speeds (above 90 km/h), whereas controlled system is capable of handling even more demanding requirements (greater input error, higher maneuver speed).

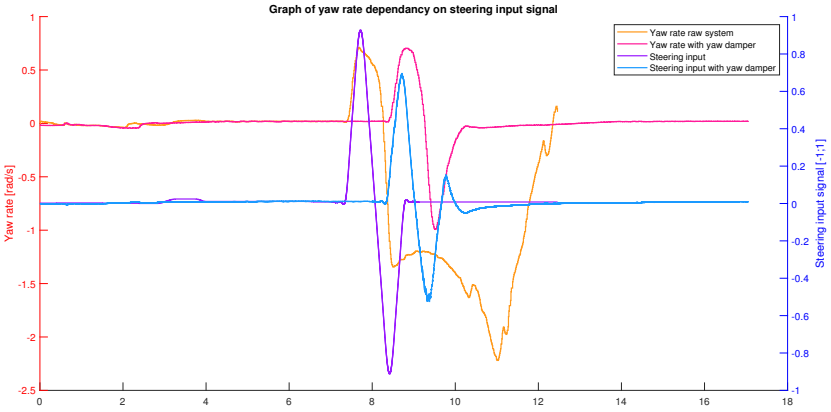


Figure 5.3: Graph displaying vehicle's yaw rate as well as steering input for raw and yaw damper systems

On this particular scenario, input signal filtration did not introduce any measurable delay; however, it may not always be the case. Depending on filter design, we may expect either constant time delay (using FIR filter) or variant time delay with peek at cut-off frequency (using IIR filter).

5.2.3 Moose Test

Moose test is standardized scenario test for vehicles, probing roll stability and maneuverability of the car. Test map(Figure 5.4) used for my experiment was built by my colleague Adam Konopiský and it is based on ISO 3888-2 Passenger Cars [32]. Width of the car used was 2 m, spacing of cones 2.5 m. Maneuver were done at speed of 65 km/h, however unlike in traditional moose test, where throttle is released at the beginning of the first cone set (Figure 5.5), I have maintained speed with cruise control system, hence putting extra emphasis on car steering dynamics at speed.

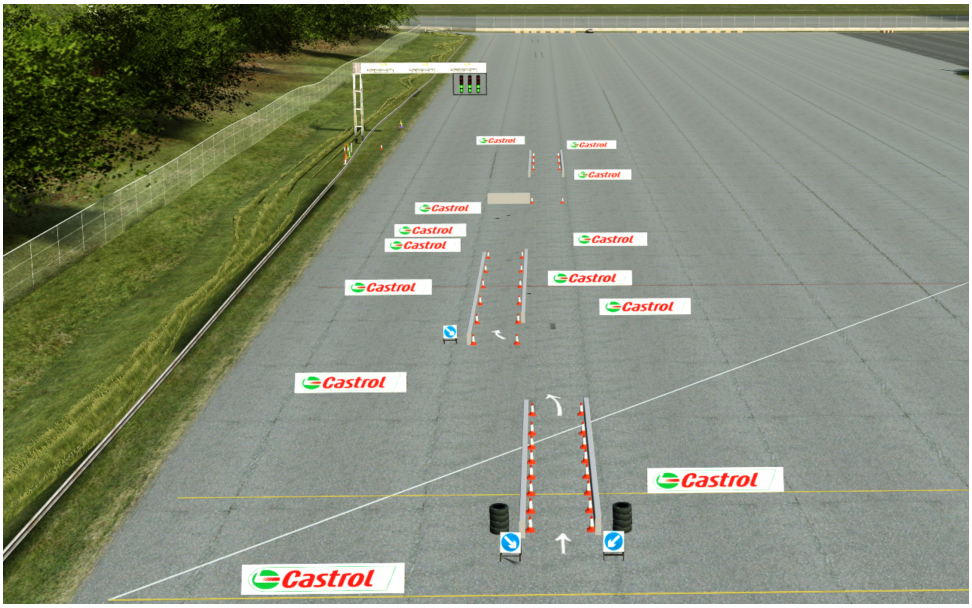


Figure 5.4: Moose test LFS custom map made by Adam Konopiský

My experiment consisted of 25 rides for both raw and yaw dampen systems. Each cone hit is undesirable event (1 negative point), hitting concrete wall is considered a highly dangerous event.

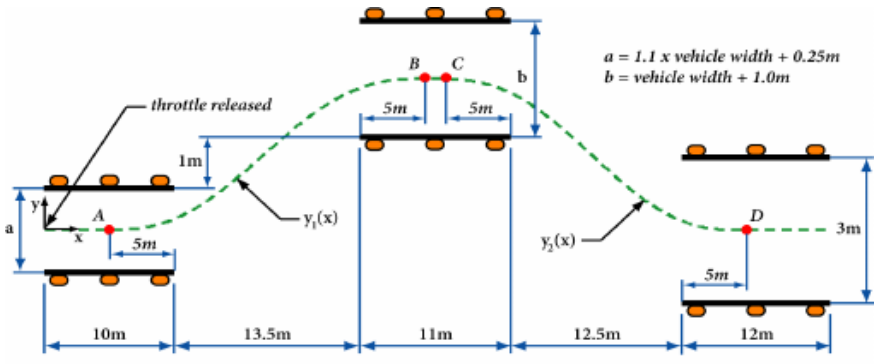


Figure 5.5: ISO standard moose test layout [32]

Both raw and yaw dampened systems had very similar performance, having on average 6.48 and 6.44 points respectively. Despite having almost same average value, there was significant difference in standard deviation, with raw system having value of 4.43 points, whereas yaw dampen system's standard deviation was 2.69 points. Furthermore, driving without yaw damper resulted in total of five crashes with concrete walls; however, while using yaw damper control, I managed to avoid any collision with walls. This may lead to conclusion that, driving with yaw damper is more predictable and consistent. Neonetheless, reliability of the testing method is flawed, having no expert driver to fully prove this statement. I believe that given more time to perfect this maneuver experiment, I would have been able to improve results for both systems, therefore I would call this experiment as inconclusive.

5.2.4 Racing Laps

Lastly, I have tried using a yaw damper to enhance my racing performance. Selected race track was Blackwood, 3.3 km long lap with a dominant flat stretch. Driving a car with the yaw damper felt considerably safer and more reliable, yielding a better performance in every single trial race. An average lap time for raw system was 1:25.25 sec, whereas an average lap time for yaw damper implemented system was 1:22.28 sec.The yaw damper seems to improve lap time by a 3.5 %, giving a clear edge over uncontrolled system. Although I must point out, that as in the previous experiment, the test driver was no professional driver, therefore results may not be conclusive.

Chapter 6

Future Development Options

While working on this bachelor thesis, I have stumbled upon a few potential improvements of the developed interface, which are out of scope of this work due to the lack of time. These options are great opportunities for a future development, and they may considerably enhance this work's value.

Firstly, my interface does not natively support FFB functionality.The FFB, as was mentioned previously, is of a high importance, nonetheless the developed interface solely relies on the proprietary Thrustmaster peripheral application to provide this functionality.

Secondly, there is a lot of room for a User Interface (UI) development. I must admit, the UI was never a priority of this work; however, it is not an excuse for a spartan-looking interface currently implemented. This may potentially lead to problems during future troubleshooting of any upcoming issues.

Next opportunity for an improvement comes from a motion platform. The motion platform safety limits do not allow for exact representation of driver impacting forces, hence I need to finely tune limitations to deliver a realistic experience, while still preserving simulator's safety. With a lack of any experience in this field, I do not believe that I am capable of solving this task on my own.

Lastly, the greatest improvement possibility lies in broadening the cooperation with the LFS game developers. I have already established a line of communication and the developer, with whom I have been working together, was open to further collaboration.

Chapter 7

Conclusions

My bachelor thesis was guided by four following objectives:

- Review of vehicle dynamics simulation software and platforms
- Modification of the existing simulation platform to a vehicle dynamics simulator and its interlinkage with a simulation software
- Vehicle dynamics control system development
- Verification of the developed systems

The first goal was met in the theoretical part of the work, where I briefly listed and introduced various simulation software and hardware options, including motion platforms.

In order to fulfil the second goal, I have chosen specific driver peripherals and worked on designing custom mounts for them. Importantly, I did not construct the mounts myself, as I had been helped by Ing. Tomáš Sommer Ph.D. and Ing. Martin Helmich, both from CTU FME.

Furthermore, I have developed interconnection interface between a simulation software LFS and both Python and MATLAB/Simulink environments. Introduced interface provides live telemetry consisting of 82 signals and allows for a control law designing. Unfortunately, the interface capabilities are hindered, due to an inability to influence any other signals than the driver input.

Subsequently, I have designed two control laws to showcase the interface functionality. Implemented algorithms were a cruise control system and a yaw damper, providing with a set speed control and a increased car stability respectively. The primary purpose of the cruise control system was purposely developed, in order to further control test conditions for the yaw damper probing. Consequently, I have experimented with the yaw damper control system, examining it's behaviour in three test scenarios: Unexpected input error test, Moose test and racing laps. The Moose test map was built by Adam Konopiský and I have only conducted my control law tests on it. The results of the test have shown meaningful improvement in predictability and stability of the test vehicle, suggesting that proposed control law is both valid and effecting. On the other hand, testing conditions and, most notably, unprofessional driver, could negatively impact credibility of the test results.

Bibliography

[1] ResearchGate GmbH, *V model for control system development and installaion*, 2006, Accessed on: 22.2.2020, [Online], Available: https://www.researchgate.net/figure/V-model-for-control-system-development-and-installation_fig5_233670302

[2] ŠKODA Auto a.s., *ŠKODA FABIA MONTE CARLO*, 5.3.2018, Accessed on: 26.3.2020, [Online], Available: https://www.skoda-storyboard.com/cs/0015_fabia_int-2/

[3] Endor AG, *Podium Wheel Base DD2*, 2020, Accessed on: 26.3.2020, [Online], Available: https://fanatec.com/eu-en/racing-wheels-wheel-bases/wheel-bases/podium-wheel-base-dd2

[4] Endor AG, *Podium Steering Wheel R300*, 2020, Accessed on: 26.3.2020, [Online], Available: https://fanatec.com/eu-en/steering-wheels/podium-steering-wheel-r300

[5] Thrustmaster, *T-LCM Pedals* , 2020, Accessed on: 27.3.2020, [Online], Available: http://www.thrustmaster.com/cs_CZ/products/t-lcm-pedals

[6] Simtag, *Simtag Simultor*, 2020, Accessed on: 3.4.bb2020 , [Online], Available: https://simtag.eu/the-products#simtagvoyager

[7] Elsaco s.r.o., *4DOF Motion Simulators*, 2016, Accessed on: 2.4.2020, [Online], Available: https://www.motion-sim.cz/new/?page=home&article=15&menu=79

[8] Wikipedia - The Free Encyklopedia, *Steward Platform*, 5.4.2020, Accessed on: 4.4.2020, [Online], Available: https://en.wikipedia.org/wiki/Stewart_platform

[9] Toyota, *Pursuit for Vehicle Safety*, 2013, Accessed on: 5.4.2020, [Online], Available: http://www.toyota.com.cn/innovation/safety_technology/safety_measurements/driving_simulator.html

[10] J Billington, *BMW builds US$116m simulation center in Munich*, 7.9.2018, Accessed on: 5.4.2020, [Online], Available: https://www.autonomousvehicleinternational.com/news/simulation/bmw.html

[11] AB Dynamics Limited., *aVDS Advanced Vehicle Driving Simulator*, 2020, Accessed on: 3.4.2020, [Online], Available: https://www.abdynamics.com/en/products/lab-testing/driving-simulators/avds

[12] IPG Automotive GmbH, *Keep your eyes on the goal with virtual test driving* , 2020, Accessed on: 11.4.2020, [Online], Available: https://ipg-automotive.com/

[13] IPG Automotive GmbH, *CarMaker: Virtual testing of automobiles and light-duty vehicles* , 2020, Accessed on: 11.4.2020, [Online], Available: https://ipg-automotive.com/products-services/simulation-software/carmaker/

[14] dSpcae GmbH, *dSpace Automotive Industry*, 2020, Accessed on: 16.4.2020, [Online], Available: https://www.dspace.com/en/pub/home/applicationfields/stories.cfm#filterterms=term-133

[15] dSpcae GmbH, *HIL Testing*, 2020, Accessed on: 17.4.2020, [Online], Available: https://www.dspace.com/en/pub/home/products/systems/ecutest.cfm

[16] Mechanical Simulation Corporation, *Vehicle Simulation Products*, 2020, Accessed on: 12.4.2020, [Online], Available: https://www.carsim.com/products/index.php

[17] Mechanical Simulation Corporation, *CarSim*, 2020, Accessed on: 12.4.2020, [Online], Available: https://www.carsim.com/products/carsim/index.php

[18] Hexagon AB, *Adams - The Multibody Dynamics Simulation Solution*, 2020, Accessed on: 11.4.2020 , [Online], Available: https://www.mscsoftware.com/product/adams

[19] Cruden, *Panthera*, 2020, Accessed on: 11.4.2020, [Online], Available: https://www.cruden.com/panthera-software/

[20] Cruden, *Panthera ADAS Tools*, 2020, Accessed on: 11.4.2020, [Online], Available: https://www.cruden.com/automotive-driving-simulators/

[21] Cruden, *Cruden Motion Simulators*, 2020, Accessed on: 11.4.2020, [Online], Available: https://www.cruden.com/hardware/

[22] rFpro, *rFpro*, 2020, Accessed on: 10.4.2020, [Online], Available: http://www.rfpro.com

[23] rFpro, *rFpro - LiDAR Survey*, 2020, Accessed on: 10.4.2020, [Online], Available: http://www.rfpro.com/digital-road-models/lidar-survey/

[24] Thrustmaster, *T 300 GT RS*, 2020, Accessed on: 27.2.2020, [Online], Available: http://www.thrustmaster.com/cs_CZ/products/t300-rs-gt-edition

[25] Samsung, *55″ Certifikovaná Ultra HD Smart TV UE55NU7372 Série 7 (2018)*, 2018, Accessed on: 10.4.2020, [Online], Available: https://www.samsung.com/cz/support/model/UE55NU7372UXXH/

[26] Live for Speed, *OutSim / OutGauge*, 25.12.2010 MediaWiki, Accessed on: 10.8.2019, [Online], Available: https://en.lfsmanual.net/wiki/OutSim_/_OutGauge

[27] Steven M LaValle, *Yaw, pitch, and roll rotations*, Cambridge University Press 2012, Accessed on: 12.8.2019, [Online], Available: http://planning.cs.uiuc.edu/node102.html

[28] S. Roberts, E. Bailey, V. van Vlaardingen, *LFS car info*, 2020, Accessed on: 12.1.2020, [Online],Available: https://www.lfs.net/programmer/carinfo

[29] Shaul Eizikovitch, *vJoy*, 2016, Accessed on: 27.8.2019 ,[Online], Available: http://vjoystick.sourceforge.net/site/

[30] D. Schramm, M. Hiller, and R. Bardini, *Vehicle Dynamics: Modeling and Simulation*, 4th Verlag Berlin Heidelberg: Springer, 2014.

[31] H. B. Pacejka, *Tire and Vehicle Dynamics*, 3rd Delft: Butterworth-Heinemann, 2012.

[32] ISO, *ISO 3888-2 Passenger cars*, 2011, Accessed on: 1.5.2020, [Online], Available: https://www.iso.org/standard/57253.html

[33] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 8th London: Pearson, 2018.

[34] R. B. G. H., *Bosch Automotive Handbook*, 10th USA: SAE International, 2018.

Appendices

Appendix A

Mathematical appendix

Matrix mentioned in Equation 4.1 is listed by indices due to its size.

$$\mathbf{R}_{1,1} = \cos(\alpha)\cos(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) \tag{A.1}$$
$$\mathbf{R}_{1,2} = \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) \tag{A.2}$$
$$\mathbf{R}_{1,3} = \cos(\beta)\sin(\gamma) \tag{A.3}$$
$$\mathbf{R}_{2,1} = \cos(\alpha)\cos(\beta) \tag{A.4}$$
$$\mathbf{R}_{2,2} = \sin(\alpha)\cos(\beta) \tag{A.5}$$
$$\mathbf{R}_{2,3} = -\sin(\beta) \tag{A.6}$$
$$\mathbf{R}_{3,1} = \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\aleph)\sin(\gamma) \tag{A.7}$$
$$\mathbf{R}_{3,2} = \sin(\alpha)\sin(\beta)\cos(\gamma) - \cos(\aleph)\sin(\gamma) \tag{A.8}$$
$$\mathbf{R}_{3,3} = \cos(\beta)\cos(\gamma) \tag{A.9}$$