Universidad del País Vasco
Euskal Herriko Unibertsitatea

FACULTY
OF ENGINEERING
BILBAO
UNIVERSITY
OF THE BASQUE
COUNTRY

Faculty of Engineering Bilbao (ESI)

Department of Automatic Control and Systems Engineering (DISA)

PHD THESIS

Real-Time Multi-Domain Optimization Controller for Multi-Motor Electric Vehicles using Automotive-Suitable Methods and Heterogeneous Embedded Platforms

Author: Martin Dendaluce Jahnke

Director: Vicente Gómez Garay
Co-director: Joshué Pérez Rastelli

May 2019

Dedicated to…

…Sally, my wife, for patiently sharing so many sacrifices over these hard years.

…my parents, for their understanding and willingness to put themselves in second place.

Acknowledgements

I wish to express my gratitude to my wife and parents, for all their support allowing to keep on with my research and conclude this work.

I also want to sincerely appreciate all the great people that have supported me in different ways over the years, particularly (in chronological order): Juan José Valera, Vicente Gómez, Miguel Allende, Eloy Irigoyen, Maider Larburu, Joshué Pérez, and many other fellow researchers and colleagues, mainly at the *Department of Automatic Control and Systems Engineering* at the *University of the Basque Country* and the *Powertrain Team* from the *Automotive Department* at *Tecnalia Research and Innovation*, as well as the leading researchers and department managers from the *Noise and Vibration Research Group* at *KU-Leuven*.

Besides I acknowledge *Tecnalia Research and Innovation* and *Fundacion Iñaki Goenaga* for providing the opportunity to (co)develop my research activities.

I also want to acknowledge in general, the Basque Government, Spanish Government and European Commission, for supporting a context, through their frameworks and funding of diverse projects, that eventually held some relation with certain research activities (see section 1.4 for further details).

Lastly, I acknowledge in particular the following projects and organizations for supporting in different ways some of the activities related to this work:

- EUNICE which has received funding under 7th RTD Framework Programme, Research & Innovation Directorate-General (DG RTD) (FP7-2011-GC-electromechanical-storage) under grant agreement n°285688.
- 3Ccar project, which has received funding under ECSEL Joint Undertaking under grant agreement No. 662192.
- ENABLES3, which received funding under ECSEL Joint Undertaking under grant agreement No. 692455-2.

Abstract

In this Thesis, an elaborate control solution combining Machine Learning and Soft Computing techniques has been developed, targeting a challenging vehicle dynamics application aiming to optimize the torque distribution across the wheels with four independent electric motors.

The technological context that has motivated this research brings together potential -and challenges- from multiple domains: new automotive powertrain topologies with increased degrees of freedom and controllability, which can be approached with innovative Machine Learning algorithm concepts, being implementable by exploiting the computational capacity of modern heterogeneous embedded platforms and automated toolchains. The complex relations among these three domains that enable the potential for great enhancements, do contrast with the fourth domain in this context: challenging constraints brought by industrial aspects and safety regulations.

The innovative control architecture that has been conceived combines Neural Networks as Virtual Sensor for unmeasurable forces, with a multi-objective optimization function driven by Fuzzy Logic, which defines priorities basing on the real-time driving situation. The fundamental principle is to enhance vehicle dynamics by implementing a Torque Vectoring controller that prevents wheel slip using the inputs provided by the Neural Network. Complementary optimization objectives are efficiency, thermal stress and smoothness. Safety-critical concerns are addressed through architectural and functional measures.

Two main phases can be identified across the activities and milestones achieved in this work. In a first phase, a baseline Torque Vectoring controller was implemented on an embedded platform and -benefiting from a seamless transition using Hardware-in-the-Loop- it was integrated into a real Motor-in-Wheel vehicle for race track tests. Having validated the concept, framework, methodology and models, a second simulation-based phase proceeds to develop the more sophisticated controller, targeting a more capable vehicle, leading to the final solution of this work. Besides, this concept was further evolved to support a joint research work which lead to outstanding FPGA and GPU based embedded implementations of Neural Networks.

Ultimately, the different building blocks that compose this work have shown results that have met or exceeded the expectations, both on technical and conceptual level. The highly non-linear multi-variable (and multi-objective) control problem was tackled. Neural Network estimations are accurate, performance metrics in general -and vehicle dynamics and efficiency in particular- are clearly improved, Fuzzy Logic and optimization behave as expected, and efficient embedded implementation is shown to be viable. Consequently, the proposed control concept -and the surrounding solutions and enablers- have proven their qualities in what respects to functionality, performance, implementability and industry suitability.

The most relevant contributions to be highlighted are firstly each of the algorithms and functions that are implemented in the controller solutions and, ultimately, the whole control concept itself with the architectural approaches it involves. Besides multiple enablers which are exploitable for future work have been provided, as well as an illustrative insight into the intricacies of a vivid technological context, showcasing how they can be harmonized. Furthermore, multiple international activities in both academic and professional contexts -which have provided enrichment as well as acknowledgement, for this work-, have led to several publications, two high-impact journal papers and collateral work products of diverse nature.

Table of contents

Table of figures

Table of tables

List of abbreviations

4WD: Four Wheel Drive
ABS: Antilock Braking System
ADAS: Advanced Driver Assistance System
ADC: Analog to Digital Conversion
AEC: Automotive Electronics Council
ANN: Artificial Neural Network
ASIC: Application-Specific Integrated Circuit
ASIL: Automotive Safety Integrity Level
ASPICE: Automotive SPICE
AUTOSAR: Automotive Open System Architecture
BSW: Basic Software
CAN: Controller Area Network
CRC: Cyclical Redundancy Checking
DBC: Database CAN
DiL: Driver-in-the-Loop
DLL: Dynamic Link Library
DNN: Deep Neural Network
DOF: Degree Of Freedom
DSP: Digital Signal Processing
ECC: Error Correction Code
ECU: Electronic Control Unit
ESP: Electronic Stability Control
FL: Front Left
FLOP: Floating-point Operations per Second
FMEA: Failure Mode and Effects Analysis
FPGA: Field Programmable Gate Array
FR: Front Right
FTA: Fault Tree Analysis
GB-RAM: Gigabyte Random Access Memory
GFLOP: Giga Floating-point Operations per Second
GHz: Gigahertz
GPGPU: General Purpose Graphics Processing Unit
GPIO: General Purpose Input Output
GPS: Global Positioning System
GPU: Graphics Processing Unit
GUI: Graphical User Interface
HAL: Hardware Abstraction Layer
HARA: Hazard Analysis and Risk Assessment
HDL: Visual Hardware Description Language
HiL: Hardware-in-the-Loop
HLS: High Levels Synthesis
HMI: Human Machine Interface

HP: Horse Power
ICE: Internal Combustion Engine
IEC: International Electrotechnical Commission
IMU: Inertial Measurement Unit
IS: International Standard
ISO: International Standard Organization
LE: Logical Element
LSTM: Long Short Term Memory
LUT: Look-up-Table
MAE: Mean Average Error
MBD: Model Based Design/Development
MCAL: Microcontroller Abstraction Layer
MFLOP: Mega Floating-point Operations per Second
MHz: Megahertz
MiL: Model-in-the-Loop
MIMO: Multiple Input Multiple Output
MISO: Multiple Input Single Output
MISRA: Motor Industry Software Reliability Association
MiW: Motor-in-Wheel
MOGA: Multiobjective Genetic Algorithm
MPSoC: Multi-Processor System-on-Chip
MPU: Memory Protection Unit
MSE: Mean Squared Error
NN: Neural Network
PWM: Pulse Width Modulation
QM: Quality Management
RF: Radio Frequency
RL: Rear Left
RNN: Recurrent Neural Networks
RR: Rear Right
RTE: Runtime Environment
RTOS: Real-Time Operating System
SDK: Software Development Kit
SoC: System on Chip
SPICE: Software Process Improvement and Capability Determination
SSD: Solid State Drive
SVM: Support Vector Machine
TCS: Traction Control System
TFLOP: Tera Floating-point Operations per Second
UDP: User Datagram Protocol
V2V: Vehicle to Vehicle
V2X: Vehicle to X
VDA: Verband Der Automobilindustrie (German Automobile Industry Association)
VHDL: Visual Hardware Description Language
vHiL: Virtual-Hardware-in-the-Loop

CHAPTER 1:

Introduction

"A goal is a dream with a deadline"

N. Hill

This first chapter starts by providing a conceptual overview, firstly discussing the engineering fields that are involved in the notably multi-disciplinary character of this work, and secondly anticipating a first insight into the technological context. This leads to the motivation and hypothesis, which consequently lead to the objectives and scope.

This chapter also presents the academic and professional context of the doctoral thesis, besides the planning of international stays and other formative activities. Finally, the structure of the document is presented, illustrating the thread that connects the subsections throughout the entire document.

1. Introduction

1.1. Introduction and technological context

1.1.1. Introduction into the research work

Automobiles and embedded technologies are fields of different nature, but close relations can be identified when discussing many of the involved topics. Some of these relations certainly arise when working with algorithms for automotive systems, thus also involving the field of control theory and software engineering.

The presented research work is mainly focused in the overlapping area of four engineering fields, each of which cover several topics that currently are subject to technological changes of notable relevance. This multi-disciplinary area is illustrated in a simplified manner by Figure 1-1. Besides well-known relations and dependencies among these fields, under deeper analysis, a series of seemingly independent topics also turn out to involve relevant multilateral implications connecting them with each other. On the upside, some of these implications present innovation potential that could be harnessed by exploiting the corresponding novel technologies. On the downside, some other implications inevitably present notable challenges and constraints to overcome. The complexity of these topics multiplies when their relations need to be carefully studied. This requires exhaustive analysis if an added value design, which furthermore is intended to be capable of evolving towards an industry-suitable solution, is pursued. This is the case for this dissertation.

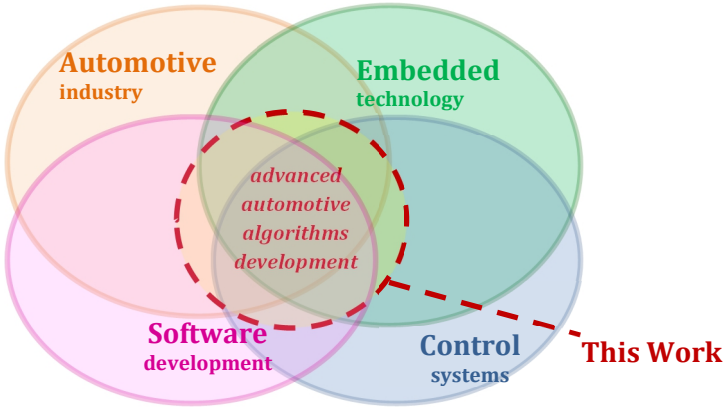

Figure 1-1. Simplified representation of the main fields involved in this research work

The fundamental points of the technological context of the aforementioned fields are briefly highlighted in the following subsection 1.1.2. These lead to the motivation and hypothesis discussed in the section 1.2 and the subsequent objectives in section 1.3.

1.1.2. Technological context

As vehicle electrification moves forward, new powertrain topologies are appearing and attractive research and innovation opportunities for developing enhanced propulsion systems can be been identified. The potential of the increased degrees of freedom and controllability of powertrains consisting of multiple electric motors can be conveniently addressed by exploiting several enabler technologies. Modern high performance heterogeneous embedded platforms can be used to implement complex algorithms, including Machine Learning and other Soft Computing algorithms. Although such advanced controller designs can be greatly supported by the means of the continuously improving model-based development solutions, major challenges appear not only on the notably complex technical layers, but also in the regulatory layer addressing safety requirements for safety-critical applications. These key points can be summarized as seen in Figure 1-2 and are introduced in the following paragraphs.

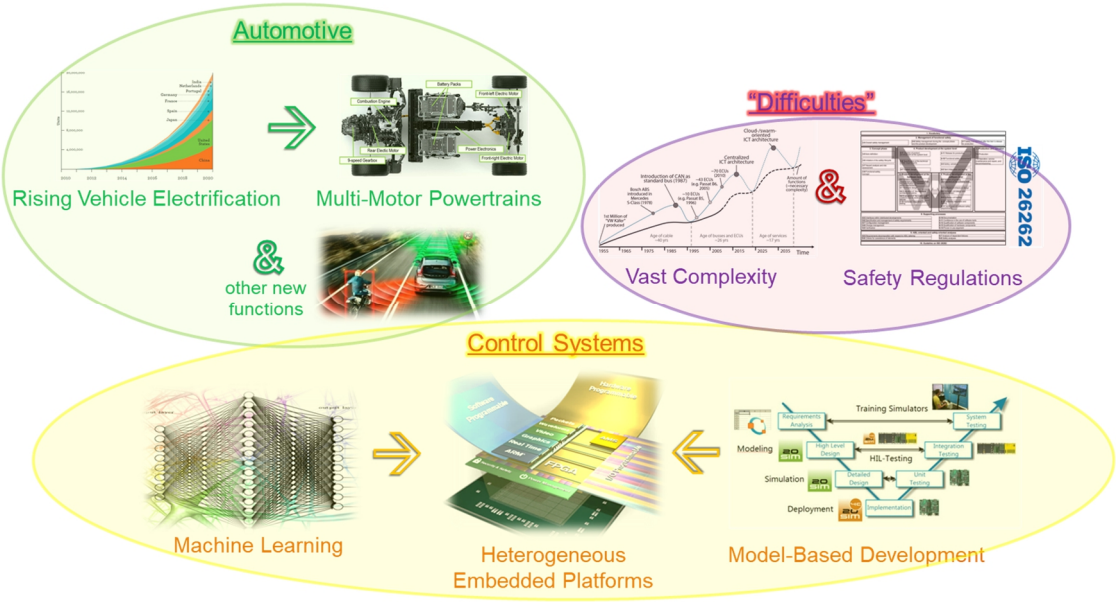

Figure 1-2. Overview over key topics [1][2][3][4][5][6][7][8][9][10][11]

Electrified vehicles -i.e. hybrid and pure electric vehicles- are showing a steadily growing sales trend. This is not only pushed by environmental, geo-political and economic considerations which have led to stringent regulations and certain scandals. It is also driven by improving acceptance rates as technology is evolving to provide better capabilities at more affordable costs. Some electrified powertrain topologies are evolving towards multi-motor configurations, beyond conventional hybrids with one motor of each kind. Powertrains with multiple electric motors and particularly with independent motors for each of the wheels on a same axis are appearing, enabling to implement advanced torque vectoring algorithms. This means, that a controller can apply additional torque to individual wheels aiming not only to enhance cornering capabilities, but also to provide additional safety and stability.

Relevant changes are also to be observed in the field of vehicle control systems in general, besides the powertrain domain. As the number of components and functions in modern vehicles increases, the complexity and derived issues in such systems are reaching

hardly sustainable levels, with over 100 million lines of source code distributed on board among over 100 ECUs (electronic control units). Besides vehicles with multiple motors and energy sources, as well as diverse comfort features, ADAS (advanced drivers assistance systems) and automated driving are strongly contributing to this issue. The technical challenge is further increased by several industry-related constraints. Firstly, some well-known aspects inherent to the big-scale vehicle industry need to be considered, such as strong cost sensitivity and diverse requirements pushing towards modularity, combined with accelerated product cycles. Secondly a major constraint is being added to this already challenging context: increasing impact from safety standards and regulations addressing safety critical systems.

One of the technological enablers with the potential to play a supporting role in the discussed context are the new generation of high-performance embedded platforms. A diversity of heterogeneous platforms offering vast computational power is appearing on the market, including multiple cores, FPGA logic and GPU units, with a computing power gain of at least one order of magnitude. These platforms are rapidly approaching towards industry-suitable levels of cost and robustness, meaning that they could eventually be exploited to cover points from both previous paragraphs. They appear as a suitable solution to implement complex algorithms for enhanced control. Furthermore, they enable to integrate several control functions into a single unit, thus reducing the ECU count and some of the associated complexity and costs.

Another enabler which can be used to cover several of the previous points is the consolidation of model-based development methodologies. This includes automated toolchains providing hardware abstraction through automatic code-generation and hardware synthesis, which can be helpful to tackle the complexity of modern embedded platforms. It also includes automated testing (from model-in-the-loop, MiL, to hardware-in-the-loop, HiL) and support for requirements traceability and documentation, facilitating to handle standards and regulations which typically involve diverse certification processes. Such tools can be very conveniently aligned with the V-model throughout the entire development process, up to the point of reaching industrialization. And, in the same way they provide valuable agility and flexibility during early development and research, they also can provide important modularity and reusability for final product development.

The final point which presents attractive applicability are advanced algorithms in general and Machine Learning in particular. Although the origins of most Machine-Learning and Soft Computing techniques are not recent, the performance and intrinsic parallelism of some of the mentioned embedded platforms enables real-time implementations of computationally demanding algorithms. These circumstances fit very conveniently to develop enhanced control solutions targeting the faster times constants, increased degrees of freedom and better controllability of the multi-electric-motor powertrains mentioned at the beginning of this subsection.

The discussed topics are the trigger for the *Motivation and Hypothesis* discussed in the following section 1.2. Besides the related overview in Figure 1-2, these topics are represented in a structured manner Figure 1-3-part1 (in blue: technology and opportunities; in purple: constraints). Each of these technological topics will be subject to a detailed discussion in chapter 2, *State of the Art*, including extensive referencing.

1.2. Motivation and hypothesis

Following the technological context highlighted in the previous subsection 1.1.2, several hot topics related to novel technologies and potential applications with remarkable research and innovation perspectives have been identified. Each of the topics does already present considerable research potential and notable challenges by itself, but beyond that, the complex relations interleaving the topics from different fields bring challenges of even greater complexity. While some of the points imply dependencies and act as enablers (e.g. powerful embedded platforms enabling sophisticated control solutions), some others impose restrictions (e.g. industrial cost and safety constraints). This scenario not only motivates, but even requires, to take a transversal approach harmonizing the different research and engineering disciplines and technical domains in a consistent research project of major proportions, in order to cover the broadness of the spectrum being addressed.

The discussed complex scenario and the underlying motivations will hereby be articulated in a concise schematic manner in the following points. It is also illustrated by Figure 1-3 where the said structured points are further supported by visual aids:

- Blue: technological aspects.
- Purple: aspects which are bound to imply constraints.
- Green: points with major research and innovation potential, which will shape the objectives and contribution of this work.
- Text next to shapes: related keyworks.

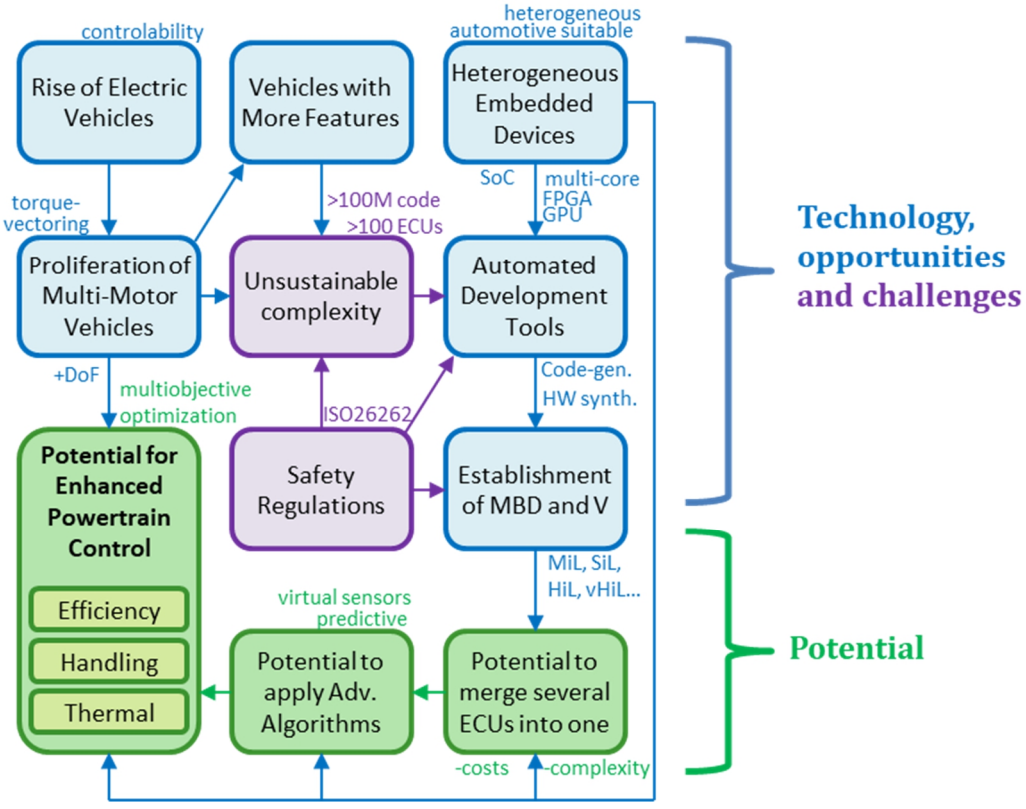

Figure 1-3. Key points of the thesis illustrating connections between concepts, technologies, opportunities, challenges and research potential

1) **Electric Vehicles** are strongly entering the market and **multi-electric-motor vehicles** are appearing as well. Such powertrains provide several relevant points offering significant improvement potential in different physical domains, driven by the additional degrees of freedom and controllability:
    - Potential 1.1: **Vehicle dynamics**: applying torque-vectoring techniques enables enhanced cornering performance and safety in critical situations.
    - Potential 1.2: **Energy efficiency**: optimizing the workload among different motors depending on their operating points can bring energy savings.
    - Potential 1.3: **Thermal stress** can be distributed and furthermore optimized as well, thus potentially minimizing transient limitations.
    - The resulting control problem is a very complex multivariable and multi-objective optimization problem. Furthermore, more refined algorithms might benefit from signals which are not measurable with conventional vehicle sensors, which would need to be estimated somehow.

    → Bottomline Potential: **applicability of Soft Computing and Machine Learning** techniques for several of the algorithmic challenges for the powertrain domain.

2) The new generation of high-performance **heterogeneous embedded platforms** is providing new levels of parallelism and at least one order of magnitude higher performance, while being increasingly suitable for vehicle industry applications.
    - Potential 2.1: Implement highly demanding **advanced algorithms** to satisfy the application potential from point 1, including Machine Learning for real-time variable estimation, prediction and even iterative optimization.
    - Potential 2.2: Integrate **several ECUs into a single** ECU, for architectural and interfacing complexity reduction, and for better variable availability for controllers by reducing bottlenecks.

3) Challenges and constraints for automotive software development are growing due to a combination of factors, where two main points consequently derive in two further factors:
    - Challenge 1: **Growing complexity of the vehicles** as a result of multi-motor powertrains and other features, such as ADAS, automated driving and other functions.
    - Challenge 2: Increased **regulatory pressure** for the development of safety-critical functions, which is also applicable to the discussed powertrain control functions.
    - Consequence 1: **MBD** is being consolidated as engineering solution aligned with the **V development methodology**. This is supported by a series of improving **toolchains with high automation and abstraction**. Besides facilitating to tackle these challenges and constraints, such tools also simplify the complex implementation tasks when working with advanced algorithms (point 1) and modern platforms (point 2).
    - Consequence 2: the **need for a complexity reduction** by implementing architectural changes of the automotive systems arises, facilitated by Consequence 1 and already mentioned in Potential 2.2.

The motivation that finally crystalizes is to conceive and implement a sophisticated solution which is capable to exploit the remarkable innovation potential of several novel technologies, maximizing the added value and real-world applicability by exploiting the relations among enabler technologies while simultaneously tackling challenging technical and industrial constraints.

The hypothesis is that this highly complex constellation, if the enabler technologies are correctly exploited and implemented, can lead to introduce highly capable Machine Learning algorithms into a field which presents notable resistance to such concepts: automotive applications of safety critical character.

The objectives concluded from this argumentation in the following section 1.3 will reflect the above discussed points, leading to a solution which as illustrated in is…

1) …constructed on 4 pillars:

   - Target multi motor electric powertrains, which represent a multivariable multi-objective multidomain optimization problem, addressable with torque vectoring.
   - Research applicability of Machine Learning methods for real-time estimations, predictions and optimization
   - Exploit new-generation of industry-suitable heterogeneous embedded platforms providing great computing power and parallelism
   - Benefit from Abstraction and automatization linked to MBD and V development methodologies provided by modern automotive-suitable tools

2) …subject to major constraints considering industry suitability, including costs, safety, regulations, methods and platforms.

3) …aiming to provide an innovative added-value contribution and baseline for future work.

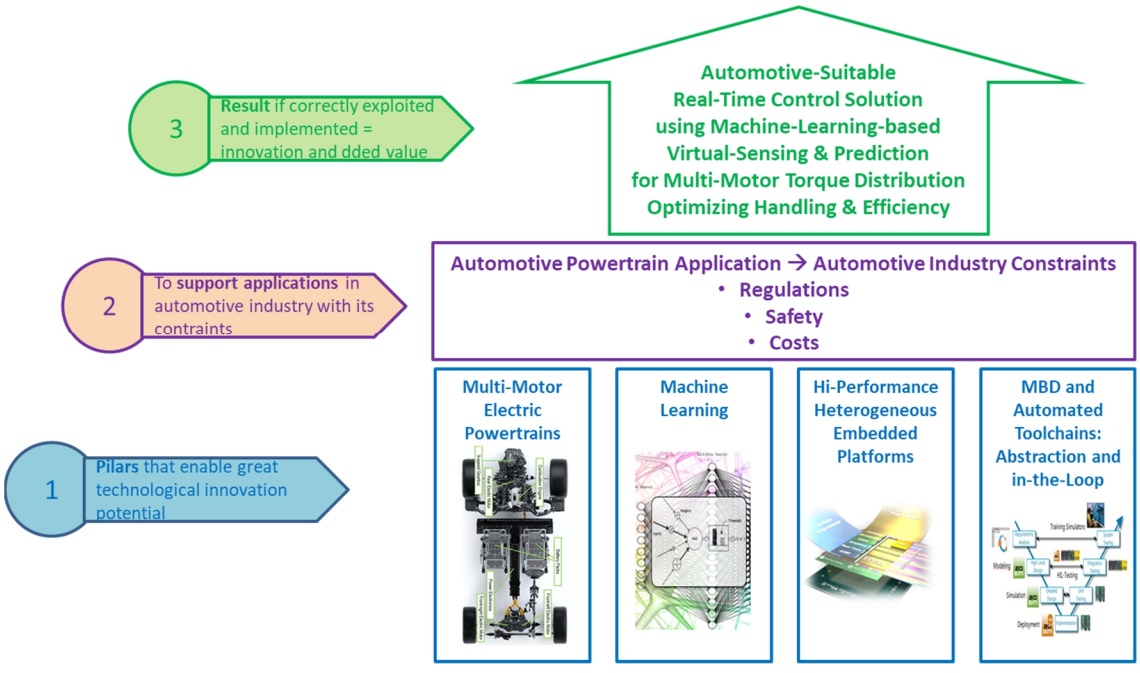

Figure 1-4. Four main pillars of the thesis supporting under the constraints that need to be overcome to reach the objectives [2][6][7][8][9][10][11]

1.3. Objectives and scope

Following the above argumentation leading to the motivation and hypothesis, the main underlying objective of this dissertation is to contribute to the progress of advanced control systems in the field automotive applications, by demonstrating not only the suitability, but also the benefits and enhancement capabilities, of applying a series of novel and non-established technologies in the automotive sector. The presented innovations, illustrated in the pillar of the previous Figure 1-4, are targeting next generation of multi-motor electric vehicles by exploiting the potential of heterogeneous embedded platforms to implement Machine Learning algorithms, supported by modern model based approaches and code-generation solutions.

Therefore the specific **objective** is to develop a control solution exploiting the abovementioned technologies to obtain an automotive-suitable integrated control system solution for real-time multi-domain optimization of a multi-motor electric powertrain, primarily enhancing its dynamic handling and energy efficiency, while also considering thermal limitations.

The underlying work can be defined through a series of tasks, which determine the **scope** of the activities to be carried out:

- Conceive a control solution for the multi-domain optimization problem.
- Conceive a Highly Integrated ECU architecture for the Powertrain Control.
- Establish a MBD framework aligned with the V Development Methodology.
- Elaborate a model of the vehicle as a system to be controlled.
- Validate the essential parts of the model with respect to real systems.
- Select and integrate Machine Learning methods, considering of safety criticality
- Test and tune the solution using the MBD framework.
- Exploit the modern toolchains for efficient development and portability.
- Provide implementation solutions on new heterogeneous embedded platforms.

The main benefits expected of the solution to be developed are the following:

- Enhanced vehicle dynamics (performance, comfort and safety).
- Enhanced energy efficiency.
- Reduced performance limitations related to thermal aspects.
- Simplification of vehicle architecture.
- Cost reduction of the control system (production and development).
- Enhanced development, testing and validation workflow agility and effort.

1.4. Context and Planning of the research

This research work has been conducted dedicating major care and effort to consistently adopt good practices and methodological approaches from both the scientific and engineering perspectives.

The most important points are discussed in the upcoming subsections, being the bottomline that the backbone of the research line was consolidated over the years, with solid foundations in the form of corresponding publications and with the additional support of other activities, providing coverage for the core contents. While the primary academic path was centred at the UPV/EHU, multiple international research activities were carried out, of both academic and professional nature. These have reflected their influence in certain aspects of the work and offered multiple opportunities which were positively exploited, but also imposed challenging boundary conditions. The intricacies of these intense but fruitful multilateral activities and their relations are also discussed below.

1.4.1. Development of the PhD and its academic framework

The presented research work has been carried out in a multilateral context involving the ***Faculty of Engineering of Bilbao* (*ESI*)** at *the* ***University of the Basque Country* (*UPV-EHU*)**. In particular, the conduction of the dissertation, as well as a major part of the academic education, was located under the umbrella of the ***Department of Automatic Control and Systems Engineering* (*DISA*)**. This department addresses the domains of Automation, Process Control, Robotics, Industrial Computer Science and Instrumentation for Control Systems.

One of the characteristic lines of this department can be said to be focusing on formal aspects of modelling and control theory of diverse advanced techniques, alongside with hard-real-time systems and embedded platforms. These topics certainly are of notable relevance in relation to the presented dissertation and are also present in the Engineering and Master of Science courses that the department offers.

The author had previously obtained his second engineering degree, followed by the Master of Science degree in *Control Engineering, Automation and Robotics* of the *DISA* department. The contents of these courses are connected to the ***Intelligent Control Research Group* (*GICI*)**, where the research activities were conducted in different phases, starting from early collaborations in 2009 up to the development of this dissertation. This research group is focuses on intelligent control and Soft Computing techniques for multiple applications and is closely linked to the specialization fields of the mentioned Master course.

Alongside with the previous activities at the *DISA* department and the *GICI* research group, the author was recruited by the private research corporation ***Tecnalia Research and Innovation***, particularly its ***Automotive Department*** of its ***Industry and Transport*** division. Activities at both institutions concurrently developed through different phases, from cooperating student to PhD candidate and ultimately a permanent contract. Further aspects related to this are discussed in subsection 1.4.2.

It must be highlighted, that the attendance to the Master of Science course over two years and the following PhD candidate position and salary were funded by a grant given by ***Fundación Iñaki Goenaga***, which was linked to the concurrent activities at *Tecnalia.*

The academic scope of the research has been additionally extended with two long stays at European universities, attending lessons and working on research activities at the ***Automotive Technology Department*** **(*FZD*)** of the ***TU-Darmstadt*** (Germany) and performing an international PhD researcher stay at the ***Department of Mechanical Engineering*** at ***KU-Leuven*** (Belgium). Further aspects related to this are discussed in subsection 1.4.3

The coordination and management of doctoral programmes and interdisciplinary research training, as well as the official masters, is handled by the ***Master and Doctoral School*** **(*MDE*)** of the *UPV/EHU*. This central institution establishes the guidelines and requirements for the research activities, acknowledgements and fundamental dissertation-related aspects.

Alongside other criteria, the established dissertation development methodology of the *MDE* requires to periodically report the progress of the activities with respect to the research plan proposed by the researcher, including publications, merits, updates and deviations. These are reviewed by the dissertation director and co-director in a first instance and then evaluated by the academic commission in a second instance. This is required to permit the continuation of the research work.

1.4.2. Concurrent professional context

As briefly mentioned above, the author got engaged in a research constellation involving educational and research activities at universities alongside with professional work at *Tecnalia Research and Innovation*.

Tecnalia is a technology corporation initially established in 2001, which has grown as a strategic alliance combining multiple research centres, currently accounting to a headcount of roughly 1500 people.

In 2009 the author got involved in this organization by reference, firstly in cooperating and working student phases over two years in total. Then he was given a PhD candidate status under the umbrella of the *Iñaki Goenaga* grant, which required to concurrently attend the master course and research and development work at *Tecnalia*. Eventually the PhD candidate program transitioned to a normal engineering contract, due to a diversity of factors.

The work carried out at *Tecnalia* involved a broad diversity of research and development activities in different contexts, ranging from purely academic work to engineering projects for the industry. The reason behind this is that this corporation is dedicated to projects of different nature, coming from both clients from the industry as well as research projects with mostly competitive public funding. The funding split between industry and public funding roughly account for one half each. The research projects range across funded European, national and regional projects.

These projects were mostly under the scope of the *Electronics and Control Systems* area and particularly in the *Motor Control and Power Electronics* group, the *Simulation and HiL test systems* group, and predominantly the *Powertrain Control Group.* Additional cooperation was also held with the *Automated Driving* group of the same area as well as with the *Safety* group from the *Software Systems* department.

Considering the corporate character and the combination of funded research with industry-oriented engineering tasks at *Tecnalia*, the bilateral academic-professional playground could eventually be perceived as challenging to harmonize at certain points. In practice, although this constellation has proven to be very demanding respecting to balancing concurrent projects leading to high workloads, it has provided exceptionally enriching knowledge acquisition and productivity at an accelerated pace.

The following bullets collect some of the most relevant projects, roughly grouped by the topics they covered, although some partly extend their scope across multiple domains. Most of them involved different embedded implementation aspects, and while some used more conservative control approaches, other more innovative research projects were directed towards more complex and advanced solutions.

- Control and algorithms for electric powertrains, closer to power electronics:
  E-Vito, IEB, BETRACTION, ECUPOW, ECUFAST, SYRNEMO, FPGA-MC, COMA.
- Control and algorithms for alternative and hybrid powertrains:
  AUTOSHIFT2, SHIFT2. EUNICE, VEMTESU, SPAIN2017, ECOCHAMPS, KT4ETRANS, VEMTESU, ECAT2, 3Ccar.
- Vehicle dynamics and modelling:
  Winter Testing, Observauto.
- Safety critical aspects and related topics:
  SafeAdapt, ENABLE-S3.
- Automated driving:
  AutoDrive.
- Other projects:
  IOSENSE.

Additional details and merits related to the most notable and relevant project, 3Ccar, are provided in subsection 7.7.

1.4.3. Planning of international stays

Placing the focus back on the academic and scientific side of research, two major international stays were set up in the context of this work. Both activities were calibrated to gain additional knowledge in selected fields by choosing not only the corresponding courses and activities, but also by targeting highly acknowledged universities and departments. The following subsections illustrate these aspects, the selection criteria followed and some information about the universities.

The results for both activities successfully fulfilled the plans. Further details regarding the outcomes of both stays, as well as information about the high ranking of these institutions, are provided in section 7.1.

1.4.3.1.TU-Darmstadt

For the purpose of deepening the knowledge in selected fields and engaging in research-oriented activities a long academic stay at a prestigious German university was planned at the beginning of the research activities. Besides the control system development topics, it was intended to place the emphasis on the automotive domain. Finally the *TU-Darmstadt* was selected and the author was accepted as a Master student to attend lessons and other activities. The stay was programmed over two entire semesters (2012-2013), firstly at the Electronics department and secondly at the Mechatronics department.

The *TU-Darmstadt* is one of the most notable German universities, particularly in the field of Mechanics engineering, where the *Vehicle Technology Department* (*FZD*) is of special relevance, and of special interest to the author. It is ranked top 3 in Germany for the field of mechanics and around the edge of top 100 in the World. [12]

Furthermore, this was intended to be combined with the participation in a so-called *Research Seminar*. Here the author chose to deepen into an exhaustive research work of notable formal character about different kinds of Machine Learning techniques and their applicability in the automotive domain.

This work was planned to extend over the entire period of one semester under the tutoring of a scientific worker of the department, involving regular review meetings (typically bi-weekly). Ultimately the work culminated with a presentation and the evaluation of the head of the *Automotive Technology* department, Prof. Dr. Hermann Winner.

Further details about the positive outcome in general, and the contents of this research work in particular, are provided in section 7.

1.4.3.2.KU-Leuven

For the purpose of achieving the international recognition status of the PhD while simultaneously enriching its scientific contents, a European stay at a University as a PhD researcher was planned for 2017, aiming to find a place at some prestigious institution. Eventually the list was narrowed down to two preferred options*: RWTH-Aachen* and *KU-Leuven*.

As a brief comparison of both universities, it can be said that while *RWTH Aachen* stands out in the field of industrial and mechanical engineering, with strong bounds to the automotive industry, while *KU-Leuven* offers attractive research activities in the fields related to Virtual Sensing on mechanical systems and Machine Learning. *KU-Leuven* can be seen as a more balanced and solid research institution for the focus of this particular work, placed at the top (50 of the World, 15 of Europe) of the typically known university rankings, as can be seen in subsection 7.4 and Figure 7-3.

Besides the recognition of the university itself, the most important part remains to align the research interests. At *KU-Leuven* the *Noise and Vibration Research Group* was through previous professional contact with its research manager Dr. Bert Pluymers. Common ground was found because this group had already been placing Virtual Sensing as one of their main focuses over the years, also addressing parallel computing challenges.

Furthermore, Dr. Francesco Cosco, with extensive expertise in GPU programming, was leading a research team around the modelling and simulation of complex mechanical systems. Consequently, under supervision the group's head Prof. Dr. Wim Desmet, an arrangement was made to join Dr. Cosco's research team by sharing his office. Driven by the notable mutual interest, the cooperation was certainly intense and fruitful, ultimately leading to a high-impact journal article.

Further details about this positive outcome, and the contents of this research work in particular, are provided in chapter 7.

1.4.4. Other formative activities

Besides the previously discussed major academic and professional activities, the author aimed to extend his interdisciplinary knowledge by attending a diversity of courses, workshops, trainings, etc.

Following, a shortened list of the most relevant points is provided, also illustrating the steady interest in fundamental topics involved in this work, mainly complex control systems, Machine Learning, embedded systems and automotive topics.

- 2008: Summer Course on Artificial Intelligence at *University of Cantabria*.
- 2010: Summer Course on Wind Turbine Control (2 Weeks, 5 ECTS) at the *University of Århus* in Denmark.
- 2010-2017: Diverse Training and Workshops with duration ranging between 1 day and 1 week: ISO-26262, AUTOSAR, TargetLink, Simulink™ C- and HDL-Generation, MCAL implementation, Model-Based-Development, amongst others. Provided by dSpace, MathWorks®, Intecs, Medini, ArcCore, amongst others.
- Since 2016: Multiple driving trainings and experience for vehicle dynamics and vehicle prototype driving.

1.5. Contributions of the work

The following list summarizes the most relevant contributions of this work, aiming to provide an overview over the different key points together with a short description, from the author's perspective. More detailed information regarding contributions and merits can be found in the concluding chapters 6 and 7, alongside with the publications and cross references. While the actual core of this work is reflected in the first bullet point, multiple outcomes, which can be exploited for related and future research activities, are highlighted as well.

- **Novel control concept application:** this work focuses on the conception, development and validation of an innovative control concept of notable complexity, exploiting cutting edge technology to tackle challenging multivariable vehicle dynamics challenges. The usage of Neural Networks as Virtual Sensors is one of its highlights. The design as a whole is proven valid to enhance stability, efficiency and thermal load. Consequently, the outcome (concepts, solutions and resources) can be used as baseline and enablers for future research.
- **Illustration of multidisciplinary technological context:** the attractive technologies arising across widespread fields of engineering, which triggered this research work, are carefully explained for a solid justification of this work, as well as to provide a better understanding for the readers.
- **Adapted to constraints from multiple domains:** the approaches and technical solutions implemented in this work prove their applicability in spite of being subject to the challenging constraints. The main restrictions come from the implications of the automotive industry, which require a more formal development process, less complex solutions and less powerful embedded platforms.
- **Implementation on different embedded platforms**: besides the processor-based SoC implementation during this work's first phase, the activities of this research have led to further elaborate implementations of Neural Networks on two intrinsically different parallel computing platforms, a GPU and a FPGA, embedded in cost-sensitive SoCs.
- **Defined a scalable and robust architecture:** the created software architecture in what respects to the controller and its algorithms, and the system architecture in what respects to the vehicle and its components, offer a solid and modular foundation to scale towards diverse applications.
- **Established a sustainable simulation framework:** the versatile model-based framework for simulation, development and implementation, ranging from MiL to HiL, has enabled not only productive iterations for this particular work, but also has been used for development activities for other projects. The vast infrastructure constructed around the main model can further be exploited for other projects. The same applies to generated data.
- **Consolidated a mature development methodology:** the framework in combination with the correct methodology and aligned with the V development model and good practices, facilitates deployment towards industrial projects. These solutions have already proven themselves in parallel customer projects and are expected to remain as reference.

- **Contributed to major international research projects:** many concepts and solutions, as well as multiple outcomes, were directly propagated into relevant European research projects in the >50M€ budget range, receiving the blessing of *European Commission* in multiple occasions.
- **Geared derived research lines across institutions:** different pieces of work have also trickled into other researchers' works with which relations have been established. For instance, inheritance occurred towards a follow-up PhD candidate across *Tecnalia* and the *University of the Basque Country*. Furthermore, while performing a stay at *KU-Leuven,* which is a Top-50 University in the World, it directly involved the local team lead, as well as a researcher from Latvia.
- **Positive results and acceptance enable future work:** the research objectives were covered and original expectations satisfied or exceeded. Consequently, the control concept, implementability, architecture, framework and methodology can be exploited for upcoming research. Good publication impact, presenting detailed technical and conceptual contents, as well as the basis of the research project as a whole, showed solid acceptance, ultimately also in a high-impact journal.

1.6. Structure of the document

This document has been structured following the workflow of the scientific research methodology of followed in the dissertation. Besides this structure, which shapes the document as a whole, the parts which contain the subset of the activity with a stronger technical core –chapters 3, 4 and 5– have been internally shaped by the development and engineering processes they present. The document consists of 7 main chapters, summarized in the following paragraphs.

The current chapter 1, ***Introduction***, has started by presenting the general introduction to the work, followed by a subsection providing a first insight into the *Technological context* and key points which will support the next two sections addressing the *Motivation and hypothesis* and the *Objectives and scope.* Following, the *Academic Framework* and the *Professional* C*ontext* of the doctoral thesis, as well as the *Planning* of the *International Stays* and *Other Activities* is discussed. Lastly, the *Contributions of the Work* are presented. The chapter concludes with the present section, explaining the *Structure of the Document.*

Chapter 2 provides a detailed insight into the ***State of the Art***, covering a wide spectrum of topics which reflect the interdisciplinary character of the work, which have already been identified in chapter 1. Each of the different topics, which constitute the essential parts of this work, are supported by a dedicated section. The first topic covers the application field this research is being targeted to, *Automotive propulsion systems* in general and *Electrified powertrain topologies* in particular. After aspects of related subcomponents are briefly discussed, the second subsection is dedicated to the closely related topic of *Automotive control systems*, which starts with several subsections discussing the main challenges and constraints such systems are facing, paying special attention to industrial topics. The next section covers the *Embedded systems*, which basically are enablers for upcoming developments, covering a wide and heterogeneous spectrum of platform types, including FPGAs, GPUs and processor-based platforms. The evolution of such platforms and their applicability to automotive systems is also discussed. The previous topics lead the thread to a section dedicated to *Software development methodology*, addressing the V-model and model-based methodologies and automated toolchains, which are a further important enabler. Finally, the chapter concludes with a section dedicated to *Advanced algorithms*.

Chapter 3 addresses the ***Definition and Modelling of the Control and Optimization Problem***, and therefore contains the first phases of the development activities. Firstly the targeted vehicle and its P*owertrain Type and Topology* and entire V*ehicle Specification* is defined. This is followed by the *Definition of Elements to be optimized*. Before proceeding to the *Definition of the Controller Requirements*, the *Design according to Automotive Constraints* is first discussed. The work finally proceeds to the *Representation of the MIMO control and optimization problem* together with the *Definition of Test-Cases*. At this point, the first block which collects all the aspect that represent the actual engineering challenge according to the objectives and technical aspects, discussed mostly in chapter 1 and partly in chapter 2, is concluded. Therefore this chapter proceeds to its second major block, which is the modelling of the previously defined system for the purpose of the controller to be designed according to the established criteria. The modelling tasks start with the most complex problem, the *Modelling of the vehicle Dynamics,* followed by the *Modelling of the Electric*

Powertrain and its *Energetic* and *Thermal* aspects is handled. To conclude, the models of secondary elements such as TCS and ABS systems are discussed.

Chapter 4 is the most extensive part of this document, as it condenses the entire ***Development and Implementation*** tasks. In practice it combines design, development and research topics with a notable technical density. The requirements from the previous chapter are propagated into the *Definition of the Control Architecture,* which reflects not only the controller, but also the targeted vehicle as a whole system. Following it proceeds to the *Definition of the Embedded Platform* as well as of the *Toolchain and Workflow.* Having established all the previous, the detailed research and development work can be executed. The *Advanced Torque Vectoring Algorithm* is conceived after describing the *Baseline Algorithm*. Then, the detailed development of each of the subcomponents that conform the entire solution follows, with the *Development of the Multi-objective Optimization* and the *Multi-objective Weighting* algorithms and ultimately the challenge of one of the core parts: the *Development of the Virtual Sensing.* This will be seen to be a Neural Network, so correspondingly the training and selection of topologies and impact of different aspects are addressed. Finally, the *Development of Mechanisms for Robustness* is addressed before processing to another major technical challenge which is the *Embedded implementation*.

Chapter 5 gathers the ***Results*** of the different aspects developed in chapter 4, covered by *Race Track Tests* with real embedded hardware in a vehicle in a first phase, and a highly representative real-time multibody vehicle simulator in a second phase. The *Final Tests of the Control System* assess the performance of the entire solution with respect to the previously defined test-cases aiming to satisfy the requirements, by performing extensive *Automated Simulation-Based Tests* before concluding with *Qualitative and Overall Assessment* including subjective evaluation. Finally *Embedded Implementation Results* include highlights from a common partnership work targeting not only processor but more outstandingly FPGA and GPU parts. To conclude, a summary over the entire *Achieved Solution* is provided with clarifying diagrams.

The work concludes with chapter 6, discussing the ***Conclusions*** and ***Future work***, including derived research lines, and with chapter 7 providing and overview of ***Other Merits, Works and Publications,*** including metrics for institutions, journals and projects related to the developed research.

CHAPTER 2:

State of the Art

"If you knew what you were doing, it wouldn't be called research"

Albert Einstein

The State of the Art in this chapter is presented by following the thread throughout a wide spectrum of interdisciplinary topics which was already introduced present diverse complex connections, involving both attractive research potential as well as challenges.

It starts with two different domains of the automotive context, firstly focusing on electrified and multi-motor powertrains, and secondly on control systems and industrial implications.

Then it proceeds to a detailed discussion of the vast diversity of embedded platform following different computation paradigms, focusing on those relevant to the targeted application: processor-based platforms, FPGAs, GPUs and their heterogeneous integration in SoCs.

The implications of the previously discussed automotive applications, industrial aspects and embedded complexity, highlight the importance of the next topic: software development methodology in general and the V development methodology, MBD and code generation, in particular.

Lastly, control algorithms and specifically Machine Learning are discussed, avoiding redundant contents already well illustrated in literature and focusing on providing a formal classification of Machine Learning before proceeding to illustrate some automotive applications.

CHAPTER 4:

Development and Implementation

"When you want to know how things really work,
study them when they're coming apart"

William Gibson,

This chapter assembles the actual implementation of the control solution presented in this work ranging from the conception of the control architecture and algorithms, to implementation topics.

It starts by defining the control and system architecture and the firstly selecting the embedded platform as well as the toolchain and workflow, before proceeding to the dense development tasks.

The development starts firstly with the baseline torque vectoring controller and secondly the entire advanced controller concept. In the following sections each of the subcomponents -i.e. the multiobjective weighting, the optimization, and most importantly, the Virtual Sensing- are extensively discussed. Robustness mechanisms are also briefly addressed. Finally, the implications and challenges of embedded implementation are discussed.

4. Development and implementation

4.1. Definition of the control architecture

As already discussed in subsection 2.2, modern vehicles are immensely complex systems which can hold up to over 100 ECUs . Even vehicles with a notably simplistic technological approach do present a non-neglectable amount of subsystems and connections.

Even when focusing exclusively on the powertrain domain, the complexity of the components and their functionality is still notable. This subsection will focus on the powertrain in general and ultimately on the implemented solution in particular.

Taking as reference the previous, Figure 4-1 illustrates with a simplified representation of a generic powertrain control architecture, where the main Powertrain ECU governs over the ECUs controlling different subcomponents and functionalities. It must be noted that this shape of the architecture does not represent the hierarchy. For instance, the ESP ECU can govern over the Brake ECU through the Powertrain ECU.

In many occasions a visual representation of the architecture, or eventually also of the electrical wiring, highlight the central role that the Powertrain ECU plays in the vehicle architecture, due to the fact that many different devices and networks are connected to it, and interact with each other through it. In other occasions, especially if a gateway is implemented, this is not equally obvious. This is the case for Figure 4-1.

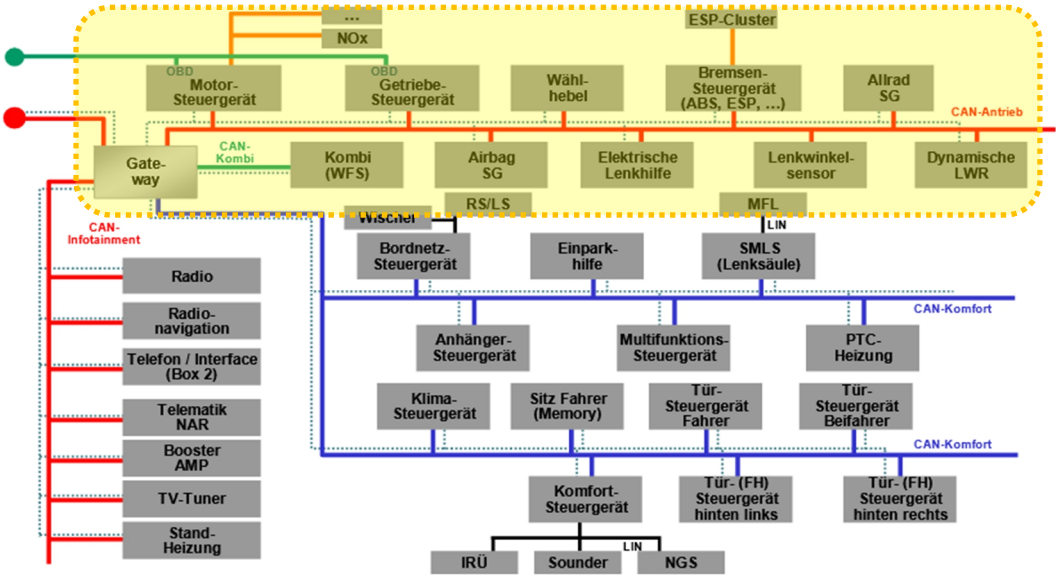

Figure 4-1. Architecture diagram with powertrain-domain and related ECUs highlighted [264]

Having a common understanding of typical automotive control architectures, at the entire vehicle level, we proceed to focus on the architecture of the powertrain domain and, in particular, on the application and solution developed in this work.

Figure 4-2 provides a baseline by illustrating a generic architecture for the powertrain domain, with the Powertrain ECU acting as centrepiece as previously discussed. Multiple specific ECUs which directly control important vehicle components and subsystems, such as the motor, the brakes, the stability systems, etc., are interfaced to it. It also acts as gateway to the rest of the vehicle, typically through a Central ECU which handles systems of less criticality, such as the HMI, including the instrument cluster and inputs such as the key, the shifter, etc.

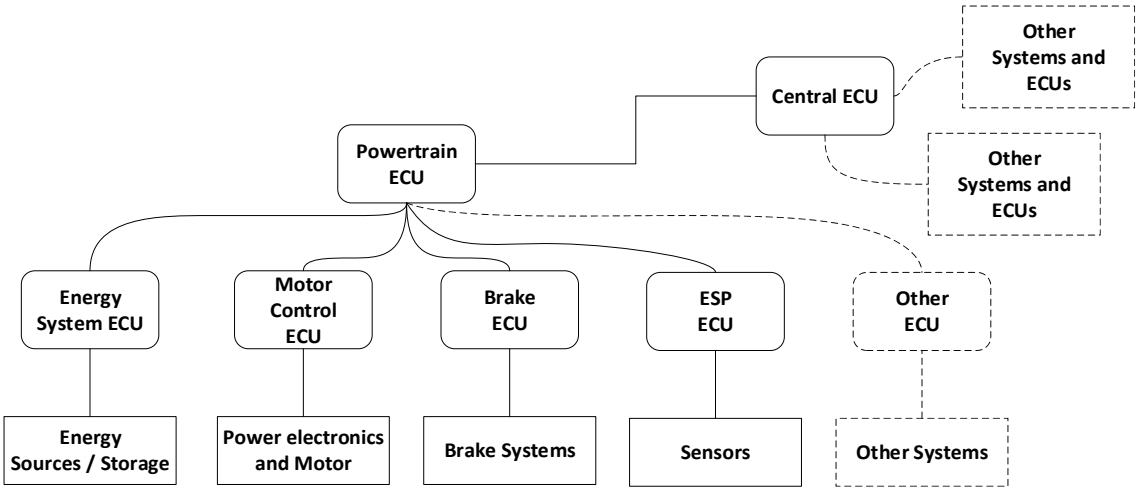

Figure 4-2. Generic architecture for the powertrain domain

Having a generic understanding of this domain, the particularization for this work can be described as follows. The first major difference is the presence of four sets of ECUs and power electronics corresponding to each of the four motors. The second is the addition of the Torque Vectoring ECU, which is the actual core of this work, on which upcoming discussions will focus. This is illustrated in Figure 4-3.

Nevertheless, it must be clarified that although the remaining components and subsystems are not objects to be developed as a solution in this work, in order to achieve representative simulations, considerable knowledge needs to be invested into their modelling. This has been already extensively described in the previous section 3.

It must also be emphasized that this architecture representation is an indicative for its topology, but does not necessarily represent the hierarchy. Following the inevitable safety concerns -such as discussed in subsection 3.3- and the general concept for stability for the developed solution, which come together with other fundamental principles in the controller requirements specified in subsection 3.4, the logic described in the upcoming paragraphs highlights the functional hierarchy, which is not effectively represented in such a diagram.

The basic principle is that the Torque Vectoring ECU is enabled to almost directly send torque requests to the motors. The nuance of almost directly doing it -instead of

directly- means that the requested set points are handled through the Powertrain ECU, which makes sure that no limit is exceeded, before forwarding these values to the motors. These limits may be constant limits given by the specification of the system components and its configuration, such as absolute figures like torque, power, speed, as well as transient magnitudes like gradients. They may also be condition-related limits, such as derived from battery performance constraints due to charge level or temperature, or the temperature of any other powertrain components as well. Even if the Torque Vectoring ECU should -and does- already anticipate such limitations in order to generate better optimized set points, the Powertrain ECU needs to act as supervisory agent and ensure that no limit is ever exceeded. But besides such limitations, a further, limiting element can come into action. These are the vehicle's own stability systems, basically the ESP, which will intervene if any critical situation is detected. This last point, is, in fact, a very important point for the safety concept in this work, as repeatedly discussed in the corresponding sections.

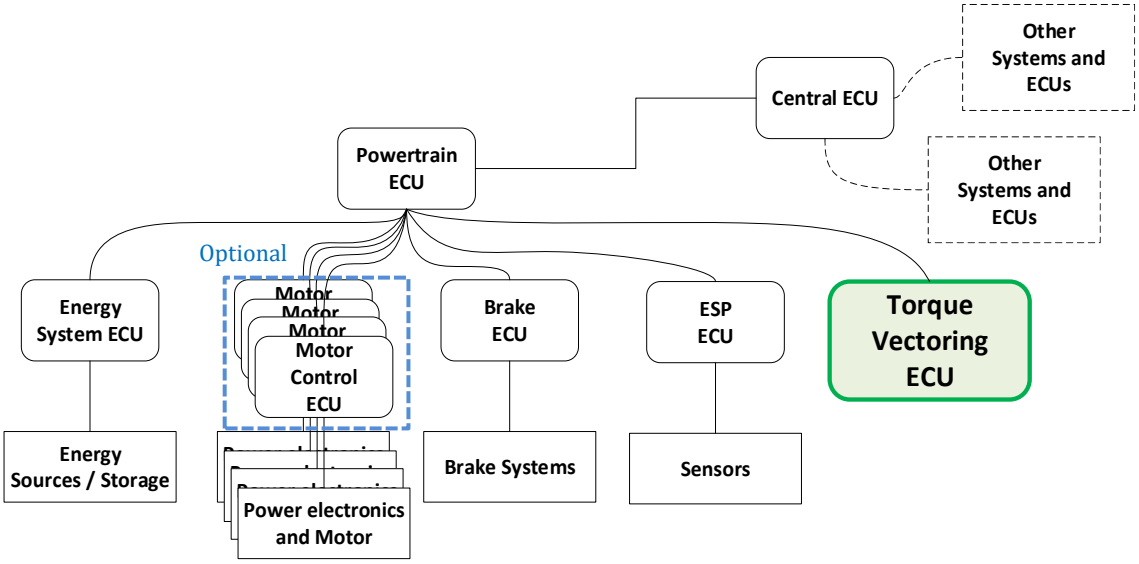

Figure 4-3. Particularized powertrain domain architecture for the targeted application

The following Figure 4-4 provides a more detailed representation of the Torque Vectoring's functionality in the context of the above discussed general architecture and functional concept. It illustrates the following two important points.

Firstly, it provides a clearer definition of the mentioned hierarchy of the control components. It matches the discussed general architecture also illustrated in Figure 4-3, emphasizing again the superior role of the Powertrain ECU. Furthermore, it explicitly indicates the already discussed fact that the torques requested by the Torque Vectoring controller are not directly applied to engines, but it can be influenced by the Powertrain ECU's control policies as well as the TCS/ESP system's limitations if necessary, in consistency with the previously cited topics from subsections 3.3 and 3.4.

Secondly, it shows the functional building blocks necessary to satisfy the said requirements. The specific algorithmic solutions contained in these blocks are yet to be defined and will be individually discussed in the upcoming subsections. But their general functional concept to provide a complete architectural understanding needs to be specified first as follows, and as illustrated in Figure 4-4.

The **Virtual Sensing** function (which could also be considered as an estimator) is conceived to provide additional inputs for the Torque Vectoring function, by providing information of values which are meaningful from the vehicle dynamics perspective, but not provided by the available sensors.

The **baseline Torque Vectoring** function can serve multiple purposes:

- Serve as reference point for the multi-objective optimization searching for the optimized torque distribution.
- Input for plausibility functions aiming to enhance robustness, stability and smoothness.
- In case of misbehaviour of the Advanced Optimizing Torque Vectoring controller, be used as fallback control function.

The **Objective Weighting function** provides an input to the multi-objective optimization function by dynamically adapting the weight of each of the objectives depending on the real-time situation of the vehicle.

The **Multi-objective Optimization function** is ultimately responsible for finding an optimized torque distribution basing on the previous inputs.

At this point of the discussion, the analysis of the system architecture is mostly left aside in order to proceed to a deeper analysis of the internal architecture of the controller itself. This will be addressed alongside with the main development of this element, in its dedicated section 4.4.

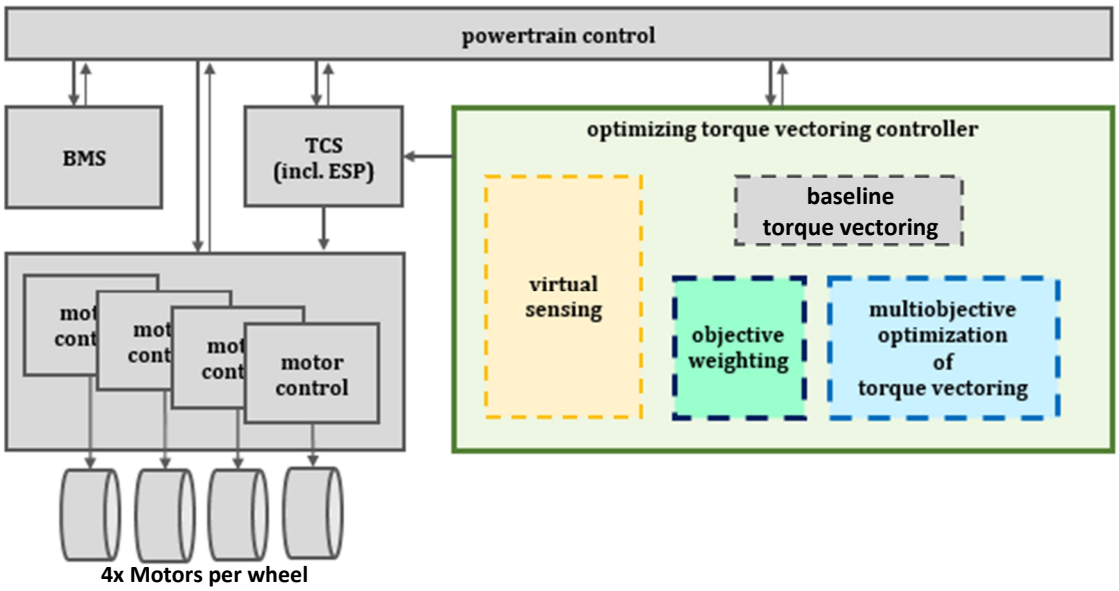

Figure 4-4. Powertrain architecture including advanced controller's fundamental functions

4.2. Definition of the embedded platform

The diversity of embedded platform solutions available on the market is notable, including devices of different nature which offer a wide range of capabilities, as well as costs (please refer to section 2.3 for an overview and explanations about relevant concepts like FPGA, GPU, lockstep, etc.). Nevertheless, with respect to the embedded platform market as a whole, the subset of devices eligible for automotive control applications is notably constraint due the aspects already discussed in section 2.2 regarding automotive control systems in general, and following section 3.4, when focusing on the developed control system in particular. Basically, the most critical aspects are the reliability and robustness linked to the functional safety considerations, while keeping a reasonable cost point. The expectation inevitably, is to obtain the greatest possible performance and parallelism under such constraints, but these inevitably are conflicting expectations.

Ultimately, the requirements that the selected embedded control platform should satisfy, crystallize down to the following specific points, which have been evaluated in a parametric manner. The following criteria is listed in no particular order: the priority of each will be discussed in the remainder of this section, where the implications of each point are elaborated. Ultimately these will be represented as weights for a parametric evaluation in Table 4-1.

- Suitability for automotive applications
- Reliability and robustness
- Redundancy
- Intrinsic parallelism
- Computing capabilities
- Interfacing and peripheral capabilities
- Development ecosystem
- Cost point

Criteria such as automotive suitability in particular, together with reliability and robustness in general, were combined under a single metric. This represents characteristics from different domains. From the purely physical point of view, it considers aspects such as the availability of the AEC-Q automotive qualification and the corresponding temperature ranges. From the embedded capabilities perspective, the availability of error prevention, detection and handling mechanisms are considered, such as ECC, MPU, etc.

The redundancy characteristic specifically is addressed separately, having to take into account that redundancy also contributes to a higher score in the previous criteria, firstly, because of the higher degree of integrity the redundancy itself provides, and secondly, because microcontrollers targeted at safety critical applications typically combine multiple additional solutions for that purpose. While single-core processors offer no redundancy at all, some multicore processors offer lockstep hardware-level “hard” redundancy down to each instruction-cycle. Nevertheless, also multicore processors which do not offer lockstep can enable the potential for software-based “soft redundancy”. Similarly, heterogeneous platforms enable other forms of redundancy to a further extent, by the means of their computational platform diversity. Besides FPGAs permit the

implementation of multiple isolated implementations of a same function for the same purpose. [141]

Parallelism is another highly relevant feature, which again holds certain relation with the previous paragraph. It might be that a device has two cores but offers no parallelism, if the second core is used for lockstep redundancy. Although some products are configurable to select either lockstep or parallelism, the first option would be chosen for the sake of this work. Besides using multiple cores, some embedded platforms also offer a certain degree of parallelism by the means of coprocessors dedicated to particular tasks, such as certain signal handling and mathematical functions. In what respects to GPUs, which still follow the instruction-based computation paradigm but with a strongly different architecture, they provide vast parallelism through their hundreds of cores, although subject to notable internal bottlenecks. Lastly, FGPAs, due to their vastly different computation paradigm, have great potential in intrinsic parallelism -and pipelining- brought by their hardware nature.

The last point of high importance in what respects to the capabilities of the device is the performance for the sake of tackling complex control algorithms, which is again partly related to the previous point. Although greater parallelism can be associated to greater performance in general, this is not necessarily the case and must be addressed as a separate criterion. On one hand, for instance, the parallelism of a low frequency multicore microcontroller might still provide lower total performance that a high clock processor with less cores. Furthermore, efficiently distributing algorithms among multiple cores brings an additional difficulty to harness the total available computing power, due to the complex relations between functionality distribution, execution time variations, bottlenecks, etc. and notable software architecture challenges. Besides the instruction set of the processor, memory also plays a major role. Firstly, in what respects to memory size. Secondly in what respect to memory bandwidth and performance, which can evidence shortcomings in runtime not only with regard to variable handling, but also in what respects to the reading of the program code. To support partial mitigation of such issues, mechanisms, such as DMA and caches are available on many platforms.

An important emphasis is to be made in relation to the two previous points, parallelism and performance: both are strongly application and implementation dependant, and should be discussed with care if considered out of context. In general terms, also with single-core platforms, the difference between the theoretical figure (like DMIPS) and the real executed instruction count per time, greatly depends on the application and its implementation. This gets even more acute on parallel embedded platforms. While certain applications -or algorithms- might be highly suitable for strongly parallel platforms, others present strong challenges to exploit the architecture. This might be, for instance, because the algorithm has a very linear character with highly coupled data dependencies, or because in spite of possible parallelism, intra-core communication bottlenecks occur. Even assuming a highly parallelizable function, the adequacy of the implementation, in what respects to structuring, programming techniques, and utilization of the particular hardware capabilities, can have a notable impact, easily of one order of magnitude -as will in fact be discussed in the results section-.

Lastly, two additional criteria of less weight are to be considered as well. The first are the interfacing capabilities, paying special attention to automotive-specific interfaces such as CAN, FlexRay and for more modern approaches, also Ethernet and its variations. Secondly development facilities are considered in the sense of the available resources and technical solutions to enable a more flexible and agile development of algorithms, especially for those platforms which represent greater challenges due to their greater complexity. Here, the more advanced solution of automatic code generation, as well as previously discussed automatic hardware synthesis (for the case of FPGAs) are of higher interest, which will be reflected in the upcoming section 4.3. Nevertheless, none of these aspects are of critical character for a good solution, as weak interfaces can be complemented with external solutions, and suboptimal development conditions can be compensated by greater development know-how -or simply time-.

To conclude, the inevitably strong constraint of the cost point is included as parameter. In practice, in real industry applications, this can be an extremely sensitive aspect, especially for bis scale products, although in certain cases the category of the product and the added value of the solution might be expected to relax this constraint. As a rule of thumb, a 50€ price (quoted for a 1000 unit order) is already considered costly, and 100€ would in most cases be prohibitive. Nevertheless, it must also be considered that in big quantity orders the unitary price might considerably drop as well, and furthermore, as technology and the market keep improving the performance/cost ratios, eventually some costlier devices might slide into the acceptable range. Subsequently, some of the devices on the high end some of the product categories are excessively costly to be considered suitable for the typical automotive price restrictions. The highest performing microcontrollers and microprocessors might have a borderline acceptable cost point, but currently only the SoCs on the lower performance end are considered to be suitable for the typical automotive cost constraints.

Table 4-1 collects the representation of the previously discussed criteria. Due to the immense diversity of embedded computing platforms available on the market, as seen in section 2.3 on which the facts discussed here are based on, the platforms are grouped into families of similar nature, for the purpose of a preselection. For each group multiple representative platforms are selected, illustrating the typical and most frequent characteristics and value ranges they offer, paying special attention to the higher end of specifications, as long as costs don not become disproportionate. This means that not all devices can be accurately represented and exceptions cannot be reflected. Each category is given a weight following the previous discussions. Additionally, for each category a minimum mark is specified as a mechanism to enforce strong constraints (i.e. the greatest platform cannot be eligible if the price is clearly excessive).

The result highlight three platforms with the highest marks, being the heterogeneous FPGA-based devices the ones with the highest score, followed by Lockstep Automotive Microcontrollers and Heterogeneous Multiple-Based platforms. Besides their score, the two later show relevant weaknesses in different critical points: the first in what respects to performance and parallelism, and the second in what respects to the excessive cost. The Heterogeneous FPGA-Based solution shows to be a very balanced solution, being the redundancy and cost point the most borderline criteria. Pure FPGAs also show relatively

well balanced parameters, with no borderline criteria but a clearly below threshold value in redundancy, besides having a lower total score. High Performance Microprocessors also fail due to reliability and redundancy, similarly to the GPU-based heterogeneous platform. Lastly, legacy industrial microcontrollers, basically combine most of the previous weaknesses.

Being the selected platform type the heterogeneous FPGA-based SoC, the decision between the two major vendors offering these kind of devices -Xilinx (Zynq® 7000 SoC) and Altera® (Cyclone® V SoC)- fell for Xilinx. Being their technical aspects and features coarsely comparable, the final decision was driven mostly considering the context of the application development, with factors being the ecosystem, available resources, integrability and the implementation approach of the toolchain, which as will be seen in the upcoming section 4.3, is most convenient to be aligned with MBD solutions already discussed, making it suitable for the overall modular philosophy of this work.

A major highlight is the heterogeneous combination of two opposed computing paradigms, enabling elaborate software architectures for robustness and exploiting strength of each application type. It integrates a dual core processor with mid-high performance -and exploitable for soft-redundancy- with a FPGA with notable performance, great throughput potential and sufficient size for many algorithms – plus additional soft-redundancy by diversity, plus redundant hardware possibility-. Figure 4-5 illustrates the architecture and main features of this device.

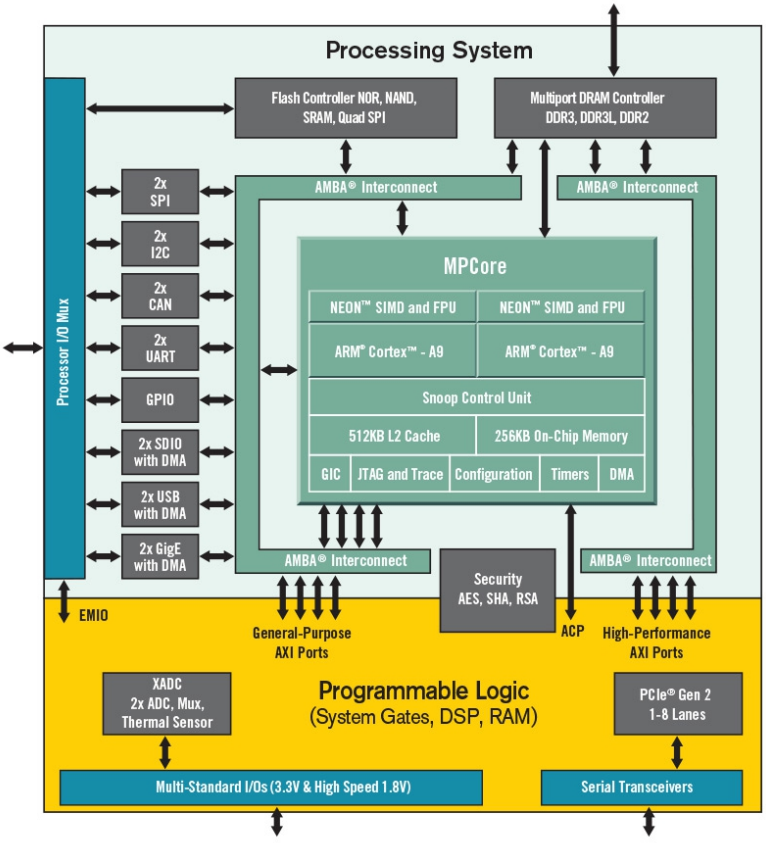

Figure 4-5. Block diagram of a Xilinx Zynq® 7000 Soc [265]

	Platform group →		Legacy Industrial Microcontrollers	Lockstep Automotive Microcontrollers	High Performance Microprocessors	FPGAs (lower end)	Hetereogeneous FPGA-based	Hetereogeneous GPU-based	Hetereogeneous Multiple-Based
	Representative examples →		* TI TSM320F28x	* TI TSM570x *NXP MPC56x *Infineon TC29x	* TI OMAP * NXP i.MX6	* Altera ... * Xilinx	* Xilin Zynq 7000 * Altera Cyclone V SoC	* Nvidia TK1	* Xilinx Zynq U+
Criteria	**Minimum**	**Weight**							
Automotive suitability, reliability and robustness	7	15	6 AEC-Q	10 AEC-Q ECC, MPU...	4	8 AEC-Q Hardware nature	8 AEC-Q Hardware nature	6	7 AEC-Q Hardware nature
Redundancy	7	15	2 Soft redundancy at most	9 Lockstep 2/4 cores	2 Soft redundancy at most	4 Single semiconductor Duplicate implementation possible	7 Multiple and duplicate implementation possible but semi-soft	6 Multiple redundancy but soft at most	10 Lockstep 2 cores Multiple and duplicate implementation possible
Parallelism	5	15	3 1-2 independent cores	4 1-4 independent cores	5 2-8 independent cores	7 Intrinsic hardware paralellism	8 2 independent cores Intrinsic hardware paralellism	9	10
Computing power	5	15	4	5	8	8	9	9	10
Interfacing and peripherals	4	5	7	9	5	6	7	6	8
Development facilities	2	5	9 Lowest complexity. Extensive resources. Best code-gen.	7 Medium complexity. Sufficient resources.	7 Medium complexity. Sufficient resources.	7 Relative complexity. Sufficient resources. Improving code-gen.	7	7	5
Cost	7	20	10 ~ 10..25 €	8 ~ 20..40 €	8 ~ 25..60 €	8 ~ 15..80 €	7 ~ 40..100€	3 ~ 100..200 €	2 ~ 100..300 €
		Total Score	**5.61**	**7.33**	**5.61**	**7.00**	**7.67**	**6.39**	**7.33**

Table 4-1. Parametric comparison of embedded platforms

The particular model finally used for this work is a mid-low model of the Xilinx Zynq® 7000, specifically the 7020 version. This is a highly capable but reasonably priced devices with adequate automotive suitability thanks to AEC-Q100 suitability and the inclusion of diverse robustness and data integrity mechanisms. The main characteristics and features are summarized in the following points, differentiating among its two different parts (FPGA first, microprocessor second) and thirdly some of their integration features [95][266].

- Artix-7 FPGA including:
    - 280 embedded memory blocks (4.9 Mb)
    - 220 DSPs (digital signal processing units)
    - 53200 LUTs (combinational logic)
    - 106400 registers (sequential logic)
- 2x ARM Cortex-A9 cores:
    - ARMv7-A architecture with Thumb2 instruction set and NEON engine
    - Single/double precision floating point
    - 667 MHz clock frequency
    - 2x 32KB L1 Instruction Cache, 2x 32KB L1 Data Cache, 512KB L2 Cache
    - 2x CAN 2.0B, 2x 1Gbps Ethernet, 2x USB 2.0, and other diverse interfaces
    - 3x Watchdog timer
    - Byte-parity support on Cache and On-Chip Memory
    - ECC double-bit error detection
- SoC FPGA and Microcontroller integration
    - Multiple bidirectional interfaces
    - 8 Direct memory access (DMA) channels (4 for FPGA)
    - Coherent and noncoherent access
    - DDR3 memory support
    - Separate power supply voltage lines

4.3. Definition toolchain and workflow

4.3.1. General methodological aspects and requirements

A MBD workflow aligned with the V development methodology -explained in section 2.4- has been followed for this work. This firstly was anticipated in and follows the argumentation related to *Development in the Automotive Context* from section 3.3 and the subsequent requirements of the controller defined from section 3.4. A relevant reason is to facilitate handling the mentioned complexity of automotive systems, which is also present in considerably intricate system models and control systems being developed in this work. Secondly it also satisfies the general modular philosophy of this work, where one of the motivations is to contribute to the State of the Art with enabler solutions and building blocks which can be used for future work.

Considering that the general aspects and details with respect to software development methodology, the V design approach and automated toolchains have already been discussed in section 2.4, redundancies will be avoided in the present section.

Instead the focus is placed on the particular tool solutions for this work, which are driven by three major factors:

- The nature of the domain and the particular application.
- The individual (sub)components being developed in this work.
- The dependencies and limitations linked to the embedded platform selected in 0.

The requirements relevant to the toolchain and its workflow, derived from the previous argumentation and methodological expectations, can be summarized as follows, basically specifying the criteria that the development solution should satisfy:

- Provide a modular, reusable and scalable solution, enabling to efficiently reuse parts of the current work, as well as take it as baseline for future work.
- Facilitate to handle complex systems, composed by a diversity of subsystems with extensive interfacing needs and with multiple abstraction layers, in an efficient and sustainable manner.
- Support documentation and certification tasks required for industrialization, such as automotive standards for safety critical applications.
- Maximize development flexibility to switch across development phases and use-cases.
- Maximize agility through development iterations to modify, analyse, calibrate and implement efficiently.
- Provide high portability of the developed control functions, especially including means of generic as well as platform and target specific code-generation.
- Facilitate the integration of third party components and models.
- Enable the interfacing with hardware components, especially physical ECUs over CAN protocol.
- Permit the execution of real-time simulations.

4.3.2. Workflows and use-cases throughout the development process

Different workflows are required depending on the approach and setup corresponding to different development tasks across different phases. The following paragraphs illustrate the fundamental closed-loop cases aligned with the previous methodological discussion and references, which shall be enabled together with the above requirements by the toolchain.

The fundamental solution is a MiL setup, illustrated in Figure 4-6. It is fundamental firstly because it covers the broadest part of the closed-loop development needs and secondly, because the following approaches can be seen as different variations of the same. In a MiL setup all the components of the system are integrated into a simulation platform, in this case a PC running a single simulation environment. This means that the controller as well as the plant model being controlled, including the multibody vehicle dynamics simulation, is running all together. It permits fully reproducible simulations, meaning that multiple runs with the same parameters will produce the same results. It also enables accelerated simulations, which are highly convenient for iterative batch simulation runs as typically used for analysis and calibration purposes.

For illustrative examples of the resulting Simulink™ models, please refer to the subsections in chapter 3; where this was already anticipated.

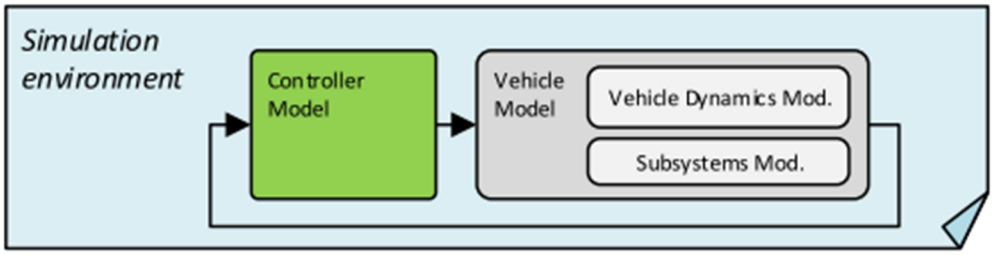

Figure 4-6. Model-in-the-loop (MiL) setup in the context of this work

The MiL setup evolves into a HiL configuration, illustrated in Figure 4-7, when the loop is closed over a real physical component which is connected to the simulation running in real time. In this work, the piece of hardware under development being the ECU running the control algorithm, the models remained in the simulation environment and the loop was closed over a real ECU prototype. At this point, real CAN messages are used. This means that a wired CAN bus is physically present and the actual CAN Database File (DBC) corresponding to the real vehicle will be applied to the interfaces. Should the case be that, unlike the system implemented in this work, analog and/or digital input/outputs should be present, corresponding interfacing modules could be added.

The productivity of a HiL approach such as this, where the ECU under development is the hardware element closing the loop, is greatly improved through the implementation of an automated process to deploy the controller model from the development environment to the hardware. This strongly streamlines each iteration. Furthermore, good calibration possibilities (i.e. adjusting controller parameters without recompiling and re-flashing, and with minimal effort) is also of considerable importance.

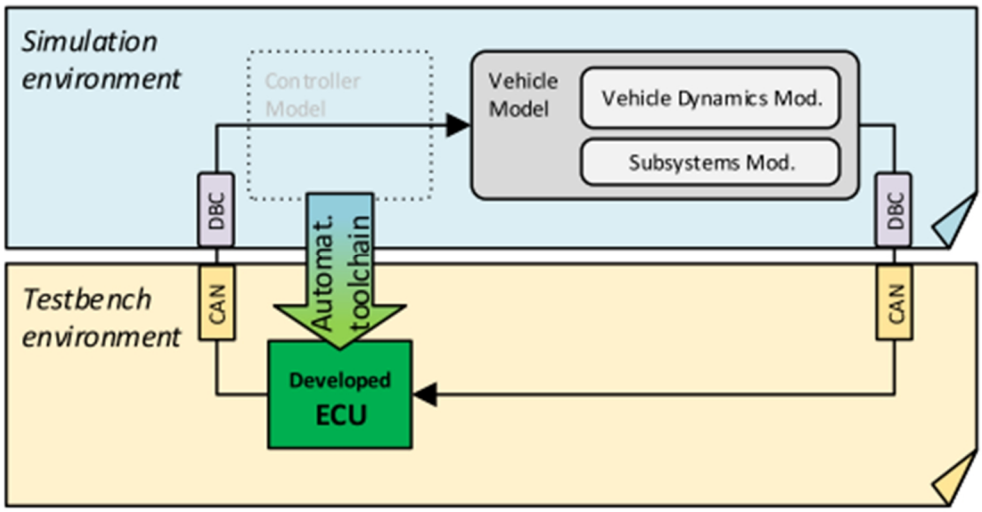

Figure 4-7. Hardware-in-the-loop (HiL) with automated deployment into ECU

The HiL setup in this work should ultimately provide a very accurate depiction of the behaviour of the vehicle and be totally transparent to the ECU. In other words: the ECU should not even "notice" that it is not connected to the real car, but to a simulated environment instead.

The bottomline is that the HiL setup should be a solid step before proceeding to test the ECU in a real vehicle. This method was used in this work: successive implementations were validated running the ECY firstly in the HiL setup and then the ECU was directly unplugged from the HiL and plugged into the car.

It must be noted that no SiL (software-in-the-Loop) nor PiL (processor-in-the-Loop) setups where chosen to be used in this work. The main reason is that, having elaborate MiL and HiL setups with highly automated infrastructure, these intermediate steps were not sufficiently relevant. Considering the higher implementation effort, ultimately the use-cases which could be covered by this complementary setups cam be covered by combining the MiL and mostly the HiL features.

Although the MiL and the HiL approaches should cover most of the use-cases and development time, a further use-case worth briefly describing is the reproduction of real recorded data towards the ECU. The purpose of this use-case is to anticipate any misbehaviour the ECU might show when exposed to the actual vehicle signals, in case there is some aspect of it which is different from the HiL setup, like for instance some unexpected signals, unstable values, etc., coming from vehicle subcomponents. Inevitably, this being an open loop setup, its coverage and representativity is limited to certain aspects. This setup is illustrated in Figure 4-8.

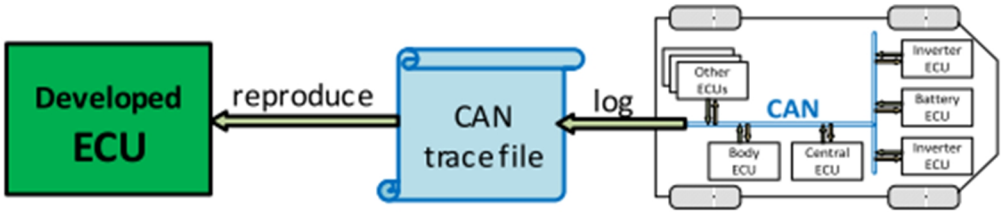

Figure 4-8. Open Loop CAN trace reproduction setup

A final setup is in practice a variation of the previous. It consists in actually connecting the ECU to the car, but blocking all the ECUs transmit signals. This can be made either by using a CAN communication gateway or router, or by disabling the transmit functionality in the ECU software. It remains being an open loop test, as the ECU will receive the same messages as with the trace-based approach but, instead of using an intermediate device under laboratory conditions, with the difference of doing so directly inside the car, consequently being exposed to the vehicle's electrical harshness, voltage variations, etc.

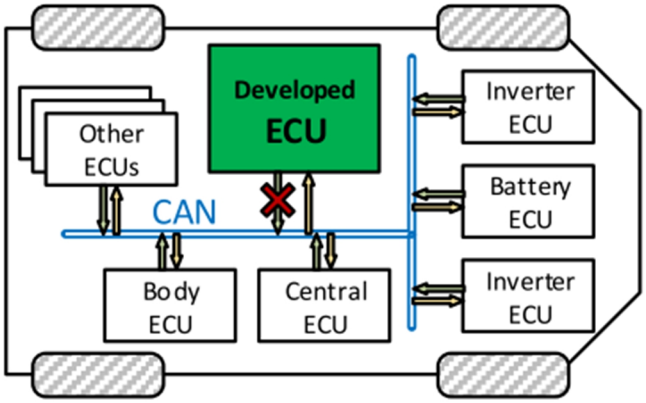

Figure 4-9. In-vehicle ECU integration with blocked transmit messages

Besides the diverse closed-loop and open-loop setups themselves, the different approaches for providing driver inputs to them are equally relevant. Basically, three different approaches are followed, as will be further seen in the upcoming sections:

- DiL (driver-in-the-loop): enables to drive the simulator in real-time using a steering wheel wit pedals together with the 3D visualization. The most common use case here is to combine it with the MiL simulation, although is also makes sense with the HiL setup. It is also to be used to record data for multiple purposes, such as arbitrary driving manoeuvres for the Machine Learning algorithms and reference paths for the autonomous driving function.
- Autonomous: an autonomous "virtual" driver function is used to drive the car over a specified path following a series of parameters, such as speed and acceleration limits. This is ideal to perform big batches of repeatable accelerated simulations, for instance, driving around a race track.
- Automated: basically is a simplified variant of the previous where the virtual driver acts in open loop and simply repeats predefined inputs, without any

control action. These inputs can be either a predefined pattern of steering, torque and brake inputs, or a recording of a previous manoeuvre. This is useful for instance for constant steering angle tests.

4.3.3. Selected toolchain

The following points provide a compact overview of the particular tools selected for the framework to develop this work and implement the solutions and methodologies described in the previous subsections.

MathWorks® Matlab™ Simulink™, is taken as central platform for the entire development. Both the system model (as illustrated in chapter 3) and the controller model (to be developed in this chapter 0) are implemented here. Matlab™ scripts are used to automate tests, data analysis and parameter handling. Code generation functions are used for efficient implementation of the functions. [267]

Dynacar™ 2.0, is the high-fidelity multibody vehicle dynamics model, extensively explained in 3.8, which is integrated into the Simulink™ model as well. [249]

Xilinx Vivado™ and SDK, is the fundamental tool offered by the semiconductor vendor for development and deployment on FPGA platforms in general, and Zynq® SoC devices in particular, by combining the general Vivado™ tool for FPGA implementation with the SDK application for microcontroller code implementation. It can be partly integrated with Matlab™ Simulink™ HDL Coder™. [268]

Xilinx HLS™, is an additional tool stands for High Level Synthesis, and permits to automatically convert generic C code into VHDL or Verilog FPGA hardware description IP blocks. [188]

Xilinx SDSoC™, is a tool which goes beyond the features of HLS™ by taking a higher level approach. It uses the same principle of converting C into hardware description languages, but additionally automates the integration of the FPGA and Microprocessor part, by automatically configuring the timing mechanisms and the interfaces for data exchange. [189]

Nvidia CUDA®, is the development environment for Nvidia GPUs and SoCs integrating them. It provides very complex performance analysis tools. [153]

Peak-Systems CAN solutions, are hardware and software solutions for working with the physical CAN bus. Peak USB-CAN interfaces are selected both for analysis, control and Simulink™ integration purposes. A CAN-GPIO module is also chosen for the purpose of in-vehicle parameter tuning on the race track. A CAN-WIFI gateway is used for telemetry on the race track. A CAN-GPS-IMU module is used for complementary race-track data acquisition. A CAN-logger device is used for complementary data logging. The software PCAN Explorer is used to plot signal and build GUIs to control and monitor during runtime. [269]

Racelogic VBOX, is a professional vehicle dynamics logging system which incorporates GPS data with a 6-DOF IMU inertial sensor, for race-track tests with the vehicle. [270]

4.3.4. Code generation approaches

The above defined toolchain enables a notable variety of implementation and code generation approaches. Due to the fact that the primary source for the function implementation is not C code -but an abstraction layer above: a model- multiple ways are viable to achieve the implementation of a function on a single platform, as illustrated in Figure 4-10.

This is particularly the case for the FPGA implementation. Here two main approaches are possible, which present fundamental differences. The first is generating directly from Simulink™ to VHDL using MathWorks® HDL Coder™. The second is using C-code as intermediate representation, to then be implemented into the FPGA using high-level hardware synthesis with Vivado™ HLS™ or SDSoC™. This has the conceptual advantage of being a more generic common ground for all three platform types (Microprocessor, FPGA and GPU), but arises efficiency concerns.

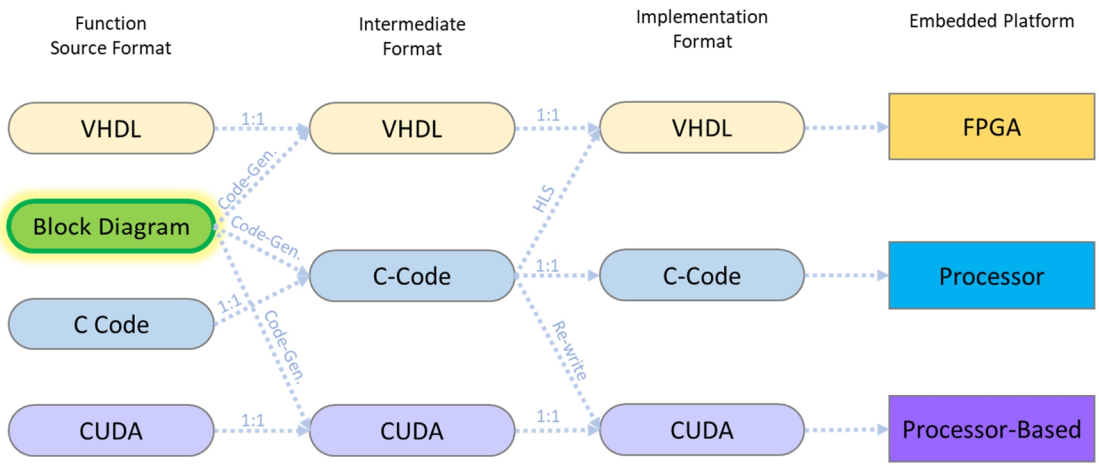

Figure 4-10. Code generation approaches

For the first stage of the work, the approach is relatively straight forward: the target platform being microprocessor-based, a generic C-code generation approach is sufficient.

For a second stage of the research work, further challenges implementing not only in FPGAs, but also in GPUs are concurrently explored. Here, the suitability and versatility of the MBD and code-generation approach is yet again validated: For both implementations, the original Simulink™ model can be taken as baseline and the output successfully propagated to the corresponding toolchains. For more details please refer to section 5.4.

4.4. Development of the Torque Vectoring controller

Before proceeding to the design of the advanced optimizing Torque Vectoring controller itself, the baseline Torque Vectoring controller, which will be a standard Torque

Vectoring controller, needs to be developed as well, following the foundations established mostly in the requirements and architecture in sections 3.4 and 4.1.

4.4.1. Development of the baseline Torque Vectoring function

4.4.1.1.Definition of the general controller concept

The baseline Torque Vectoring solution is intended to consist of a simple, robust, efficient and predictable algorithm, as brought by the discussion from the multiple previous sections. Consequently, purely arithmetic approach of primarily open-loop nature and without any temporally dynamic components (i.e. integrative parts) is preferred. This means that the design needs to base on conventional arithmetic operations and look up tables. Still, sufficient behavioural complexity can be achieved in desired to adjust it to the perform as desired, by combining multiple look up tables and variable gains to amplify or attenuate the vectoring effect depending on the desired behaviour, following a criteria-based behavioural expression form which conceptually has similarities to fuzzy logic semantics.

Alternative approaches were certainly considered as well, especially using closed-loop yaw rate controller concepts and Fuzzy Logic. Torque Vectoring controllers involving such a solution were developed in a later ramification of this research line leading to concurrent research activities (explained in chapter 7) were a Fuzzy yaw rate controller is implemented with positive results [271][56].

Nevertheless, to comply with the requirements defined in this work, the hypothetical benefit of such a Soft Computing-based controller is considered insufficient to trade of the more straight-forward design initially discussed, intended to act both as a baseline as well as a fall-bac solution. Consequently, the look-up-table based approach was selected.

4.4.1.2. Initial development for real circuit testing with a 2 MiW vehicle

A standard look-up-table based Torque Vectoring controller, such as the one discussed, was developed as part of an dedicated specialization project exploiting the demonstrator vehicle from the Eunice research project [227]. This was a fundamental part for the first major stage of this work, and is as well an important building block for the further advanced algorithms in the seconds phase.

This Torque vectoring controller, illustrated in Figure 4-11, is conceived as a widely tuneable algorithm with open-loop as well as closed-loop operation capabilities, aiming to enhance the development and testing flexibility. The core control variable is based on a theoretical calculation of the centripetal acceleration corresponding to the curve that is being driven. This is calculated using the vehicle's speed and the radius that would correspond to the steering wheel angle under optimal traction conditions. Considering this theoretical centripetal force as a representative of the magnitude of the curve being driven, a Torque Vectoring value is determined. The control law applies low values for soft cornering forces (under 0.15g), rapidly increasing the value for medium corners and limiting it to a 10/90% distribution for the strongest corners (over 0.55g).

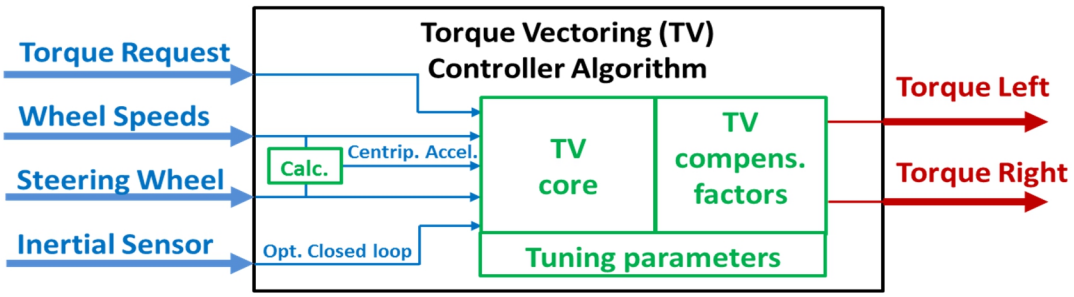

Figure 4-11. Baseline Torque Vectoring design for two wheel in motor Eunice vehicle [50]

Taking this as reference value, a series of factors are overlaid to increase or decrease the Vectoring value. For instance, it is attenuated at very low and very high speeds. It is also attenuated for close to zero wheel angles. An additional factor is the steering wheel velocity, which can be adjusted to increase if agile and incisive reactions are desired, or to attenuate if stability and safety have highest priority.

Furthermore, this controller was conceived to permit the addition of a closed-loop behaviour using feedbacks from the inertial sensor, either with the lateral acceleration or the yaw rate. The theoretical value of the later corresponding to the driver's inputs can be calculated in a simple manner similarly to the centripetal value. If these values are far below or beyond the theoretical ideal values, amplification or attenuation factors are applied. Nevertheless, under consideration of the achieved results and for the sake of simplicity, stability, and sensor inaccuracy and noise immunity, these functions were chosen not to be activated in this first phase.

4.4.1.3.Final Standard Torque Vectoring controller design

Consequently, this look-up-table based design approach was taken as baseline for the Torque Vectoring of the vehicle with four independent motors presented in this work. Furthermore, it will also serve as runtime fall-back solution, if any kind of malfunction is suspected. The fundamental principles remain the same, as illustrated in Figure 4-12: it uses static look-up-tables and standard arithmetic functions in combination with variable attenuation and amplification factors to implement a straight forward open loop control approach.

Fundamentally, the only major change with respect to the baseline Torque Vectoring is the fact that, with the introduction of two additional motors, two new degrees of freedom are added as already explained and mathematically formulated in section 3.5: three torque distribution values are needed, one between the front/rear axis, and then the left/right distribution on each of the axes, which in combination with the total torque demand input, determines the specific torque for each wheel.

An additional enhancement made to the original concept is that the main look-up-tables, those that control the distribution basing on the theoretical lateral force, were extended with an additional dimension. The third dimension is the requested torque, which permits a better handling of the limits of the engine, strongly reducing the incidence of the

saturation functions in the constraint blocks at the output of the Torque Vectoring. Such constraints blocks enforce the limitations of the power and torque specifications of the motors and could cause the values of the vectoring to get truncated (please see Figure 4-16 and related explanation in subsection 4.4.1). Should this undesired event occur, a reassignment function is put in place to, identically as in the original Eunice Torque Vectoring, add the truncated torque to one of the other wheels. This is aimed to fulfil the delivery of demanded torque.

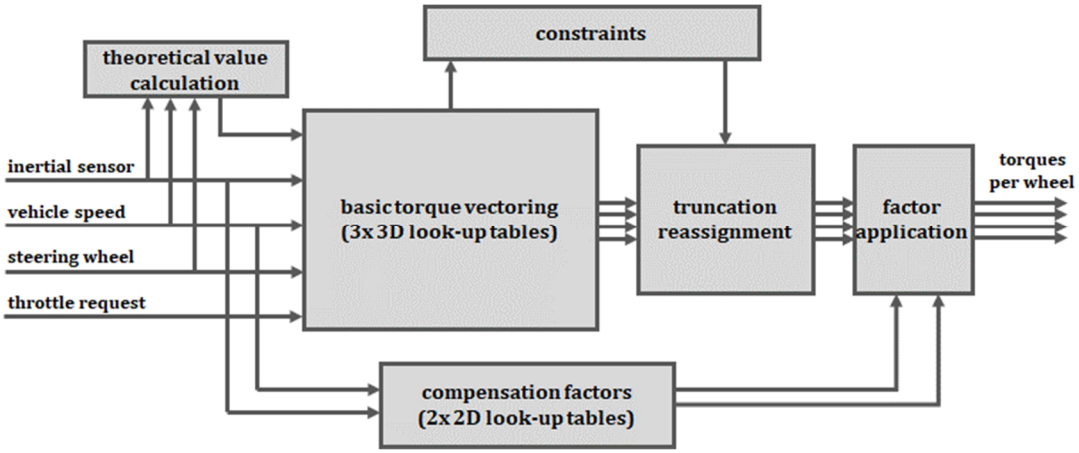

Figure 4-12. Block diagram of the baseline Torque Vectoring controller

The calibration of these 3D look-up-table values, shown in Figure 4-13 and Figure 4-14, also follows the same principle and even similar values to the original controller, especially in what refers to the lateral distribution (left/right) among each axis. A difference is brought by the longitudinal distribution, where more torque goes to the rear axis for two reasons. Firstly, because this axis is dimensioned to provide greater power, and consequently a 50/50 distribution cannot be kept at all times. Secondly, because excessive torque on the front axis tend to lead to understeer, in contraposition to the rear axis, where it tends to cause oversteer. Consequently, by default a rear distribution proportional to the power difference is applied by default. This also avoids subjective inconsistency feeling when, while accelerating, the front motors would arrive at their power limit earlier than the rear motors.

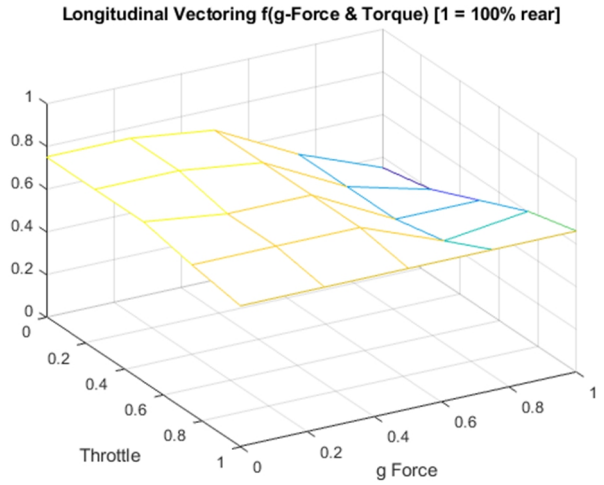

Figure 4-13. 3D Look up table for longitudinal torque distribution

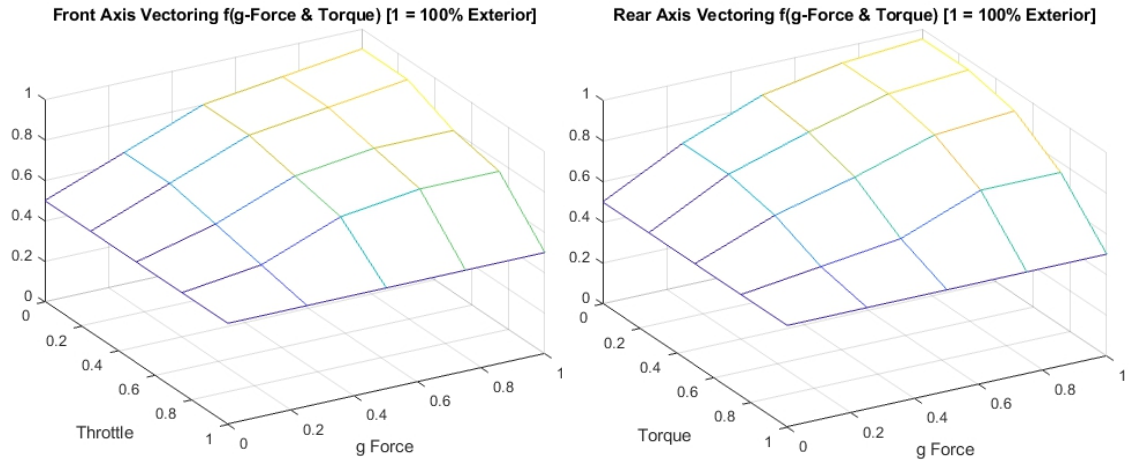

Figure 4-14. 3D Look up tables for lateral torque distribution

4.4.1. Development of the advanced Torque Vectoring controller

Having established the baseline Torque Vectoring function, Figure 4-15 represents all the elements of the control function as whole, represented as the green box. Here, the actual advanced part, which provides the MIMO optimization an represents the one of the main engineering challenges, is represented in a blue dotted box. The elements on the left side of this box, architecturally speaking, provide inputs to the advanced functionality, while the grey boxes can be seen as standard elements.

This diagram also highlights another element which represents a major challenge: the Virtual Sensing inputs. While the definition of the needed signals for the intended Torque Vectoring functionality will be established in this subsection, the development of the said Virtual Sensing building block itself will be developed in the dedicated section 4.7.

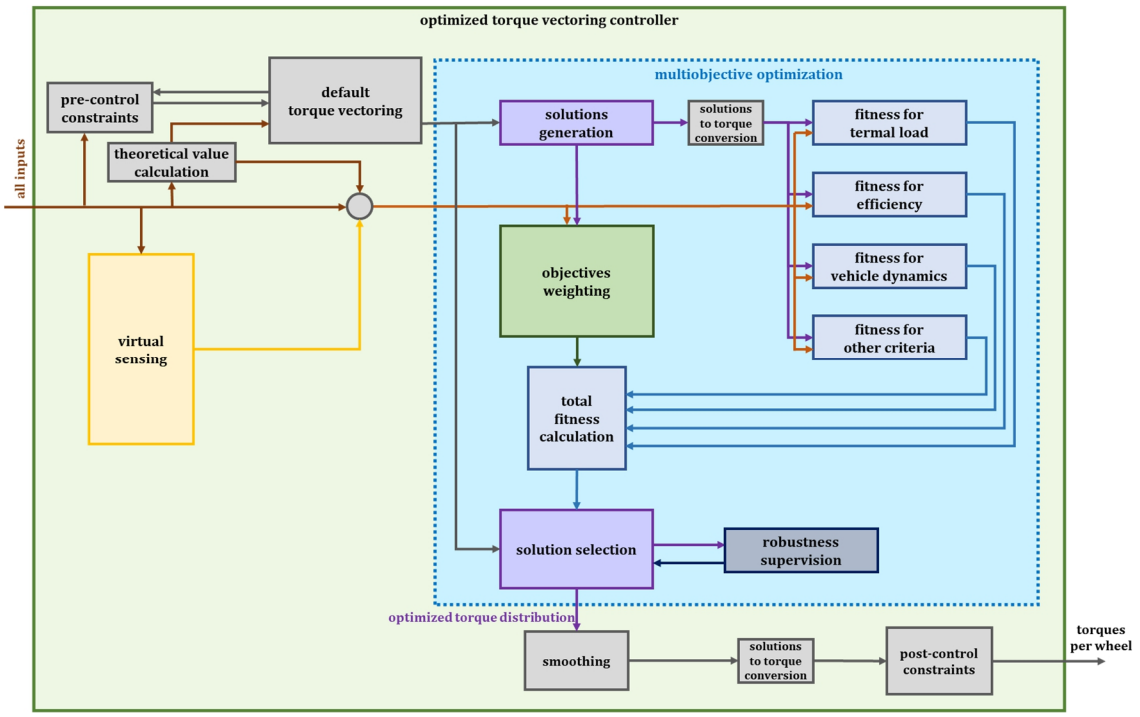

Figure 4-15. Functional architecture of the advanced optimizing Torque Vectoring controller, combining multiple functions and subfunctions from multiple functional and depth layers

There are two conceptually different kinds of **inputs** for the optimization function of the advanced Torque Vectoring controller.

1) **Signals** that provide information about the dynamic situation of the vehicle itself. This information can come from multiple sources, although the physical sensors are the primary source for all of them:
    a. Sensors directly providing physical values such as vehicle speed, wheel speeds, acceleration forces and rotational velocities of the vehicle.
    b. Theoretical real-time calculated value(s) representing to severity of the cornering, as previously discussed for the standard Torque Vectoring.
    c. Virtual sensing real-time signals providing values that are not measurable using the actual sensors.
2) **Reference value** from the baseline Torque Vectoring algorithm. This is to be used as starting point for the optimization function of the advanced algorithm.

With these inputs, the actual core functions, which are the multi-objective weighting function and ultimately the multi-objective optimization function itself composed by a variety of subfunctions, come into action. As these building blocks are major components with an entity of their own, the corresponding extensive explanations will be addressed in the two additional dedicated sections 4.5 and 4.7.

Nevertheless before proceeding, it is worth mentioning some secondary functionalities shown at multiple points of the diagrams. The one requiring some more explanation is the constraint functionality, which in the previous Figure 4-15 appears at the beginning and at the end of the chain. The purpose of the first function is to anticipatedly feed the effect of the specified power and torque limits of the motor, and take these into

account in order to avoid optimizing towards a non-applicable value, which would get truncated at the end of the chain with the post-control constraints, illustrated in Figure 4-16, thus reducing the optimality of the set-point.

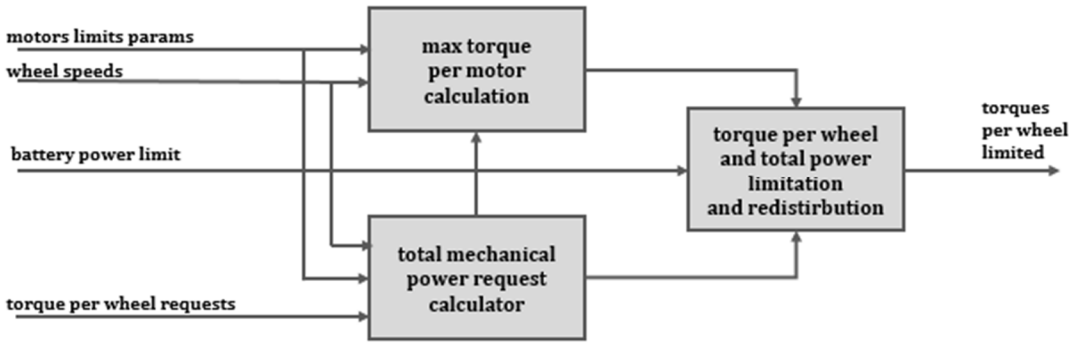

Figure 4-16. Constraints function, applied as pre-control function

Another of two further minor secondary functionalities is the smoothing function, which basically is a first order filter. This has been calibrated to be little restrictive, as the nature of the vehicle dynamics might require fast actions, and smoothness. Therefore its main purpose here is to smoothen the discrete variations between consecutive samples.

The last secondary block to be mentioned is the conversion from the solutions expressed as three distribution variables, into torque for each of the wheels according to the demanded total torque. This function and its inverse are actually present at multiple points of the discussed diagrams, but are usually considered among the trivial blocks to be omitted for the sake of simplicity and clarity.

Besides the architecture and the functionality discussed, a fundamental parameter also needs to be defined: the sampling time of the controller. Observing the dynamics of the system from the models and race track data acquisition, finally a value of 5 ms has been defined. This is a suitable compromise between the time constant of the actuator, which is the motor, the slower behaviour of the vehicle dynamics and slip, and the data flow on the CAN communications protocol, which is not faster than a 10 ms cycle for most signals.

4.5. Development of multi-objective objective weighting

The purpose of this function, conceptually speaking, is to determine in real-time the importance of each of the objectives, in accordance to what has been defined in sections 3.4 and 3.5. This means, that the four outputs of the objective weighting are the weights for the each of the objectives:

- Vehicle dynamics
- Energy efficiency
- Thermal load
- Comfort/Smoothness

Architecturally speaking, as illustrated in Figure 4-15, this function is very closely linked to the general multi-objective optimization function, as it can in fact be considered a part of it, and the objective weighting output is needed for the multi-objective optimization's total fitness evaluation and final solution selection. These other functional elements will be discussed in the upcoming section 4.6.

The nature of the objective weighting function, which needs to decide in real-time the importance of each of the four objectives depending on the current driving variables, was identified to be highly suitable for the utilization of a Soft Computing such as Fuzzy Logic, as it is principally based on qualitative criteria which can best be expressed through the formulation of rules based on expert knowledge. The explicit accuracy of a numerical approach is not only considered as unnecessary, but might even be counterproductive, for instance for (re)calibration task efforts, besides greater difficulty in expressing the knowledge and experience-based rules.

One alternative could have been following a look-up-table based approach, similarly to the baseline Torque Vectoring described in subsection 4.4.1. Actually, the reasoning behind those Torque Vectoring rules expressed as look-up-table curves, could eventually also have been expressed as Fuzzy rules, and vice versa. Nevertheless, it was determined that due to the amount of needed input and output variables, and the complexity of their relations, expressing and adjusting the wanted behaviour would have been notably more challenging and, as discussed in the previous paragraph, it was not considered necessary nor adequate.

Having determined the usage of a standard Fuzzy-Logic algorithm for the real-time determination of each of the four objective weights, under consideration of the vehicles behaviour regarding vehicle dynamics, as well as the other aspects as a system as a whole, six inputs were selected for the Fuzzy Logic function. These are enumerated in the upcoming bullet points, together with the dedicated functions which provide the corresponding information.

The calculation of these input signals that are provided to the Fuzzy Logic objective weighting function is simpler when compared to many other functions in the controller. Therefore, they are better explained than represented graphically. Some of them are even basically just forwarding of the information already available from the previous layers. An additional element present in each of these functions is a scaling factor added to normalize

but which also can be used to alter the sensitivity of the Fuzzy Logic controller. In other words, these are additional calibration parameters. The resulting function is reflected in Figure 4-17.

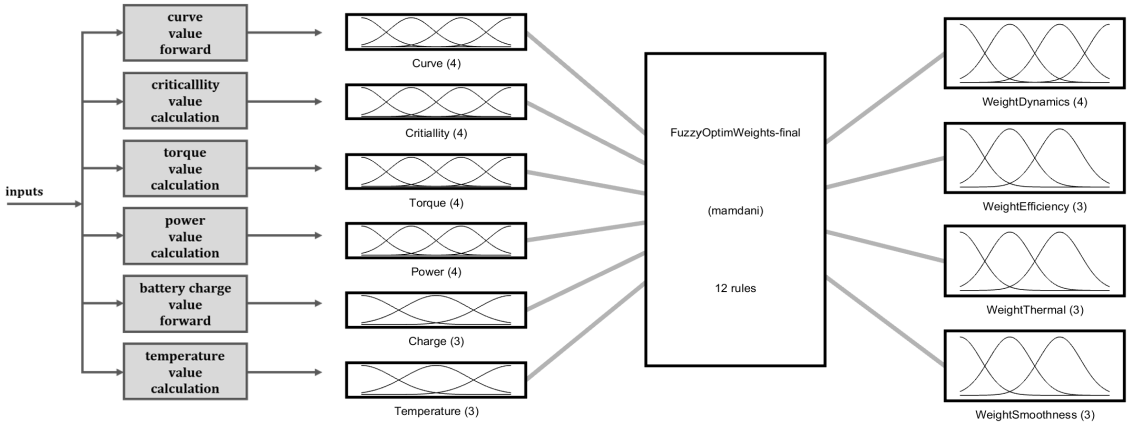

Figure 4-17. Fuzzy Logic function with 6 inputs and 4 outputs, with membership functions and their number, plus the functions to provide the values to the inputs

- Curve: Forwards the numerical values already used for other functions representing the theoretical yaw rate and lateral acceleration the vehicle should be theoretically facing under optimal grip, basing on the steering angle and longitudinal speed. These values represent the severity of the curve being negotiated and therefore higher values will mean a more demanding situation, although not necessarily critical.
- Criticality: This is the only function that involves some complexity if compared to the others. Fundamentally it bases on the same theoretical values from the previous "curve" function, but the value grows if the lateral acceleration and/or yaw values significantly differ from the theoretical values. This means it follows an equivalent logic to the one discussed in 4.4.1. The meaning is, that the vehicle is starting to slip and that therefore the situation is critical, this value will increase.
- Torque: Simply forwards the total requested torque value. It is meant to represent how demanding the acceleration request is, as higher values are more likely to represent a dynamically challenging situation and also to saturate the wheels.
- Power: Calculates the total mechanical power value basing on the wheel speeds and current standard Torque vectoring distribution. It is related to the previous due to the common element of the torque, but simultaneously represents the magnitude of speed. This means that the behaviour at higher speeds can be indirectly assigned differently.
- Charge: Simply forwards the charge status of the battery. The reason is that it should take into account to prioritize efficiency as far as possible, if battery level gets reduced.
- Temperature: Provides information if any of the four power electronics modules or motors are overheating, to correspondingly provide higher priority to the heat redistribution objective.

The membership functions were determined to be organized in an intuitive equidistant manner aiming to slightly reduce the significant calibration complexity, as the representativeness for their physical meaning can be conveniently calibrated anyway through the previously mentioned gains, besides the rules themselves. Another reason was to ensure smooth operation with progressive transitions which ae perceived as natural for the driver and passengers. As computational load was not determined to be a relevant constraint in this case subject to relatively few evaluations, a gaussian membership function was used aiming for the same reason. Depending on the physical magnitudes involved, three or four membership functions were considered as representative. So conclusively, the input membership functions were defined as illustrated by Figure 4-18. Similarly, the output membership functions were defined as in Figure 4-19.

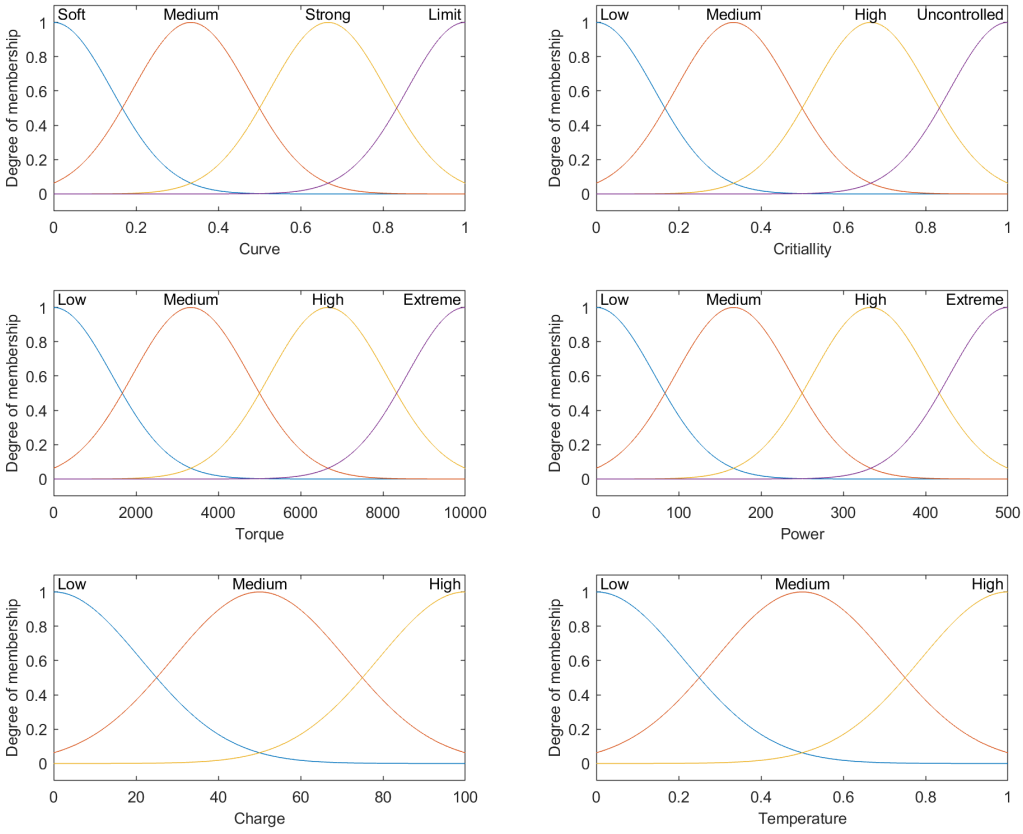

Figure 4-18. Membership functions for Fuzzy Logic inputs

Finally, the rules were defined to reflect the qualitatively expected behaviour under consideration of expert experience. Certain iterative adjustments were performed following the common practice of this work by objectively evaluating metrics following the use-cases. Effort was invested into reflecting the intended behaviour and the most important criteria with the minimum amount of rules possible, for the sake of simplicity and clarity. The resulting rules are showed in Figure 4-20, which were implemented using Mamdani inference with centroid Defuzzification method, leaving the remaining parameters on default values. Basically, the operation principle as always follows the *Requirements* from section 3.4: Vehicle dynamics is generally the top priority objective, as it is also the most safety critical aspect, especially when the driving state is getting closer to critical limits.

Otherwise, energy efficiency is kept predominant, unless temperatures get closer to the functional limits, which progressively increases the weight of the corresponding objective.

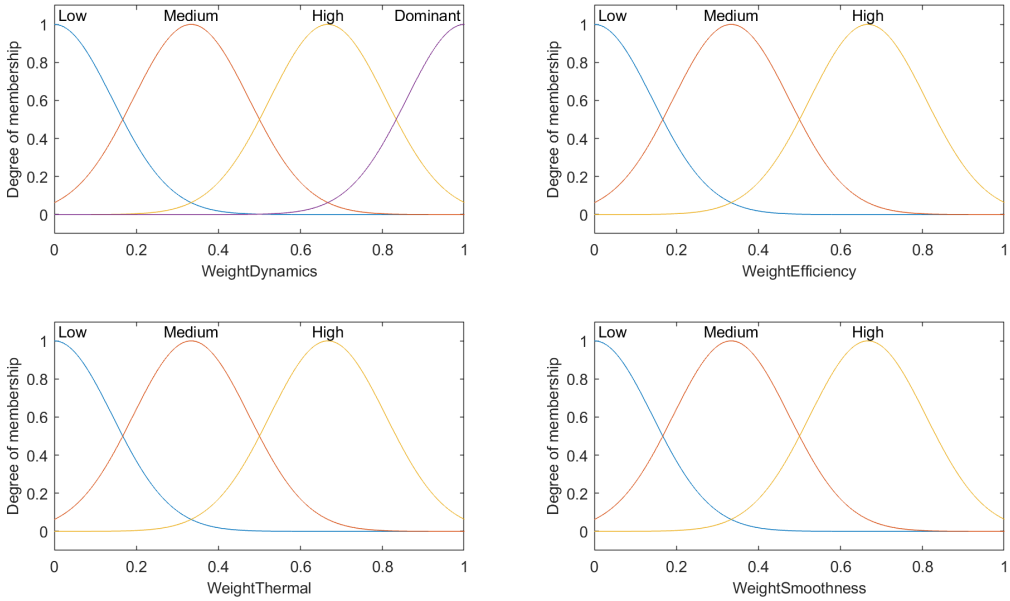

Figure 4-19. Membership functions of Fuzzy Logic outputs

```
1. If (Curve is Soft) and (Critiallity is Low) and (Torque is Low) then (WeightDynamics is
   Low)(WeightEfficiency is High)(WeightThermal is High)(WeightSmoothness is Medium) (1)

2. If (Curve is Strong) or (Critiallity is Medium) or (Torque is Medium) then (WeightDynamics is
   High)(WeightEfficiency is Medium)(WeightThermal is Low)(WeightSmoothness is Low) (1)

3. If (Curve is Limit) then (WeightDynamics is Dominant)(WeightEfficiency is Low)(WeightThermal is
   Low)(WeightSmoothness is Low) (1)

4. If (Critiallity is Uncontrolled) then (WeightDynamics is Dominant)(WeightEfficiency is
   Low)(WeightThermal is Low)(WeightSmoothness is Low) (1)

5. If (Torque is Extreme) then (WeightDynamics is Dominant)(WeightEfficiency is Low)(WeightThermal is
   Low)(WeightSmoothness is Low) (1)

6. If (Power is Extreme) then (WeightDynamics is Dominant)(WeightEfficiency is Low)(WeightThermal is
   Low)(WeightSmoothness is Low) (1)

7. If (Power is Medium) or (Charge is Medium) then (WeightEfficiency is Medium)(WeightThermal is
   Medium)(WeightSmoothness is Low) (1)

8. If (Charge is Low) then (WeightEfficiency is High) (1)

9. If (Curve is Medium) and (Critiallity is Medium) and (Power is High) then (WeightDynamics is
   Low)(WeightEfficiency is High)(WeightThermal is Medium)(WeightSmoothness is Low) (1)

10. If (Curve is Soft) and (Critiallity is Low) and (Torque is Low) and (Power is High) then (WeightDynamics
    is Low)(WeightEfficiency is High)(WeightThermal is Medium)(WeightSmoothness is Low) (1)

11. If (Curve is Soft) and (Critiallity is Low) and (Torque is Low) and (Power is Medium) and (Temperature
    is Medium) then (WeightDynamics is Low)(WeightEfficiency is Medium)(WeightThermal is
    High)(WeightSmoothness is Low) (1)

12. If (Temperature is High) then (WeightThermal is High) (1)
```

Figure 4-20. Rule set of the Fuzzy Logic function

4.6. Development of multi-objective optimization

The purpose of this function, conceptually speaking, is to perform the optimization of the torque distribution itself. Having -in the previous sections- established the surrounding functions which define and provide the relevant inputs, as well as the objectives to optimize in real time with adapting weights, this concluding building block for the scope of the optimization functionality can be developed. At this point, some major design decisions still remain to be made, fundamentally, the optimization search area and consequently the optimization algorithm type itself, and with certain dependency on this, the implementation and complexity of the fitness evaluation functions.

Architecturally speaking, this function block actually is a cluster of functions, as is illustrated in Figure 4-15, meaning that in spite of already having addressed multiple functions in the previous subsections, multiple partially independent blocks are still to be implemented. The three fundamental parts are:

- Solutions generation
- Fitness evaluation
- Optimization per se (including solution selection)

The first and most important decision is selecting an optimization approach itself. For this decision to be made, the dimensionality of the problem needs to be known. It is already known, from sections 3.4 and 3.5, that the function has:

- 3 control variables (D_{long}, D_{front}, D_{rear})
- 4 optimization objectives and corresponding fitness functions (dynamics, efficiency, thermal, comfort)

The optimization search area is still to be defined. From the vehicle dynamics perspective and considering the meaning of the control variables and their effect on the behaviour of the car, it is determined that it is not reasonable to allow an excessively widespread search area, as the value coming from the baseline Torque Vectoring is already considered as a high quality value from which no excessive deviations are desirable. Finally, a search space in a region of ±0.1 in each axis is selected. This value needs to be interpreted in the context of the internal distribution value convention of the algorithms, given by Eq. 3-1 in section 3.5. In other words, it represents a notable search window of 20% for the longitudinal distribution, and 40% for the lateral distribution on each axis. This is explained by the fact that by definition, the longitudinal distribution may vary from in the range [0, 1] at any time, but the lateral distributions have half the range, either [0, 0.5] or [0.5, 1] depending on the direction of the curve, because it is not wanted to generate a vectoring force towards in the opposite direction of the rotation of the curve. In practice, the effective search space is even smaller, because additional constraints apply: it is not allowed to apply a 100% vectoring on any side, instead, it is limited to [0.2, 0.9] for longitudinal, and [0.05, 0.5] or [0.5, 0.95] for lateral. Furthermore, additional constraints might apply, due to torque and power limitations, further compressing the search area. Consequently, the effective resolution is enhanced in most of the cases because, as will be later explained later and is illustrated in Figure 4-21, steps will be smaller for many possible solutions.

This means, that the search space and the amount of possible solutions is relatively small, especially, considering that with the fast control cycles of 5 m and vehicle dynamics in combination with the output filtering, a smaller step resolution would provide questionable benefit at the cost of considerable computational cost. Consequently, a resolution of 0.05 (worst case) is considered sufficient, which means 5 solutions on each dimension. As the number of corresponding solutions is given by the power of 3 (number of control variables) the resulting total possible solutions is restricted to:

- 5^3 = 125 possible solutions

This is a relatively small and simple search space, which hardly would justify the implementation of some kind of iterative method such as Genetic Algorithms, and not even simpler Gradient Descent and similar methods. Consequently, a “one shot” method which evaluates all the finesses in a single iteration is established. This will provide a kind of Pareto front, of which then the preferred solution will be selected.

The generation of the potential solution batch inside the search space, illustrated in Figure 4-21, is performed by a dedicated block which is constructed using a combination of Simulink™ blocks and, exceptionally, a small coded function, for the sake of efficiency of operations of vectorial nature.

One important peculiarity of this function can be noticed in the said figure: if any of the search dimensions hits a constraint, the steps separating the solutions will be reduced, in order to stay inside the boundaries without effectively losing the possible solution count. Here the additional resolution gain discussed before can be seen.

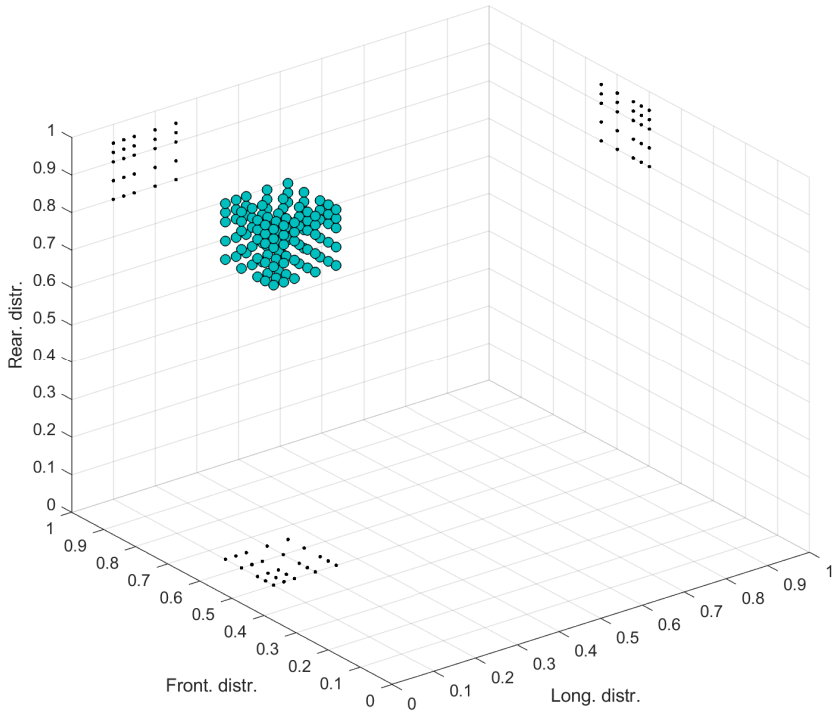

Figure 4-21. 3D representation of the search area near to the limits, where boundaries of 0.50, 0.95 and 0.90 are hit, accordingly adapting steps

Each of these 125 solutions needs to be evaluated for the four defined objectives, and its total fitness calculated according to the real-time objective weight values. The four fitness functions were implemented as different subsystems, which will be explained in the upcoming subsections. It must be noted, that many of the operations inside these functions are especially challenging to represent graphically, even in a simplified block diagram. In fact, the dimensions of the actual Simulink™ diagrams rendered the functions to be incomprehensible even on big screens, because it was not possible to represent them in a 4-dimensional vectorial manner to handle all wheels as a batch as usual, because the vectorial nature was used to handle the 125 solution arrays, resulting in massive Simulink™ diagrams. Consequently, the four fitness functions and their operation principles are described in textual form in the following paragraphs.

The **vehicle dynamics fitness function** bases its principle of operation on determining how close to the limit the grip of each wheel is. But due to the massive dynamic load transfer during cornering and acceleration manoeuvres (transiently change of the normal force the wheel is holding, as already discussed in 3.8) which directly affects the grip capacity of the wheel, this would not make sense if the value of the normal force was unknown.

As the measurement of the normal force on the wheel is not available from any conventional sensor, this is where the need for a Virtual Sensor arises, which will be developed in section 4.7. Additionally, the friction coefficient with the ground could also be estimated, but for the scope of this work the assumption was made that all four wheels are rolling on identical floor with respect to each other, which should be the case in most non-exceptional situations (such as driving over leaves, or a manhole cover, or some split-mu situation).

Receiving the input about the normal force estimation from the Virtual Sensing function, a balanced torque distribution is intended to be achieved. Here, a reasonable balance between providing more torque to the exterior wheel to enhance the yaw moment while not getting to close to the wheel's grip limit is intended. The grip limit is typically approximated by the "traction circle" (occasionally also referred to as "Kammscher Kreis" from German) illustrated in Figure 4-22, which represents the vectoral decomposition of the lateral and longitudinal forces on the wheel, being the module the equivalent to the total grip. Ideally the total grip should provide somewhere near 1g acceleration, assuming friction mu=1, which in practice would be somewhat less. [272]

Consequently, to satisfy the expectation of enhancing the yaw while not risking slip, following the principle of the vectorial decomposition of the traction circle, the function internally calculates values such as the total lateral force and longitudinal force request, with the corresponding distribution among each individual wheel. Using this information together with the vertical force, the relative margin until the traction limit is calculated. Basing on this information, two look-up-tables are used to evaluate the balance between yaw generation and slip avoidance for each wheel, which together with their relative load generates the total fitness of each particular solution from the vehicle dynamics point of view.

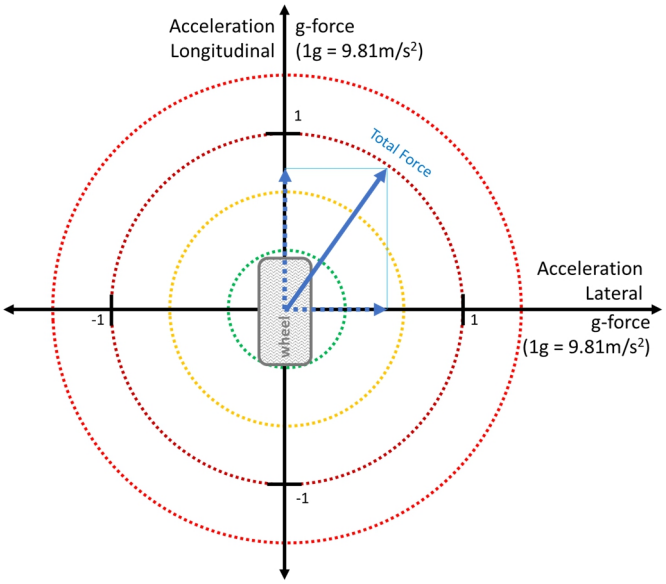

Figure 4-22. "Kammscher Kreis" representing the friction circle with value of 1g

The **energetic efficiency fitness function** is a relatively simple subsystem, especially when compared to the vehicle dynamics function. It basically implements static 3D-look-up-tables with an off-line characterization (the same as in section 3.9) of each engine's efficiency at each operating point (with respect to speed and demanded torque). From here, depending on each motor's current power load, the current global efficiency is calculated, which is directly normalized to the fitness value.

The **thermal load fitness function** is somewhat more complicated again, as it involves finding a good compromise for 8 temperatures (power electronics module and motor for each wheel). Consequently, the current situation of each of these values needs to be considered to determine a scalar fitness value.

The fundamental principle relies on a look-up-table for each of the value, which provides an exponential-like slope as the component gets closer first to its warning temperature, and then to its maximum temperature. The resulting value is not directly applied to the total fitness output summation, but instead, an additional factor is applied to consider the relative load of each of the motors. This is designed to reflect the fact that even if one motor is closer to the limit than another one, if it is receiving a much lower power demand at the moment, it should not be as relevant as some other motor which might be marginally cooler, but withstanding multiple times the power load.

The **comfort fitness function** ultimately has been implemented with a quite simplistic approach, which combines multiple different metrics basing on the control action and its previous values. Basically, the main criteria are the variation of the solution with respect to the torque distribution value currently being applied, and the trend over the previous samples. In other words, these values basically are mathematically represented by a series of filters and derivatives, which reflect both fast dynamics as well as mid-term changes, to avoid oscillations. Their calibration to achieve the desired behaviour, once again, is performed by a series of look-up-tables.

The **total fitness** is simply calculated by multiplying the normalized objective weight values with each of the fitness function outputs, and adding them together.

Finally, a relatively simple **solutions selection function** (the second of two functions exceptionally programmed in coded language), selects the preferred solution. This function selects a kind of Pareto front the three best solutions and then selects the two that best satisfy the currently most important objective. Of the remaining solutions, the solution which is closest to the value provided by the standard Torque Vectoring algorithm is selected as final solution.

This last solution selection functionality could potentially be even further enhanced by adding some additional intelligence, for instance with some Soft Computing method to reflect expert knowledge again, but this was not implemented in the scope of this work.

4.7. Development of Virtual Sensing

4.7.1. Virtual sensing function

The purpose of this function, conceptually speaking, is to provide the values of a physical magnitude for which no sensor is available in the car. Particularly, it must provide the value of the normal forces on each wheel, for the purpose of the fitness evaluation for vehicle dynamics optimization, basing on the closeness to the critical grip limit of each wheel, as discussed in the previous sections.

Consequently, this section will address another major challenge in this work: the estimation of a highly non-linear physical magnitude subject to fast variations under the influence of a wide diversity of variables of different nature, as seen in section 3.8.

4.7.2. Selection of the Virtual Sensing Solution

For the estimation of the normal forces, multiple algorithmic solutions where considered, which should satisfy the following criteria, mainly derived from the requirements in section 3.4:

- Rely on a relatively simple structure, accessible for statistical analysis.
- Not involve computationally excessively costly calculations.
- Perform with good but not necessarily excessive accuracy, prioritizing consistency (avoiding big deviations) and the previous two bullets in general.
- Offer reasonable flexibility to adapt to differing vehicle configurations, supporting a systematic re-configuration.
- Be suitable to be executed in highly parallel and non-sequential-code-based platforms.

It must be noted that an additional optional requirement was also kept in mind: thinking of future works and concurrent research activities, it would be ideal for this function to be also suitable for the prediction of values with a horizon of around 30-60ms. Such predicted values could firstly be used to adapt to delays due to the time constant of the motors, and secondly -and more interestingly- to optimize different solutions according to the predicted effect of their action. In fact, this second concept was finally pursued in a parallel research activity. This leads to notable major complexities and multiplies the function evaluation needs, all of which is not relevant for the approach in this work and certainly not in the present subsection. These concepts have been addressed in a dedicated publication [273] as discussed in 7.2, being also related to the solutions in section 5.4.

With respect to these concepts, it is understood that using the same algorithm for two different kinds of estimations is not necessarily optimal from the purely algorithmic point of view, but it would be highly convenient from the implementation efficiency point of view, having the possibility of using the same algorithm structure, especially in what respects to an FPGA implementation. Being able to estimate and predict different values with the very same implementation on the FPGA hardware (logical gates, flips-flops, etc.) by just changing the parameters, would enable a greatly improved cost efficiency of the solution. Although for this purpose theoretically potentially more adequate solutions could be considered as well, such as instance Recurrent Neural Networks (RNN) [274]–[276] or

Long Short Term Memory (LSTM) Networks [277]–[279], embedded hardware resource optimization motivates such a multi-purpose NN concept.

Basically, the possible estimation -and also eventually prediction- approaches can be divided in two big groups: those with a modelling or analytical basis on one hand, and those based on some kind of Machine Learning or similar approach.

Following diverse implications and considering their remarkable suitability for the described physical magnitude problem -and the alignment with the motivation and technological context described in chapter 1- a Machine Learning based approach is selected.

This choice was also partly supported by a fundamental comparative analysis task in cooperation with other researchers -as explained in section 5.4- which provided relevant inputs for the purpose and consequently brought additional confidence with respect to adequacy of the decision.

In particular, two representative algorithms of the Machine Learning domain were compared together with a mathematical formulation based estimation. This does not mean that other approaches were not considered as well, especially Kalman-filter based approaches [280][281][282]. But ultimately, the research was oriented to the Machine Learning concepts. Thus, the following approaches were compared:

- Standard Feed-Forward Neural Network (Perceptron)
- Standard Support Vector Machine
- Analytical simplification of weight distribution

A representative resulting plot is shown in section 4.7.3, where the outputs are compared to the ground-truth provided by the multibody simulation model described in 3.8. The particular driving situation is not the most challenging one, but still is sufficiently complex to highlight the strength and weaknesses of each method, although all can be said to provide reasonably good results.

It must be noted, that the Neural Network counted with 17 inputs and two hidden layers of 35 and 20 neuron (which is a medium size in relative terms, as will be seen in the upcoming subsections). The SVM implied a lower computational cost, with its 18 inputs and the simple subjacent vector operations involved, assuming the constant values were precalculated offline and the block-based implementation was adequately optimized. In principle the analytical model was computationally even simpler, as it just involved a few dozen of algebraic operations. But it must be noted, that it relied on the roll angle as input signal, which is a value which cannot be directly obtained either from the available signals. This means, that in order to implement it in practice in the vehicle, this signal would need to be estimated as well, which would involve firstly additional computational cost, depending on the selected method, and secondly would have an undetermined impact on the accuracy. This option was not further analysed.

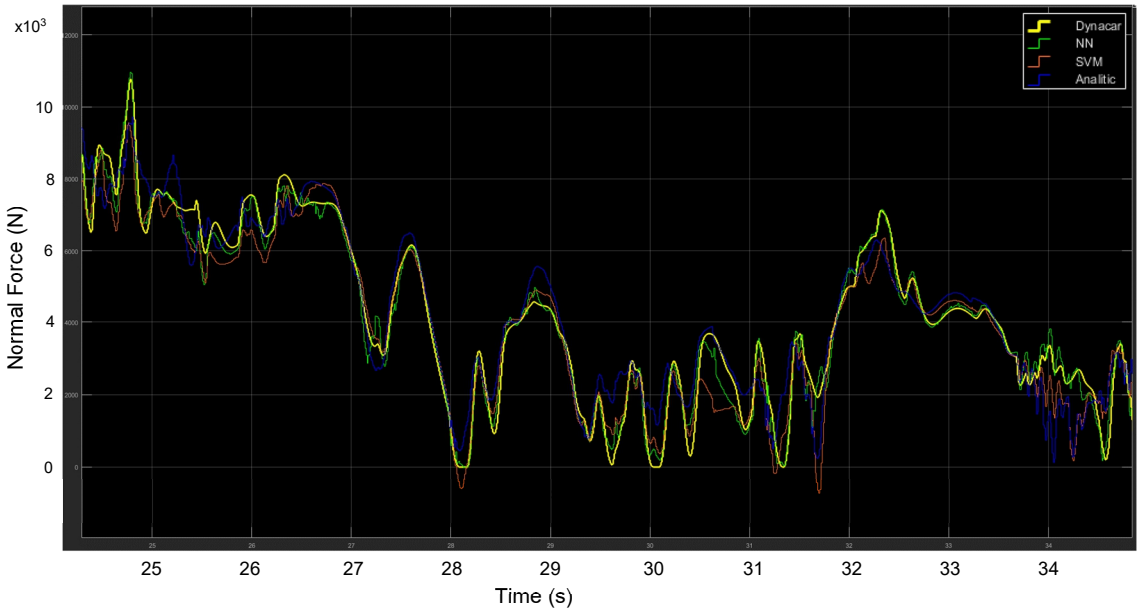

Figure 4-23. Comparison of estimation solutions: NN, SVM, mathematical model

4.7.3. Description of the selected Neural Network

Ultimately, the relatively simple Feed Forward Multilayer Perceptron Neural Network was chosen as Virtual Sensor solution for the normal forces of the four wheels. This was considered as more than adequately accurate, as upcoming results will show, and it maximizes the principle of prioritizing simplicity (although not so much dimensionality) for the multiple reasons already discussed, mainly embedded implementation and robustness/safety.

This section briefly describes the fundamentals of the proposed NN inference algorithm, the Multilayer Perceptron [193][199], for the purpose of providing a more accurate understanding for the upcoming embedded implementation discussions.

As illustrated in the flow chart in Figure 4-24, once the input signals are received, they need to be scaled to normalized values, before the execution of the actual NN can start with the hidden layers. For each hidden layer, each of its neurons needs to apply a weight to each of the signals from the previous layer (or the inputs, in the case of the first layer) and calculate the resulting sum. This can be represented as a multiplication of a vector with a matrix, but can be also implemented as dot products. Having the intermediate results of the previous layer, the bias values are added to each output. The final output of the layer is obtained after applying the activation function for each neuron. This process is repeated for each hidden layer, until reaching the output layer. This last layer does not have an activation function and will provide the results to be scaled to get the actual NN outputs.

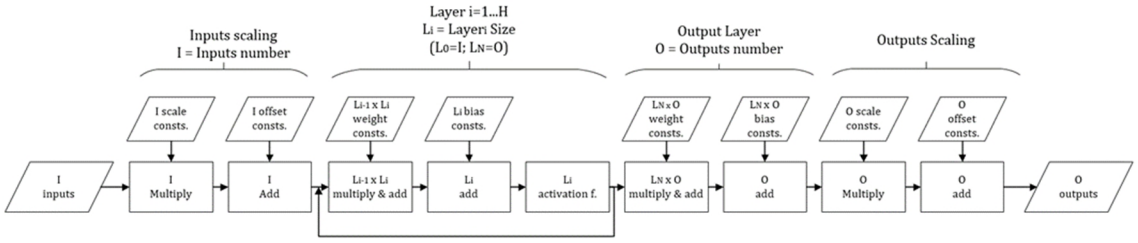

Figure 4-24. Flow chart of the generalized version of the implemented NN inference algorithm, indicating the dimensions of the data and the amount of operations

It is relevant to understand that due to the vectorial and matricidal character of the operations, the computational complexity and spatial complexity are not neglectable. The dimensions are represented as L_0 for the amount of inputs and O for the output count. Each hidden layer numbered from 1 to H with the index i, has L_i neurons, meaning that the dimension of the weight matrix will be $L_i \; x \; L_{i-1}$ and the bias vector L_i. Consequently, the count of parameters (weights and biases) and basic math operations are expressed in Eq. 4-1 and Eq. 4-2 respectively.

$$NN_{Params} = 2L_0 + \sum_{i=1}^{H} \left(L_i(L_{i-1} + 1)\right) + O(L_H + 3) \qquad \text{Eq. 4-1}$$

$$NN_{Opers} = 2L_0 + \sum_{i=1}^{H} \left(L_i(2\,L_{i-1} + 1)\right) + O(2L_H + 3) \qquad \text{Eq. 4-2}$$

The mathematical operations from Eq. 4-2 are decomposed in multiplications and additions in Eq. 4-3 and Eq. 4-4.

$$NN_{MultOpers} = L_0 + \sum_{i=1}^{H} L_i\,L_{i-1} + O(L_H + 1) \qquad \text{Eq. 4-3}$$

$$NN_{AddOpers} = L_0 + \sum_{i=1}^{H} \left(L_i(\,L_{i-1} + 1)\right) + O(L_H + 2) \qquad \text{Eq. 4-4}$$

To the previous basic arithmetic operations, the operations for the computation of the activation function needs to be added. As each hidden neuron includes an activation function, the number of activation functions equals the number of hidden neurons, as in Eq. 4-5.

$$NN_{hidden\_neurons} = \sum_{i=1}^{H} (L_i) \qquad \text{Eq. 4-5}$$

In this work a NN with a sigmoidal activation function, has been selected, expressed as in Eq. 4-6.

$$\text{Sigmoid(x)} = \frac{2}{1 + e^{-2x}} \qquad \text{Eq. 4-6}$$

Besides increasing both the addition and multiplication count by one per hidden neuron, the reciprocal division and especially the exponential function can represent a notable computation cost, depending on the platform and the implementation. To tackle this issue, an approximate function could be implemented by the means of a look-up table (LUT) with a specific amount of data points. Therefore these operations are counted separately.

The following Table 4-2 collects the theoretical complexity magnitudes –parameter and operation counts- for a few representative NN topology examples. A NN with three hidden layers L_{1-3} of 32, 16 and 8 neurons respectively with a reduced input set In of 16 is taken as reference topology. As for all other cases, the output count Out is 4 –one per wheel-, thus the topology is {16, 32|16|8, 4}. This is also taken as baseline to normalize relative complexity for other topologies. The same topology with an extended input set of 24 is also considered in the second row. The third row reflects a smaller topology, thinking of space constricted FPGAs. The remaining rows show examples for bigger topologies, illustrating the notable growth of complexity. It also illustrates the fact, that even for the same total quantity of neurons, these magnitudes change depending on their distribution.

Topology						Complexity		
In	Hidden Layers				Out	Param.	Oper.	Activation f.
(L0)	L1	L2	L3	L4	(O)			(hid. neurons)
16	**32**	**16**	**8**	**0**	**4**	**1284**	**2468**	**56**
24	**32**	**16**	**8**	**0**	**4**	**1556 (+21%)**	**2996 (+21%)**	**56 (+0%)**
8	**16**	**12**	**8**	**0**	**4**	**512 (-60%)**	**960 (-61%)**	**36 (-36%)**
32	32	16	8	0	4	1828 (+42%)	3524 (+43%)	56 (+0%)
32	32	24	0	0	4	2020 (+57%)	3908 (+58%)	56 (+0%)
32	64	32	16	0	4	4860 (+279%)	9532 (+286%)	112 (+100%)
32	128	32	16	0	4	9020 (+602%)	17788 (+602%)	176 (+214%)
32	64	64	32	16	4	9020 (+602%)	17788 (+602%)	176 (+214%)

Table 4-2. Theoretical complexity values of different NN topologies

It must be noted that the impact of the scaling operations is almost negligible, accounting for $L_0 + O$ multiply and addition operations (e.g. just 40 operations for the baseline NN), and can typically be optimized away by inlining them in some interface operation outside the Neural Network, consequently, they are not taken into account.

4.7.4. Training, testing and evaluation process

The training of Neural Networks can be said to be a challenging task which has fed the publication of different kinds of work for already over two decades now [283]–[285]. Even for the relatively small NNs developed in this paper, it can represent a considerable engineering challenge due to the many design parameters that can be tuned. And even with the high computational power of modern computers, training jobs can take multiple days to complete on a high-end personal computer. Besides, these activities involve considerable amounts of data and results that are generated and which need to be properly organised and analysed in order to achieve a solid solution.

The two most fundamental questions with major implications, 1) the input signal selection and 2) the NN topology itself (i.e. the number of layers and the amount of neurons on each layer), will be discussed in the upcoming subsections, 4.7.6 and 4.7.7. In any case, their adjustment requires an empirical approach involving abundant iterations to find the best suited combination of settings.

Besides the design of the NN itself, a few other options are also to be selected regarding the training process, mainly the data partitioning and the training algorithm, as well as the performance evaluation function. In this sense, during the first iterations of the training process, computationally less intensive options were taken: Levenberg-Marquardt backpropagation training method instead of Bayesian Regularization Backpropagation, and a smaller sample subset of the training datasets. This accelerated the training until getting a rough approach to the design dimension range that is suitable. Once the potential design ranges here identified, a more refined adjustment and analysis process was started. The basic principle is to train an extensive variety of combinations of different topologies and input signals, searching for the best possible compromise between estimation accuracy, NN size and computational effort.

For the purpose of achieving a productive and reliable design work in what respects to the training, analysis and decision making, notable effort was invested into the creation of a systematic training infrastructure. Furthermore, this should be provided as a reusable design resource for future work. For the purpose of better understanding of the upcoming steps, as well as due to the magnitude and importance of the Neural Network training, some additional details will also be discussed in the context of this section.

The training infrastructure was built aligned with the iterative steps of the development process around the Neural Network training, being the major building blocks the following, also illustrated in Figure 4-25:

- **Master configuration template**. This includes a section for the training process with documented parameters and option lists, and selectable training a validation datasets. Identically, for the definition of the design of the NN itself, it includes selectable and tuneable topology and input signal configuration lists, which are vectorized. This permits to, for instance, configure an array of topologies and another array of input signals, and launch a batch training and evaluation of the matrix resulting from each of their combinations. To ensure a stable design workflow, all the adjustments are centralized in this single file, and each batch job

configuration is explicitly linked to the outcoming NNs and result with the corresponding sequential numbering, prefixes and naming patterns.

- **Batch job launcher script**. This is the main script which launches all the other scripts which implement the dataset preparation, training, storing, analysis, documentation, and auxiliary functions, which are automatically launched.
- **Dataset preparation script**. It handles the generation of the data that is fed directly into the training algorithm. It extracts the desired signals from each of the raw data logs in the dataset, reshapes them and concatenates them into a single data matrix. It offers some options, such as the reduction of the data samples to accelerate the training without eliminating variety, as well as resampling. The same script is called also for the generation of the validation and testing dataset.
- **Driving data mirroring script**. It is designed to be called by the previous data preparation script only one time for each new item added to the dataset. Considering that the car is symmetric, it is thought to ensure that both sides (left/right) of the car are trained consistently, avoiding biases which could for instance come from asymmetric circuits and different occurrences in time.
- **Old standard dataset reprocessing script**. This script was created to update older training datasets when different changes were introduced into the newer datasets, (for example, adding derivatives or other calculated values derived from existing signals, or correcting a unit conversion error in the model).
- **NN performance evaluation script**. It enables using different ad-hoc written functions for the evaluation and metric obtention of the trained networks, against the selected testing dataset.
- **NN storage script**. It embeds the generated NN into a structure, together with all the relevant information about the settings of the NN itself, as well as the training procedure and the involved datasets and variables. This information not only facilitates the manual handling of the NNs, but it also simplifies the internal functions of the relatively complex reporting and summarizing scripts.
- **Old standard NN updating script**. Similarly to the dataset reprocessing script, this script was used to update legacy trained NNs when the structure into which information is embedded was updated. They were also used to add the resulting performance evaluation metrics, in order to facilitate comparative analysis without having to re-process test datasets and execute metric calculations again.
- **Equivalent NN performance averaging script**. As NN training is subject to randomness, identical NNs trained with the same datasets can deliver different results. Consequently, in order to avoid an unfortunate NN to distort the analysis, this script permits to combine the performance metrics of multiple equivalent NNs into a single, which can also be used for statistical analysis of different kinds. This script can additionally also be used for combining the metrics of equivalent NNs which were trained with different datasets, if they are tested with the same set.
- **Report generation script.** It calls multiple functions to generate different outputs for the reporting of the training results. It can be either called automatically after finishing a batch training job, or on demand by manually providing a list of selected previously trained networks. It provides different plots and subplots, and it also generates a detailed Excel spreadsheet which very conveniently summarizes the most important information about each network in a one-page view.

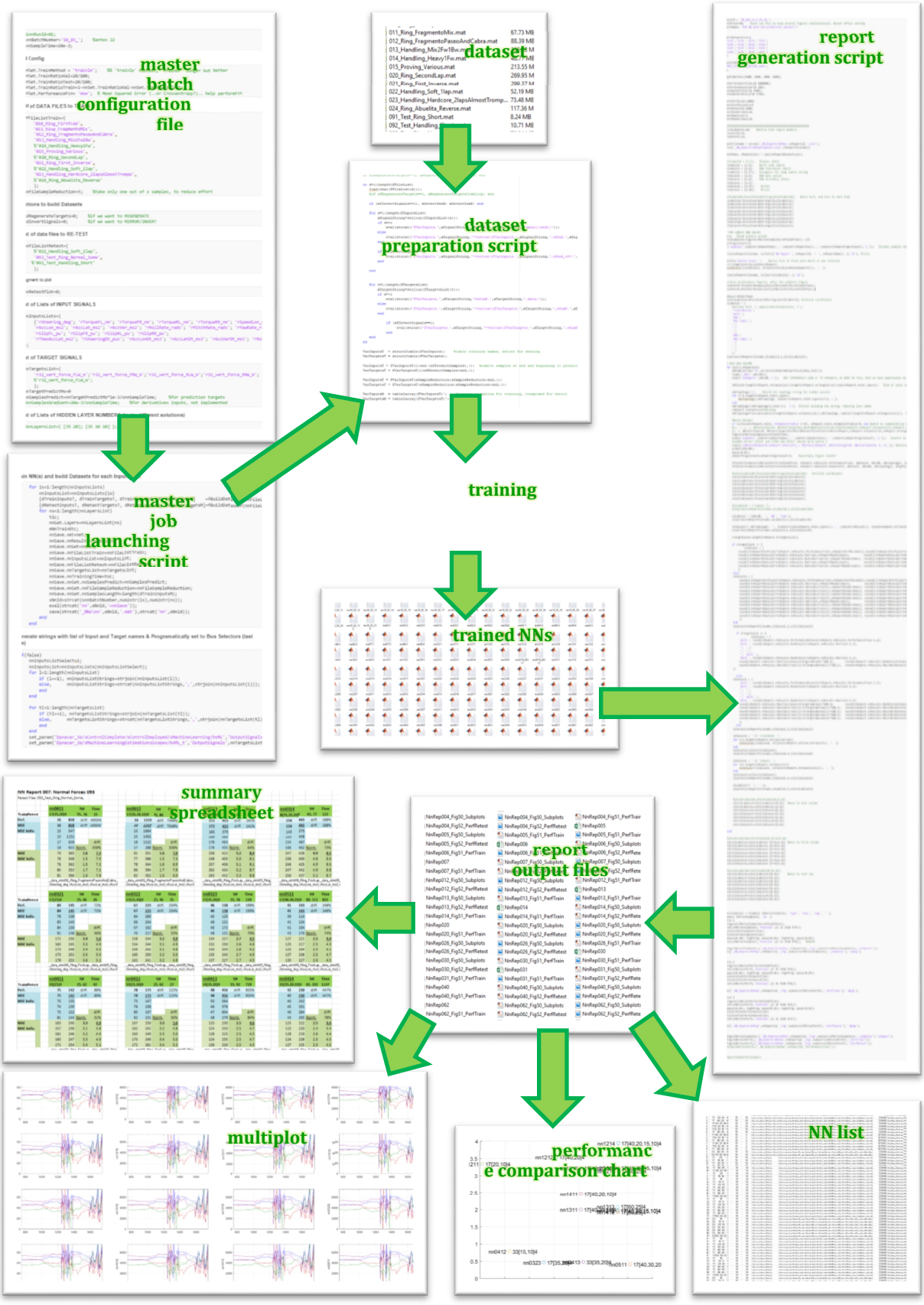

Figure 4-25. Screenshots of main parts of the training infrastructure and the outcomes to illustrate the process in a simplified manner, excluding multiple scripts (e.g. performance evaluation)

- **NN list creation script**. It creates an excel file with one row for each of the found (or selected) NNs providing a clear overview over the prediction horizon, topology, estimated variables, training inputs, training dataset, etc. of each.
- **NN deployment script**. It launches the automated generation of the selected NN in form of Simulink™ block and in form of code.
- **Model update script.** It inserts the selected NN into the general simulation model and automatically reassigns the corresponding bus selector to map the input signals required for the NN to operate.

The last relevant point to be discussed regarding training obviously is the dataset on which the entire training, testing and validation activities were based. As already anticipated in section 3.8, a highly accurate multibody vehicle dynamics model was used. This firstly permitted generating a vast dataset in a consistent manner representing a wide variety of driving situations. Furthermore, a simulated vehicle is necessary anyway, as it was not possible to equip any test vehicle with normal force sensors on each wheel, which is necessary to train the NNs. In other words, the only way to train the NNs with an unmeasurable signal, is using the multibody simulator as highly representative ground truth.

In order to give the possibility for the NN to have good generalization capabilities and adapt to the widest possible of driving circumstances, an extensive dataset covering many different scenarios, road types, manoeuvres and driving styles was created by many hours of driving by different drivers on the multibody simulator with 3D visualization and steering wheel setup. The scenarios or race tracks driven where the following, some of which are digitalized versions of real race tracks, while others are synthetically generated virtual tracks for testing purposes:

- Nürburgring (Germany) race track
- Inta (Spain) race track
- Flat (synthetic) race track
- Oval (synthetic) race track
- Proving ground (synthetic) race track

All the tracks were driven in different styles to cover the widest possible spectrum of the highly non-linear behaviour of the vehicle dynamics:

- “Slow”: normal street driving speed or below, far away from any dynamically critical limit.
- “Fast”: fast but smooth driving, aiming to achieve good lap times, with some wheel squeak but trying to avoid sliding.
- “Borderline”: driving on the stability borderline and beyond, with the vehicle sliding and partly out of control, requiring strong recovery manoeuvres.
- “Arbitrary”: performing diverse driving manoeuvres, such as driving in a slalom-like S shape, in circles, in figure eight and evasive manoeuvres, at different speeds on different grounds.

Additionally, in order to enhance the diversity of the training, test and validation data, all tracks were also driven in reverse direction, roughly duplicating the size of the training data. Besides that, in order to ensure symmetry, all datasets were mirrored (as explained in "driving data mirroring script" in the above bullets), which further doubles the dataset.

4.7.5. Selection of the input signals

The amount of available input signals for the NN is relatively extensive, as there are dozens of signals available from the vehicle sensors and its subsystems, besides a variety of signals obtained from these, such as derivatives and calculated values.

Input variable selection can turn into a surprisingly complex problem, especially when handling highly non-linear systems with an extensive amount of eligible signals, as is the case addressed in this work with the NN. The optimization of input variables with respect to criteria such as relevance, computational effort, training difficulty, dimensionality and comprehensibility, can be approached with different methods and algorithms, as discussed in [286]. In relation to this, it must be noted, that the impact of the input variable quantity and topology size not only affect the runtime computational effort, but also the offline training effort and required dataset size.

In this work, the previous considerations were included in the design and decision process, in combination with a knowledge and observation-based selection approach under consideration of the meaningfulness of the physical magnitudes represented by the variables. Therefore, no formal approach involving a concise optimality criterion. The avoidance of correlations and direct linear combinations was addressed by analysing the physically meaningful relations of the variables.

In alignment with the given reasoning, the following principles were followed:

- All vehicle dynamics signals of high relevance are to be included (i.e. vehicle speed, inertial sensor signals, wheel slips, etc.)
- Tightly coupled (or directly correlated) signals should be avoided.
- Multiple derivative signals are to be included, to reflect dynamics behaviour in time.

This design decision holds a relevant relation with the discussion from the next subsection, which is the selection of the topology itself. As changes on either side affect the outcome of the other, these evaluations where in practice combined in an iterative manner.

Ultimately, three different input sets were defined and analysed through extensive batch-wise training and performance analysis:

A. 17 inputs: minimal set
B. 24 inputs: reasonable set
C. 32 inputs: extended set

The detailed list of inputs is provided in Table 4-3.

It must be noted that the two bigger input sets were adjusted to a size which is a multiple of 8 to facilitate embedded implementation in general, probably being of special interest for a GPU implementation, due to its internal kernel and warp organization.

For an exhaustive analysis including plots regarding the performance of each option, please refer to the results in section 5.

Ultimately, a topology with the intermediate variant of 24 inputs was concluded to be the most adequate.

	Steering	Torque FL	Torque FR	Torque RL	Torque RR	Wheel Speed FL	Wheel Speed FR	Wheel Speed RL	Wheel Speed RR	Speed Long.	Acc. Long.	Acc. Lat.	Acc. Vert.	Roll Rate	Pitch Rate	Yaw Rate	Slip FL	Slip FR	Slip RL	Slip RR	Theo. Curvature	Theo. Acc. Lat.	Theo. Acc. Lat. Diff	Theo. Yaw Rate	Theo. Yaw Rate Diff	Steering Dt	Acc. Long. Dt	Acc. Lat. Dt	Acc. Vert. Dt	Roll Rate Dt	Pitch Rate Dt	Yaw Rate Dt	Theo. Curvature Dt.
A (17)	x									x	x	x	x	x	x	x					x		x			x	x	x	x	x	x	x	
B (24)	x	x	x	x	x					x	x	x	x	x	x	x					x	x	x	x	x	x	x	x	x	x	x	
C (33)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 4-3. Three different variants of NN input signal sets

4.7.6. Selection of the topology

A further aspect of the NN configuration of the NN to be defined is the topology. This involves the number of layers, the number of neurons, and the distribution of the number of neurons on each layer.

Here many considerations can be made from the NN theory point of view and the impact of different topologies for instance for the representation of non-linearities, generalization capacity, and so on.

Ultimately, keeping these aspects in mind, an empirical approach was taken to evaluate the topology, together with the input signal set discussed in the previous subsection, and the MISO/MIMO decision from the upcoming subsection. This was achieved in an efficient manner exploiting the highly automated infrastructure described in 4.7.4, enabling to launch big batch training and evaluation jobs.

Assuming that a single NN should provide the information for the normal forces on all four wheels, a multi-layer topology was considered as interesting. Nevertheless, single layer topologies were taken into consideration in the analysis as well. Multiple topology sizes were evaluated across an adequate range of dimensions, for each of which the same amount of neuron was distributed across one or multiple layers in different manners (i.e. hidden layers as [50] or [30 20] or [15 15 0]).

For an exhaustive analysis including plots regarding the performance of a selected subset of the vast amount of possible solutions, please refer to the results in section 5.

The results illustrate the entire boundary of reasonable solutions, including lower and higher extremes in what respects to neuron count and network depth. Ultimately, the best balanced middle region, aiming to keep a good compromise between accuracy and computational cost, is addressed. The selected topology being: 24 inputs, hidden layers with 30 and 20 neurons, 4 outputs.

4.7.7. Selection of MIMO or MISO

Before concluding, it may be mentioned as well that it was evaluated to re-structure the NNs, from one single MIMO NN for all wheels, into four MISO NNs, one for each wheel.

It was thought that this could have enhanced the estimation capacity of each of the NNs while reducing their size. Another consideration in favour of multiple MISO NNs could be the implementation on FPGA. Especially under the risk, that the NN might face issues to fit onto the selected SoC, as will in fact be seen in the upcoming section 5.4.

Nevertheless, the empirical results discussed above show that this approach is in principle not worth it. The savings in computational cost ($\sim 1/2$) are not sufficient to compensate the fact that the NN would be needed to run four times.

The interpretation of this conclusion is that the physical magnitudes of all four wheels are relatively closely coupled from the vehicle dynamics point of view. Simplifying, the four wheel present complementary magnitudes: when some goes up, others go down. Besides, two wheels on of different sides fluctuate similarly: when turning to one side, the wheels on that side get unloaded while the ones on the other get both loaded. Idem when braking and accelerating, with the front and rear end. Of course, to this simplified behaviour description, the challenging effects of the transients need to be added, but from a structural point of view the coupling can be assumed as such. Correspondingly it is understood that a significant part of the internal numbers being calculated in the hidden layers are to an extent similar for all cases, meaning that using a same topology for all wheels simultaneously should turn out to be more efficient, as similar calculations can be reused.

For an analysis including plots regarding the performance of each option, please refer to the results in section 5.

4.8. Development of mechanisms for robustness

The system architecture, the controller architecture and the individual subfunctions have been carefully conceived to reduce the degree of uncertainty, facilitate their analysis and mitigate the impact of an eventual prediction malfunction, as repeatedly discussed in multiple previous sections. This inevitably induced the algorithms to be designed avoiding unnecessary complexity, not only to enhance computational efficiency and reduce embedded device costs, but also to enhance robustness and furthermore facilitate the supervision of their intended behaviour.

In what respects to concerns regarding the usage of a Machine Learning algorithm, a fundamental consideration to be highlighted is the following: a malfunction of a Virtual Sensor should be considered similarly to a malfunction of any conventional physical sensor: both can potentially suffer some kind of malfunction and need the corresponding support mechanisms to detect, contain and mitigate the effect. For instance, limiters of different kind and plausibility checks to detect excessive value variations or unrealistic values.

A further fundamental consideration, of architectural nature, is that the designed optimizing algorithm is conceived to provide an enhanced set-point over the baseline Torque-Vectoring algorithm, aiming to reduce the slip, but it must not be permitted to cause critical situations. It might happen that anyway certain slip occurs, in the same way slip can happen with a generic Torque-Vectoring or simply a normal torque distribution. The worst-case scenario would be some other unwanted effect which could eventually affect the stability of the vehicle. But in such an unlikely situation, a superior layer of conventional traction and stability control functions –i.e. TCS and ESP- must override the torque set-points.

Nonetheless, aiming to mitigate the risk of stability systems having to intervene in such a worst case scenario and following discussions from sections 3.4, 4.1 and 4.4, a diversity of functions have been placed across the controller, in order to implement such mechanisms and enhance the robustness and stability both of the estimations and the control actions.

The functions that have been implemented include partly rather simple elements such as saturators and rate-limiters, but also more elaborate functions that detect excessive fluctuations in multiple time windows. Furthermore, a simplified mathematical model to estimate wheel forces is used to enable plausibility checks. Whenever unexpected values are detected, logical functions trigger an error and smoothly fall back to the default Torque-Vectoring values. The threshold to trigger an error is three bad samples in a window of 5 samples.

In conclusion, besides the diverse functional blocks to handle the values of diverse signals, such as absolute saturators, dynamic saturators, rate limiters, an additional top-level layer has been placed for purely supervision and intervention purposes. In other words, this block is not part of the control flow per se, in the sense that it does not directly modify the value of the signals. Instead, it checks the calculated signals for correctness and triggers a fallback to the baseline controller if unexpected values are found.

For simplicity and better clarity, the implemented checks are enumerated as follows, with some additional clarification. It must be emphasized, that the checks are applied at two points: one is the output of the NN normal force estimation values, to avoid implausible data to be fed into the controller. The second, is the output of the controller itself, to avoid any erratic behaviour brought by any of its inputs or its optimization functions.

Supervision of NN signals:

- Lowpass filters (basic blocks) + Rate limiters (basic blocks)
- Absolute saturators (basic blocks)
- Mathematical plausibility function (subsystem)
- Oscillation detections (subsystem)
- Bidirectional integrators (basic blocks) + dynamic saturators (basic blocks)

Control action:

- Rate limiters (basic blocks)
- Absolute saturators (basic blocks)
- Dynamic saturators (basic blocks)
- Bidirectional integrators (basic blocks) + dynamic saturator
- Oscillation detection (subsystem)
- Bidirectional integrators (basic blocks) + dynamic saturators (basic blocks)

All the functions have been calibrated basing on data analysis and tests in order to avoid false positives but provide sufficient detection sensibility.

For validation purposes, the functions where successfully validated with multiple fault injection tests, meaning that erroneous values were spontaneously fed into the signals during runtime, in order to test the detection.

4.9. Embedded implementation

4.9.1. Processor-based implementation: Phase 1, baseline Torque Vectoring

The first embedded implementation effort, which was fully in the scope of this work, was made during the first phase of the Torque Vectoring development. Basically three fundamental milestones were accomplished here, and it consequently served as a proof of concept for multiple points and as basic enablers for future tasks of this work and potential future works.

The first major milestone was to establish the framework for the execution on the selected Xilinx Zynq® 7020 SoC embedded platform. This means, achieving an operational implementation of the elemental building blocks around the actual algorithm to be implemented, which is later to be integrated into some kind of periodically called function with input and output interfaces.

A low-level bare-metal approach was followed for this implementation, relying on periodical timer-based function calls with different priorities, implemented using the Xilinx Vivado™ and its additional SDK environment.

This meant that additionally low-level CAN functions needed to be implemented for the CAN-based input and output interfacing, because there was no availability of Simulink™ blocks or automatically generated functions, and the available Xilinx CAN-libraries were for Linux runtimes. Besides adapting sample code for the processor's CAN module, the CAN frame encoding and decoding functions had to be written. This implies that the raw data frame with up to 64 bits that is transmitted by CAN needs to be correctly handled to physically meaningful variables in the software. In other words, basing on the bit mapping and signal scaling definition in the DBC file, the corresponding value offset and scaling factors need to be applied for every frame, decoding each message for reception, and encoding for transmission.

Additionally, CAN calibration and monitoring capabilities were also implemented, which in contrast to the previous might be seen as optional, but are highly convenient for a productive development workflow, especially on the race track. Otherwise, for each parameter change, code would need to be re-compiled (and probably even re-generated) and re-programmed onto the target. Having these calibration functions enabled to adjust selected parameters directly over CAN during runtime.

Both previous building blocks were harmonized under the umbrella of an additional enhancement to the infrastructure: a self-written code generator the described CAN encoding, decoding, initialization, calibration and monitoring routines, plus for some additional functions, such as consistent automatic model and HiL parameter initialization. Taking as input a CAN signal specification file, this code generator created 5 output files of different kind -source code .c and .h files and Matlab™ .m scripts- containing all the previous functions. A relevant benefit of this is that it facilitates any modification to the interfaces and furthermore guarantees consistency across multiple work products implemented in different places -Vivado™ and Simulink™-, clearly accelerating the development workflow and reducing exposure to inconvenient error variables.

Besides the peripheral CAN communication solution, internal communication also needed to be implemented for such a heterogeneous embedded platform. AXI interfaces were configured to interface the data exchange between the processor cores and the FPGA. Besides, a very basic FPGA test function was implemented for solution validation purposes.

Besides the achieved implementation discussed above, multiple alternative implementation approaches were evaluated and even partly developed as well. One of them based on using the fully Simulink™-based deployment approach, which is based on a specific Linux runtime. Another approach also used Linux, but being entirely Xilinx-based. Besides determinism and criticality concerns, as well as toolchain immaturity at the time, both implementation attempts were deprecated when the bare-metal solution succeeded. This involved greater effort than the other higher level solutions, but also provided greater control about many implementation and runtime aspects, and forced to acquire a better understanding of the inner workings of the platform.

The second milestone was the establishing of the entire simulation and development framework, supporting up to real-time HiL with the physical ECU. The simulation setup and the associated methodology have already been extensively discussed, across the corresponding sections of chapter 3 and 0, thus redundant discussions in the subsection will be avoided.

The third milestone was the actual implementation of the first algorithm iteration, namely the baseline Torque Vectoring Controller. For this to be possible, the two previous milestones were a necessary technical dependency. Here the model-based Simulink™ framework is exploited to automatically generate functional program code of the controller. Following successful deployment and implementation on the embedded platform, the correctness of the solution was firstly validated using the real-time HiL framework.

Lastly, this solution was deployed in the car for race track tests, actually achieving the final milestone.

Please note that additional details about the achieved results related to these milestones are further discussed in the upcoming chapter 5.

Inevitably, as in any real-world embedded implementation activity, in spite of the sophisticated and highly automated toolchain, a notable amount of time was invested into this first iteration as diverse unexpected technical and tool-related obstacles had to be overcome. In any case, it was considered a necessary part of this work. Besides, it highlighted the fact of the notable initial effort -or learning curve- that needs to be invested into such kind of tasks, and the fact that sophisticated toolchains can be a double edged sword: they enable great development potential once the setup is consolidated, but as long as the setup -or the product themselves- is not fully mature, time-consuming major setbacks can arise. In fact, a very clear example of such a situation can be seen in the next subsection.

4.9.2. FPGA-based implementation: Phase 1, Advanced Controller

As anticipated in the previous subsection, when using the Matlab™-Simulink™ HDL-Coder based approach for FPGA hardware description code generation, several limitations

where detected, which serve as a notoriously representative example of the intricacies such ambitious implementation approach can imply.

These limitations are attributed to the fact that the HDL-Coder toolbox was originally rather oriented to the implementation of simpler algorithms, typically involving bitwise arithmetic, logical operations, basic math operations and at most signal filtering using fixed point variables.

Although, probably driven by the current technological context discussed along this work, MathWorks® is clearly making an effort to extend relevant features, important capabilities for the scope of this work are still missing. This means that achieving an efficient implementation is currently excessively complicated, but the expectation is that this should not be such a blocker issue in upcoming releases. Although this cannot be discussed due to NDA, even looking at release notes it can be assumed that some relevant implementation possibilities, are upcoming. In fact, some such features have started to be available post-2017 releases.

The first relevant example for very recently added fundamental capabilities is the AXI-slave interface, which was included in release 2017a. Several floating point operations and capabilities where also introduced in 2017a, as well as the capability to reuse certain blocks. Many important interfacing and basic matrix and vector handling capabilities where also introduced in 2017a and 2016b. It also wasn't before 2016b, that native floating point and optimized implementations for many elemental operations and filters were added. In other words, until shortly before the closure of the first phase of this work and the corresponding implementation tests, not even the most rudimentary vector-based algorithm could have been implemented.

Focusing again on the limitations impeding a reasonable implementation for this work, one major limitation is the fact the Simulink™ "while iterator" block is not supported. This was the most convenient solution to implement the layer reutilization in order to achieve a significant resource utilization reduction.

Another limitation is the lack of support for non-element-wise matrix multiplications.

M-functions, which can be used as workaround for some other limitations (especially the while-loop limitation) present yet a further limitation: they are incompatible with floating-point signals.

The matrix multiplication limitation could be overcome by using and array of dot product blocks, one per neuron in each layer. The downside of this is that it is very prone to generate a very resource-consuming implementation.

A better solution for the previous is presumed to be available in some upcoming Matlab™ release, by using the new multiply-accumulate and multiply-add blocks. This would be functionally equivalent but provide an optimized implementation, as these blocks are designed for an efficient mapping on the DSP slices of the FPGA. Furthermore, these blocks would permit to configure serial implementation for enhanced resource usage

accelerated by pipelining, or parallel implementation, for greatest throughput. But still the resource utilization may not necessarily be ideal.

The matrix signal handling limitation can be overcome by serializing the matrixes and reconstructing them using the blocks.

The floating-point interfacing limitation can be conveniently avoided by converting the model to a fixed-point implementation. This would furthermore enhance the resource utilization and timing performance. But this would slightly reduce the accessibility of the solution, as it would require additional steps to deploy different neural networks and it limits the NN implementation reusability.

This shows that it is technically not impossible to implement the solution, but it can be anticipated that implementing such workaround will inevitably generate some new drawbacks, such as questionable efficiency and the loosing many of the benefits of the MBD approach (excessive complexity, strongly ad-hoc solution, loss of modularity and significantly lower abstraction). In other words, it would be probably contradictory to do an MBD implementation in this manner, making it more reasonable either to:

- Wait for these limitations to be overcome with one of the following Matlab™ releases.
- Use a C code-based implementation in combination with high-level synthesis as alternative solution.

The second point is the most reasonable approach to obtain a widely optimal and restriction-free NN implementation on the FPGA. It is expected to permit to keep good modularity and the model-based abstraction by using the automated IP implementation methodology. In fact, a solution aligned with this approach was eventually explored in a joint research activity, as further explained in section 5.4.

CHAPTER 5:

Results

"Your data is just as important as your model:
If bullshit in, then bullshit out"

H. Winner

The results chapter kicks off with a recap of already published results from the MBD-HiL workflow that culminated with race track test of the real embedded implementation of the baseline Torque Vectoring algorithms, integrated in a motor-in-wheel vehicle.

Followingly the results focus on the second development phase and the core contributions brought by this work beyond those first phase tests, firstly in what respects to the Neural Networks by themselves and their capabilities as Virtual Sensors for vehicle dynamics.

Secondly results focus on the analysis of the entire algorithmic solution as a whole, constituting the advanced real-time optimizing Torque vectoring controller, showing striking enhancements to performance and behaviour.

The chapter is wrapped up with relevant highlights of embedded implementation on highly parallel heterogeneous devices from a partnership work, and finally providing a summary and graphical overview of the achieved solution.

5. Results

5.1. Race-track tests with real MiW vehicle

As mentioned in various occasions across this work, validating and showing real-world applicability of the proposed solutions was considered of importance and correspondingly notable effort was invested exploiting the available resources for the purpose of algorithm testing/calibration/validation and data acquisition for modelling.

For these purposes, the demonstrator vehicle of the Eunice European project was reused. In the context of an internal follow-up project, the author lead an upgrade stage to the vehicle was in order to integrate torque-vectoring. Additionally, instrumentation was enhanced, a panel with knobs and switches for calibration was built and most importantly, the power was slightly increased.

Figure 5-1. Strong cornering applying baseline Torque Vectoring on the Eunice vehicle in its original state at Tecnalia's facilities (left) and in upgraded at Los Arcos in Navarre-Spain (right)

This vehicle was taken as reference for the modelling of fundamental components of this work, namely the vehicle dynamics and the powertrain model, including thermal behaviour and efficiency. This has been extensively discussed in the corresponding subsections 3.1, 3.8 and 3.9. It must be emphasized, that these tests correspond to the first phase of this work, i.e. the baseline. The more powerful 4WD vehicle including the 3Ccar rear axis which was defined for the advanced controller in phase two, an entirely model-based approach was followed, with results discussed in the remaining sections.

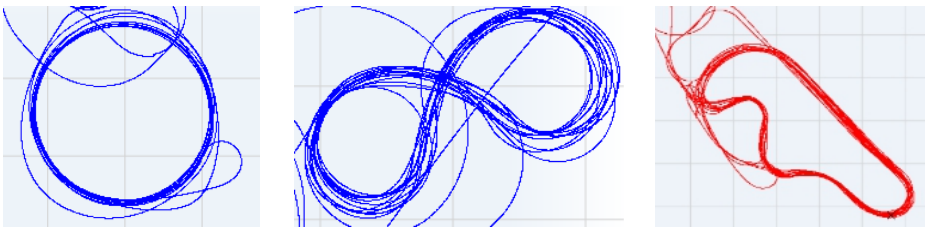

Figure 5-2. Plot of the GPS traces of different tests at Los Arcos Circuit

The achieved race-track test results were widely positive. The vehicle's handling improved as expected. The notably understeering character of the original vehicle was enhanced to a mostly neutral character. Under aggressive manoeuvres it even turned slightly oversteering. The responsiveness and cornering capability was notably enhanced. This was especially perceivable under subjective assessment. Furthermore, slip in the curve interior wheel was drastically reduced and the exterior wheel was forced to exploit the transiently available extra normal force given by the dynamic weight redistribution.

Measurement graphics also confirmed the enhanced handling behaviour. The yaw rate plot in Figure 5-3 shows the sharper responsiveness and slightly higher values in general. Table 5-1 illustrates the improvement of all numeric metrics. The weaker point was the speed, but this is to be attributed to the relatively low maximum torque and power of this vehicle in particular, being this the limiting factor rather than the available grip, which was relatively high and often could not be not fully exploited.

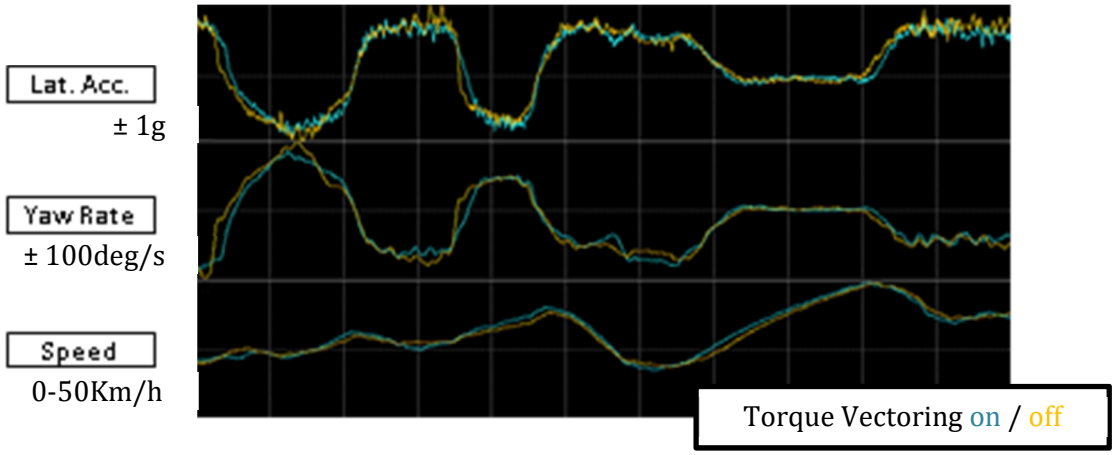

Figure 5-3. Baseline Torque Vectoring performance on Eunice vehicle at high grip handling circuit

Metric	**Test (dry)**					
	Skid Pad		***Figure 8***		***Handling Circuit***	
Torque Vectoring:	off	on	off	on	off	on
Lap Time (s)	-	-	13.63	13.37	24.61	24.68
Max Speed (km/h)	52.2	53.1	-	-	-	-
Max. Peak Lat. Acc. (g)	-	-	0.92	0.95	0.91	0.93
Max. Sust. Lat. Acc. (g)	0.83	0.85	0.82	0.84	0.80	0.83
Max. Yaw Rate (deg/s)	33	34	51	58	63	71
Advantage for TV	yes		yes		yes	

Table 5-1. Baseline Torque Vectoring performance on Eunice vehicle (Los Arcos)

A further step was taken for additional testing of the vehicle's and the baseline algorithm's performance under extreme and low grip conditions, again upgrading the vehicle and taking it to OEM and TIER1 Winter Testing facilities in Sweden. These tests provided complementary data for the discussed purposes.

Besides, these low-grip conditions where used to validate the TCS control function developed, which has been integrated into the model as discussed in section 3.10.1.

Figure 5-4. Baseline Torque Vectoring testing on Eunice vehicle under extreme conditions at the Colmis Winter Testing Track in Arjeplog-Sweden

On snow, as expected from more extreme conditions, the vehicle highlighted the character of both cases more acutely: stronger understeer without Torque Vectoring and noticeable oversteer when turning the controller on. The detailed analysis of this assessment is shown in Table 5-2 and Table 5-3.

Handling criteria	**Dry tarmac testing conditions**			
	No Throttle	***High Throttle***		
		No TV (50/50)	***Standard TV***	***Aggress. TV***
Cornering speed	N/A	Medium	High	High
Curve entry	Slight Underst.	Understeer	Neutral	Incisive
Handling	Slight Underst	Understeer	Neutral	Oversteer
Wheelspin (in/out)	N/A	Slip Grip	Grip Grip	Grip Grip
Acceleration exit	N/A	High	Med-High	Medium
Subjective perform.	N/A	Normal	Very High	High
Subjective agility	N/A	Normal	High	Very High

Table 5-2. Qualitative criteria for baseline Torque Vectoring on tarmac (Los Arcos)

Handling criteria	**Slippery surface testing conditions**			
	No Throttle	***High Throttle***		
		No TV (50/50)	***Standard TV***	***Aggress. TV***
Cornering speed	N/A	Lowest	Highest	Average
Curve entry	Understeer	Strong Underst.	Slight Underst	Understeer
Handling	Understeer	Strong Underst.	Slight Underst	Oversteer
Wheelspin (in/out)	N/A	Slip Slip	Grip Grip	Grip Slip
Acceleration exit	N/A	High	High	Medium
Subjective perform.	N/A	Low	High	Good
Subjective agility	N/A	Low	High	Good

Table 5-3. Qualitative criteria for baseline Torque Vectoring on snow (Winter Testing)

A further positive outcome of these tests was the successful implementation basing on the conceived MiL/HiL/Deployment as discussed in section 4.3. The resulting practical result is that algorithm modifications could directly be deployed into the ECU using code generation. Then the infrastructure permitted the implementation to be tested with the entire vehicle model including multibody simulation in real-time with HiL. If this test was successful, the ECU could be directly unplugged from the desk and plugged into the car.

Further details and deeper analysis around these results have been published in [49] and [287].

5.2. Neural Networks

In order to achieve the presented Neural Network solution, a wide variety of different solutions and adjustments were evaluated in this work. For instance, adjustments for aspects such as training method, data selection, performance metrics, stopping criteria, and so on, are to be made. The variant number further grows with the design decisions corresponding to the NNs themselves, fundamental choices being the input signal selection, network topology and dimensions of each layer, as already discussed in section 4.7.

Many challenges -of similar nature to those faced when working with vehicle dynamics models and complex controllers- arise when digging into exhaustive analysis regarding NNs. The most obvious is the mentioned vast amount variables and corresponding variant combinatory, which further increases the problem of long computation times, as training times tend to be an inconvenient with NNs. Besides the amount of inputs to the training jobs, the amount of outputs -i.e. the generated information- promptly reaches borderline unmanageable magnitudes. In contrast to the case of vehicle dynamics simulations, disk usage is not so critical with NN, but keeping overview might turn into a problem instead. This work resulted in training over 1300 NNs, for each of which dozens of settings and output metrics had to be analysed in order to take design decisions for the next development iteration.

An additional challenge comes from the usual randomness of the NN training process. This implies that -in contrast to vehicle model and controller simulations- running a same input dataset and parameters multiple times, generates a different output each time. Consequently, each NN solution was programmed to be trained multiple times, calculating the average performance representing multiple results of the same topology. Inevitably, results often showed outliers with exceptionally bad performance. Such an example is illustrated in Figure 5-5, showing and outlier not only for small -and potentially less accurate- topologies, but also for a big NN. Such outliers certainly distort the average representations as well and, therefore, need to be handled and discarded.

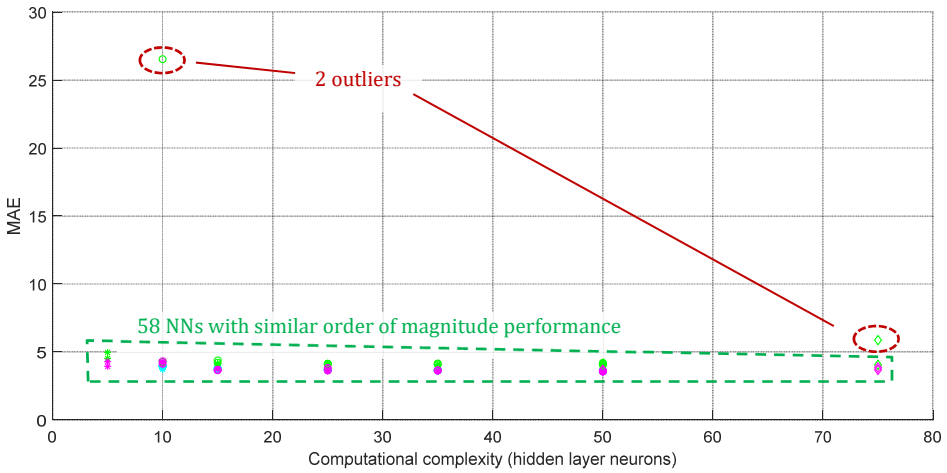

Figure 5-5. Unwanted NN training effect: outlier NNs with exceptionally poor performance for both small and big solutions (left and right) while good solutions lay on lower side

Another typical issue also resulting from the training randomness, is that occasionally the training fails to converge even after an excessive number of iterations. The consequence is that the training job, which typically takes between 5 minutes and 2 hours -depending on a variety of factors such as the size of the NN and the dataset- can get stuck for over 12 hours. This not only generates a disposable result, but also disrupts batch training jobs.

To address some of these issues and implement a relatively efficient and systematic workflow -in line with the repeatedly discussed philosophy of this work- a wide variety of scripts, functions and subroutines of different nature were created for the training, analysis, plotting, documentation and organization tasks. The resulting infrastructure reached 27 .m MatLab™ files containing the over 2000 lines of functional code implementing the necessary functionalities. The most relevant fraction of these is briefly explained in section 4.7. For the sake of result analysis, the automatic generation and storage of figures, reports, summaries, overview spreadsheets and even logs, is fundamental. An example NN overview table can be seen in Figure 5-6, as well as in Figure 4-25 of the said section.

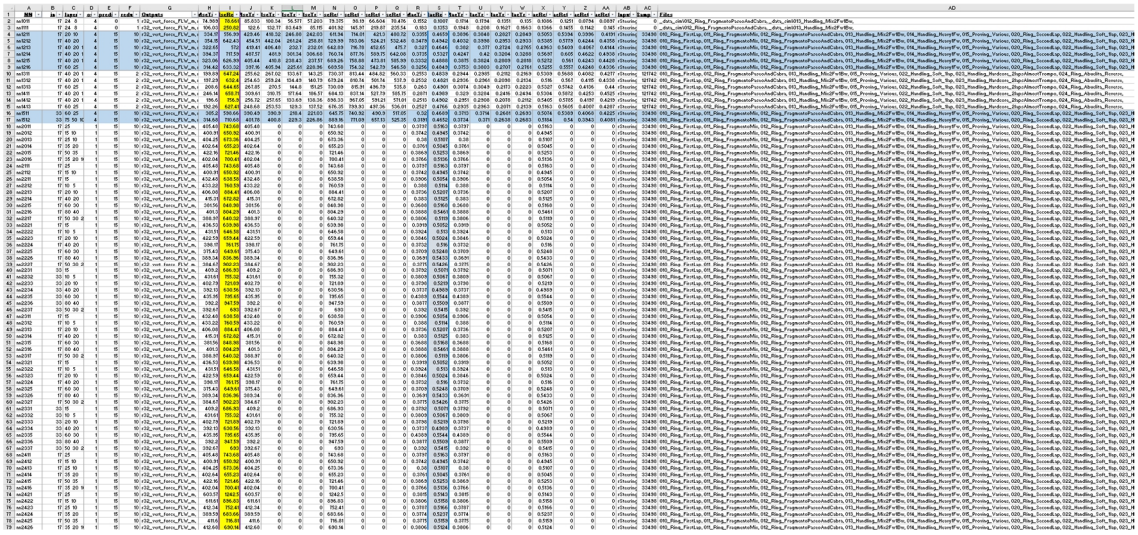

Figure 5-6. Example of dimensions of automatically generated Excel spreadsheet containing roughly 5% of the total NNs trained and subset of the analysis criteria (columns)

Proceeding to the results of main interest, in what respects to the NNs themselves, Figure 5-7 illustrates a collection of a variety of network topologies selected from different batches but trained under the same conditions. These can be seen as the most relevant results from the NN development, which lead relevant conclusions.

An important note is that as for all other analysis and metrics, different datasets were used to firstly train, and secondly re-test the NNs. The following graphics and metrics were generated with this second independent re-test dataset.

For the major part of the analysis the MSE (mean squared error) metric was payed primary attention to, mostly in order to apply a greater penalty on bigger errors, besides its frequent convenience for visualization purposes.

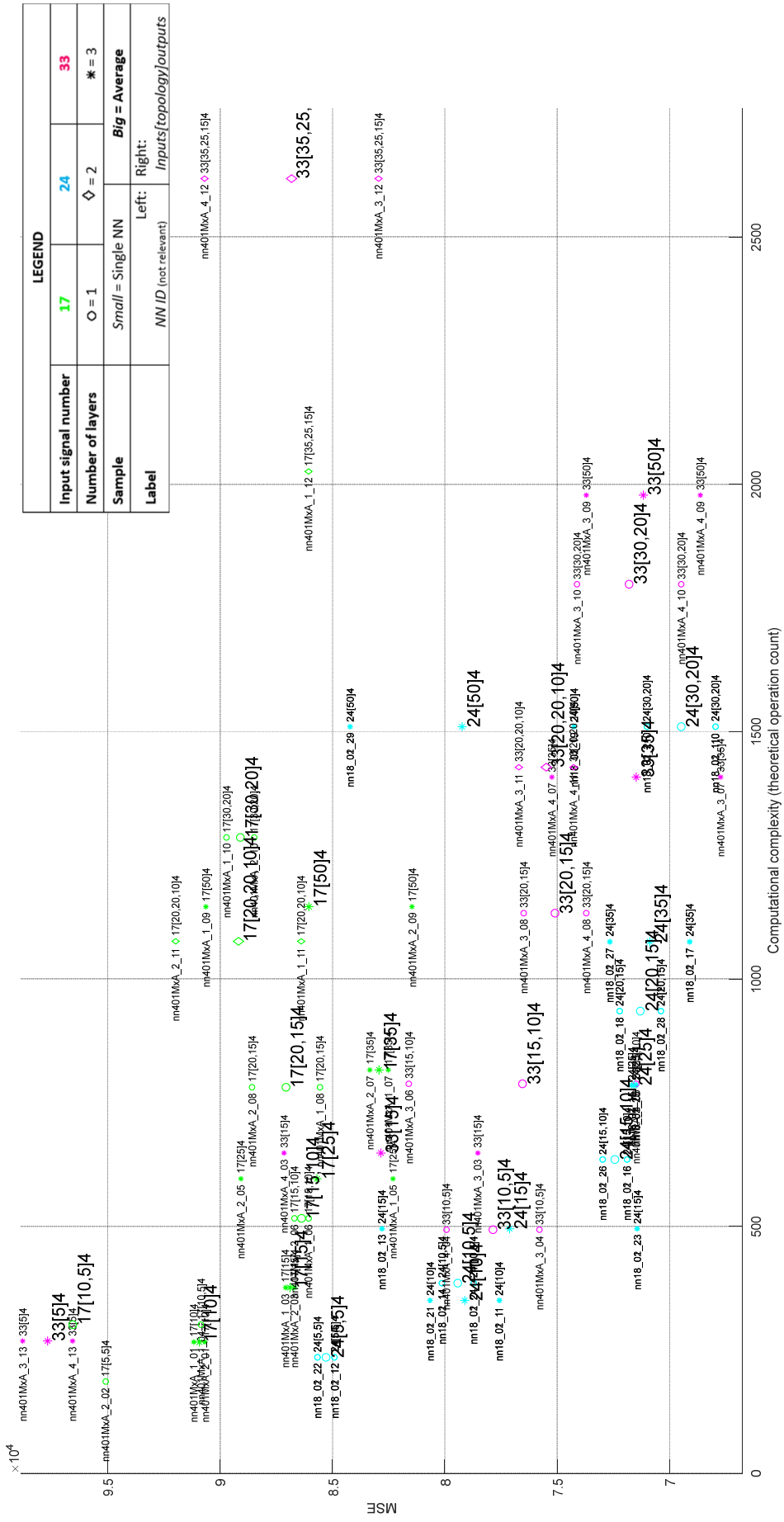

Figure 5-7. Performance of NN topologies with four outputs (bigger labels represent average values for multiple samples with identical configuration)

The previous visualization in particular actually is a crop for legibility purposes. This means that multiple NN samples and average values mostly of the smaller NN solutions -which are less capable- and even some of the bigger and deeper, are beyond the upper plot boundary.

Another point to provide a more relevant visualization is to represent the complexity of the NNs -always shown in the X axis- according to the corresponding theoretical computational operation count. This computational complexity is extensively discussed in subsection 4.7.3. An alternative representation, for instance used in Figure 5-5, is to represent the NN complexity by the sum of the neuron count, considering the hidden neuron topology.

Proceeding to the overall result analysis, the first observation is the fact that a bigger network is not necessarily better, although many smaller topologies present dominantly worse results. In this figure, a clear trend can be observed: initially error tends to decrement as NNs get bigger, but after a relatively flat region, results tend to worsen again. And several big and deep NNs do in fact show surprisingly poor accuracy.

A relevant observation is that the topologies with 24 inputs consistently show better results. It can be seen to be a balanced compromise with respect to the insufficient 17 input approach. In what regards to the 33 input approach, unexpectedly inconsistent results can be found. In general terms, these additional inputs are concluded to not be worth the additional computational cost and could even be said to be counterproductive.

Another observation from these results is that the topologies with two layers seem to show the most consistent results and appear as the best compromise solution.

Nevertheless, in order to provide a clearer perception of the behaviour of the involved signals, as well as their physical meaningfulness, Figure 5-8 shows the real-time estimations of a selected subset of NNs in a dynamically challenging situation, which means that highly non-linear phenomena are happening and estimations become significantly more challenging for the NN.

This subset of NNs selected for their representativeness is summarized in Table 5-4. It includes not only the best but, to provide a contrast, also some less good examples.

NN Topology	Comp. Operations	Δ	MSE	Δ	MAE	Δ
17[35]4	816	-46%	82517	21%	202	14%
17[30,20]4	1286	-15%	88480	30%	202	13%
17[35,25,15]4	2026	34%	86060	27%	203	14%
24[15]4	495	-67%	71444	5%	181	2%
24[35]4	1075	-29%	69106	2%	180	1%
24[30,20]4	**1510**	**0%**	**67956**	**0%**	**178**	**0%**
33[35]4	1408	-7%	67732	0%	180	1%
33[35,25,15]4	2618	73%	82948	22%	192	8%

Table 5-4. Subset of particularly representative NNs with main attributes and metrics, including delta (Δ) with respect to selected solution in bold

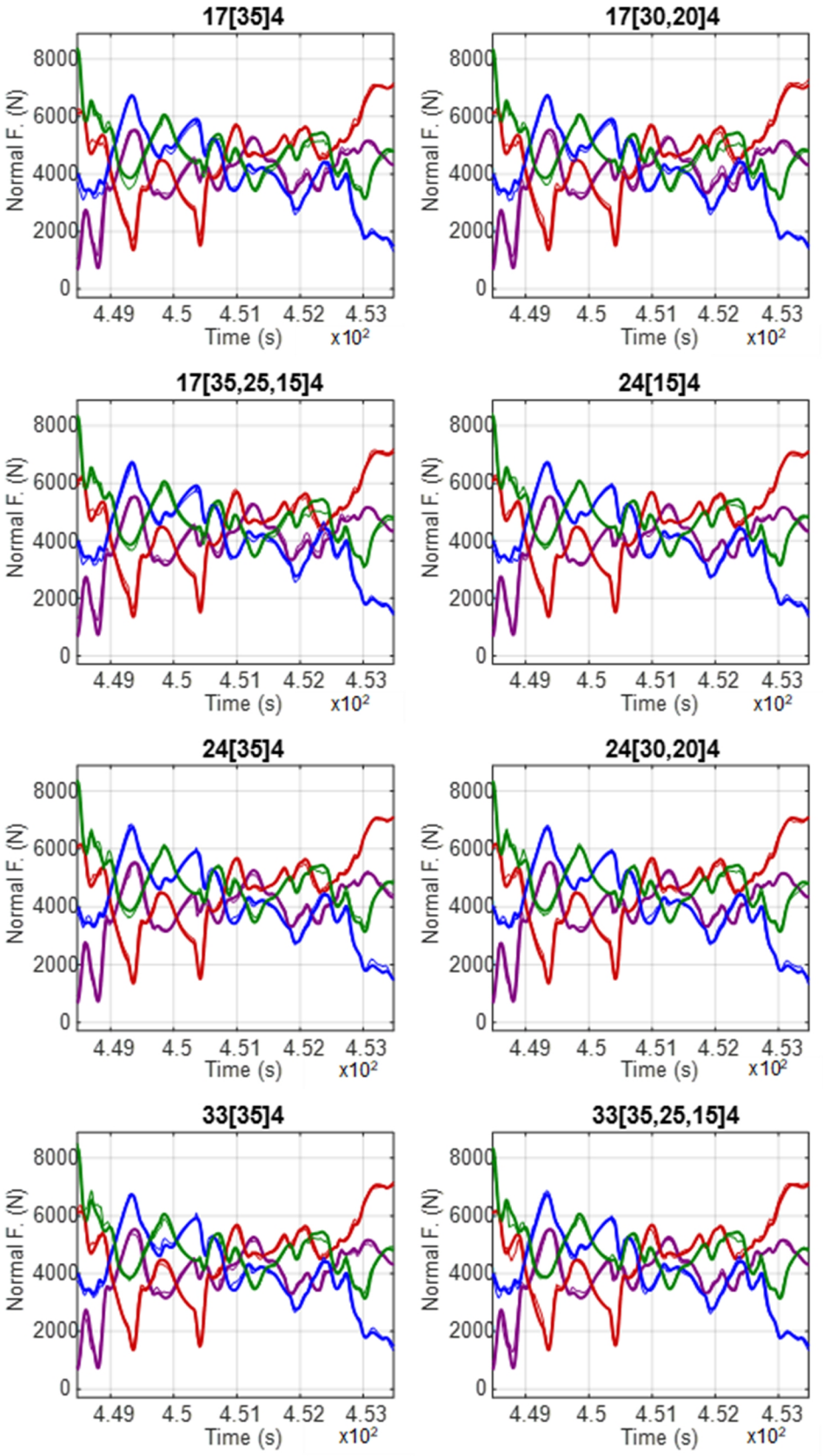

Figure 5-8. Runtime estimations of different NN topologies

Another observation of physical meaningfulness is given by the MAE (mean average error) metric. Ultimately, the key figure is that the MAE is in the range of 180 N shows a notably good performance, considering that the physical magnitude of this variable is around the range of 2000-7000 N depending on the driving situation (in average roughly 4300 N on the front axis and 4600 N on the rear). In other words, this represents a relative error of under 4 % with respect to the average load on the wheels.

The most important conclusion that can be drawn from these results, especially under observation of the plots in time, is that all the selected NNs provide widely satisfying results and roughly similar behaviour, even under dynamically challenging situations and occasionally physical oscillations, as plotted. Minor deviations can be observed especially at dynamically intensive transients and peaks. But even in such cases, on the wheel supporting less weight -where the relative error for the same absolute error increases- the error figure only exceptionally exceeds 10%. In most cases, it can be seen to be inside a 1-2 % error boundary. The fundamental consideration here is that that level of accuracy is considered more than sufficient for the application targeted in this work, were the controller should anyway be calibrated to offer a margin of at least 10-15 % with respect to the physical limit of the wheels.

In what respects to the compromise between size and accuracy, a clear non-proportionality can be observed. For the given examples, minor accuracy gains come at the cost of notably greater computational cost. In any case, only one evaluation per control cycle is needed in this application. Taking as reference previous work, were a smaller NN with roughly 1/6 of the computational complexity took just 34.7 μs to run on an even more constraint embedded platform with roughly 1/4 of single-core performance [288], optimization needs at this point are estimated as negligible.

It may also be observed, that among the selected NN subset another outlier can be identified, but of opposite nature of the previously discussed badly performing outliers. This particular example of NN appears to provide exceptionally good accuracy for its size. Nonetheless, diverse other trained NNs with the same topology show clearly worse performance. Therefore the interpretation in this case would be, that the randomness in the training data pushed it to some kind of overfitting which coincidentally happens to be well suited for the test driving scenario. Nevertheless, such a NN should not be trusted for a solid solution, as its performance under other input data clearly degenerates. Besides, it lacks reproducibility, meaning that if a similar NN would be wanted to be trained again, it probably would not work as expected either.

Ultimately, following the previous argumentation, the topology with 24 inputs and hidden layers as [30,20] is selected as final solution for this work. It shows near the best metrics overall in spite of its computational cost being in the middle range. Its behaviour in time provides a consistently reliable impression, with no unusual deviations. Furthermore, multiple examples of the same topology showed low variance with respect to the performance, while multiple other topologies presented greater inconsistencies among each other.

Before concluding, another analysis of great interest, which was actually addressed prior to the elaboration of the final results above, as already discussed in section 4.7, is the

comparison of the results with respect to a NN approach with only one output. In other words, for the concept of using four smaller NNs with one output instead of one bigger one with four outputs. The corresponding results are illustrated in Figure 5-10.

Basically these results lead to identical conclusions as those already discussed above. Some nuances may be observed though. For instance, the 17 input solutions shows a clearer disadvantage for this sample of NNs, while the 24 input variant does show a somewhat clearer advantage this time. This further strengthens the previous selected solution.

Nevertheless, the bottomline of these results is that the trade-off of using smaller NNs is, in principle not worth it. At least not for a sequentially executed platform. Roughly similar results can be achieved for 4 outputs with a complexity of around 1000 operations, while for the 1 output case, most of the similarly performing solutions are in the 500-750 complexity range, which would need to be multiplied x4, needing one NN execution per wheel. This would mean that in global, the benefit is questionable considering that 2-3x computations would be required in total.

To conclude, a brief comment in what respects to another aspect in the scope of future work and parallel works, as already briefly suggested in other sections, may be made. This is with regard to the applicability of a similar NN topology for predictive purposes. Particularly, for batch predictions -in the order of 1000 per cycle- of future slip values, in order to evaluate the fitness of diverse possible torque set-points. This is part of a further research line and further discussed in section 7.2 and [273]. Figure 5-9 provides the most representative plot of this use-case, in a worst case dynamical situation with a very ambitious 50 ms slip prediction horizon.

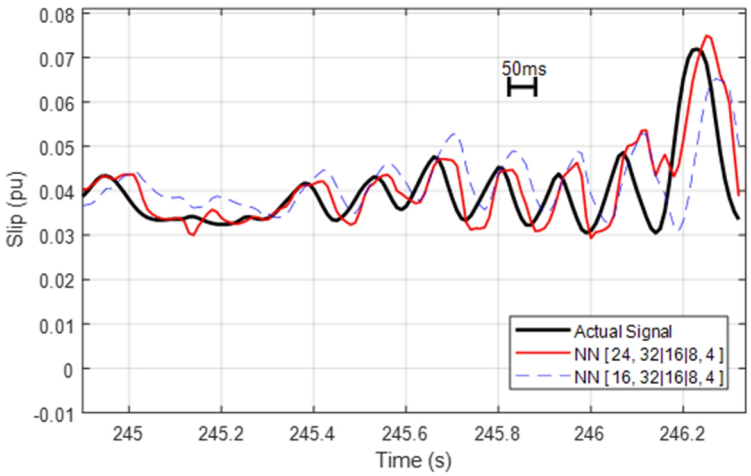

Figure 5-9. NN Prediction capacity for slip values with 50 ms horizon under very unstable conditions as in [273]

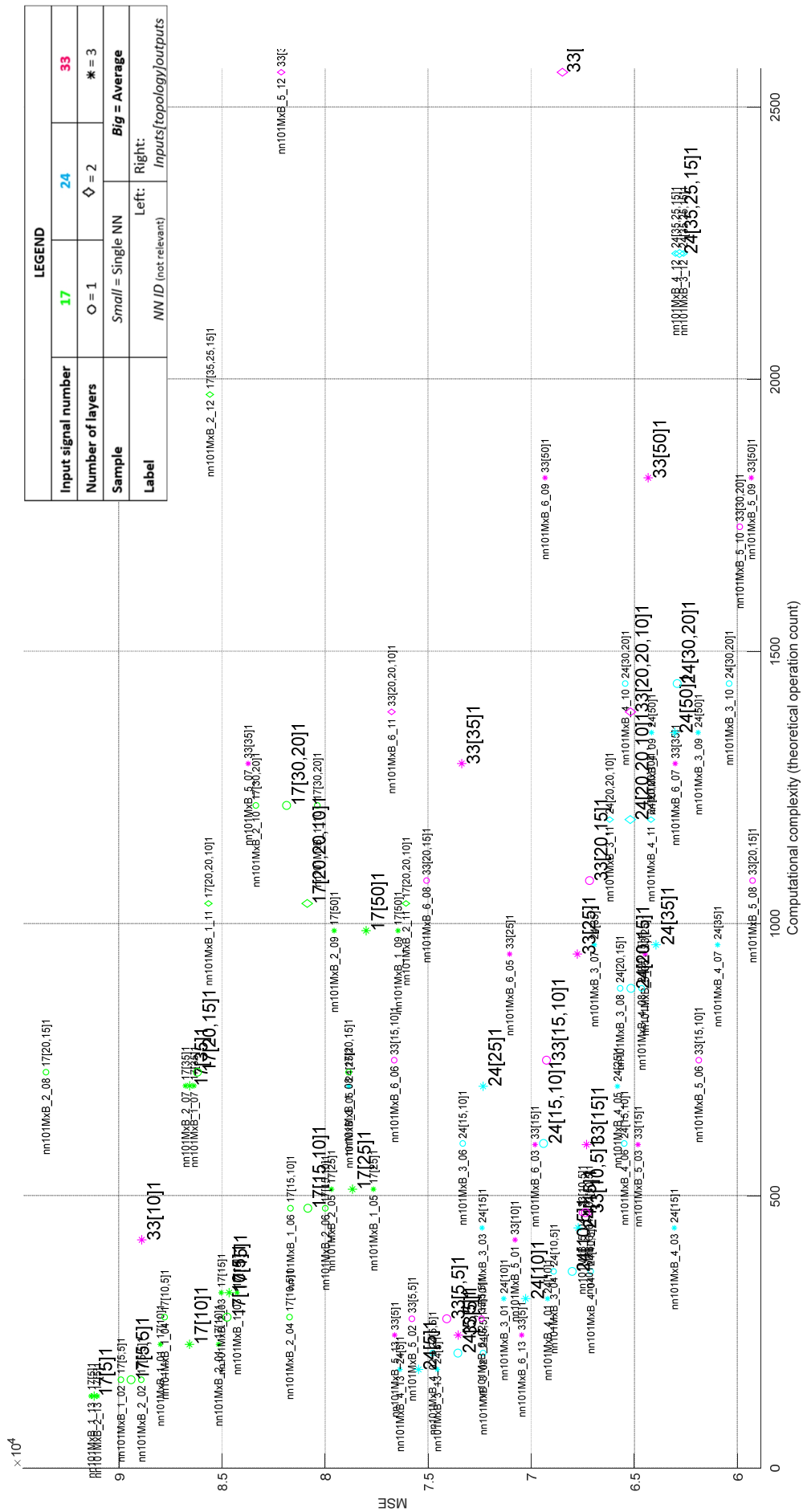

Figure 5-10. Performance of NN topologies with four outputs (bigger labels represent average values for multiple samples with identical configuration)

5.3. Final test of the full control system

The final overall performance of the vehicle -as a system being controlled by the advanced controller- will be analysed in this section. This involves addressing not only the most challenging part of vehicle dynamics, but also criteria representing the other objectives of this solution.

As already extensively discussed, vehicle dynamics are a field of great complexity, involving an extensive variety of physical magnitudes related by non-linear magnitudes and time behaviour which involve major challenges. This makes such systems not only difficult to model, but they are also outstandingly challenging to control and laborious to analyse, especially when the control and optimization problem involves multiple domains and multiple objectives.

A similar problem to the one just discussed for NNs is faced when developing the control solution, in what respects to the high number of adjustable parameters and correspondingly borderline unmanageable permutations, which actually is even worse in this case. Besides, the analysis task turns even more complicated, with a high amount of output variables to analyse, which often can be of difficult interpretation as roughly illustrated in Figure 5-11.

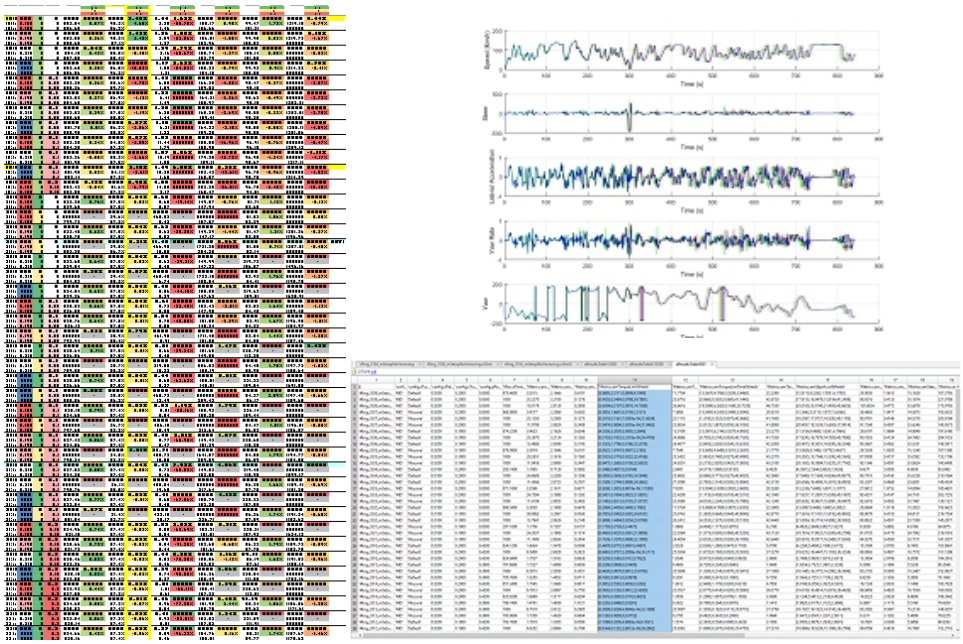

Figure 5-11: Screenshot of a typical 3-monitor setup to illustrate the magnitudes of a typical batch of data and report analysis after each simulation job involving a subset of calibration variants

The following paragraphs will follow a multiple-step approach to address the most important performance aspects for the developed controller. Following the test-cases defined in 3.6, the most important criteria are illustrated with selected figures, tables and derived explanations to facilitate the understanding of the behaviour of the system for the relevant points of interest in particular.

5.3.1. Automated simulation-based tests

The following tests rely on an automated approach using a combination of metrics and plots, mainly for the sake of firstly reproducibility, and secondly iterative and systematic calibration approaches. For this purpose, a third party automated driver function was used with Dynacar™, which follows a defined path.

In every plot, the vehicle's behaviour with the advanced controller turned off and turned on is illustrated, the legend being respectively red and blue coloured lines. In the tables, for each metric the delta that the advanced controller provides is indicated, in green if it is an improvement, or red otherwise.

5.3.1.1.Constant Radius Circle

The first collection of relevant results is obtained by starting with the most basic test manoeuvre, the Constant Radius Circle as defined in section 3.6.1.

This test shows grip limits and might bring up understeer or oversteer, but it does not represent a dynamically extremely critical situation. It is a quite stationary behaviour which can be observed. Conveniently, this facilitates analysis with less risk for confusion through other complex phenomena which appear in upcoming tests. It does not highlight some of the advantages of the controller though, which will become more obvious in those tests, where the focus will be particularly placed in analysing those additional aspects.

Figure 5-12 shows the torque set points provided by the controller. None of this set points had to be limited at any point by the TCS system. The controller sends clearly more torque to the rear right wheel, which helps to turn left. The front right wheel does not receive much additional torque, which would probably saturate it and risk more understeer.

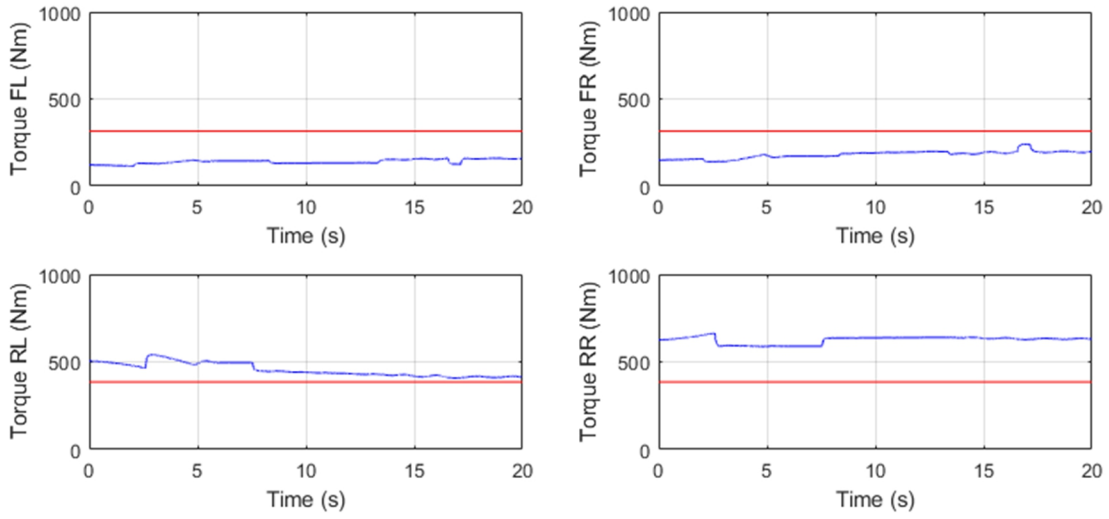

Figure 5-12. Constant radius test: Torque set points on each wheel

The advanced controller shows stable behaviour, with smooth control actions and few switching between different optimization steps. This can be said to show the intended ideal behaviour, which suits well a stable situation as the one in the test case.

Figure 5-13 shows the achieved speeds. The advanced controller manages to accelerate faster and achieve a higher final speed. This is attributed to less friction losses, as slip is minimized. Consequently it also finishes the first full 360º turn 0.9 seconds faster.

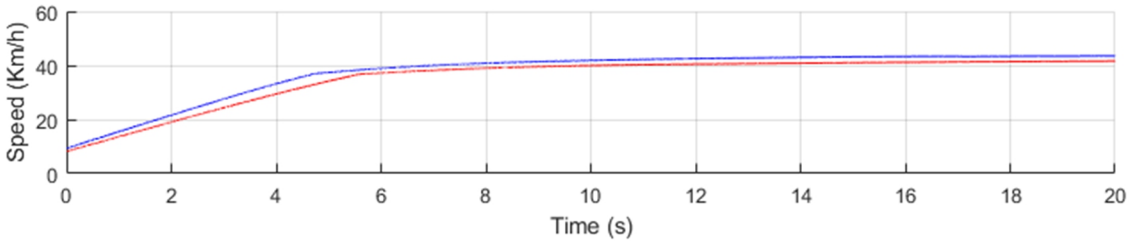

Figure 5-13. Constant radius test: Speed

Figure 5-14 illustrates the lateral dynamics performance with the lateral acceleration and yaw rate criteria, which probably are among the points of major interest. The advanced controller provides greater values for both cases, in consistency with the greater speed observed above.

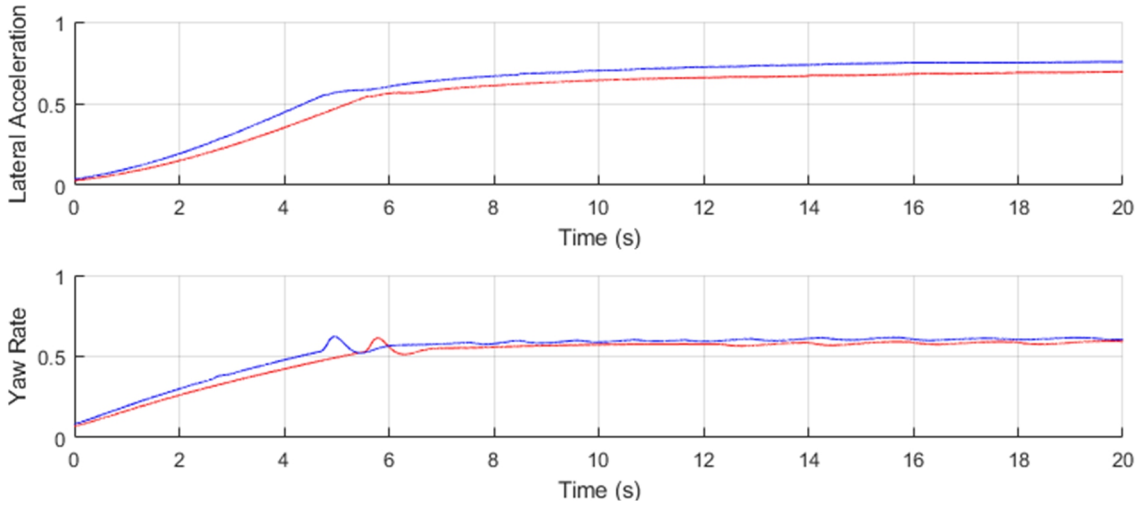

Figure 5-14. Constant radius test: Lateral acceleration and yaw rate

It must be noted that the glitch seen roughly after second 5 can be ignored. This is introduced by a minor unintended behaviour bug of the autonomous driver function. The practical effect of this glitch on the test run is negligible. Unwanted oscillations of another kind occasionally also appear at the first fractions of second in each run, when the vehicle is placed on the ground. For this purpose, the first fractions of second of some plots are truncated for cleaner visualization purposes.

Figure 5-15 shows the slips on all four wheels. While the most critical wheel -i.e. the front right one which takes most of the steering load- suffers from less slip, the rear right one shows slightly higher slip. Surprisingly, the front left wheel suffers from more slip in this particular run. One explanation is that greater lateral acceleration makes it receive less vertical force, so it eventually starts gaining some additional rotational speed. In any case, a value of under 0.04 is a very low and unproblematic slip. Nevertheless, this is a good

example of multiple phenomena in this complex domain that might be perceived as counterintuitive. Furthermore, it must be clarified that the left wheels show lower values -partially out of scope- because of the way slips are calculated in Dynacar™, relative to the longitudinal speed of the vehicle. As this test describes a very narrow radius, the curve inner wheels are turning noticeably slower, slightly distorting the metric, which is mostly noticeable when zooming in to such small values.

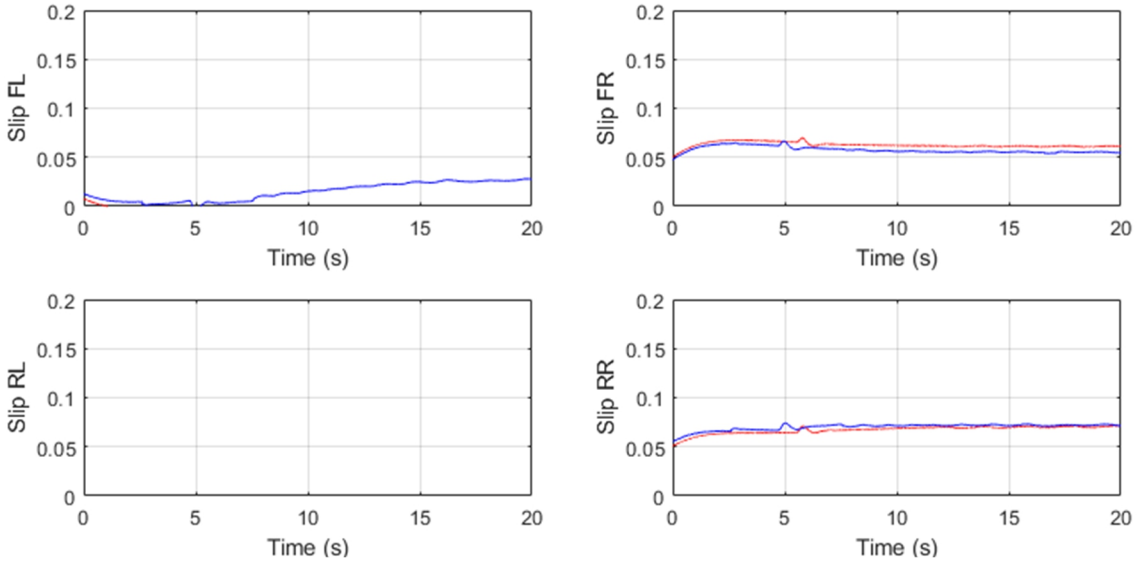

Figure 5-15. Constant radius test: Wheel slip rates

The main metrics from the previous figures can be summarized into Table 5-5. It provides a clear view of how the advanced controller notably provides performance enhancement for all the most relevant criteria, as discussed above.

Controller	360º time (s)		Speed Max (km/h)		Speed Average (km/h)		Lat. Accel. Max (m/s^2)		Lat. Accel. Average (m/s^2)		Yaw Rate Max (rad/s)		Yaw Rate Max (rad/s)	
On	20.9	-4.4%	43.50	4.0%	33.85	6.0%	0.76	7.7%	0.54	11.0%	0.62	1.3%	0.48	5.6%
Off	21.8		41.81		31.93		0.70		0.49		0.61		0.45	

Table 5-5. Summary of the most relevant results for the constant radius test

The previous table includes the average lateral acceleration, which in this particular test probably is of less interest than the maximum value, as it includes the speeding up phase starting from standstill, where the lateral grip is not yet at the limit.

As a concluding observation, a few significant effects can be highlighted when the torque request is further increased. For instance, increasing it to 1800 Nm in total, to force a more extreme situation. Figure 5-16 pays special attention to the slip on the right wheels, which are supporting the greatest vertical load and correspondingly should provide the greatest contribution to following the curve. It shows how the advanced controller consistently keeps the slip of the front right wheel below the symmetric distribution, which facilitates this wheel to keep steering the vehicle around the curve.

Furthermore, at some point, without the advanced controller, the vehicle starts to slip and get unstable, which can be appreciated with increasing slip values and finally even oscillations.

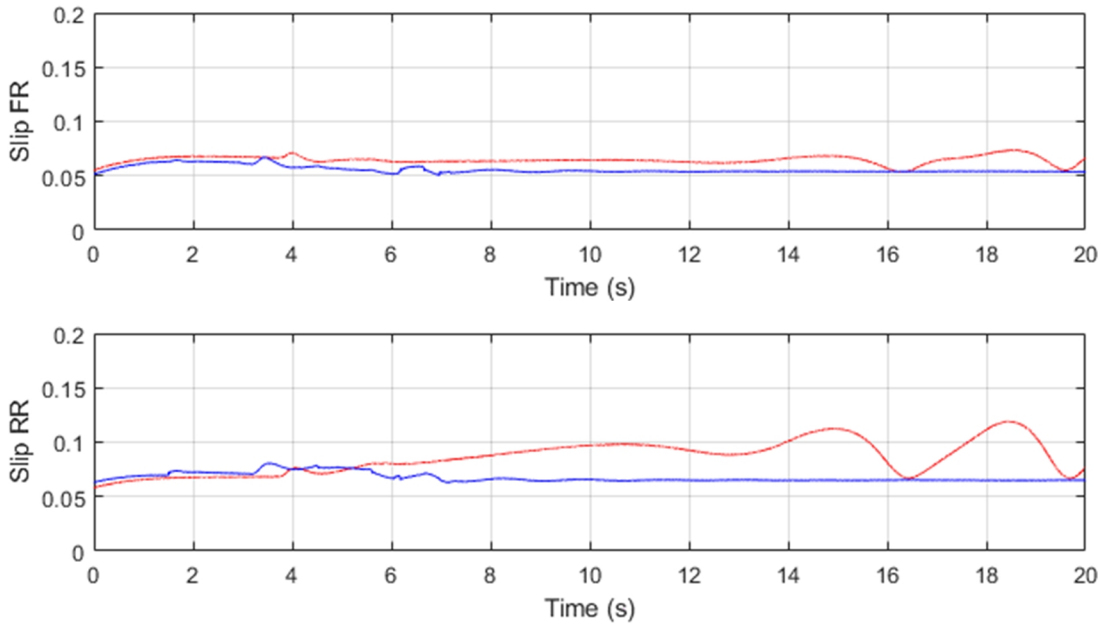

Figure 5-16. Constant radius test under extreme conditions: Outer wheel slip rates

The result is that the vehicle without advanced controller eventually slides to a wider curve than 20,, with a deviation of roughly 0.6 metres, as illustrated in Figure 5-17.

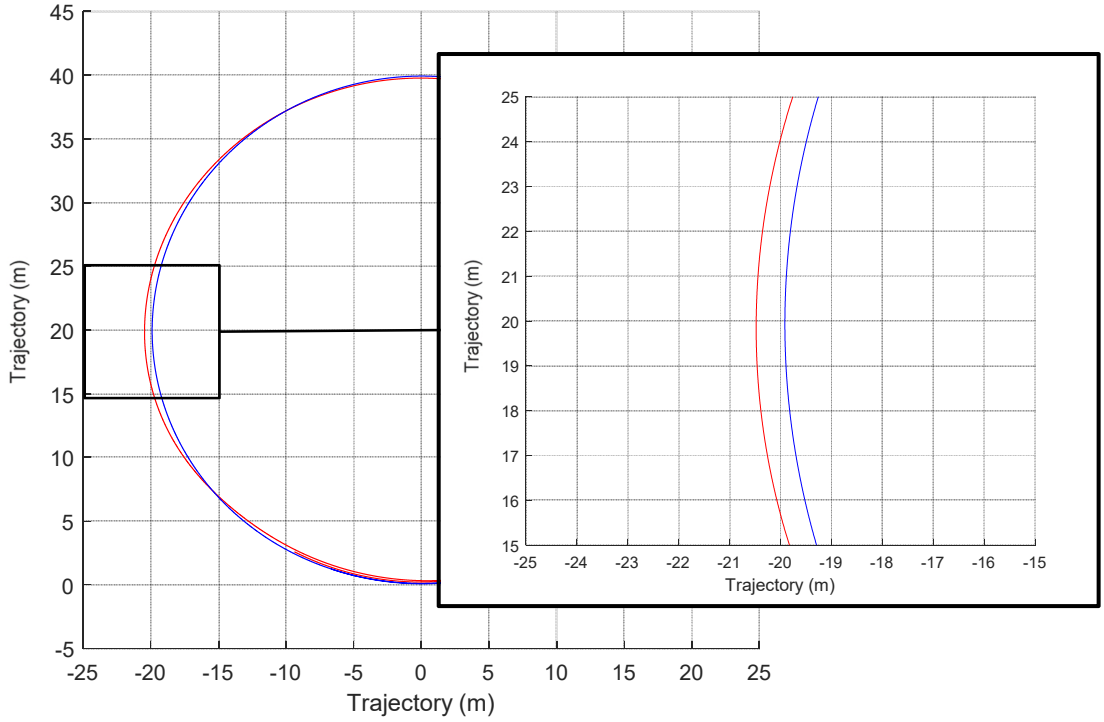

Figure 5-17. Constant radius test under extreme conditions: Trajectory deviation

5.3.1.2.Elk test

The second test, the Elk Test as defined in section 3.6.2, provides a more dynamically challenging situation, prone to bringing the vehicle to a critical situation from the stability point of view. It involves multiple transients and therefore provides a couple of points to be further analysed and added to the previous results and criteria.

The torque set points are illustrated in Figure 5-18. The advanced controller adapts to the situation as expected, with no abrupt variations, but when necessary making minor adjustments, as multiple optimization steps can be seen.

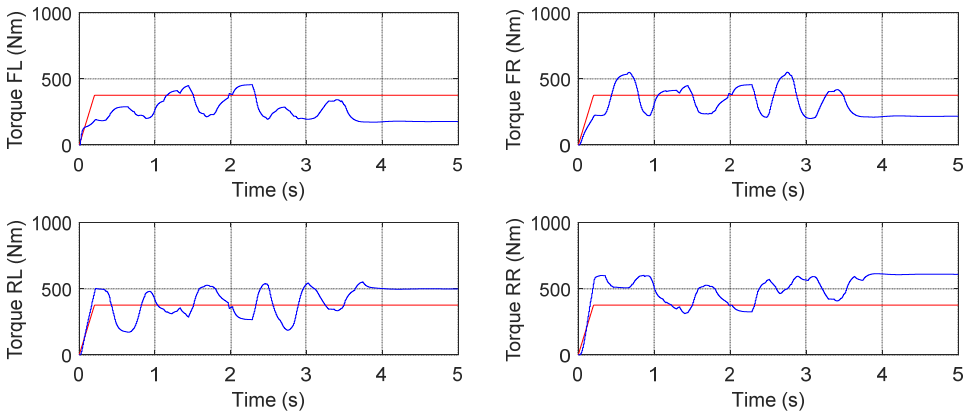

Figure 5-18. Elk test: Torque set points on each wheel

The resulting trajectory is shown in Figure 5-19. While the path for the first part is surprisingly similar, in the second part greater sliding and a noticeable instability can be appreciated with the advanced controller turned off, making the vehicle miss the path and forcing the driver to strongly correct.

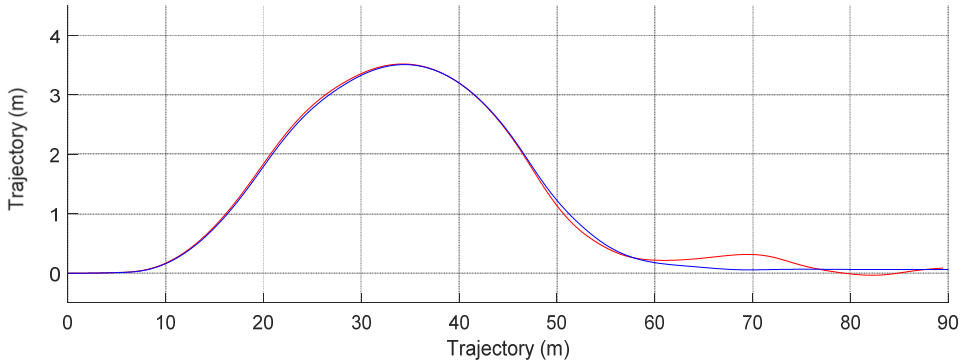

Figure 5-19. Elk test: Trajectory

The way the vehicle is struggling to keep the trajectory is considerably more noticeable when looking at the yaw angle in Figure 5-20. Without advanced controller, the vehicle reaches high rotation over its own vertical axis, being on the edge of spinning and fully losing control.

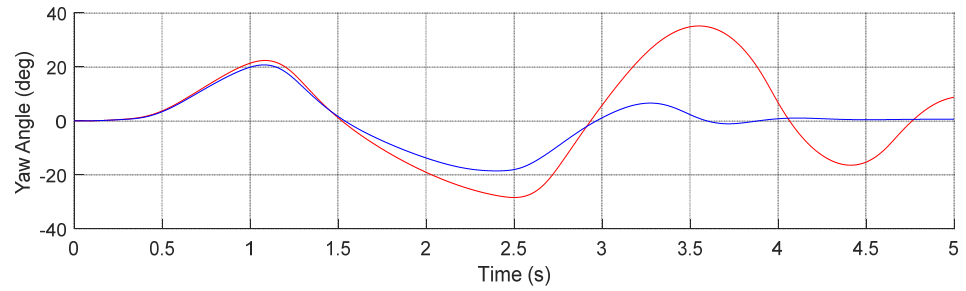

Figure 5-20. Elk test: Yaw rate

The effect of the Torque Vectoring can be clearly seen when zooming into this second part of the manoeuvre, as shown in Figure 5-21. At that moment, both cases have the vehicle sliding sideways -somewhat more without advanced controller- but the reaction to the steering wheel input with the controller on is noticeably quicker in time and with a greater gradient.

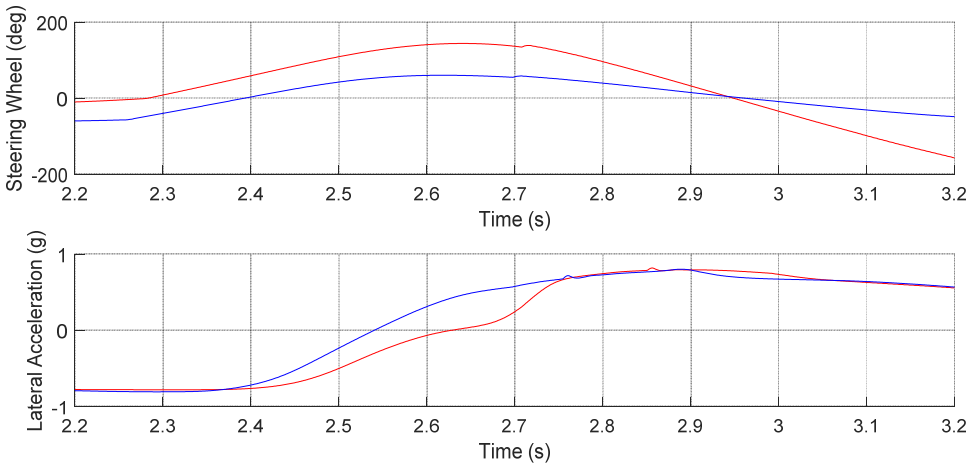

Figure 5-21. Elk test: Steering wheel input and lateral acceleration

An even more noticeable advantage is seen when looking at the slips, in this case focusing the analysis on the front wheels, depicted in Figure 5-22, which are steering the vehicle in both directions in this test. The slip rate with the advanced controller is consistently lower and also more stable. In fact, eventually without advanced controller the slip peaks at 0.2, where the TCS system, adjusted for that particular value, happens to successfully intervene.

As a result of the sliding, the vehicle loses considerable speed without advanced controller, as illustrated in Figure 5-23

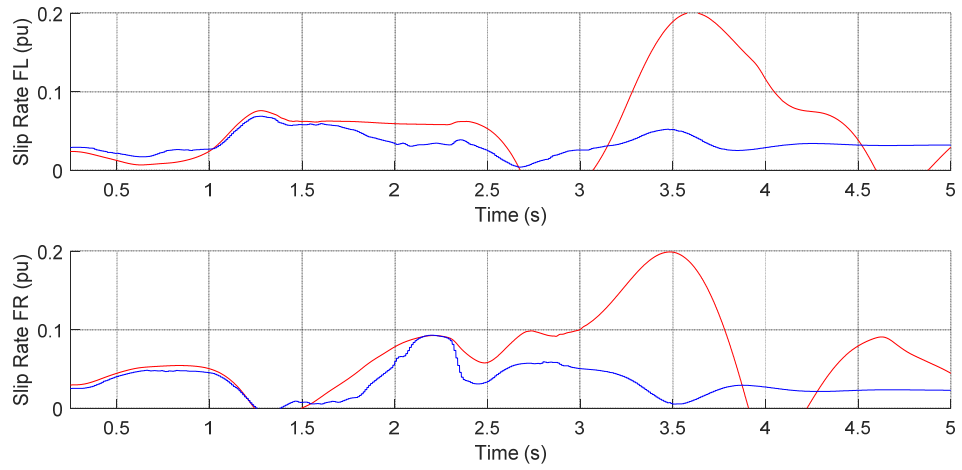

Figure 5-22. Elk test: Steering wheels slip rates

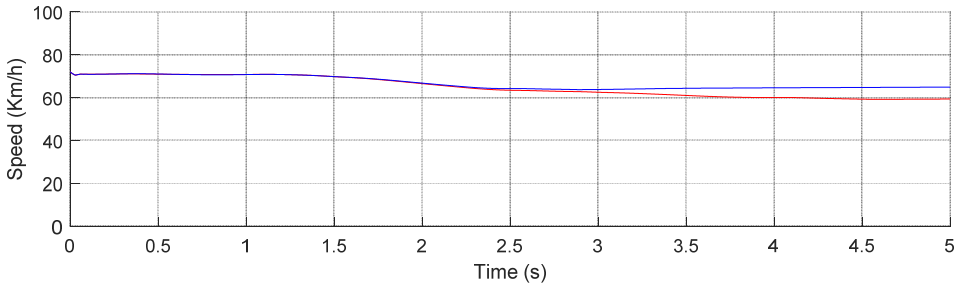

Figure 5-23. Elk test: Speed

Following the above discussion, the main additional enhancements that are highlighted through this particular test, are summarized in Table 5-6.

Controller	Speed Min (Km/h)		Speed Average (km/h)		Slip Rate Front Max (pu)		Slip Rate Front Average (pu)		Yaw Angle Max (deg)	
On	63.69	7.7%	66.76	2.8%	0.093	-53.7%	0.034	-45.2%	20.7	-41.0%
Off	59.15		64.93		0.201		0.062		35.2	

Table 5-6. Summary of the most relevant results for the elk test

While the general lateral dynamics -and collaterally also the longitudinal dynamics- keep showing notable enhancements in the higher end of one-digit percentages -except for the yaw angle, which shows an even bigger difference-, the slip values are roughly half for both metrics, which is a very satisfying result as well.

Finally, a few additional observations may again be made by pushing the test to a more extreme situation. In both cases, when the advanced controller is not active, the autonomous driver loses control over the vehicle and ends up spinning the car and sliding of the track.

This happens in different scenarios. Firstly, the total torque request is doubled. To compensate, the speed is decreased to 62 Km/h. The resulting trajectory is illustrated in Figure 5-24

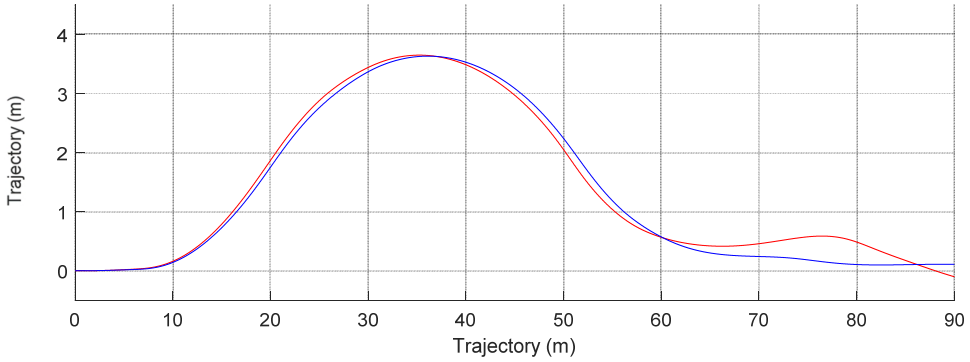

Figure 5-24. Elk test under extreme conditions: Trajectory and loss of control in absence of advanced controller

Another even more extreme scenario is illustrated in Figure 5-25, where with an also high torque request of 2500 Nm, the vehicle is driven into the curve with a clearly excessive speed. First, in absence of the advanced controller, the vehicle makes a much wider curve. And ultimately again, the vehicle ends up strongly sliding, spinning, and flying off the track. With the advanced controller, slip is also high, but the vehicle remains controllable even for the autonomous driver. Inevitably, the specified path cannot exactly be followed and the vehicle gets back to its original lane over 20m later than intended.

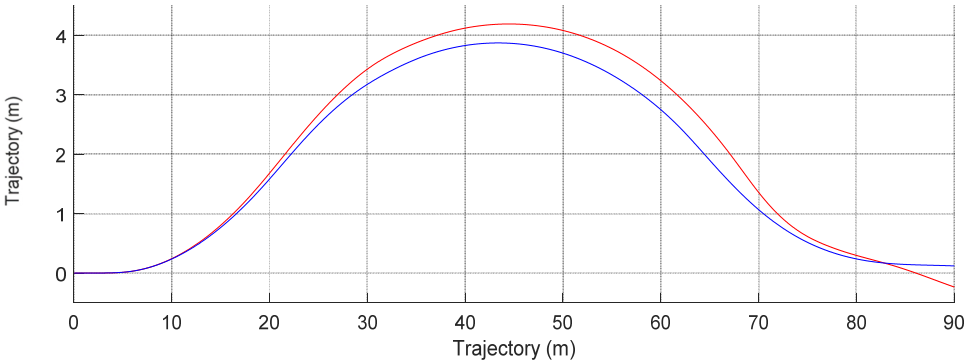

Figure 5-25. Elk test under extreme conditions: Trajectory and loss of control without advanced controller

5.3.1.3. The Nürburgring

To conclude, the long-distance Nürburgring race-track test, as discussed in 3.6.3, provides a few final conclusions and additional metrics with a wider scope, providing a complete image about the performance of the advanced controller and enhancements also for other -less microscopic- metrics.

It must be noted though, that once again the black-box autonomous driver caused a few restrictions here. In many cases excessive accelerations and speeds caused autonomous driver to lose control over the vehicle. Correspondingly, the maximum power was maintained, but the accelerator pedal was truncated to half. Furthermore, speed was limited to 130 Km/h -consequently leading to very humble lap times-. Besides, a few extra functional blocks had to be added to the autonomous driver's commands, in order to impose constraints to mitigate misbehaviours. Nevertheless, for the runs without advanced controller, a few track departures happened, as highlighted in Figure 5-26.

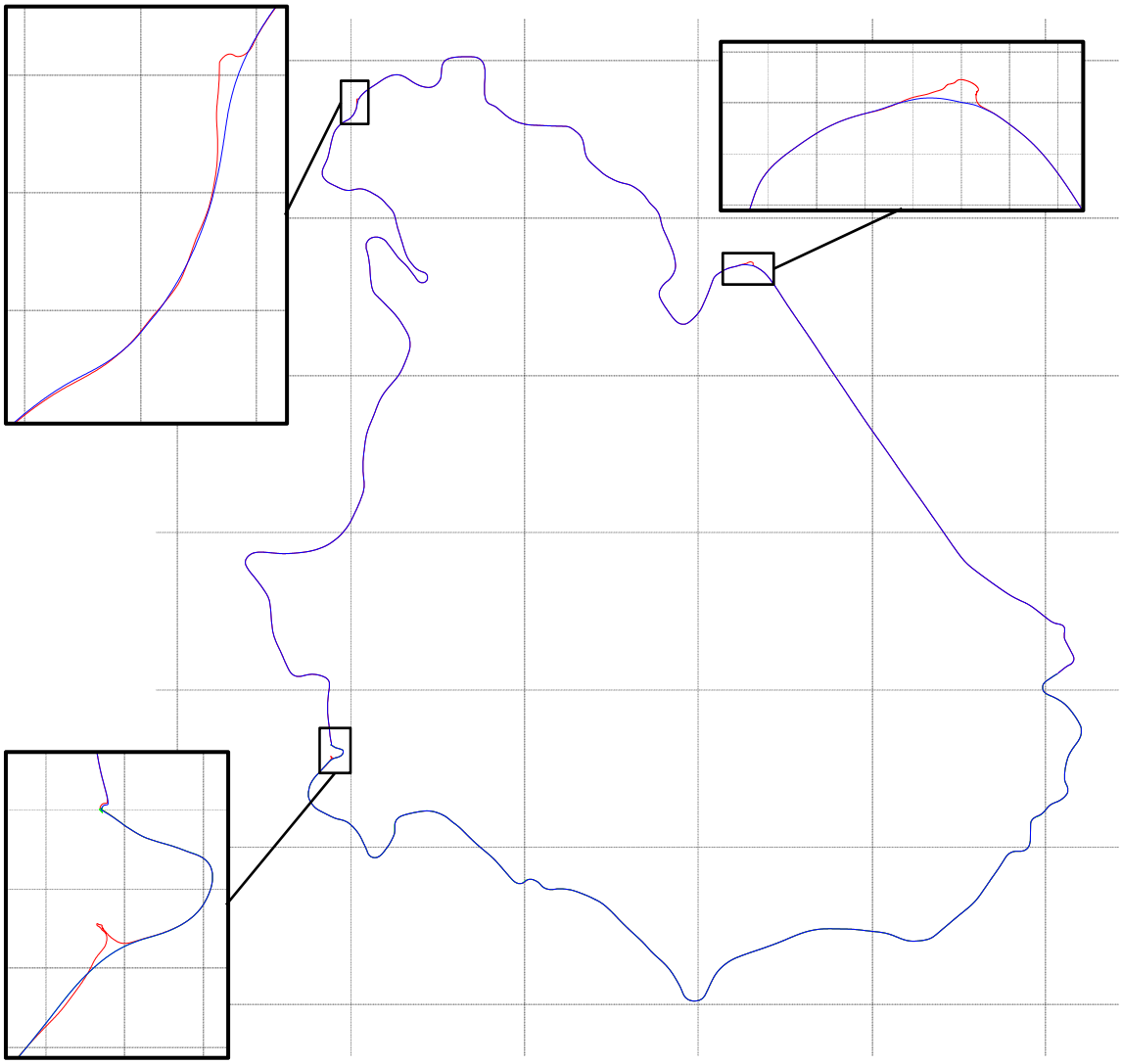

Figure 5-26. Nürburgring trajectory, highlighting track departures without advanced controller (scale on main plot: 1 square = 1000 m, zoom plots scale: n/a)

The fact that the autonomous driver crashed a multiple of times without advanced controller is consistent with the observations in previous tests. The interpretation for this is that -although the advanced controller generated additional yaw moment which could hypothetically cause oversteer- it is not so prone to exceed the traction capacity of the wheels, consequently causing less critical and unstable situations.

Before proceeding to the remaining bigger scope metrics, a final detailed view of the response of the yaw rate in dynamically challenging situations is highlighted in Figure 5-27. Here again, the strong gradient and higher peaks of the yaw rate can be seen under the effect of the Torque Vectoring implemented under the advanced controller.

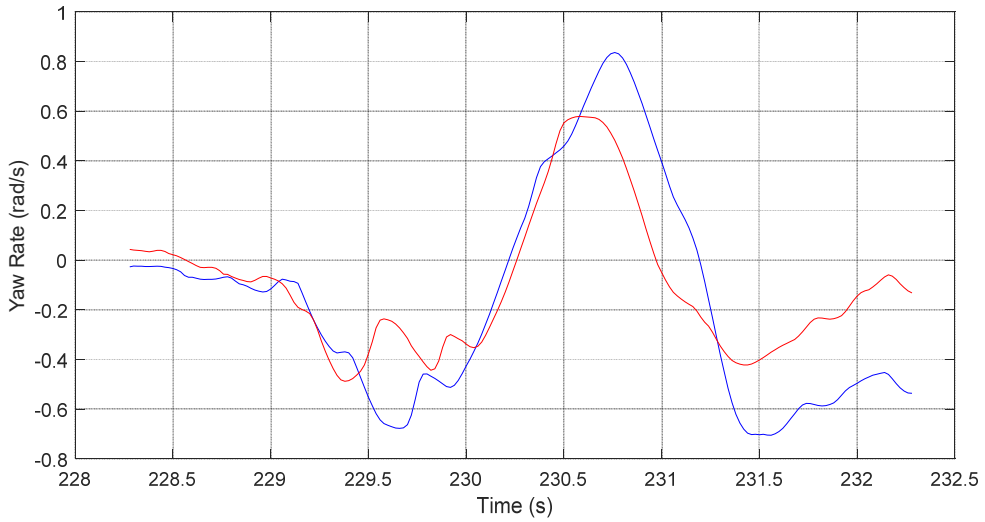

Figure 5-27. Nürburgring test: Yaw rate

Ultimately, Table 5-7 provides the macroscopic metrics of relevance -for which corresponding plots with a scope of around 800 seconds would not be useful-. In this case, the focus is placed firstly on energetic values. Secondly on overall metrics related to slip. And ultimately, on some additional overall lateral dynamics metrics related to the steering behaviour and slip angle.

Controller	Lap time (s)		Ele. Energy total (kWh)		Mec. Energy total (kWh)		Eff. (%)		Slip total (n/a)		Tq Lim. Time avg./wheel (s)		Tq. Lim total (n/a)		Steering total (n/a)		Slip Angle total (n/a)	
On	793.8	-6.3%	35.82	-1.7%	30.77	1.1%	85.9%	2.8%	10.77	-30.7%	15.9	-23.3%	31.1	-40.0%	119	-10.9%	1737	-25.4%
Off	847.2		36.44		30.45		83.6%		15.54		20.8		51.9		133		2330	

Table 5-7. Summary of the most relevant results for the Nürburgring test

Interestingly the total mechanical power used to make the entire lap is the only of the discussed metrics which provides a worse value. It is assumed to be related to higher speeds around curves, and correspondingly higher aerodynamic and friction losses. But nevertheless the advanced controller's efficiency gains are able to make up for this and end up consuming less electrical power anyway.

The total power consumption is slightly over 35 kWh. This might appear somewhat high for the Nürburgring's slightly under 21 km, but considering that other big vehicles like for instance the Jaguar I-Pace claim 27 kWh/100km under optimal economic driving conditions [289], it certainly is plausible. Furthermore, it must be considered, that on the race track the vehicle is constantly under high throttle, full braking and also burning energy in friction and slips. Besides, regenerative braking is not enabled.

Further relevant metrics are related to slip. The first metric indicates the total accumulated slip of all wheels. For the sake of simplicity it is provided as unitless magnitude, in contrast to the previous cases, as it is not calculated as average (because the total time is different). A further metric is added to indicate for how much time per wheel the TCS had to intervene to reduce slip, in average across the four wheels. Similarly to the total slip metric, a unitless total limitation metric is calculated as well. This metric indicates how strongly the TCS had to limit the requested torque values.

To conclude with these metrics, two additional unitless total metrics are provided. One indicates the amount of steering that the autonomous driver had to provide as input, where more steering is considered as worse, as an indication of understeer and strong corrective manoeuvres. Furthermore, understeer and lateral slip of the steering wheels can also be identified by its slip angle, for which another accumulated metric is provided.

Having extensively addressed vehicle dynamics and now also efficiency, the final focus is placed on the thermal aspects, which were also exhaustively modelled. To provide better comparability of thermal plots on the time axis, another simulation was run with less aggressive driver settings. The result is, that the total time advantage for the advanced controller is reduced to under 2 seconds, thus the plots are better aligned. Furthermore, the difference of mechanical power -which has the most direct impact on the thermal model- is smaller.

Thermal behaviour is illustrated in Figure 5-28 and summarized in Table 5-8. Left and right temperatures without advanced controller obviously show almost identical values -minor differences being due to the effect of different rotations speeds and slip-. In contrast, clear differences over time can be appreciated amongst motors due to the effect of the vectoring.

The impact of vectoring on temperature divergences is probably not as strong as might have been thought. At some point nevertheless the difference between left and right sides reaches roughly 10ºC. Although the optimization rules of the controller must have contributed to reducing this difference, in this test, where critical driving situations are predominant, the contribution of the optimization algorithm is limited, as maximizing the grip for the sake of keeping stability needs to be prioritized.

In overall, no major impact is noticed in what respects to average values. While the final temperature with advanced controller was notably lower on the power electronics modules side -which is attributed to a coincidence and their fast dynamics- the mean temperature for the motors was roughly 1% higher, which another minor surprise but not concerning anyhow.

Criteria	Controller	Motor Temperatures				Modules Temperatures				Average	
		FL	FR	RL	RR	FL	FR	RL	RR	Motors	Modules
Mean	On	62.0 ºC	65.6 ºC	79.8 ºC	83.4 ºC	38.9 ºC	39.5 ºC	41.6 ºC	41.8 ºC	72.7 ºC	40.5 ºC
	Off	63.6 ºC	63.2 ºC	79.9 ºC	81.3 ºC	39.4 ºC	39.4 ºC	41.6 ºC	41.7 ºC	72.0 ºC	40.5 ºC
Final	On	73.0 ºC	76.4 ºC	95.5 ºC	92.2 ºC	43.2 ºC	43.3 ºC	52.5 ºC	32.6 ºC	84.3 ºC	42.9 ºC
	Off	75.6 ºC	75.3 ºC	94.8 ºC	95.8 ºC	48.4 ºC	48.3 ºC	53.1 ºC	53.5 ºC	85.4 ºC	50.8 ºC

Table 5-8. Main figures of the thermal behaviour from the Nürburgring test

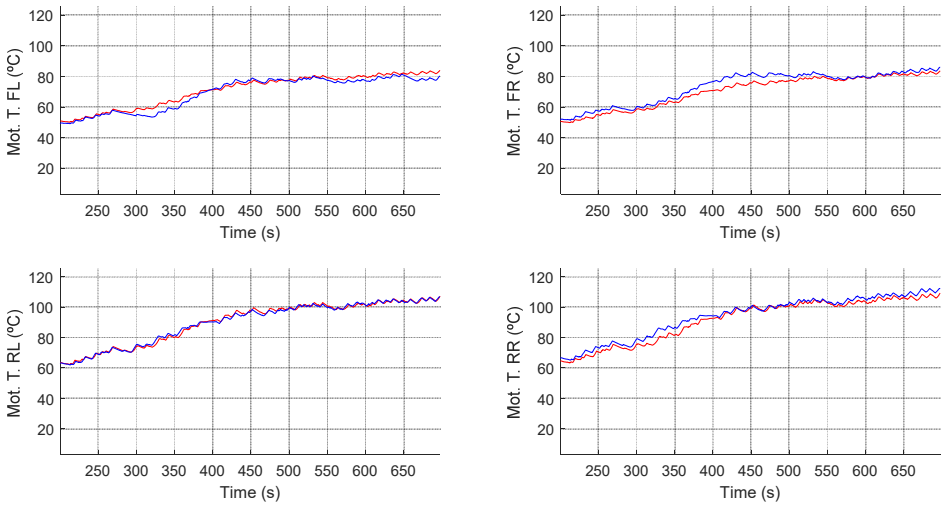

Figure 5-28. Nürburgring test: Motor temperatures

A final observation is that surprisingly the autonomous driver had even bigger difficulties to keep the vehicle on the track, when it was tested by activating only the baseline Torque Vectoring controller, without the advanced controller's estimation and optimization features. In fact, it was only capable of finishing a lap unharmed under considerably restrictive throttle and TCS limitations. The most plausible explanation -besides the driver's shortcomings- is that it was designed for a front wheel drive vehicle with notably less power. Apparently, in this upgraded powerful 4WD vehicle, and in absence of an ESP system, strong oversteer frequently occurs, which would probably require a more skilful driver at the steering wheel.

Nevertheless, it is notorious that the advanced controller is capable of mitigating the instabilities that are caused by the baseline Torque Vectoring controller, considering that the baseline vectoring provides the initial values for the optimization function. This is another highlight in what respects to the good performance of the implemented estimation and optimization functions, and the developed concept overall.

5.3.2. Qualitative and overall assessment

Besides the previous mainly objective automated tests, which are basically based on automatically generated metrics in combination with expertise-based qualitative assessment of the plots -as seen above-, additional subjective criteria are also to be incorporated for a final overall assessment of the control solution.

Firstly, real-world test-track tests already have proven the attractiveness and benefits of the torque-vectoring concept un general, as discussed in section 5.1. In spite of being for a different vehicle setup, the objective assessment certainly shows fundamental similarities between the performance of the real vehicle on track and the upgraded vehicle model ultimately used. For instance, key figures such as yaw rate showed identical improvements, and the nature of the improvement of the vehicle's behaviour overall is of similar nature in both cases.

The subjective aspects with the most perceivable gain are better cornering responsiveness when cornering -also viewable in the plots, mostly through the yaw rate-.

Secondly, a subjective assessment was also performed using the simulator setup itself, representing the high power 4WD vehicle from the second phase. Although inevitably the subjective perception in what respects to driving forces and other nuances are not fully available here, force-feedback on the steering wheel provides certain feedback nonetheless. And in any case, the vehicle's reactions to the driver's inputs with different controller setups are visible in the 3D visualization of the virtual race tracks, which in any case was a relevant supporting tool to gain better understanding of how the vehicle reacts. This is in fact applicable to the DiL tests as well as to the automated test, as watching a 3D visualization facilitates the understanding of phenomena that are not always self-explanatory when watching a plot, especially when the vehicle slides out of control.

The results of this assessment widely coincide with the dynamical and performance enhancements discussed in the above sections.

5.4. Embedded implementation: common FPGA & GPU work

In the second development phase leading to the final advanced controller, it was not expected -and would have not been reasonable- to aim to achieve a complete embedded implementation of the fully functional system, including the entire set of algorithms that compose the advanced controller. This is consistent considering the scope of the work itself, besides some of the constraints discussed in section 4.9.

Nonetheless, multiple building blocks of different nature were effectively implemented on the targeted embedded Zynq® FPGA/SoC-platform. Aiming for a solid proof of concept, milestones with a triple purpose were pursued: achieve validation of relevant concepts, enable further development steps and provide solutions for future work. These achieved results are further detailed in section 4.9 as well, but can be summarized as following milestones which were successfully achieved:

- Write CAN functions for embedded implementation and runtime calibration
- Build a framework for algorithm implementation on the embedded platform
- Set up a code-generation solution
- Implement the baseline Torque Vectoring code on the embedded platform
- Exploit the full system simulation framework for MiL and HiL validation
- Integrate the embedded platform on a real vehicle
- Perform race-track tests

The bottomline is that a fully functional baseline Torque Vectoring implementation was efficiently achieved in an heterogeneous embedded platform, by exploiting the established simulation, development and implementation frameworks to successfully integrate and test it on a real prototype vehicle, illustrated in 5.1. Figure 5-29 illustrates this in a simplified manner.

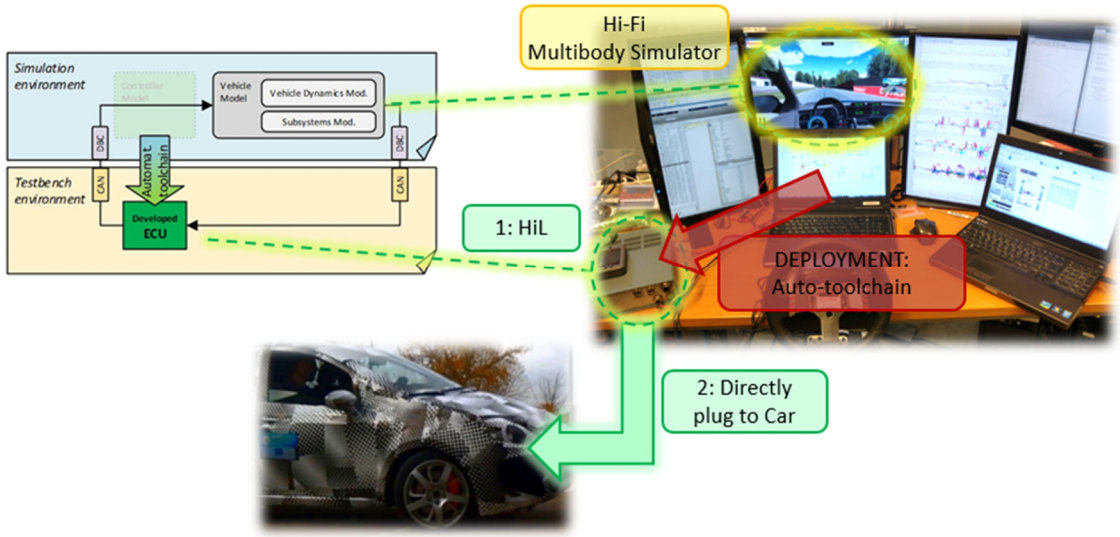

Figure 5-29. Streamlined workflow from MBD development to HiL and vehicle

Secondly, a couple of highly relevant heterogeneous embedded implementations of the NN was pursued in partnership with outstanding fellow researchers: a post-doctoral researcher and another doctoral researcher. Their expertise and contributions not only led to the results that will be discussed in the following paragraphs, but also to a high-impact journal publication, as further discussed in section 7.2 [273].

In this common work, a variation of the NN from this dissertation was implemented on two highly parallel heterogeneous SoC platforms. The code for the NN was automatically generated and implemented on the two embedded platforms identified as most relevant and industry suitable: the same Xilinx Zynq® 7020 FPGA-SoC-platform discussed along this work, and the NVIDIA Tegra K1 GPU-SoC also previously mentioned.

The use-case targeted in this work is basically a step towards one of the proposed future research lines by further evolving the advanced Virtual Sensing-based Torque Vectoring Concept established in the second phase of this dissertation. The extended use-case is to provide, as an input to the Torque Vectoring optimization algorithm, a vehicle dynamics prediction of the effect of different possible control actions. This means, that at each control cycle, a big batch of 512-1024 NN instances needs to be evaluated, each with different values for different possible solutions.

These implementations required exhaustive optimizations in order to exploit the parallelism of the platforms and avoid internal bottlenecks, which required the platform-specific expertise of the research partners.

The most representative results are illustrated in Table 5-9. The main conclusions in what respects to the embedded platforms is that both FPGA and GPU devices widely outperform the processor-based capabilities. Furthermore, the GPU outperformed the FPGA. The FPGA mostly suffered a notable performance impact because, for a fully pipelined solution, the implementation approach consumed more hardware resources than available. Correspondingly, alternative FPGA implementations were evaluated using smaller NNs, look-up-tables for the sigmoidal function, and 32 bit fixed point arithmetic instead of floating point.

Platform	NN Topology	Implem. Variant	NN Batch Size	Time per NN (µs)	GPU Occupancy	FPGA Resource utilization
CPU	16 [32 16 8] 4	Baseline	1024	~12.0	-	-
GPU	16 [32 16 8] 4	Const. Mem.	512	0.05	25%	-
GPU	16 [32 16 8] 4	Const. Mem.	1024	0.03	x	-
FPGA	8 [16 12 8] 4	Fix32 Pipelined	1024	~0.19	-	0%, 521%, 132%, 406%
FPGA	8 [16 12 8] 4	Fix32 LUT1k	1024	3.25	-	19%, 80%, 17%, 29%
FPGA	16 [32 16 8] 4	Fix32 LUT1k	1024	4.20	-	37%, 83%, 25% 76%

Table 5-9. Extract of results of NN implementations on FPGA and GPU as in [273]

In any case, in absence of pipelining the GPU scaled up notably better and increased the performance gap. Besides, it can be assumed that the GPU would also have performed notably better if the algorithm would have grown in complexity.

It must be emphasized, that these remarkable computation time results correspond to big batches of parallelly executed NNs. In other words, the resulting outstandingly low average execution times per NN are achieved because many are computed simultaneously. This means that reducing the amount of NNs will not scale linearly. In fact, below a certain number of NNs, the total computation time would remain virtually unaltered for the GPU implementation. So, executing a single NN would take almost the same time as executing for example 10. For the FPGA implementation, the scalability depends on the pipelining implementation. For further details and extensive results and analysis, please refer to the publication [273].

5.5. Summary of Achieved Solution

As a wrap up of the most important parts from the previous chapters, it can be summarized that a successful development setup and controller implementation have been achieved, establishing a model-based automated framework. This is illustrated in Figure 5-30 and Figure 5-31.

This framework enables to develop algorithms with notable abstraction, by benefiting from real-time HiL and DiL simulations, automatic code-generation and automated tests and analysis, amongst other features. In a first project phase, where a prototype vehicle with a 2xMiW electric powertrain was taken to race track tests, this solution enabled a seamless implementation of the baseline Torque Vectoring algorithm, firstly on the HiL, and secondly directly plugging the same ECU in the real vehicle. This represents a major improvement to the productivity and flexibility of the workflow.

A highly representative system model representing the targeted vehicle has been developed, this being fundamental for a second fully simulation-based phase. The core is a high-fidelity multibody vehicle dynamics model. Furthermore, complementary subcomponent models have been created, mainly a powertrain model integrating elaborate thermal and efficiency models, besides secondary models for mechanical aspects, TCS and ABS. Most of these models were adjusted by exploiting data acquired in race track tests. The ultimately targeted vehicle is defined to have a powerful powertrain with four independent electric motors.

Figure 5-30. HiL setup with physical ECU prototype and Simulink™ based framework

The developed advanced controller itself implements a real-time optimization approach, extending the baseline Torque Vectoring algorithm from the first phase. Multiple refined algorithms have been integrated to enhance its capabilities and realize the real-time

multivariable multiobjective optimization concept. A Fuzzy Logic algorithm provides the priorities of the different objectives basing on the driving situation. The most important objective is related to vehicle dynamics, namely avoiding slip by preventing excessive torque on the wheels. For this purpose, a Neural Network is implemented as Virtual Sensor for an unmeasurable physical magnitude: normal force on each wheel.

Multiple implementation approaches were followed for the most important building blocks of this work, namely the Torque Vectoring and the Neural Network. These targeted heterogeneous embedded platforms, including microprocessor, FPGA and GPU parts. This was partly done in cooperation with fellow researchers.

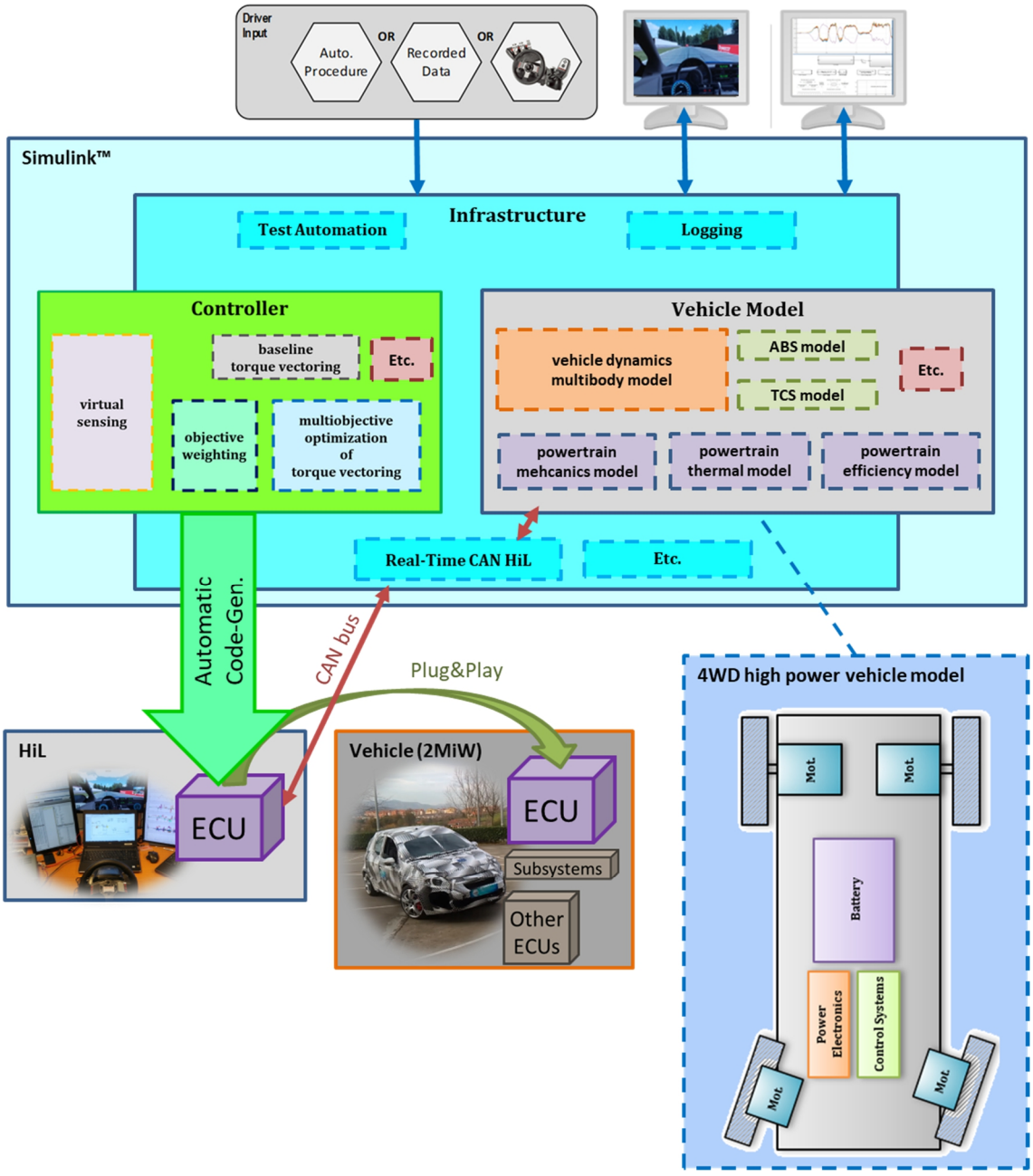

Figure 5-31. Wrap up of the building blocks and relations of the implemented solution

CHAPTER 6:

Summary and Conclusions

"Learning is not a spectator sport"

D. Blocher

This chapter presents the conclusions and future work. The main conclusions in general are firstly summarized over the two first pages, before proceeding to a more detailed discussion on per-topic basis.

CHAPTER 8:

References

8. References

[1] IEA, Electric Vehicles initiative, and Clean Energy Ministerial, "International Energy Agency - Global EV Outlook to 2020." 2013.
[2] "Acura NSX 2017 | Acura.com." [Online]. Available: http://nsx.acura.com/. [Accessed: 24-Feb-2016].
[3] Charley Cameron, "100 Self-Driving Cars Set to Hit Sweden's Public Roads in 2017," 2017. .
[4] Roland Berger Strategy Consultants, "Consolidation in vehicle electronic architectures", 2015.
[5] ISO, "ISO 26262-8:2011(en), Road vehicles — Functional safety — Part 8: Supporting processes," 2011.
[6] M. A. Nielsen, "Neural Networks and Deep Learning," 2015.
[7] "#neuralnetwork - DeviantArt." [Online]. Available: http://www.deviantart.com/tag/neuralnetwork. [Accessed: 12-Feb-2016].
[8] K. Gurney, *An Introduction to Neural Networks*. CRC Press, 1997.
[9] "Zynq UltraScale+ MPSoC." [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html. [Accessed: 03-Mar-2016].
[10] A. Mihalache, F. Bedoucha, Y.-M. Foll, and M. Foucault, "Assurer la SdF d'un logiciel existant ayant de nouvelles fonctionnalités spécifiées en Model Based Design," in *Congrès Lambda Mu 20 de Maîtrise des Risques et de Sûreté de Fonctionnement*, 2016.
[11] "Model Based Design." [Online]. Available: https://www.hil-simulation.com/home/model-based-design.html. [Accessed: 12-May-2019].
[12] "World University Rankings 2019 by subject: engineering and technology," *Times Higher Education (THE)*, 08-Oct-2018. [Online]. Available: https://www.timeshighereducation.com/world-university-rankings/2019/subject-ranking/engineering-and-IT. [Accessed: 06-May-2019].
[13] European Commission, "All facts and figures - Facts and figures - Transport - European Commission," 2016. [Online]. Available: http://ec.europa.eu/transport/strategies/facts-and-figures/all-themes/index_en.htm. [Accessed: 03-Mar-2016].
[14] E. Jaffe, "Far Beyond Rush Hour: The Incredible Rise of Off-Peak Public Transportation," *CityLab*, 2014. .
[15] ITDP, "The Rise of Shared Mobility, A Key Complement to Public Transit," *Institute for Transportation and Development Policy*, 12-May-2014. .
[16] "Railway passenger transport statistics - quarterly and annual data - Statistics Explained," EC. [Online]. Available: http://ec.europa.eu/eurostat/statistics-explained/index.php/Railway_passenger_transport_statistics_-_quarterly_and_annual_data. [Accessed: 08-Feb-2016].
[17] Linklaters, "Set to revive: investing in Europe's infrastructure - Linklaters," 2014.
[18] IPTS, "Dynamics of the Introduction of new Passenger Car Technologies." 2003.
[19] European Commission, "The implementation of the 2011 White Paper on Transport 'Roadmap to a Single European Transport Area –towards a competitive and resource-efficient transport system' five years after its publication: achievements and challenges," 2016. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/themes/strategies/doc/2011_white_paper/swd%282016%29226.pdf. [Accessed: 13-May-2019].
[20] OICA, "Sales Statistics | OICA." [Online]. Available: http://www.oica.net/category/sales-statistics/. [Accessed: 08-Feb-2016].
[21] "Eurostat - Tables, Graphs and Maps Interface (TGM) table." [Online]. Available: http://ec.europa.eu/eurostat/tgm/table.do?tab=table&init=1&language=en&pcode=tsdpc340&plugin=1. [Accessed: 08-Feb-2016].
[22] K. Bodek and J. Heywood, "Europe's Evolving Passenger Vehicle Fleet: Fuel Use and GHG Emissions Scenarios through 2035." MIT, 2008.

[23] Cummins, “Worldwide Emissions Regulations.” 2014.
[24] BBC, “Volkswagen: The scandal explained,” 2015. .
[25] Roland Berger Strategy Consultants, “Diesel controversy –Temporary shock or paradigm shift in powertrain?,” 2015.
[26] Roland Berger Strategy Consultants, “The future of diesel: Automobile manufacturers need a paradigm shift – new opportunities for automotive suppliers,” 2015.
[27] FTI, “Regulation and Competitiveness of the EU Automotive Industry,” 2015.
[28] ICCT, “The International Council on Clean Transportation - European Vehicle Market Statistics: Pocketbook 2014,” 2014.
[29] European Commission, “European Commission Press release: Commission welcomes Member States’ agreement on robust testing of air pollution emissions by cars,” 2015. [Online]. Available: http://europa.eu/rapid/press-release_IP-15-5945_en.htm. [Accessed: 08-Feb-2016].
[30] “The VW Emissions Scandal and Its Effect on the Global Auto Industry,” *Global Research*. [Online]. Available: http://www.globalresearch.ca/the-vw-emissions-scandal-and-its-effect-on-the-global-auto-industry/5498738. [Accessed: 03-Mar-2016].
[31] I. W. H. Parry, J. Darmstadter, and others, *The costs of US oil dependency*. Resources for the Future, 2003.
[32] “Imports and secure supplies - Energy - European Commission,” *Energy*. [Online]. Available: https://ec.europa.eu/energy/en/topics/imports-and-secure-supplies. [Accessed: 03-Mar-2016].
[33] Roland Berger Strategy Consultants, “Fuel cells – A realistic alternative for zero emission?,” 2013.
[34] “Electric Car Range Comparison -- Top 11.” [Online]. Available: http://evobsession.com/electric-car-range-comparison/. [Accessed: 03-Mar-2016].
[35] “The Tesla Battery Report-Nov-2014-V12-extract.pptx - Extract-from-the-Tesla-battery-report.pdf.” [Online]. Available: http://www.totalbatteryconsulting.com/industry-reports/Tesla-report/Extract-from-the-Tesla-battery-report.pdf. [Accessed: 03-Mar-2016].
[36] E. Helmers, “Possible Resource Restrictions for the Future Large-Scale Production of Electric Cars,” in *Competition and Conflicts on Resource Use*, S. Hartard and W. Liebert, Eds. Springer International Publishing, 2015, pp. 121–131.
[37] ACEA, “European Automobile Manufacturers Association - New passenger car registrations by alternative fuel type in the European Union Q4 2015,” 2016.
[38] “Europe EV Sales Up 49%,” *CleanTechnica*. [Online]. Available: http://cleantechnica.com/2015/12/27/europe-ev-sales-up-49/. [Accessed: 08-Feb-2016].
[39] “Horizon 2020 - European Commission,” *Horizon 2020*. [Online]. Available: https://ec.europa.eu/programmes/horizon2020/. [Accessed: 03-Mar-2016].
[40] “Shaping Digital Innovation | ECSEL Joint Undertaking.” [Online]. Available: https://www.ecsel.eu/. [Accessed: 29-Apr-2019].
[41] J. Dulac, “Global transport outlook to 2050: Targets and scenarios for a low-carbon transport sector,” 2013.
[42] J.-P. Louis, *Control of Synchronous Motors*. New York, NY: John Wiley & Sons, 2013.
[43] R. Isermann, *Engine Modeling and Control*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014.
[44] “Torque Vectoring: The Hyper-Smart, Fuel-Efficient Future of All-Wheel Drive,” *Popular Mechanics*, 01-Oct-2009. [Online]. Available: http://www.popularmechanics.com/cars/news/4225886. [Accessed: 08-Feb-2016].
[45] E. Siampis, M. Massaro, and E. Velenis, “Electric rear axle torque vectoring for combined yaw stability and velocity control near the limit of handling,” in *52nd IEEE Conference on Decision and Control*, 2013, pp. 1552–1557.
[46] Tomas Smetana and et al., “Schaeffler active eDifferential: The active differential for future drive trains,” presented at the Schaeffler Symposium, 2010.
[47] “Audi. Torque vectoring,” *YouTube*. [Online]. Available: https://www.youtube.com/watch?v=cFBMiwWaks8. [Accessed: 08-Feb-2016].
[48] “Torque Vectoring and Active Differential.” [Online]. Available: http://torque-vectoring.belisso.com/. [Accessed: 26-Feb-2016].
[49] M. Dendaluce, M. Allende, J. Pérez, P. Prieto, and A. Martin, “Multi Motor Electric Powertrains: Technological Potential and Implementation of a Model Based Approach,” in *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, Florence, 2016.

[50] M. Dendaluce, I. Iglesias, A. Martin, P. Prieto, and A. Peña, “Circuit testing of a Torque Vectoring Algorithm on a Motor-in-Wheel car using a Model-Based Methodology, HiL Setups and Novel Embedded Implementation Solutions,” Rio de Janeiro, 2016.
[51] H. Huchtkoetter and H. Klein, “The Effect of Various Limited-Slip Differentials in Front-Wheel Drive Vehicles on Handling and Traction,” 1996.
[52] T. Gassmann and J. Barlage, “VISCO-LOK: A Speed-Sensing Limited-Slip Device with High-Torque Progressive Engagement,” in *SAE International Congress & Exposition*, 1996.
[53] N. H. Mohd Zainal Abidin, S. Imamori, and A. Alexander, “Development of High Efficiency Next-Generation SH-AWD Rear Drive Unit,” in *SAE 2015 World Congress & Exhibition*, 2015.
[54] L. D. Novellis, A. Sorniotti, and P. Gruber, “Wheel Torque Distribution Criteria for Electric Vehicles With Torque-Vectoring Differentials,” *IEEE Trans. Veh. Technol.*, vol. 63, no. 4, pp. 1593–1602, May 2014.
[55] B. Heissing and M. Ersoy, Eds., *Chassis handbook: fundamentals, driving dynamics, components, mechatronics, perspectives*, 1st edition. Wiesbaden: Vieweg + Teubner, 2011.
[56] A. Parra, A. Zubizarreta, J. Pérez, and M. Dendaluce, “Intelligent Torque Vectoring Approach for Electric Vehicles with Per-Wheel Motors,” *Complexity*, Feb. 2018.
[57] “SMARTGREENS 2014 - 3rd International Conference on Smart Grids and Green IT Systems.” [Online]. Available: http://www.smartgreens.org/KeynoteSpeakers.aspx?y=2014. [Accessed: 08-Feb-2016].
[58] Siemens AG, “Siemens RACE Reliable Automation and Control Environment,” 2015.
[59] “fromsiliconvalley.files.wordpress.com/2015/07/post.png.” .
[60] Dendaluce Jahnke, Martin, “System-on-Chip-based highly integrated Powertrain Control Unit for next-generation Electric Vehicles: harnessing the potential of Hybrid Embedded Platforms for Advanced Model-Based Control Algorithms,” presented at the Electric Vehicle Symposium 28, Goyang, Korea, 2015.
[61] “3Ccar Project.” [Online]. Available: https://assrv1.oth-aw.de/3Ccar/index.php/home. [Accessed: 03-Mar-2016].
[62] Renesas Electronics Europe, “Functional Safety System Developments from MCU Vendor Point of View White paper.” 2016.
[63] M. Hillenbrand, *Funktionale Sicherheit nach ISO 26262 in der Konzeptphase der Entwicklung von Elektrik/Elektronik Architekturen von Fahrzeugen*. Karlsruhe: KIT Scientific Publishing, 2012.
[64] ISO, “ISO 26262-6:2011 - Road vehicles - Functional safety - Part 6: Product development at the software level,” 2011.
[65] National Instruments, “What is the ISO 26262 Functional Safety Standard? - National Instruments,” 2016.
[66] “ISO 26262 | Functional Safety | Quality-One.” .
[67] H. Guo, J. Liu, S. Yu, and J. Li, “Iso 26262 Impact on Development of Powertrain Control System,” in *Proceedings of the FISITA 2012 World Automotive Congress*, SAE-China and FISITA, Eds. Springer Berlin Heidelberg, 2013, pp. 705–715.
[68] B. Schätz, “Certification of embedded software–impact of ISO DIS 26262 in the automotive domain,” in *Leveraging Applications of Formal Methods, Verification, and Validation*, Springer, 2010, pp. 3–3.
[69] VDA QMC Working Group 13 / Automotive SIG, “Automotive SPICE Process Assessment / Reference Model,” 2015.
[70] “Automotive SPICE - Verband der Automobilindustrie e. V. (VDA).” [Online]. Available: https://vda-qmc.de/software-prozesse/automotive-spice//. [Accessed: 29-Apr-2019].
[71] “AUTOSAR, Home.” [Online]. Available: http://www.autosar.org/. [Accessed: 30-Jan-2015].
[72] H. Bo, D. Hui, W. Dafang, and Z. Guifan, “Basic Concepts on AUTOSAR Development,” in *2010 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, 2010, vol. 1, pp. 871–873.
[73] G. Sandmann and R. Thompson, “Development of AUTOSAR software components within model-based design,” SAE Technical Paper, 2008.
[74] B. Jäger, “AUTOSAR METHODOLOGY @BMW.,” p. 16.
[75] MISRA Consortium, “MISRA C:2012 - Addendum 2: Coverage of MISRA C:2012 against ISO/IEC TS 17961:2013 ‘C Secure.’” 2016.
[76] “Code Generation - Embedded Coder - Simulink - MathWorks España.” [Online]. Available: http://es.mathworks.com/products/embedded-coder/. [Accessed: 14-Mar-2016].

[77] “TargetLink.” [Online]. Available: https://www.dspace.com/de/gmb/home/products/sw/pcgs/targetli.cfm. [Accessed: 14-Mar-2016].
[78] A. Roychoudhury, *Embedded Systems and Software Validation*. M. Kaufmann, 2009.
[79] D. Coulon, “Future of the Automotive Industry, Electronics is the Key,” *tti MarketEYE*, 2014. .
[80] Decision Etudes Conseil, “Extract from the report World Electronic Industries 2012–2017. General outlook: Towards more professional electronics, a chance for North America and Europe.” 2014.
[81] PwC, “Automotive Perspective 2015.” 2014.
[82] F. Milella, “Problem-Solving by Immersive Virtual Reality: Towards a More Efficient Product Emergence Process in Automotive,” vol. 2, no. 4, p. 9, 2015.
[83] “Green Car Congress: Audi to use Gen 2 zFAS controller in production e-tron quattro; piloted driving; moving to domain-controlled architecture.” [Online]. Available: http://www.greencarcongress.com/2016/01/audi-to-use-gen-2-zfas-controller-in-production-e-tron-quattro-piloted-driving-moving-to-domain-cont.html. [Accessed: 19-Apr-2016].
[84] A. Tyagi, “Audi zFAS (central driver assistance controller) Piloted Driving,” *Automotive Electronics*, 04-Jul-2018. .
[85] “Infineon-TriCore_Family_BR-BC-v01_00-EN.pdf.” [Online]. Available: https://www.infineon.com/dgdl/Infineon-TriCore_Family_BR-BC-v01_00-EN.pdf?fileId=5546d4625d5945ed015dc81f47b436c7. [Accessed: 30-Apr-2019].
[86] “M580,” *Pi Innovo*, 05-Oct-2018. [Online]. Available: https://www.pi-innovo.com/product/m580/. [Accessed: 30-Apr-2019].
[87] “Freescale MPC574xP.” [Online]. Available: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC574xP. [Accessed: 27-Jan-2015].
[88] “ZF and NVIDIA Announce Artificial Intelligence System for Autonomous Cars, Trucks and Industrial Applications.” [Online]. Available: https://press.zf.com/press/en/releases/release_2637.html. [Accessed: 30-Apr-2019].
[89] N. Newsroom, “NVIDIA Boosts IQ of Self-Driving Cars With World’s First In-Car Artificial Intelligence Supercomputer,” *NVIDIA Newsroom Newsroom*. [Online]. Available: http://nvidianews.nvidia.com/news/nvidia-boosts-iq-of-self-driving-cars-with-world-s-first-in-car-artificial-intelligence-supercomputer. [Accessed: 30-Apr-2019].
[90] “Xilinx.” [Online]. Available: http://www.xilinx.com/. [Accessed: 31-Jan-2015].
[91] E. Nurvitadhi *et al.*, “Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Neural Networks?,” in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, New York, NY, USA, 2017, pp. 5–14.
[92] S. M. Trimberger, “Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology,” *Proc. IEEE*, vol. 103, no. 3, pp. 318–331, Mar. 2015.
[93] D. Strenski, P. Sundararajan, and R. Wittig, “The Expanding Floating-Point Performance Gap Between FPGAs and Microprocessors,” *HPCwire*, 22-Nov-2010. [Online]. Available: http://www.hpcwire.com/2010/11/22/the_expanding_floating-point_performance_gap_between_fpgas_and_microprocessors/. [Accessed: 22-Jan-2015].
[94] Altera Corporation, “Cyclone V SoC Brochure,” 2014.
[95] Xilinx Inc., “Xilinx Automotive Zynq-7000 All Programmable SoCs Brochure.” 2014.
[96] “NVIDIA Tegra: The World’s Fastest Mobile Processors.” [Online]. Available: http://www.nvidia.com/object/tegra.html. [Accessed: 14-Jul-2017].
[97] S. Che, J. Li, J. W. Sheaffer, K. Skadron, and J. Lach, “Accelerating Compute-Intensive Applications with GPUs and FPGAs,” in *2008 Symposium on Application Specific Processors*, 2008, pp. 101–107.
[98] Infineon Technologies AG, “Highly integrated and performance optimized 32-bit microcontrollers for automotive and industrial applications.” Infineon Technologies AG, 2017.
[99] Valerie Bernon-Enjalbert, Mathieu Blazy-Winning, Regis Gubian, David Lopez, Jean-Philippe Meunier, and Mark O’Donnell, “Safety Integrated Hardware Solutions to support ASIL D Applications.” NXP, 2013.
[100] Texas Instruments (last), “DSP - Overview - Processors.” [Online]. Available: http://www.ti.com/processors/digital-signal-processors/overview.html. [Accessed: 31-Dec-2018].

[101] B. C. Ward, J. L. Herman, C. J. Kenna, and J. H. Anderson, "Outstanding Paper Award: Making Shared Caches More Predictable on Multicore Platforms," in *2013 25th Euromicro Conference on Real-Time Systems*, 2013, pp. 157–167.
[102] N. Hemsoth, "Latest FPGAs Show Big Gains in Floating Point Performance," *HPCwire*, 16-Apr-2012. [Online]. Available: http://www.hpcwire.com/2012/04/16/latest_fpgas_show_big_gains_in_floating_point_performance/. [Accessed: 22-Jan-2015].
[103] M. I. and Strategy, "A Machine Learning Landscape: Where AMD, Intel, NVIDIA, Qualcomm And Xilinx AI Engines Live," *Forbes*. [Online]. Available: http://www.forbes.com/sites/moorinsights/2017/03/03/a-machine-learning-landscape-where-amd-intel-nvidia-qualcomm-and-xilinx-ai-engines-live/. [Accessed: 14-Jul-2017].
[104] A. Guzhva, S. Dolenko, and I. Persiantsev, "Multifold acceleration of neural network computations using GPU," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5768 LNCS, no. PART 1, pp. 373–380.
[105] A. A. Huqqani, E. Schikuta, S. Ye, and P. Chen, "Multicore and GPU Parallelization of Neural Networks for Face Recognition," *Procedia Comput. Sci.*, vol. 18, pp. 349–358, 2013.
[106] "Zynq UltraScale+ MPSoC Data Sheet: Overview." .
[107] "How to Reduce FPGA Logic Cell Usage by >x5 for Floating-Point FFTs," *Design And Reuse*. [Online]. Available: https://www.design-reuse.com/articles/42344/hardwired-floating-point-fpga-based-ffts.html. [Accessed: 11-Jul-2017].
[108] T. Trader, "Comparing Peak Floating Point Claims," *HPCwire*, 17-Jun-2014. [Online]. Available: http://www.hpcwire.com/2014/06/17/comparing-peak-floating-point-claims/. [Accessed: 22-Jan-2015].
[109] S. S. L. Oskouei, H. Golestani, M. Kachuee, M. Hashemi, H. Mohammadzade, and S. Ghiasi, "GPU-based Acceleration of Deep Convolutional Neural Networks on Mobile Platforms," *Distrib. Parallel Clust. Comput.*, 2015.
[110] Infineon Technologies, "Infineon Aurix TC297TA." [Online]. Available: http://www.infineon.com. [Accessed: 17-Apr-2016].
[111] W. M. Johnson, "Super-Scalar Processor Design," Stanford University, Technical Report CSL-TR_89_383, 1989.
[112] J. E. Smith and G. S. Sohi, "The microarchitecture of superscalar processors," *Proc. IEEE*, vol. 83, no. 12, pp. 1609–1624, 1995.
[113] A. Vajda, "Multi-core and Many-core Processor Architectures," in *Programming Many-Core Chips*, Springer US, 2011, pp. 9–43.
[114] "File:Superscalarpipeline.png - Wikimedia Commons." [Online]. Available: https://commons.wikimedia.org/wiki/File:Superscalarpipeline.png. [Accessed: 08-Mar-2016].
[115] S. Vassiliadis, S. Wong, and T. D. Hämäläinen, *Embedded Computer Systems: Architectures, Modeling, and Simulation: 6th International Workshop, SAMOS 2006, Samos, Greece, July 17-20, 2006, Proceedings*. Springer Science & Business Media, 2006.
[116] R. Zurawski, *Embedded Systems Handbook*. CRC Press, 2005.
[117] I. T. AG, "KIT_TC1797_SK - Infineon Technologies." [Online]. Available: https://www.infineon.com/cms/en/product/evaluation-boards/kit_tc1797_sk/. [Accessed: 12-May-2019].
[118] I. T. AG, "AURIX™ Family – TC297TA (ADAS) - Infineon Technologies." [Online]. Available: https://www.infineon.com/cms/en/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc2xx/aurix-family-tc297ta-adas/. [Accessed: 12-May-2019].
[119] "TMS320F28335 Delfino™ 32-bit MCU with 150 MIPS, FPU, 512 KB Flash, EMIF, 12b ADC | TI.com." [Online]. Available: http://www.ti.com/product/TMS320F28335. [Accessed: 12-May-2019].
[120] "TI TMS570LC4357." [Online]. Available: http://www.ti.com/product/tms570lc4357. [Accessed: 27-Jan-2015].
[121] "MPC564xL|32-bit MCU|Chassis-Safety | NXP." [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/power-architecture-processors/mpc5xxx-55xx-32-bit-mcus/ultra-reliable-mpc56xx-32-bit-automotive-and-industrial-microcontrollers-mcus/ultra-reliable-dual-core-32-bit-mcu-for-automotive-and-industrial-applications:MPC564xL. [Accessed: 12-May-2019].

[122] "MPC567xK|32-bit MCU|ADAS | NXP." [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/power-architecture-processors/mpc5xxx-55xx-32-bit-mcus/ultra-reliable-mpc56xx-32-bit-automotive-and-industrial-microcontrollers-mcus/ultra-reliable-mpc567xk-mcu-for-automotive-industrial-radar-applications:MPC567xK. [Accessed: 12-May-2019].
[123] "SPC58 E Line Multi-Core Microcontrollers (MCUs) for Critical Automotive Applications - STMicroelectronics." [Online]. Available: https://www.st.com/en/automotive-microcontrollers/spc58-e-line-mcus.html. [Accessed: 12-May-2019].
[124] "STM32F407/417 - STMicroelectronics." [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f407-417.html. [Accessed: 12-May-2019].
[125] "RH850 Family (Automotive only) | Renesas Electronics." [Online]. Available: https://www.renesas.com/in/en/products/microcontrollers-microprocessors/rh850.html. [Accessed: 12-May-2019].
[126] "i.MX 6QuadPlus Applications Processors | Quad Arm® Cortex®-A9 with extreme graphics performance and enhanced power management | NXP." [Online]. Available: https://www.nxp.com/products/processors-and-microcontrollers/arm-based-processors-and-mcus/i.mx-applications-processors/i.mx-6-processors/i.mx-6quadplus-processor-quad-core-high-performance-advanced-3d-graphics-hd-video-advanced-multimedia-arm-cortex-a9-core:i.MX6QP. [Accessed: 12-May-2019].
[127] "Infineon." [Online]. Available: http://www.infineon.com/. [Accessed: 31-Jan-2015].
[128] "Texas Instruments." [Online]. Available: http://www.ti.com/.
[129] "Freescale." [Online]. Available: http://www.freescale.com/. [Accessed: 31-Jan-2015].
[130] Xilinx, "What is an FPGA?," 2015. [Online]. Available: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm. [Accessed: 15-Jan-2015].
[131] I. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges," *Found. Trends® Electron. Des. Autom.*, vol. 2, no. 2, pp. 135–253, 2007.
[132] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*. Tallahassee, Florida: Springer, 2007.
[133] "Can FPGAs Beat GPUs in Accelerating Next-Generation Deep Learning?," *The Next Platform*, 21-Mar-2017. [Online]. Available: https://www.nextplatform.com/2017/03/21/can-fpgas-beat-gpus-accelerating-next-generation-deep-learning/. [Accessed: 14-Jul-2017].
[134] "Altera." [Online]. Available: http://www.altera.com/. [Accessed: 31-Jan-2015].
[135] Altera Corporation, "Cyclone V Product Table," 2018.
[136] Xilinx Inc., "7 Series Product Selection Guide," 2018.
[137] "Top FPGA Companies For 2013 - SourceTech411." [Online]. Available: http://sourcetech411.com/2013/04/top-fpga-companies-for-2013/. [Accessed: 31-Jan-2015].
[138] Xilinx Inc., "Xilinx All Programmable Functional Safety Design Flow Solution Product Brief (PB015)," 2014.
[139] Altera, "A Safety Methodology for ADAS Designs in FPGAs." Sep-2013.
[140] "TÜV-Qualified FPGAs for Functional Safety Designs." [Online]. Available: http://www.altera.com/end-markets/industrial/functional-safety/ind-functional-safety.html. [Accessed: 29-Jan-2015].
[141] Xilinx Inc., "Xilinx Isolation Design Flow." [Online]. Available: http://www.xilinx.com/applications/isolation-design-flow.html. [Accessed: 12-Jan-2015].
[142] Xilinx, "Spartan-6 FPGA Dual-Lockstep MicroBlaze Processor with IDF." 2012.
[143] "Exploring the Floating Point Performance of Modern ARM Processors." [Online]. Available: http://www.anandtech.com/show/6971/exploring-the-floating-point-performance-of-modern-arm-processors. [Accessed: 24-Jan-2015].
[144] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance FPGA-based accelerator for large-scale convolutional neural networks," in *Field Programmable Logic and Applications (FPL), 2016 26th International Conference on*, 2016, pp. 1–9.
[145] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *ACM SIGARCH Computer Architecture News*, 2010, vol. 38, pp. 247–257.
[146] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 161–170.

[147] N. Suda *et al.*, “Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks,” 2016, pp. 16–25.
[148] S. Hariprasath and T. N. Prabakar, “FPGA implementation of multilayer feed forward neural network architecture using VHDL,” in *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, 2012, pp. 1–6.
[149] A. Youssef, K. Mohammed, and A. Nasar, “A Reconfigurable, Generic and Programmable Feed Forward Neural Network Implementation in FPGA,” 2012, pp. 9–13.
[150] S. Sahin, Y. Becerikli, and S. Yazici, “Neural network implementation in hardware using FPGAs,” in *Neural Information Processing*, 2006, pp. 1105–1112.
[151] N. Buonaiuto *et al.*, “Satellite Identification Imaging for Small Satellites Using NVIDIA,” *AIAAUSU Conf. Small Satell.*, Aug. 2017.
[152] NVIDIA Corporation, “GPU-Based Deep Learning Inference: A Performance and Power Analysis.” 2015.
[153] NVIDIA Corporation, “CUDA Toolkit Documentation,” 2017. [Online]. Available: http://docs.nvidia.com/cuda/. [Accessed: 20-Oct-2017].
[154] Eric Brown, “Nvidia unveils Tegra K1 with 192 GPUs, preps 64-bits,” *LinuxGizmos.com*, 06-Jan-2014. .
[155] K. Morris, “Driver’s Ed for FPGAs: Will Altera FPGAs Drive Your Future Audi?,” 06-Jan-2015. [Online]. Available: http://www.eejournal.com/archives/articles/20150105-audi. [Accessed: 08-Jan-2015].
[156] K. Kim and K. Choi, “SoC Architecture for Automobile Vision System,” in *Algorithm & SoC Design for Automotive Vision Systems*, J. Kim and H. Shin, Eds. Springer Netherlands, 2014, pp. 163–195.
[157] Altera Corporation, “Altera Automotive Brochure.” 2015.
[158] Altera, “Cyclone V Hard Processor System Technical Reference Manual.” 2014.
[159] “Microsemi.” [Online]. Available: http://www.microsemi.com/. [Accessed: 31-Jan-2015].
[160] S. Jose, “Architecture Matters: Choosing the Right SoC FPGA for Your Application White Paper,” p. 43, 2013.
[161] “Breaking Moore’s Law,” *I, Cringely*, 15-Oct-2013. [Online]. Available: https://www.cringely.com/2013/10/15/breaking-moores-law/. [Accessed: 06-May-2019].
[162] A. D. Horowitz Kyle Kelley, James Mao, John P. Stevenson, Mark, “CPU DB: Recording Microprocessor History.” [Online]. Available: https://cacm.acm.org/magazines/2012/4/147359-cpu-db-recording-microprocessor-history/fulltext. [Accessed: 06-May-2019].
[163] “Intel’s former chief architect: Moore’s law will be dead within a decade - ExtremeTech.” [Online]. Available: https://www.extremetech.com/computing/165331-intels-former-chief-architect-moores-law-will-be-dead-within-a-decade. [Accessed: 06-May-2019].
[164] “A Look Back at Single-Threaded CPU Performance.” [Online]. Available: https://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/. [Accessed: 06-May-2019].
[165] S. Posey, “NVIDIA HPC Directions for Earth System Modeling,” p. 50.
[166] N. Navet and F. Simonot-Lion, *Automotive Embedded Systems Handbook*. CRC Press, 2008.
[167] M. Broy, S. Kirstan, H. Krcmar, B. Schätz, and J. Zimmermann, “What is the benefit of a model-based design of embedded software systems in the car industry?,” *Softw. Des. Dev. Concepts Methodol. Tools Appl. Concepts Methodol. Tools Appl.*, p. 310, 2013.
[168] J. L. Boulanger and V. Q. Dao, “Requirements engineering in a model-based methodology for embedded automotive software,” in *IEEE International Conference on Research, Innovation and Vision for the Future, 2008. RIVF 2008*, 2008, pp. 263–268.
[169] M. Conrad, I. Fey, and S. Sadeghipour, “Systematic Model-Based Testing of Embedded Automotive Software,” *Electron. Notes Theor. Comput. Sci.*, vol. 111, pp. 13–26, Enero 2005.
[170] “A Design Methodology for Safety-Relevant Automotive Electronic Systems.” [Online]. Available: http://papers.sae.org/2004-01-1665/. [Accessed: 26-Feb-2016].
[171] G. N. Vo, R. Lai, and M. Garg, “Building Automotive Software Component within the AutoSAR Environment - A Case Study,” in *9th International Conference on Quality Software, 2009. QSIC ’09*, 2009, pp. 191–200.
[172] “AUTOSAR: Basics.” [Online]. Available: http://www.autosar.org/about/basics/. [Accessed: 26-Feb-2016].
[173] K. Forsberg and H. Mooz, “The relationship of system engineering to the project cycle,” in *INCOSE International Symposium*, 1991, vol. 1, pp. 57–65.

[174] S. Balaji and M. S. Murugaiyan, “Waterfall vs. V-Model vs. Agile: A comparative study on SDLC,” *Int. J. Inf. Technol. Bus. Manag.*, vol. 2, no. 1, pp. 26–30, 2012.
[175] ISO, “ISO 26262-6:2018 - Road vehicles - Functional safety - Part 6: Product development at the software level,” 2018.
[176] M. Broy, S. Kirstan, H. Krcmar, B. Schätz, and J. Zimmermann, “What is the benefit of a model-based design of embedded software systems in the car industry?,” *Softw. Des. Dev. Concepts Methodol. Tools Appl. Concepts Methodol. Tools Appl.*, p. 310, 2013.
[177] K. Pohl, H. Honninger, R. Achatz, and M. Broy, Eds., *Model-based engineering of embedded systems: the SPES 2020 methodology*. Berlin ; New York: Springer, 2012.
[178] T. Rizano, R. Passerone, D. Macii, and L. Palopoli, “Model-based design of embedded control software for hybrid vehicles,” in *2011 6th IEEE International Symposium on Industrial and Embedded Systems*, 2011, pp. 75–78.
[179] J. L. Boulanger and V. Q. Dao, “Requirements engineering in a model-based methodology for embedded automotive software,” in *IEEE International Conference on Research, Innovation and Vision for the Future, 2008. RIVF 2008*, 2008, pp. 263–268.
[180] M. Conrad, I. Fey, and S. Sadeghipour, “Systematic Model-Based Testing of Embedded Automotive Software,” *Electron. Notes Theor. Comput. Sci.*, vol. 111, pp. 13–26, Enero 2005.
[181] *Modeling with UML*. New York, NY: Springer Berlin Heidelberg, 2016.
[182] F. Oquendo, J. Leite, and T. Batista, *Software architecture in action: designing and executing architectural models with SysADL grounded on the OMG SysML Standard*. Cham: Springer, 2016.
[183] ETAS, “ETAS - ASCET Software Products - Software Products & Systems - Product Search - ETAS Products,” 17-Dec-2009. [Online]. Available: http://www.etas.com/en/products/ascet_software_products.php. [Accessed: 14-Mar-2016].
[184] Esterel Technologies, “SCADE Suite KCG C & ADA Code Generator,” *Esterel Technologies*, 23-Mar-2015. .
[185] MathWorks Inc., “HDL Coder, Brochure,” 2018. .
[186] “GPU Coder.” [Online]. Available: https://es.mathworks.com/products/gpu-coder.html. [Accessed: 06-May-2019].
[187] “Using Vivado HLS C, C++, System-C Block in System Generator,” Dec. 2012.
[188] Xilinx Inc., “Xilinx, HLS.” [Online]. Available: http://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html. [Accessed: 30-Jan-2015].
[189] Xilinx Inc., “SDSoC Environment Platform Development Guide,” 2019.
[190] K. J. Åström and T. Hägglund, “The future of PID control,” *Control Eng. Pract.*, vol. 9, no. 11, pp. 1163–1175, Nov. 2001.
[191] S. J. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” *Control Eng. Pract.*, vol. 11, no. 7, pp. 733–764, 2003.
[192] S. Olaru, A. Grancharova, and F. L. Pereira, *Developments in Model-Based Optimization and Control: Distributed Control and Industrial Applications*. Springer, 2015.
[193] C. Sammut and G. I. Webb, *Encyclopedia of Machine Learning*. London: Springer, 2010.
[194] L. Li and F.-Y. Wang, *Advanced motion control and sensing for intelligent vehicles*. New York: Springer, 2007.
[195] F. Lamnabhi-Lagarrigue, A. Loría, and E. Panteley, Eds., *Advanced topics in control systems theory: lecture notes from FAP 2004*. London: Springer, 2005.
[196] M. Dendaluce, J. J. Valera, V. Gómez-Garay, E. Irigoyen, and E. Larzabal, “Microcontroller Implementation of a Multi Objective Genetic Algorithm for Real-Time Intelligent Control,” in *International Joint Conference SOCO’13-CISIS’13-ICEUTE’13*, Springer, Cham, 2014, pp. 71–80.
[197] M. Larrea, E. Larzabal, E. Irigoyen, J. J. Valera, and M. Dendaluce, “Implementation and testing of a soft computing based model predictive control on an industrial controller,” *J. Appl. Log.*, vol. 13, no. 2, Part A, pp. 114–125, Jun. 2015.
[198] M. Allende, P. Prieto, B. Heriz, J. M. Cubert, and T. Gassman, “Advanced Shifting Control of a Two Speed Gearbox for an Electric Vehicle,” presented at the Electric Vehicle Symposium 28, Seoul, Korea, 2015.
[199] E. Alpaydin, *Introduction to Machine Learning*, 2nd Revised edition. New. The Mit Press, 2010.
[200] D. K. Chaturvedi, *Soft Computing*, vol. 103. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.
[201] A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, 1959.
[202] S. Marsland, *Machine Learning: An Algorithmic Perspective*, 1st ed. Crc Pr Inc, 2009.

[203] A. Ng, “Standforg University - Course CS229 - Machine Learning Notes,” 2016.
[204] O. Chapelle, B. Schölkopf, A. Zien, and others, “Semi-supervised learning,” 2006.
[205] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *J. Artif. Intell. Res.*, pp. 237–285, 1996.
[206] M. J. Matarić, “Reinforcement learning in the multi-robot domain,” in *Robot colonies*, Springer, 1997, pp. 73–83.
[207] J. A. Lasserre, C. M. Bishop, and T. P. Minka, “Principled hybrids of generative and discriminative models,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, vol. 1, pp. 87–94.
[196] Simon Haykin, “Neural Networks: A Comprehensive Foundation”, 1994.
[209] A. De la Escalera, J. M. Armingol, and M. Mata, “Traffic sign recognition and analysis for intelligent vehicles,” *Image Vis. Comput.*, vol. 21, no. 3, pp. 247–258, 2003.
[210] L. Zhao and C. E. Thorpe, “Stereo-and neural network-based pedestrian detection,” *Intell. Transp. Syst. IEEE Trans. On*, vol. 1, no. 3, pp. 148–154, 2000.
[211] Y.-J. Zhai and D.-L. Yu, “Neural network model-based automotive engine air/fuel ratio control and robustness evaluation,” *Eng. Appl. Artif. Intell.*, vol. 22, no. 2, pp. 171–180, Mar. 2009.
[212] *Machine Learning in Computer Vision*, vol. 29. Berlin/Heidelberg: Springer-Verlag, 2005.
[213] K. Bennett, A. Demiriz, and others, “Semi-supervised support vector machines,” *Adv. Neural Inf. Process. Syst.*, pp. 368–374, 1999.
[214] Z. Sun, G. Bebis, and R. Miller, “On-road vehicle detection using Gabor filters and support vector machines,” in *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, 2002, vol. 2, pp. 1019–1022.
[215] T. H. Thi, K. Robert, S. Lu, and J. Zhang, “Vehicle Classification at Nighttime Using Eigenspaces and Support Vector Machine,” 2008, pp. 422–426.
[216] C.-M. Fu, C.-L. Huang, and Y.-S. Chen, “Vision-based preceding vehicle detection and tracking,” in *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, 2006, vol. 2, pp. 1070–1073.
[217] X. Li, D. Lord, Y. Zhang, and Y. Xie, “Predicting motor vehicle crashes using Support Vector Machine models,” *Accid. Anal. Prev.*, vol. 40, no. 4, pp. 1611–1618, Jul. 2008.
[218] Y. Chen, G. Liang, K. K. Lee, and Y. Xu, “Abnormal behavior detection by multi-svm-based bayesian network,” in *Information Acquisition, 2007. ICIA’07. International Conference on*, 2007, pp. 298–303.
[219] L. A. Zadeh, “Fuzzy logic and approximate reasoning,” *Synthese*, vol. 30, no. 3–4, pp. 407–428, 1975.
[220] L. A. Zadeh, “Fuzzy Logic, Neural Networks, and Soft Computing,” *Commun ACM*, vol. 37, no. 3, pp. 77–84, Mar. 1994.
[221] A. S. Cherry and R. P. Jones, “Fuzzy logic control of an automotive suspension system,” *Control Theory Appl. IEE Proc. -*, vol. 142, no. 2, pp. 149–160, Mar. 1995.
[222] S.-I. Son and C. Isik, “Application of fuzzy logic control to an automotive active suspension system,” in *, Proceedings of the Fifth IEEE International Conference on Fuzzy Systems, 1996*, 1996, vol. 1, pp. 548–553 vol.1.
[223] M. N. Aris Triwiyatno, “Engine Torque Control of Spark Ignition Engine using Fuzzy Gain Scheduling,” *TELKOMNIKA Telecommun. Comput. Electron. Control*, vol. 10, no. 1, 2012.
[224] C. von Altrock, “Fuzzy logic technologies in automotive engineering,” in *WESCON/94. Idea/Microelectronics. Conference Record*, 1994, pp. 110–117.
[225] D. E. Goldberg and J. H. Holland, “Genetic algorithms and machine learning,” *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, 1988.
[226] A. Tsakalidis and C. Thiel, “Electric vehicles in Europe from 2010 to 2017: is full-scale commercialisation beginning?,” European Comission, 2018.
[227] Eunice Project, “EUNICE Project Homepage,” 2015. [Online]. Available: http://eunice-project.eu/#!prettyPhoto. [Accessed: 17-Apr-2016].
[228] “3Ccar Project Homepage.” [Online]. Available: http://3ccar.eu. [Accessed: 17-Apr-2016].
[229] M. Duff, “2016 Ford Focus RS,” *Car and Driver*, 18-Jan-2016. [Online]. Available: https://www.caranddriver.com/reviews/a15102367/2016-ford-focus-rs-first-drive-review. [Accessed: 13-May-2019].
[230] “Mercedes-Benz AMG A45,” *Top Gear*, 13-Jan-2015. [Online]. Available: https://www.topgear.com/car-reviews/mercedes-benz/amg-a45. [Accessed: 11-Apr-2019].

[231] M. Hillenbrand, M. Heinz, N. Adler, J. Matheis, and K. D. Müller-Glaser, "Failure mode and effect analysis based on electric and electronic architectures of vehicles to support the safety lifecycle ISO/DIS 26262," in *Proceedings of 2010 21st IEEE International Symposium on Rapid System Protyping*, 2010, pp. 1–7.
[232] The mathworks Inc., "Automotive Industry Standards - ISO 26262 Support in MATLAB and Simulink." [Online]. Available: https://www.mathworks.com/solutions/automotive/standards/iso-26262.html. [Accessed: 20-Sep-2017].
[233] H. Kopetz, *Real-time systems: design principles for distributed embedded applications*, 2nd edition. New York Dordrecht Heidelberg: Springer, 2011.
[234] F. Paetsch, A. Eberlein, and F. Maurer, "Requirements engineering and agile software development," in *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.*, 2003, pp. 308–313.
[235] K. Schwaber, "SCRUM Development Process," in *Business Object Design and Implementation*, 1997, pp. 117–134.
[236] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61–72, May 1988.
[237] "ISO 3888_2_2011." .
[238] "Opel Insignia Grand Sport 2017 - Maniobra de esquiva (moose test) y eslalon | km77.com - YouTube." [Online]. Available: https://www.youtube.com/watch?v=P3U8WQrDEUk. [Accessed: 06-May-2019].
[239] nuerburgring.de, "Nordschleife." [Online]. Available: https://www.nuerburgring.de/fans-info/rennstrecken/nordschleife.html. [Accessed: 17-Apr-2019].
[240] P. Ruch, "Nürburgring, Nordschleife: Lieblingskurve Bellof-S, Hasskurve Wippermann," 24-May-2017.
[241] M. Ersoy and S. Gies, *Fahrwerkhandbuch: Grundlagen – Fahrdynamik – Fahrverhalten– Komponenten – Elektronische Systeme – Fahrerassistenz – Autonomes Fahren– Perspektiven*. Springer-Verlag, 2017.
[242] M. Mitschke and H. Wallentowitz, *Dynamik der Kraftfahrzeuge*. 2004.
[243] "Model longitudinal dynamics and motion of two-axle, four-wheel vehicle - MATLAB." [Online]. Available: https://www.mathworks.com/help/physmod/sdl/drive/longitudinalvehicledynamics.html. [Accessed: 26-Jul-2017].
[244] "CarSim." [Online]. Available: https://www.carsim.com/products/carsim/. [Accessed: 31-Jul-2017].
[245] I. A. GmbH, "CarMaker," 28-Jul-2017. [Online]. Available: https://ipg-automotive.com/products-services/simulation-software/carmaker/. [Accessed: 31-Jul-2017].
[246] "veDYNA." [Online]. Available: https://www.tesis-dynaware.com/en/products/vedyna/overview.html. [Accessed: 31-Jul-2017].
[247] "rFactor Pro." [Online]. Available: http://www.rfpro.com/. [Accessed: 31-Jul-2017].
[248] M. Blundell and D. Harty, *The Multibody Systems Approach to Vehicle Dynamics*. Elsevier, 2014.
[249] "Dynacar by Tecnalia." [Online]. Available: http://www.dynacar.es/en/home.php. [Accessed: 20-Jul-2016].
[250] Tecnalia Research and Innovation, "Dynacar 2.0 - Release Notes and User Manual." 2017.
[251] J. Cuadrado, D. Vilela, I. Iglesias, A. Martin, and A. Peña, "A Multibody Model to Assess the Effect of Automotive Motor In-wheel Configuration on Vehicle Stability and Comfort," in *ECCOMAS Multibody Dynamics*, 2013.
[252] R. Pastorino, F. Cosco, F. Naets, W. Desmet, and J. Cuadrado, "Hard real-time multibody simulations using ARM-based embedded systems," *Multibody Syst. Dyn.*, vol. 37, no. 1, pp. 127–143, May 2016.
[253] H. B. Pacejka, *Tire and Vehicle Dynamics, 2nd edition*, 2 edition. SAE International, 2005.
[254] "Pacejka - MapleSim Help." [Online]. Available: http://www.maplesoft.com/support/help/MapleSim/view.aspx?path=TireComponentLibrary/PacejkaTire. [Accessed: 31-Jul-2017].
[255] S. Garatti and S. Bittanti, "Parameter estimation in the Pacejka's tyre model through the TS method," *IFAC Proc. Vol.*, vol. 42, no. 10, pp. 1304–1309, 2009.
[256] "Pacejka '94 parameters explained – a comprehensive guide," *Edy's Projects*, 24-Dec-2011. .

[257] "Genetic fuzzy self-tuning PID controllers for antilock braking systems," *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/222555060_Genetic_fuzzy_self-tuning_PID_controllers_for_antilock_braking_systems/figures?lo=1. [Accessed: 31-Jul-2017].
[258] A. Pena, I. Iglesias, J. J. Valera, and A. Martin, "Development and validation of Dynacar RT software, a new integrated solution for design of electric and hybrid vehicles," *EVS26*, pp. 1–7, 2012.
[259] B. O. Varga, F. Mariasiu, D. Moldovanu, and C. Iclodean, *Electric and Plug-In Hybrid Vehicles*. Cham: Springer International Publishing, 2015.
[260] S. Onori, L. Serrao, and G. Rizzoni, *Hybrid Electric Vehicles*. London: Springer London, 2016.
[261] X. Chen, J. Wang, and A. Griffo, "A High-Fidelity and Computationally Efficient Electrothermally Coupled Model for Interior Permanent-Magnet Machines in Electric Vehicle Traction Applications," *IEEE Trans. Transp. Electrification*, vol. 1, no. 4, pp. 336–347, Dec. 2015.
[262] S. K. Chowdhury, "A Distributed Parameter Thermal Model for Induction Motors," in *2005 International Conference on Power Electronics and Drives Systems*, 2005, vol. 1, pp. 739–744.
[263] "ISO 2631-1:1997 - Mechanical vibration and shock -- Evaluation of human exposure to whole-body vibration -- Part 1: General requirements." [Online]. Available: https://www.iso.org/standard/7612.html. [Accessed: 10-Oct-2018].
[264] Thomas Scharnhorst, "Systemarchitektur Gesamtfahrzeug, AUTOUNI," Wolfsburg, 2006.
[265] "Zynq-7000 All Programmable SoC." [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. [Accessed: 31-Jul-2016].
[266] Xilinx Inc., "Zynq-7000 SoC Data Sheet: Overview (DS190)," 2018.
[267] "MATLAB - El lenguaje del cálculo técnico - MATLAB & Simulink." [Online]. Available: https://es.mathworks.com/products/matlab.html. [Accessed: 06-May-2019].
[268] "Xilinx Software Development Kit (XSDK)." [Online]. Available: https://www.xilinx.com/products/design-tools/embedded-software/sdk.html. [Accessed: 06-May-2019].
[269] "Home: PEAK-System." [Online]. Available: https://www.peak-system.com. [Accessed: 06-May-2019].
[270] "RACELOGIC - Racelogic | Experts in Data Logging, Video and GPS Simulation." [Online]. Available: http://www.racelogic.co.uk/index.php/de/. [Accessed: 06-May-2019].
[271] A. Parra, M. Dendaluce, A. Zubizarreta, and J. Pérez, "Novel Fuzzy Torque Vectoring Controller for Electric Vehicles with per-wheel Motors," presented at the XXXVIII Jornadas de Automática, Gijón, 2017.
[272] M. Hecht, T. Keilig, U. Kleemann, O. Polach, U. Seiffert, and R. Voit-Nitschmann, "Fahrzeugtechnik," in *Dubbel: Taschenbuch für den Maschinenbau*, K.-H. Grote and J. Feldhusen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. Q1–Q117.
[273] M. Dendaluce Jahnke, F. Cosco, R. Novickis, J. Pérez Rastelli, and V. Gomez-Garay, "Efficient Neural Network Implementations on Parallel Embedded Platforms Applied to Real-Time Torque-Vectoring Optimization Using Predictions for Multi-Motor Electric Vehicles," *Electronics*, vol. 8, no. 2, p. 250, Feb. 2019.
[274] J. T. Connor, R. D. Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 240–254, Mar. 1994.
[275] M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schluter, and H. Ney, "Comparison of feedforward and recurrent neural network language models," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, 2013, pp. 8430–8434.
[276] M. N. Karim and S. L. Rivera, "Comparison of feed-forward and recurrent neural networks for bioprocess state estimation," *Comput. Chem. Eng.*, vol. 16, pp. S369–S377, May 1992.
[277] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[278] K. S. Tai, R. Socher, and C. D. Manning, "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks," *ArXiv150300075 Cs*, Feb. 2015.
[279] "Optimizing Recurrent Neural Networks in cuDNN 5," *NVIDIA Developer Blog*, 06-Apr-2016. [Online]. Available: https://devblogs.nvidia.com/optimizing-recurrent-neural-networks-cudnn-5/. [Accessed: 28-Sep-2018].
[280] C. K. Chui and G. Chen, *Kalman Filtering: with Real-Time Applications*, 4th ed. Berlin Heidelberg: Springer-Verlag, 2009.

[281] M. S. Grewal, “Kalman filtering,” in *International Encyclopedia of Statistical Science*, Springer, 2011, pp. 705–708.
[282] M. Doumiati, A. Charara, A. Victorino, and D. Lechner, *Vehicle Dynamics Estimation using Kalman Filtering: Experimental Validation*. John Wiley & Sons, 2012.
[283] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.
[284] D. J. Montana, L. Davis, and M. St, “Training Feedforward Neural Networks Using Genetic Algorithms,” p. 6.
[285] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: closing the generalization gap in large batch training of neural networks,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 1731–1741.
[286] R. May and G. D. and H. Maier, “Review of Input Variable Selection Methods for Artificial Neural Networks,” *Artif. Neural Netw. - Methodol. Adv. Biomed. Appl.*, 2011.
[287] M. Dendaluce, I. Iglesias, A. Martin, P. Prieto, and A. Peña, “Race-track testing of a torque vectoring algorithm on a motor-in-wheel car using a model-based methodology with a HiL and multibody simulator setup,” in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 2500–2505.
[288] M. Dendaluce, J. J. Valera, V. Gómez-Garay, E. Irigoyen, and E. Larzabal, “Microcontroller Implementation of a Multi Objective Genetic Algorithm for Real-Time Intelligent Control,” in *International Joint Conference SOCO’13-CISIS’13-ICEUTE’13*, Á. Herrero, B. Baruque, F. Klett, A. Abraham, V. Snášel, A. C. P. L. F. de Carvalho, P. G. Bringas, I. Zelinka, H. Quintián, and E. Corchado, Eds. Springer International Publishing, 2014, pp. 71–80.
[289] “Jaguar I-Pace: Test, Crash, Daten, Videos | ADAC.” [Online]. Available: https://www.adac.de/der-adac/motorwelt/reportagen-berichte/auto-innovation/jaguar-i-pace-2018/. [Accessed: 06-May-2019].
[290] M. Dendaluce, V. Gómez, and E. Irigoyen, “Potential for applying Machine Learning in the context of Upcoming Automotive Technology,” presented at the XII Simposio CEA de Control Inteligente (SCI 2016), Gijón, 2016.
[291] M. Dendaluce Jahnke, “Ausarbeitung Forschungsseminar Maschinelles Lernen.” 16-Jul-2013.
[292] “Portal de Acceso a la Web of Knowledge.” [Online]. Available: https://www.recursoscientificos.fecyt.es/factor/. [Accessed: 12-May-2019].
[293] “Scopus.” [Online]. Available: https://www.scopus.com. [Accessed: 27-Feb-2019].
[294] “Scimago Journal & Country Rank.” [Online]. Available: https://www.scimagojr.com/. [Accessed: 27-Feb-2019].
[295] “Scimago Journal & Country Rank - MDPI Electronics Journal.” [Online]. Available: https://www.scimagojr.com/journalsearch.php?q=21100829272&tip=sid&clean=0. [Accessed: 27-Feb-2019].
[296] “The World University Rankings - Times Higher Education,” *Times Higher Education (THE)*. [Online]. Available: https://www.timeshighereducation.com/. [Accessed: 27-Feb-2019].