Novel Functional Safety Monitoring Software in Hybrid and Electric Vehicles

Botes, Frederik

Award date:
2019

Awarding institution:
University of Bath

Novel Functional Safety Monitoring Software in Hybrid and Electric Vehicles

Frederik F. Botes

A thesis submitted for the degree of
Doctor of Philosophy

University of Bath

Department of Mechanical Engineering

May 2019

This page is intentionally left blank.

Copyright Notice

Restrictions on use and Licensing

Access to this thesis/portfolio in print or electronically is restricted until

……. .

Signed on behalf of the Doctoral College

……..

Declarations of any previous submission of the work

The material presented here for examination for the award of a higher degree by research has / has not been incorporated into a submission for another degree.

Candidate's signature: 

Declaration of authorship

I am the author of this thesis, and the work described therein was carried out by myself personally.

Candidate's signature: 

Declaration of Published Work

At the time of submission, the author has published the following papers:

John Birch, Frederik Botes, Paul Darnell, David McGeoch. *Development of an Adaptive Safety Monitoring Function*. in SSS '16. 2016. Brighton: Safety-Critical Systems Club.

Frederik Botes, David McGeoch, Paul Darnell, Sam Akehurst, Andrew Hillis. *Development and Optimisation of an Adaptive Safety Monitor*. SAE World Congress 2018. K. Inniger. Detroit, SAE International.

Contents

List of Figures

Chapter 1 - Introduction

Chapter 2 – Literature Review

Chapter 3 – Safety Requirements, Ideal Monitoring Attributes, and Test Environment

Chapter 4 – Adaptive Safety Monitor

Chapter 5 – Principal Component Analysis Safety Monitor

List of Tables

Chapter 4 – Adaptive Safety Monitor

Chapter 6 – Validation & Verification Considerations and Concept Evaluation

Acknowledgements

My industrial supervisor, Dr. David McGeoch, for making me a functional safety engineer. Thank you for every way you have helped me to develop both my thinking and my practice, and for setting me off on this career path. Thank you for not always giving me the answer in my hand, but guiding me such that I learn the “why”.

My co-supervisor, Dr. Sam Akehurst, for supervising on my MSc first, and now also my PhD. Thank you for offering useful perspectives as the automotive academic.

My lead supervisor, Dr. Andrew Hillis, for taking me on as your PhD student. Thank you for mentoring me and developing me to the doctoral standard. Thank you for your patience, and always putting yourself aside to ensure I had what I needed to succeed.

Those that helped directly with this project. Paul Darnell, for taking interest in this project and supporting my own personal development; I look up to your curiosity and can-do spirit about everything. John Birch, for being the author on my first published paper at my first conference.

The support staff. Gillian, for making sure the office runs smoothly, and always being a friendly face. Kye, for helping with every imaginable IT problem, and for being a good friend.

My colleagues, peers, and close friends. Kevin, for embarking on the doctoral journey with me, everything that has come with it. Martin, for tolerating life with two PhD students. You both have made the process such a joy, and I am grateful for the friendships that were forged in this period of my life.

My parents, Lynette and Ryk, for giving me every opportunity to succeed, for pushing me towards greater goals, and for always giving me a place of respite when I needed it most.

My wife, Anna, who I fell in love with during this important time of my life. Thank you for being my perfect helper, encourager, and motivator when my stamina was at its lowest. You have made it all worthwhile and given me the push I needed to finish the job.

Lastly, I want to thank the Almighty Lord, Jesus Christ, who has given me the opportunity to further His Kingdom through my work.

Dedicated to my wife, Anna, whose selfless, patient, and ceaseless encouragement has been my driving motivation towards completion.

Abbreviations and Units

Abbreviation	Meaning
2D	Two-Dimensional
3D	Three-Dimensional
AC	Alternating Current
ACE	Autonomy, Connectivity, Electrification
ANN	Adaptive Neural Network
ASIL	Automotive Safety Integrity Level
BEV	Battery Electric Vehicle
C	Controllability
DC	Direct Current
DSAL	Data Safety Assurance Level
DSMP	Data Safety Management Process
E	Exposure
E/E	Electronic/Electrical
ECU	Electronic Control Unit
EKF	Extended Kalman Filter
EV	Electric Vehicle
FSC	Functional Safety Concept
FSR	Functional Safety Requirement
FTTI	Fault-Tolerant Time Interval
HARA	Hazard Analysis and Risk Assessment
HAZOP	Hazards and Operability Analysis
HEV	Hybrid Electric Vehicle
ICE	Internal Combustion Engine
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronic Engineers
IP	Intellectual Property
ISO	International Standards Organisaton
MIMO	Multiple-Input Multiple Output
MISO	Multiple-Input Single-Output
N/A	Not Available
NHTSA	National Highway Traffic Safety Administration
NPP	Nuclear Power Plant
OEM	Original Equipment Manufacturer
PC	Principal Component
PCA	Principal Component Analysis
PHEV	Plug-in Hybrid Electric Vehicle
PID	Proportional, Integral, Derivative
QM	Quality Management
S	Severity

SAE	Society of Automotive Engineers
SIL	Safety Integrity Level
SPC	Statistical Process Control
SVD	Singular Value Decomposition
TSC	Technical Safety Concept
TSR	Technical Safety Requirement
V&V	Validation and Verification
VSC	Vehicle Supervisory Controller

Unit Symbol	**Name**	**Domain**
%	Percentage	-
C	Celsius	Temperature
Hz	Hertz	Frequency
I	Current	Electrical flow
kg	Kilogram	Weight
kgm^2	Inertia	Moment of inertia
kph	Kilometres per hour	Velocity
kWh	Kilowatt Hour	Energy
m	Metre	Length
MIPS	Million instructions per second	Processing speed
Nm	Newton-Metre	Torque
rpm	Revolutions per minute	Angular Velocity
s	Second	Time
V	Volts	Electrical potential
W	Watt	Power

Abstract

Fuel economy and emission challenges are pushing automotive OEMs to develop alternative hybrid-electric, and full-electric powertrains, increasing variation in potential powertrain architectures, which exacerbates the complexity of control software used to coordinate various propulsion devices. Safety of this control software must be ensured through high-integrity software monitoring functions that detect faults and ensure safe mitigating action is taken. This monitoring functionality has itself become complex, requiring extensive modification for each new powertrain architecture to develop, calibrate, and verify the software to ensure safety as defined by ISO 26262. However, it must also be robust against false fault-detection, thereby maximising vehicle performance availability to the customer. It is therefore desirable to investigate whether novel approaches for software safety monitoring can address the manufacturer's complexity and calibration burden whilst robustly achieving safety with minimal effect on availability.

In this thesis, two novel functional safety monitoring concepts have been conceived and developed. First, an Adaptive Safety Monitor is introduced that aims to directly reduce the necessary safety software fidelity through new reasoning surrounding safety and driver expectation. An improvement in robustness is demonstrated by successfully using a low-fidelity safety software coupled with an adaptive safety monitor instead of a conventional high-fidelity model approach, and was shown to both accurately detect erroneous torque demand and prevent false-positive detection. Secondly, a novel Principal Component Analysis based safety monitor is introduced. An automated PCA model derivation process is developed that derives safety software automatically from the control software, with minimal effort from the OEM, and is shown to both quantitatively and qualitatively detect software faults within 5Nm of torque demand. These concepts are supported by a literature review, development context in relation to the ISO 26262 standard, and a set of ideal monitoring attributes derived from ISO 26262, ISO 25010, and expert opinion. A Matlab/Simulink electric vehicle model was created to serve as a simulation test-bed for both concepts. Lastly, considerations for verification and validation are explored before both developed concepts are evaluated according to the derived ideal attributes.

Chapter 1

Introduction

1 – Summary

The introduction will outline the motivation behind this project by discussing the demands of the automotive market and pressures on automotive OEMs, recent technology advances, and the trends that these advances have yielded in the automotive sector. The new challenges these trends bring to the automotive OEM will be discussed with particular attention drawn to increased vehicle control software complexity, leading to increased development costs for the manufacturer. From this, the research question will be asked, along with the aim and objectives this project seeks to achieve. Finally, the structure of this thesis will be outlined.

1-1 – Motivation

1-1.1 – Manufacturer Development Burdens

Modern vehicle manufacturers (OEMs) are being challenged more than ever before to develop new technologies to meet the demands of the market. Governments around the world are implementing ever-more stringent emissions legislation, such as the upcoming Euro VII terms in Europe, along with a replacement for the New European Drive Cycle that aims to provide emissions figures that better match real world driving, through the World Harmonized Light Vehicle Test Procedure [1]. To comply with these new emissions targets, manufacturers must develop their vehicles with the environment in mind. The penalties can be severe, as not meeting these targets for a mass-market vehicle will lead to heavy fines imposed on the manufacturer. Simultaneously, more demanding customer expectations in performance – seemingly at odds with reducing emissions – have forced manufacturers to improve vehicle technology in most other areas to remain competitive. Customers in most markets are seeking cars with better driving performance, refinement, features, and fuel economy, and new vehicle technologies are important to satisfying consumer desire in purchasing new vehicles [2].

1-1.2 – Recent Technology Advances

Technology advances in recent years have transformed vehicle engineering design capabilities and added features to vehicles in the large automotive sector that is still expanding. Underwood [3] highlights a number of key technological advances that have fostered the most change.

1-1.2.1 – Computing Power

Developments in microprocessors have, in general, followed Moore's Law [4] under which the transistor density in mainstream microprocessor circuits double roughly every 18 months. In turn, this means processor speed in mainstream microprocessors is continuously increasing. Figure 1-1 shows the result of Moore's Law since the 1970s with respect to how transistor density has increased average microprocessor clock speed.

Figure 1-1: A recent visualisation of the impact of Moore's Law, yielding exponentially increased average clock speed in mainstream microprocessors [5, 6]

Couple this trend of increased performance with the decreased manufacturing and energy costs per unit of performance, and the use of persistently more powerful processing devices in consumer vehicles remains feasible. Figures 1-2 and 1-3 show the reduction of manufacturing per transistor cycle and energy consumption per million instructions per second, respectively.

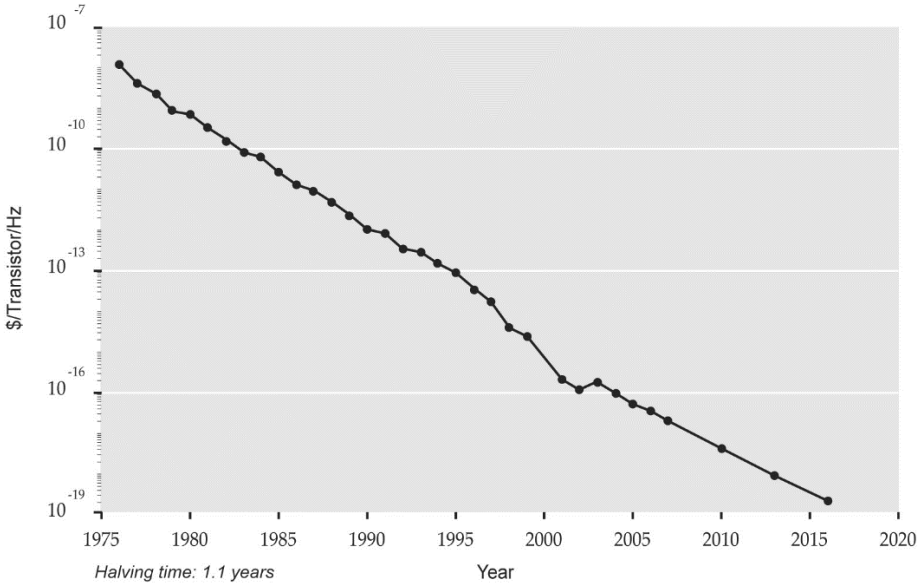

Figure 1-2: falling microprocessor manufacturing cost per transistor runtime cycle [6]

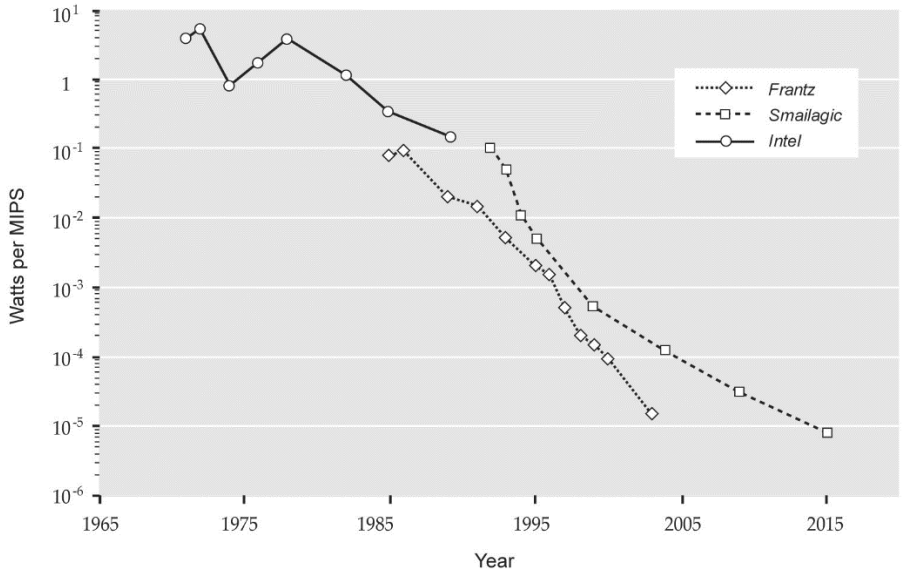

Figure 1-3: Improved energy efficiency per microprocessor instruction [6]

Indeed, established microprocessor companies have developed high performance computing processors specifically for new software applications in the automotive, such as Engine Control Units (ECUs) [7].

1-1.2.2 – Networking

Communication technology, specifically wireless communications, has increased exponentially too. Over time, Nielsen's Law of Internet Bandwidth [8] has described this trend as a 50% annualised growth in internet bandwidth. The introduction of high speed wireless internet protocols, such as 4G and the upcoming 5G service, can deliver wireless speeds of up to 20 Gbits/s [9]. Primarily driven by the consumer cellular phone demands, this communication service is already being used by vehicle entertainment systems. Already the wireless network is impacting consumer driving habits, as live traffic updates are being produced using information from clients connected to the wireless cellular network. With network capabilities seeing continuous investment in both first and third world countries, automotive manufacturers are seeking ways to utilise these newfound communication capabilities through new vehicle features that meet consumer demands. Further to this is the development of high-speed short-range communication protocols such as Dedicated Short-Range Communication, aimed at being able to quickly connect and share data with local transmitter/receivers [3].

1-1.2.3 – Battery Technology

Battery electric accumulators used for electric vehicles (EVs) are another trend that is seeing great strides in advancement. An obstacle to electric vehicles has – until recently – been the performance limitation of providing enough energy density (kWh/kg), and the cost limitation of reducing battery price per kWh ($/kWh). Recent breakthroughs in battery technology, and the establishment of more raw material suppliers and mass production battery manufacturers, has addressed both limitations, as average power density has risen and cost per kWh has fallen [10]. Indeed, based on 2012 forecasting, major automotive manufacturers have, in fact, been able to exceed predictions in reducing battery cost, shown in Figure 1-4 through analysis conducted by Wood Mackenzie

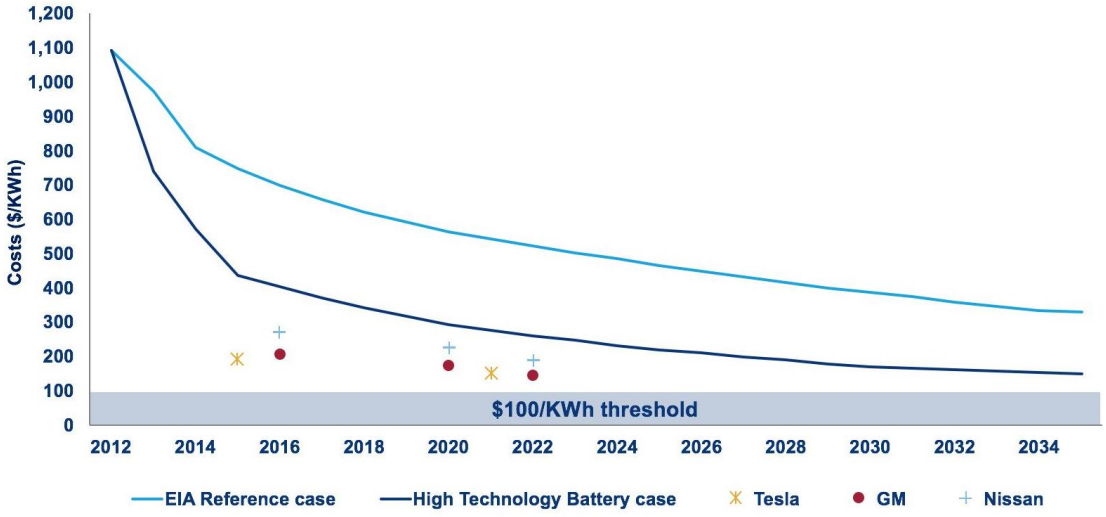

Figure 1-4: Wood Mackenzie 2012 battery cost forecast, with recent data and near future prediction by some major automotive OEMs [10].

Currently, lithium-ion batteries have been the main technology used in the automotive industry, but next generation research is already underway, with the much more widely abundant sodium-ion being a favourable avenue to drastically reduce the cost per kWh. Many predict these could be widely available as soon as 2020 [11].

1-1.3 – Automotive Industry Trends: Autonomy, Connectivity, Electrification.

The key technology drivers in the automotive industry has resulted in three identifiable technology trends emerging in the automotive sector: Autonomy, Connectivity, and Electrification [12], known as the 'ACEs' trends.

1-1.3.1 – Autonomy

Autonomy refers to some level of vehicle intelligence, whereby the vehicle can take over functions typically performed by the driver. This primarily includes control of acceleration, braking, and steering, but can extend to making driving decisions based on the environment, such as changing lanes, taking an exit, merging with the flow of traffic, and obstacle avoidance. SAE J3016 [13] is a published standard for categorising various 'levels' of autonomous capabilities a vehicle has; this is shown in Table 1-1.

Table 1-1: Summary of automated driving levels from SAE J3016 [13]

SAE Level	Name	Execution of Steering and Acceleration/ Deceleration	Monitoring of driving environment	Fallback performance of dynamic driving task	System Capability (driving modes)
0	No Automation	Human Driver	Human Driver	Human Driver	N/A
1	Driver Assistance	Human Driver and System	Human Driver	Human Driver	Some Driving Modes
2	Partial Automation	System	Human Driver	Human Driver	Some Driving Modes
3	Conditional Automation	System	System	Human Driver	Some Driving Modes
4	High Automation	System	System	System	Some Driving Modes
5	Full Automation	System	System	System	All Driving Modes

Many manufacturers have sold Level 2-capable autonomous vehicles, but the 'tipping point' comes when Level 3 is reached, as this moves the responsibility of monitoring the driving environment away from the driver and onto the vehicle, meaning liability shifts to the

manufacturer. Currently ‘autonomy’ is most widely seen in automated emergency braking systems which are expected to be mandated in Europe in the near future [14]. From a safety perspective, autonomy is seen as being able first to result in crashless vehicles to reduce road traffic accidents significantly through the aid of warnings and interventions for the driver, before eventually becoming ‘driverless’ vehicles, whereby the vehicle is able to fully control itself on the road and keep the occupants safe. Public opinion of driverless vehicles is still split, with recent high-profile fatalities being blamed on self-driving systems [15]. However, Underwood’s [3] forecast predicts the first stepping stone of crashless vehicles will lead to more widespread public acceptance of driverless systems, with these systems set to improve with more testing enabled by the recent technology advancements in networking and computing power.

1-1.3.2 – Connectivity

Technology advancements in wireless internet bandwidth and new short-range communication protocols have opened the possibility for the development of ‘connected vehicles’. These are vehicles that are connected to a network capable of supplying useful information that can affect driving decisions by either the driver or the autonomous vehicle, either through short range communication between nearby vehicles, between the vehicle and a greater infrastructure network, or generally between the vehicle and a connectivity cloud [3]. The information available to a connected vehicle can help aide infrastructure traffic flow, through the use of ‘intersection move assist’, ‘cooperative adaptive cruise control’ (or ‘platooning’), and live traffic information in order to reduce journey travel times and improve fuel efficiency [3]. It can also be used as a safety tool, utilising features such as ‘do not pass warning’, ‘control loss warning’ and ‘forward collision warning’ to help make better driving decisions.

A safety pilot has been conducted by the US Department of Transportation over a three year period starting in 2015, called The Ann Arbor Connected Vehicle Test Environment, whereby ~2800 vehicles were equipped with short range communication send/receivers [16]. These ‘vehicle awareness devices’ transmit ‘basic safety messages’ continuously, sending location and velocity data to a central network, which is then shared with nearby vehicles. Interestingly, the basic safety message is communicated over 5.9 GHz channel frequency, which was specifically set aside by the US Federal Communications Commission to support intelligent transportation systems, as far back as 1999; clearly, then, connected vehicles have been a long-term development, poised for growth over coming years.

1-1.3.3 – Electrification

Electrification of vehicle powertrains – of the three ACEs trends – is the furthest developed trend so far, yet still in its infancy in terms of development potential and market share. Electric powertrains were in fact first seen at the dawn of the motorcar era, but never found mass market appeal due to limitations on performance development versus the internal combustion engine [17]. Primarily, battery technology has been the greatest roadblock to mass market ownership. Over the past two decades, however, electrified powertrains have become a popular choice amid rising fuel prices and as battery technology has progressed to make such vehicles more commercially feasible in the mass [17]. Typically, the electric motors are still coupled to an internal combustion engine (ICE) in a hybrid-electric vehicle (HEV) or plug-in hybrid-electric vehicle (PHEV) application, but as battery technology advances more fully-electric vehicle (EV, also known as battery-electric vehicles, BEV) models are expected to be offered by manufacturers.

The State of Innovation 2017 report by Clarviate Analytics [18] looks at novel patent families granted across technology sectors in order to track research activities and technology advancement, with the most recent results shown in Table 1-2.

Table 1-2: State of Innovation 2017 tracking of patent subsectors submitted globally within the automotive sector.

%	Subsector	2016	2015	% Change
23%	Alternative powered vehicles	42880	37844	13%
11%	Navigation systems	21568	19753	9%
11%	Transmission	20299	20175	1%
10%	Seat, seatbelts and airbags	19754	18165	9%
10%	Safety	19076	18551	3%
7%	Suspension systems	13431	12827	5%
6%	Steering systems	11925	10841	10%
6%	Pollution control	10667	10114	5%
5%	Security systems	9217	8627	7%
4%	Braking systems	8262	7654	8%
4%	Engine design and systems	8048	7845	3%
2%	Entertainment systems	4591	4659	-1%

Within the automotive sector, the subsector that saw both the greatest percentage and year-over-year change in novel patent families was alternative powered vehicles, which the report notes comprise of mostly of electrified powertrain innovations. Tightening emissions is leading to some countries imposing national bans on new vehicles sold with non-electrified powertrains. These countries reportedly include the UK, China, India, Germany, France, among others, imposing such a ban between 2030 and 2040 [19]. Most drastically, the Netherlands has announced a ban on any non-zero-emission vehicle sold from 2030 [20]. The 2017 European market share of EV and BEVs is shown in Figure 1-5.

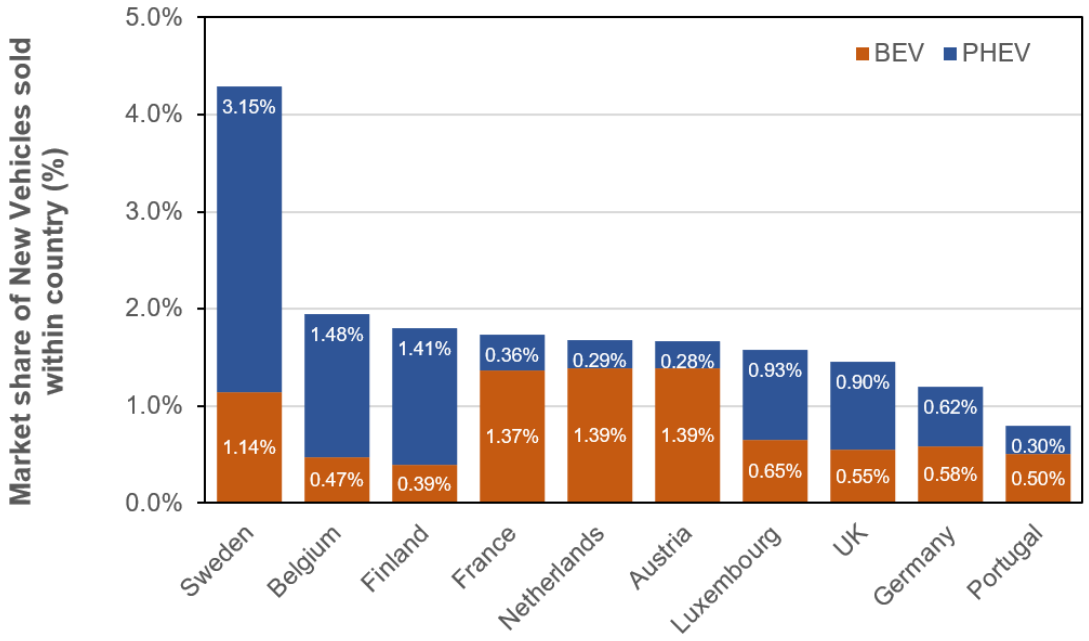

Figure 1-5: Market share percentage of BEV and PHEV of new cars sold within major European countries in 2017 [21].

With a combined EV and BEV new car market share less than 2% for most major European countries, and the emissions legislations being imposed by their governments, the market capitalisation potential is very high for electric vehicles over the coming years and into the future.

1-1.4 – ACEs and Software Complexity

While the ACEs trends advance the quality of product for consumers, meeting the demands of the market through greater convenience, performance, efficiency, and connectivity, there is an opportunity cost to each of these trends. The prospect of vehicle autonomy has prompted many

questions surrounding consumer lifestyle habits with regards to vehicle ownership. Many companies are investing in shared ownership business models, with on-demand vehicle hailing for a regular subscription fee. Autonomous vehicles are seen by many experts as the biggest fundamental change to the automotive industry since the introduction of the car itself. Integration of autonomous technology with the current road traffic rules and human drivers is also a concern: autonomous systems have already been blamed for many fatal incidents in the US and Europe [15], [22]. though most can attribute improper use and lack of driver intervention as the ultimate enabler. The connectivity trend will require significant investment from either private automotive companies, or local, national, and even international governments to improve the wireless communication infrastructure of vehicle-to-grid and vehicle-to-vehicle networks [3]; in either respect, the cost will most likely be passed onto the consumer through added purchase price or through taxation. Widespread vehicle electrification will require significant upgrades in the power grid capability to be able to cope with the demands of vehicle charging, resulting in more expensive electricity unit prices from the power suppliers, and possibly increased taxation due to national infrastructure upgrades [23].

Aside from the given research and development costs associated with any new technology, the ACEs trends all share a common denominator in increased software complexity. Each of the ACEs trends require significantly more information to be processed; autonomous vehicles process large amounts of visual data from cameras, as well as Lidar and radar systems, to understand the state of the environment around the vehicle. It must then process this data into the most suitable action/reaction, calling on much stored data [13]; this complexity is further compounded if the autonomous system is continuously learning through performance analysis.

Connectivity requires wireless and information processes to be incorporated with the vehicle software system, requiring additional computational power to process the numerous streams of information that is used to change performance and aide in driving decisions and driver recommendations. Autonomous vehicles can rely on vehicle-to-anything data to make driving decisions [24], and with the introduction of 5G networks capable of 1 to 20 Gbit/s data transfer [9] potentially huge amounts of network data can be made available for the vehicle control software to use. The network data could provide information of every local vehicle, pedestrian and object detected by the network of cars, all of which needs to be processed by the vehicle for the best course of action.

Electric powertrains introduce a fundamentally different method of delivering drive torque than mainstream vehicles have historically utilised up until recent years, with either petrol or

diesel ICEs. The software that controls the torque delivery, and management of the various subsystems of the electric powertrain (such as the inverters and accumulators), is both complex and substantially different from the software that controls torque delivery through an ICE. A single vehicle model may have had a range of different petrol or diesel powertrain variants, probably shared between some vehicle models, however, with the introduction of electrified powertrains in the mainstream automotive market the number of possible powertrain varieties for a single vehicle model similarly increases. In all, the cumulative software complexity and development effort required for a single vehicle model across all powertrain variants increases substantially. Furthermore, new driveability, efficiency and performance control features continue to push up software complexity over all hybrid and electric variants.

1-1.4.1 – Powertrain Control Software

The software that is primarily responsible for the control of all powertrain functions, and which contains all driveability, performance, and efficiency features, is called the 'functional software'. The functional software takes vehicle states, environmental measurements, and driver input, specifically, the accelerator pedal. Using these signals, the functional software is able to produce a 'torque demand', which is the overall powertrain torque that is requested from the powertrain [25]. This torque demand is then distributed between the various propulsion actuators for actualisation, with software components dedicated to controlling the individual actuators. When one considers all the factors that go into determining the appropriate torque demand, compounded with the trend of electrification, clearly the functional software that will become increasingly complex.

1-1.4.2 – Safety Software

Due to the software separating driver from powertrain control (i.e., there no longer exists a mechanical linkage between accelerator pedal and powertrain actuators), the functional software becomes a safety critical component. As such, it is essential that manufacturers verify that any unreasonable risk is mitigated within the functional software, as a fault or malfunction in the functional software can lead to a hazard. Indeed, the recent introduction of ISO 26262 Road Vehicles – Functional Safety [26] has warranted that manufacturers who aim to be ISO-compliant must prove that freedom from unreasonable risk is ensured in all electronic and electrical systems, including software.

With the complexity in the functional software, however, the verification of such freedom is a significant undertaking for the manufacturer, and is typically commercially infeasible.

Therefore, OEMs instead develop a simpler set of software which acts as a watchdog or comparator to the functional software, and which can be much more easily verified; this is called the 'safety software'. The safety software is a simplified version of the functional software, containing only the most important components of the functional software to ensure that safety is being achieved. It checks to see whether the torque demand being asked by the functional software is plausible, expected, and safe; the safety software is a fault detection and fault reaction mechanism used to ensure the vehicle meets the safety requirements set out by the manufacturer using ISO 26262. The context of functional and safety software is shown in Figure 1-6.

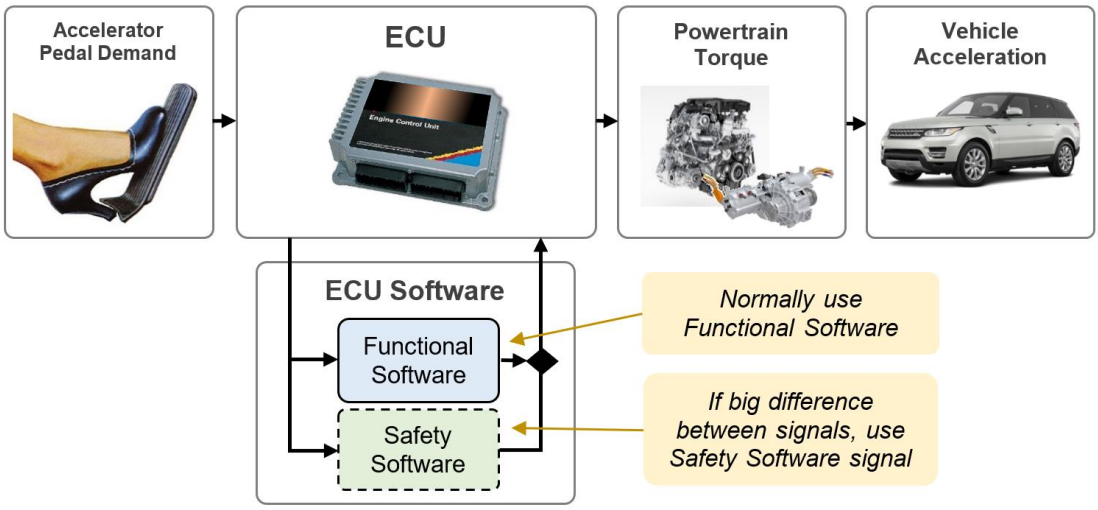

Figure 1-6: Functional and Safety Software in the context of driver controlling vehicle acceleration.

Compared to the functional software, the safety software is of a reduced complexity to enable feasible verification of safety, but also comes at the cost of some level of reduced accuracy when compared to the functional software. The safety software errs on the side of caution, but doing so can result in reduced system robustness, leading to an unnecessary loss of vehicle availability to the driver due to omitted functional software components in the safety software. Therefore, it becomes an issue of software fidelity: high fidelity safety software will be more accurate and robust, but its complexity will result in much higher development and verification efforts, whereas low fidelity software is less complex with reduced development and verification efforts, but at the cost of robustness. OEMs aim to develop the lowest cost safety software that meets robustness requirements.

Unfortunately, with the anticipation of much more complex functional software due to the ACEs trends, the safety software will in turn also start becoming more complex, as otherwise the fault detection system becomes much less robust. This leads to increased development and verification costs per powertrain variant. Furthermore, with the wider range of powertrain variants available, the cost of development and verification for the safety software for all the variants within a single vehicle model offering cumulatively increases the cost far beyond what it has been before, with the current safety software approaches. Safety software in particular is seeing a great amount of innovation and research, as can be seen in Table 1-2, where the automotive subsector of 'safety' (which excludes 'seats, seatbelts and airbags') is responsible for 10% of novel patent families being submitted in 2016 [18], putting it in the top five most-researched automotive subsectors according to this metric of innovation.

1-2 – Research question

Given the prospect of significantly increased software development, verification, and implementation costs to the OEM as the ACEs trends will tend to increase software complexity, the following research question should be asked:

- *Could a new approach or method of implementing safety software reduce the cost burden on the manufacturer, while still ensuring the safe operation of the functional software in vehicle powertrains?*

1-2.1 – Aims and Objectives

The research question has prompted this investigation into the current method(s) used by OEMs to achieve safety through safety software. Jaguar Land Rover, an external automotive OEM, has provided funding for this project, as well as made available expertise from its safety team. Using these resources, and expertise at the University of Bath, the aim of the project is as follows:

- *Investigate new safety software concepts and evaluate them against the current state of practice.*

To satisfy the aim, and answer the research question, the following five objectives are set:

1) **Objective 1:** Understand current state of practice, and identify new concepts from literature.

   A literature review will be conducted into fault detection methods, considering the current state of practice in the automotive industry, and other industries such as aerospace, locomotive, nuclear power plants and process engineering.

2) **Objective 2:** Identify ideal concept attributes and score candidates against them.

   The application within which the concept will be applied, and ISO 26262 standard which governs the topic, needs to be understood such that the intended application can be appropriately scoped. Furthermore, with expertise available from the OEM, as well as other applicable standards governing the software, a list of desirable attributes can be identified, against which candidate concepts and the benchmark can be scored against. These scores can be weighted according to desirability, and are used to narrow down the concept candidates to only those that provide the highest chance of success before being investigated.

3) **Objective 3:** Establish a testing method for testing performance of the concepts.

   To test whether the concepts are able to detect faults and perform satisfactorily, a valid testing method must be devised, including the development of a testing environment within which the concepts can be evaluated.

4) **Objective 4:** Conduct detailed investigation into candidate concept(s).

   With the concepts selected from the literature review, ideal concept attributes identified, and a testing environment established, the detailed investigation and development of the concept candidates can be undertaken.

5) **Objective 5:** Determine if candidate concepts could be suitable for a future production vehicle using ideal concept attributes.

   After the concept investigations take place, they are re-evaluated and re-scored against the ideal concept attributes post-investigation, using the outputs of the detailed concept investigations. The final evaluation will discuss each concept's score with each attribute in a discussion.

1-3 – Thesis Structure

The contents of each of the seven chapters of this thesis is detailed below:

1) **Chapter 1:** Introduction, containing the motivation for the project. Here, the current automotive technology trends of autonomy, connectivity, and electrification are highlighted, and the problems they bring to the OEM are discussed, specifically, increased software complexity. The software used in automotive vehicles is discussed, with focus on safety software. The research question is then asserted, and aims and objectives highlighted.

2) **Chapter 2:** The literature review, looking at the current state of practice with regards to safety software and fault detection in the automotive industry. This is then expanded to other industries, such as aerospace, locomotive, and nuclear power plants to identify possible concepts. A map of concepts is created and categorised, and these concepts explained and discussed. A number of these are subsequently selected based on their initial potential.

3) **Chapter 3:** An overview of the ISO 26262 standard, discussing mostly the concept phase of the standard, where the safety lifecycle process begins. In the first part of this chapter, a walk-through of the concept phase is conducted, leading to the functional safety requirements that provide the context within which the concepts are being developed within. Safety nomenclature and their meaning for the subsequent chapters are defined here. The next part of this chapter contains the identification of the ideal monitoring, using ISO 26262, ISO 25010, and OEM expertise from Jaguar Land Rover to determine which attributes are desirable in a safety concept. These are then weighted according to desirability. The benchmark concept and concept candidates identified in the literature review are then scored using knowledge gained from the literature review to eliminate low scoring concepts. The last part of this chapter introduces the test method, through the use of an electric vehicle model made in Matlab/Simulink software environment, ending with the testing method for the safety concepts being established at the end of the chapter.

4) **Chapter 4:** A detailed investigation into the adaptive safety monitor concept, this chapter provides a novel justification for the use of an adaptive module in the benchmark system. The basis of what constitutes an unexpected hazard is challenged,

and the way this new reasoning may allow for some adaptive function to be used. Furthermore, this adaptive function can allow a less complex set of safety software to be used while maintaining and/or improving concept reliability. The concept is then explained, and then tested using the test environment from Chapter 3, specifically with the use of a vehicle creep controller, and the results analysed. An automated optimisation technique is finally described and tested that aides in parameter tuning, and forms the basis of what could constitute future work with automated derivation.

5) **Chapter 5:** The second concept candidate is investigated, a principal component analysis (PCA) based safety concept. The development of the concept is documented, starting from fundamentals being applied to a linear system in a vehicle program to detect torque errors. A novel method of being able to identify fault direction as well as magnitude through output signal offsetting is introduced. Non-linear systems are then investigated, leading to the conception of the Local PCA concept which divides the system operating ranges and fits local PCA models to them. After this, an automated subdivision PCA process is created, which enables automated PCA-based safety software model generation based only the training data inputs and outputs. These models are then stored and used in real-time as a successful safety monitor to estimate torque error in the control software, demonstrated through a vehicle simulation. Finally, the limitations of the concept are discussed, and alternative PCA module architectures are presented.

6) **Chapter 6:** With the two key concepts investigated primarily for functional suitability, the verification and validation concerns unique to each concept is discussed in this chapter, along with their potential solutions. The two concepts are then evaluated against the original ideal concept attributes, with an updated post-investigation score and discussion for each attribute leading to final concept scoring and recommendation.

7) **Chapter 7:** In this final chapter, a summary of the research activities undertaken in this thesis is presented. The original research question is revisited and answered through the discussion of how the research activities meet the objectives set in this chapter. An overall conclusion is drawn, and future research activities originating from this investigation are described.

1-4 – Conclusion

Functional software used to control vehicles and their powertrains are a significant burden to OEMs in the process of product development. New automotive technology trends, autonomy, connectivity, and electrification (ACE) are set to exacerbate this burden through necessary increased functional software complexity, which drives increased safety software complexity, using current safety software approaches. It is therefore worthwhile to examine if novel approaches could reduce this burden.

With the project motivation outlined, and the research question proposed with the aim and objectives of the project, a literature review will be conducted to provide context within the field of fault detection safety systems, the current state of practice in the industry, and to identify possible safety concept candidates for further investigation.

Chapter 2

Literature Review

2 – Summary

A literature review needs to be undertaken to understand which safety concepts have seen successful implementation. This literature review will seek to cover the automotive industry to identify a concept to benchmark future concepts against, and to identify a number of preliminary concepts for future investigation. The fault detection strategies in the automotive sector will first be explored, then other industries such as locomotive, industrial plants and aerospace. Finally, safety concepts will be identified and discussed before the concept candidate shortlist is presented.

- **Objective 1**: Understand current state of practice and identify new concepts from literature.

2-1 – Fault Detection Strategies in the Automotive Sector

First, the current state of practice of safety monitoring in the automotive industry needs to be understood. This section will cover the benchmark concept against which future concept candidates will be evaluated.

2-1.1 – ISO 26262 Overview

ISO 26262 [27], released in 2011, is a new standard by which functional safety in road vehicles can be verified. A second edition was released in 2018 [28],but as this project was undertaken under the first edition, that will be used going forwards. The first edition contains ten parts that cover a traceable safety lifecycle process from the Concept Phase (Part 3 [27]), through the Product Development Phases (Parts 4-6 [29-31]), to the Production & Operation Phase (Part 7 [32]). Management guidelines (Part 2 [33]) are provided, as are Supporting Process information (Part 8 [34]), ASIL-Oriented and Safety-Oriented Analysis recommendations (Part 9 [35]). Vocabulary (Part 1 [26]) and Informative Guidelines (Part 10 [36]) make up the rest of the standard. Chapter 3 goes into more detail with regards to applying ISO 26262 to this project, but a few important ideas and concepts need to be introduced for this chapter.

ISO 26262 uses a ‘Hazard Analysis and Risk Assessment’ to identify potential vehicle-level hazards resulting from item-level malfunctions, with the item in this project being the powertrain functional software. Each hazard is classified by its severity, exposure, and controllability, using the Automotive Safety Integrity Level (ASIL) to determine how much risk a particular hazardous situation poses to safety. Subsequently, the commensurate rigour of development and verification for the item is required by the OEM to show that unreasonable risk has been mitigated. The ASIL-ratings range from lowest priority with Quality Management (QM, an acceptable level of risk), through to ASIL A, B, C, and finally with D requiring the most development rigour. A ‘safety goal’ is then assigned to each ASIL-rated hazard, which is then realised through a ‘functional safety concept’; this is what the safety software is for the powertrain functional software, a functional safety concept that mitigates unacceptable risk due to possible malfunctions in the functional software.

2-1.2 – E-Gas Safety Software Concept

In 1995, a consortium of German manufacturers (BMW, Daimler, VW, Porsche, and Audi) established the E-Gas Working Group, a safety-driven collaboration for establishing safety

standards for electronically controlled throttle bodies that were beginning to see widespread use [25]. As of 2002, the workgroup had released a standardised functional safety monitoring concept for gasoline and diesel ECUs, and their concept is designed to implement a high-level safety requirement, specifically the 'prevention of unintended acceleration' (which, according to their own hazard analysis and risk assessment, yielded an ASIL-B rating). It uses a central functional monitoring architecture with three levels: Level 1 is the 'functional level', which contains all of the complexities of normal operation, calculating torque demand, which includes all the variables that account for drivability, fuel efficiency, performance, driver modes, powertrain transient compensation etc. Here, a faulty torque demand could lead to a violation of safety, and therefore the safe operation of the functional level needs to be ensured. This task is carried out by Level 2, the functional monitoring level, which calculates the 'permitted torque demand', given some of the same information. Level 2 houses the safety software which calculates the permitted torque demand, and should the permitted torque demand exceed the torque demand from the functional software, a fault will be flagged, and the appropriate fault reaction or limitation will take place. Finally, to ensure that some hardware fault has not compromised the ability of the Level 2's monitoring integrity, a Level 3 subroutine is implemented, spanning over both the primary integrated circuit on the functional controller, and a separate integrated circuit called the monitoring controller. Figure 2-1 shows the E-Gas three-level concept for powertrain control.

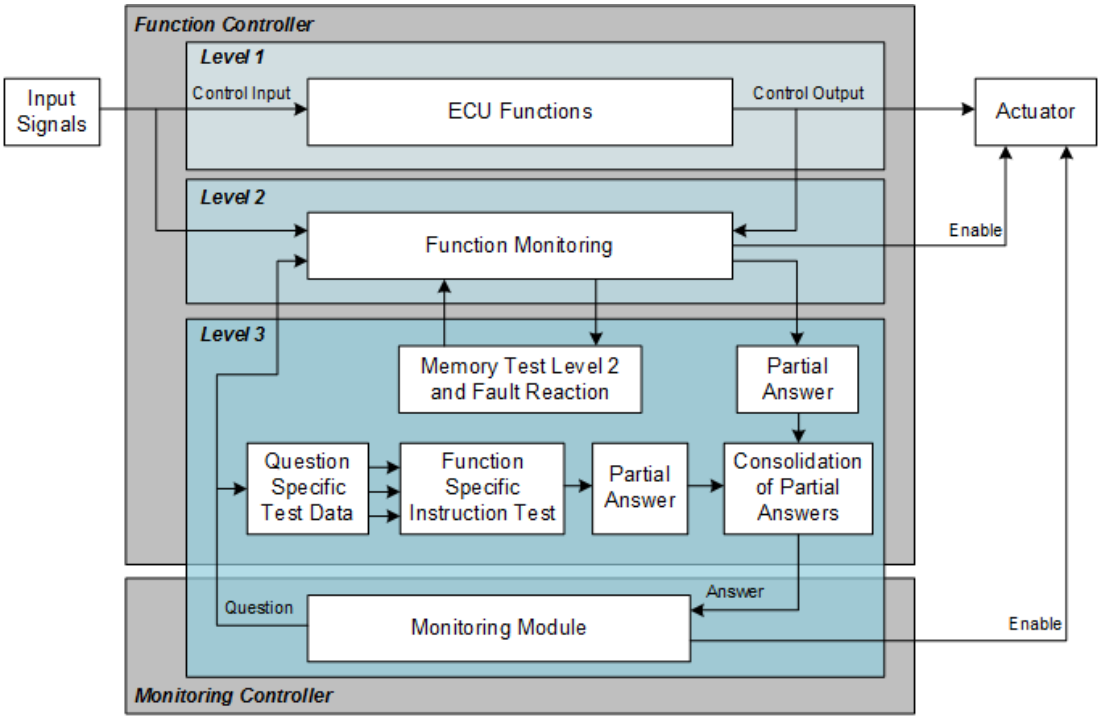

Figure 2-1: Interaction between three levels in E-Gas safety concept for ECU [25]

2-1.2.1 – Continuous Torque Demand Monitor

From the E-Gas Continuous Torque Monitor [25], a software-specific application of the safety concept can be created, called the ‘continuous torque demand monitor’. Instead of measuring directly the components of the vehicle powertrain in a feedback loop, the continuous torque demand monitor specifically looks at the cumulative torque demand desired by the system, before being interpreted by the propulsion actuators; this makes the monitor actuator independent, as it limits the scope to just the functional software.

The safety software typically uses a simplified mathematical model of the functional software to estimate whether the functional software behaves as it should. The safety software model takes in some of the same input signals as the functional software, including driver inputs (e.g. accelerator pedal position, steering angle, mode selection), powertrain and vehicle states (e.g. actuator rotational speed, gear selection, differential engagement status, battery state of charge), and environmental conditions (e.g. ambient temperature, humidity) to calculate the permitted torque demand. Not all signals are used, owning to the safety software being simplified. The actual torque demand is measured from the functional software, and the permitted torque demand subtracted from the actual torque demand to calculate the difference between the two, called the ‘nominal torque error’, which is sent to a fault decision block. If the nominal torque error is above some acceptable threshold, it is assumed that a malfunction has occurred, and because the safety software is more rigorously verified for safety, it is assumed that the malfunction has occurred in the functional software, and a fault is flagged by the system. When this happens a fault reaction mechanism is triggered to bring the vehicle back to a safe state. For example, if an excessive amount of torque is detected from the functional software, the fault reaction mechanism may limit torque to the maximum safe limit such that the driver retains control of the vehicle. Figure 2-2 shows the architecture of the continuous torque demand monitor [37].

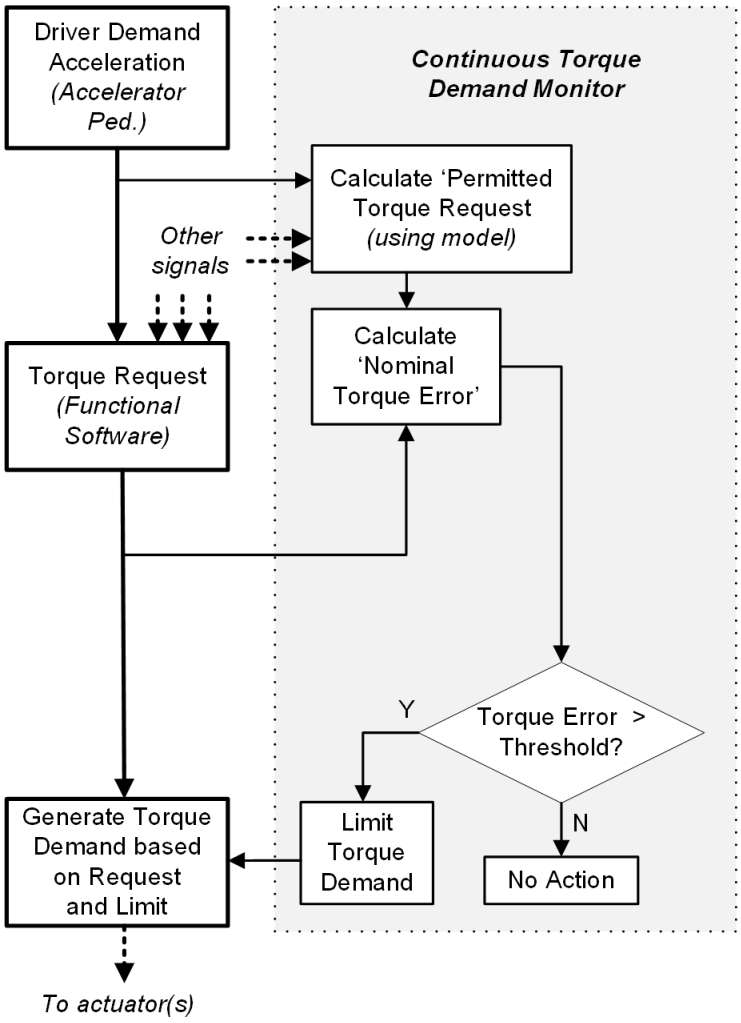

Figure 2-2: Continuous torque demand monitor, an example of model-based safety software design.

Figure 2-2 is the basic concept of how the safety software ensures that the functional software is operating correctly. In practice, there are many extra complexities bespoke to each vehicle and OEM, including monitors within the safety software model around different parts of the torque structure. The fault-check block is also not simply the instantaneous error of the measured and expected torque outputs, but often incorporates some level of statistical analysis, taking into account historical variables and counters. The fault check block also depends on other variables, as a fault in one context does not necessarily mean a fault in another, for the same calculated error.

The E-Gas software architecture is commonly used in the automotive industry for functional safety, especially among the Workgroup members and their suppliers. But as powertrain systems become increasingly diverse, particularly with the introduction of hybrid systems and

electrification, the interdependency and relations between new and existing functions and features increase the chance of a fault developing. In order for the E-Gas three-level approach to achieve the same level of robustness, the Level 2 monitoring functions need developing at a greater effort and cost, such that all possible fault causes and noise factors are accounted for.

2-1.2.2 – Safety Software Complexity

With the release of ISO 26262, the key question arises for the safety software engineer: 'how does one know the system is safe enough?' The straightforward answer is simply that if there is any error between the measured torque output and the expected torque output, then the system is not safe. In order to detect with 100% accuracy the occurrence and magnitude of any fault, the safety software model in Figure 2-2 needs to be a perfect representation of the functional software. Such a safety software model would yield an 'ideal expected torque output', which feeds into the error calculation block to produce the 'actual torque error'. This has zero error under normal operating conditions, but under malfunctioning conditions a fault occurs, an error value is produced. To have a perfect safety software model that could quantify the true magnitude of the malfunction with 100% accuracy would require that all signals and processing code for every possible noise factor be taken into consideration. A perfect set of safety software, however, is far too complex to feasibly verify, and so its fidelity needs to be reduced to ease the burden of development and verification; doing so introduces noise factors.

Noise factors are variables and underlying processes that, left unaccounted for, reduce the accuracy of the error calculation. The degree to which they affect the error calculation is dependent on the significance of that noise factor. Safety software complexity is reduced by choosing those noise factors that have the least significance, and would not reduce the ability of the concept to detect the presence of a safety-critical malfunction. Should the complexity be reduced too far, robustness of the safety monitoring concept is impacted, as it will incorrectly determine a fault has occurred – based on the safety model – when a true fault has not. Such a scenario is termed a "false positive", whereby safety is maintained, but availability of the function is unnecessarily reduced. Causes of noise factors usually fall into the following categories [38, 39]:

- **Unstructured Uncertainties**: Plant, component, or environment dynamics and variables that are not accounted for *a priori* in the safety software, leading to discrepancies between the mathematical safety software model and the plant itself.

For example, not accounting for vehicle mass when estimating torque error by measuring vehicle acceleration.

- **Process Noise**: Limitations of the embedded software environment to perform perfectly accurate calculations of the physical domain. For example, accumulation of rounding errors due to using software variables with low bit depth.
- **Measurement Noise**: Inaccuracies in the sensor measurement of the physical domain. For example, the high-frequency noise often found in accelerometer measurements, which often needs to be filtered and approximated.

Attempting to account for every possible noise factor in such a highly dynamic plant – while keeping the safety monitor robust – is infeasible. Besides this, checking if any fault occurs (no matter how small), is usually over and beyond the requirement for meeting even the most stringent ASIL D rated safety goal. While that is the ultimate goal for the safety software, achieving this state in reality proves to be more effort than is required to ensure safety of the item; this is why engineers simplify the safety software by making the safety software model less representative of the functional software. This results in a system that estimates the expected torque output rather than perfectly imitates it, thus it is called the 'estimated expected torque output', and the corresponding error calculation yields the 'estimated torque error'. Because it is not fully representative, there is some degree of signal variance where noise factors have not been fully modelled and accounted for.

It is important to remember that having a safety software model that is not fully representative of the functional software does not necessarily mean the safety goal would not be met. Indeed, as long as the safety goal is met, the safety software model can be as simple as needs be. What could change is the probability of a false positive. Engineers overcome this by incorporating an error threshold in the fault decision block that will ignore small error values, attributing them to the fact that a simpler, less representative safety software on its own will have some error variance under normal conditions. Even if there actually is a small fault, it would not be a safety risk to the driver as long as the safety objective is still met. A more representative safety monitor can generally use a smaller error threshold without false positives occurring, whereas a less representative safety monitor typically needs a larger threshold as normal error variance is greater due to more unaccounted noise factors being present. As with most things in engineering, a balance needs to be found, where the error threshold in the safety software is small enough to reliably detect if the safety goal is close to being missed (without misdiagnosing a fault where there isn't one), but such that the safety software is developed at a low cost (possibly meaning greater signal variance). Development effort is therefore directed

towards compensating for noise factors that have the greatest effect on variance and signal noise, as these are the most likely to give a false reading.

2-1.2.3 – Adaptive Safety Monitor Concept

The problem with rising safety software complexity due to the ACEs trends [3] has been stated. If, however, there were some way to use a simplified safety software model and still achieve safety, this would directly reduce the complexity of the safety software. The problem, naturally, is that this would also incur many more false positives due to noise factors that are left unaccounted for in the simpler safety software. However, if there were some way to adapt to these noise factors through an additional software function, while still achieving safety, robustness could be regained. Therefore, this theorised 'adaptive safety monitor' is a direct answer to the problem of safety software complexity by achieving the same result, with enough robustness, all while using a simpler safety software model that requires less development, verification, and implementation effort [40]. The adaptive safety monitor is discussed further in Chapter 3 during concept selection, and throughout Chapter 4.

2-1.2.4 – Safety and Electrified Powertrains

It is worth briefly discussing the effect that hybrid and electric vehicle and system design has on safety. Conventional ICE vehicles use only one torque source, whose power is split between two or four wheels through one or more differentials. Vehicle handling can be affected by which axle(s) are being driven, and the position of the engine mass with regards to these axles. Besides these arrangements, and active differential control (i.e. torque vectoring), there is not much else in a conventional powertrain that can significantly affect vehicle dynamics in the event of unintended acceleration or deceleration. With the introduction of electric motors, however, the number of powertrain system variants become much greater [41]. Because of this, some powertrains are susceptible to new hazards that require consideration, and new safety goals.

In addition to the ICE powertrain topologies, electric motors can be placed on the engine crankshaft, after the transmission but before a differential, on the previously un-powered axle before a new differential, a motor per wheel. These options are not exclusive either, so a combination of many of these can be seen. The 2016 Honda NSX hybrid vehicle, for example, uses two motors independently driving the front wheels, and a third motor aiding the ICE with torque fill and regenerative braking [42]. From a safety point of view, independently driven wheels can not only lead to unintended acceleration or deceleration when a torque malfunction

occurs, but can also cause the car to change course through unintended lateral yaw [43, 44]. The 2016 Honda NSX powertrain is shown in Figure 2-3.

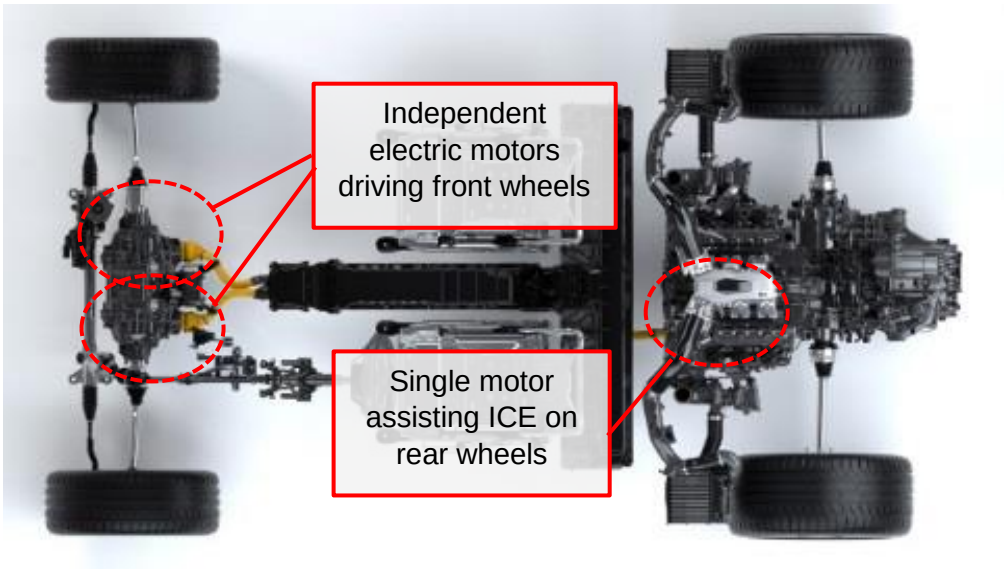

Figure 2-3: 2016 Honda NSX, an example hybrid powertrain with independent electrically driven front wheels. [45]

Additionally, depending on system architecture, a fault in one of the wheels could cause it to generate current and drive the other motor(s), further worsening the unintended steering effect [46]. Using this powertrain layout brings with it the added safety goal of ensuring that the cross-axle difference in torque from the motors is not enough to cause unintended vehicle instability. Achieving safety could require that steering angle, vehicle heading estimation and wheel slip be accounted for in the safety software.

In summary, therefore, while having a variance of powertrain topologies bring with them exciting opportunities for better vehicle dynamics, it also presents an opportunity for new hazards. To the OEM, this stresses the importance of functional safety in the products it sells, and the need to ensure powertrain malfunctions do not lead to vehicle level hazards in all powertrain variants. The extra effort to ensure safety is now more difficult than ever due to these unique powertrain characteristics, hence why new safety concepts are of interest. It is possible that other industries could use safety concept methods that are now worth examining in the new era of vehicle powertrains.

2-2 – Fault Detection Strategies in Other Sectors

Safety concepts span a great number of industries and fields where ensuring the safe operation of electronically controlled systems is vital. There are many different methods used in practice, but a select few are typically favoured by certain industries; this is because some methods lend themselves better to some applications than others, be it due to flexibility, cost effectiveness, accuracy or robustness, available resources, packaging constraints etc. Moreover, engineers specialising in a certain industry use well established safety concepts that are both familiar to them, and are proven in the field. Indeed, ISO 26262 Part 4 Section 7.4.3.4 [29] encourages the use of "well-trusted automotive systems design principles" in order to reduce systematic failures. Oftentimes, standards agencies, institutes and organisations that govern codes of practice for a particular industry issue safety monitoring standards, endorsing a handful of safety monitoring concepts for their field. This was not the case with ISO 26262, as ISO intentionally provided a broad risk-based verification and assessment process for validating whatever method an OEM would like to use for the desired application, rather than a one-method-fits-all approach. In doing so, ISO has left it up to the engineers to tailor their approach for their specific application in the widening scope of items that are electronically controlled in a vehicle. For this reason, it makes sense to assess the safety monitoring concepts found in industries outside of automotive as well.

When changing focus to other industries, it is important to remember that the frame of reference of the environment in which safety concepts and monitors operate has also changed. Vehicles are consumer products used in a dynamic and noisy environment, whereas the nuclear industry's massive plants operate within tightly controlled, low-tolerance environments. Nevertheless, there are elements of safety concepts that could be adapted for a hybrid or EV application.

2-2.1 – Locomotive

In many ways, road vehicles are similar to trains. Both operate in dynamic environments, both have self-contained drive units (both ICE and electric motors), and both carry occupants. Trains are, however, commercial modes of transit that are often supported and operated by the government. More importantly, trains operate on fixed routes. Because of this, trains have been using standardised Train Running Diagnostic Systems for over 40 years, consisting of a 5T system [47, 48]. The 5T system uses five independent monitoring systems:

- Train Coach Running Diagnostics System.
- Trace Hotbox Detection System.
- Trackside Acoustic Detection System.
- Trouble of Moving Freight Car Detection System.
- Track Performance Detection System.

Of the five diagnostics systems, only the Train Coach Running Diagnostics System is located on the train itself, whereas the rest are trackside. Analysis for trackside diagnostics takes place away from the track at the railway control centre through 'Communication-Based Train Control' [49, 50]. For road vehicles, this is not yet a feasible concept, as the infrastructure has not yet developed for Cloud-based diagnostics to be implemented in every vehicle, but the future of connected vehicles may bring with it the telemetry infrastructure to implement an external safety concept [50]. [51] uses an external monitoring method using cameras to capture data on rail tracks, which are then processed using trained ANNs to detect cracks in the railroad. However, the reliability is hampered by numerous noise factors such as glare, weather and obstacles. In 2018, Renault released the SYMBIOZ concept, a level 4 prototype vehicle that uses special highway telemetry to aid in the autonomous vehicle capabilities [24].

The Train Coach Running Diagnostics System is part of the main 'Train Control and Monitoring System' that handles all vehicle controls within the 'Train Consist Level' [52].The system architecture includes two Vehicle Control Units (one active, one redundant), and a Safety Control System consisting of a Safety Supervisor and Wheel Slide Protection. The latter two are SIL (Safety Integrity Level) validated and are required to comply with IEC 61508 [53]. Since trains are commercially operated, and since packaging is less of a constraint than in a road vehicle, the monitoring system can use more sensors at greater cost, measuring current flow to electric motors.

Guo et al. [54] submitted a patent for detecting a system fault in an electric power locomotive, whereby multiple data inputs are analysed and compared against a fault knowledge library. If abnormal data was found, it would match an entry in the library, and a fault-reaction would occur. This is called an Expert System, as a knowledge base is pre-programmed before commission (see Figure 2-6). While this can be useful to some degree in automotive safety software, the sheer number of variants and environmental noise factors mean that developing this knowledge base will require significant effort. An interesting train diagnostics paper was written by Sunder et al. [55] where they developed a safety concept for detecting component faults in an ICE-2 trailer type by analysing trailer vibrations, though not for the purpose of torque monitoring. A similar paper was written by Chao et al. [56], and a patent based on this

concept was applied to identifying bearing cage failures in electric motors by [57]. Data was collected with accelerometers, and analysis was conducted in the time-domain, in the frequency-domain, and with statistical discriminants in parallel, finding that each were useful in detecting different types of faults. For road cars, then, it could be possible to study vibrations in parts of the powertrain to detect if torque is being applied, with whole vehicle vibration measured as a control measurement to isolate what is happening in the powertrain alone. In reality, vibration-type torque monitoring is better suited for trains as there are fewer noise factors to account for, and the level of fidelity needed to accurately measure torque rules out this approach. Component vibration characteristics could also change over time with wear, which would need to be tested and modelled. However, some of the statistical techniques seen in this paper such as examining the variance, skewness, low-band Kurtosis and high-band Kurtosis could be used for identifying errors through a noisy signal [55].

2-2.2 – Industrial and Nuclear Power Plants

Industrial plants are of a different nature to the topics seen so far. The purpose they serve is not a transport mission, and the plant environment itself is tightly controlled so as to reduce external noise factors. In an effort to mitigate risk, reduce errors, and increase efficiency, the majority of the normal operation is controlled either automatically or by a human-operated control centre, particularly in the nuclear field where radiation poses a significant threat to life. Despite this, power plants and process engineering plants are still staffed by human workers whose safety is of the utmost importance, and as such require safety measures to be implemented in the control systems to ensure this. Many of the safety objectives are easily ensured with regular quality management tasks such as scheduled component maintenance, sensor calibration, actuator tests, equipment cleaning and verification [58]. Different types of plants have different maintenance requirements and operational safety precautions, typically based on the SIL associated with the risk involved. These risks vary as much as the purpose of the plant; a nuclear power plant (NPP) needs to ensure safe nuclear material handling, whereas a pharmaceutical chemical processing plant must ensure that the product is manufactured correctly.

Ensuring safety, in many ways, does not make business sense in the automotive world. Leading automotive manufacturers invest a lot of money developing redundant fault detection systems that may not even be needed over the course of the vehicle's lifetime. Adding a new feature introduces cost, both through developing its functionality, but also through developing its corresponding safety mechanism. Safety systems are implemented, therefore, for the prime purpose of ensuring the safety of occupants and ensuring the safe and dependable operation of

the vehicle. For industrial plants, however, safety monitoring is inherent to the business model. While a failure may or may not cause harm to workers, it will always affect production i.e. loss of profit. A catastrophic failure is usually due to either an extraordinary external circumstance such as an earthquake [59] (something that can be planned for but not avoided), or the compounding effect of a series of multiple small failures within the plant. Since chemical plants and NPPs are commercially operated, more money is typically spent on the safety systems than in a consumer product. Some examples of failure consequences could include:

- Nuclear reactor meltdown.
- Incorrect chemical ratios.
- Fire or Explosion.
- Pollutant escaping into the environment.

To avoid these, then, the application of safety monitoring techniques in industrial plants are imperative. Of all accidents and safety system failures, NPPs receive the greatest amount of attention due to the nature of the material being handled, and the resulting effects of a failure. The most infamous incident occurred in 1986 at the Chernobyl power plant in Ukraine [60] after a reactor meltdown resulted in an explosion, releasing large amounts of radiation into the atmosphere. More recently, the 2011 Fukushima disaster was a result of an off-shore earthquake and subsequent tsunami that damaged electrical power supply lines and safety infrastructure, causing loss of cooling function to all the reactors [59].

NPPs adhere to a strict maintenance schedule on all measurement equipment to reduce the likelihood of a sensor or data transmission fault occurring. IEC/IEEE 61582 governs safety standards in NPP instrumentation and control important to safety [58]. NPPs require very accurate measurements in order to tightly control reactor temperatures. Testing the conditions of electronic equipment is important to safety, and includes standards for indenter modulus tests [61], elongation at break tests [62], oxidation induction time tests [63], and optical time domain reflectometry [64]. In the past two to three decades, NPPs have shifted away from analogue control to digital electronic automated instrumentation and control. This has increased functionality, but also complexity, leading to greater risk of latent and common cause faults [65]. Electrical/electronic functional safety is covered generally by IEC 61508 so similar terminology such as Safety Integrity Level (SIL) is used in this context. The International Atomic Energy Agency's (IAEA) safety policy promotes the use of redundancy, diversity and independence in safety-critical measurement and signals [65]. Diversity means the way in which the parameter is measured is different for each redundant signal, and could

include varied design and measurement principles, and/or different components bought from different vendors. Redundancy means multiple measurements and sensors for a safety-critical parameter, such as reactor temperature. The multiple signals are then processed via voter software to that tests for faulty signals or signal carriers. A triple modular redundancy chip was developed and tested by Hiari et al. for an automotive application, reducing cost by containing it on a single microchip [66]. Figure 2-4 shows their design of triple modular redundancy architecture, an example of diverse redundancy.

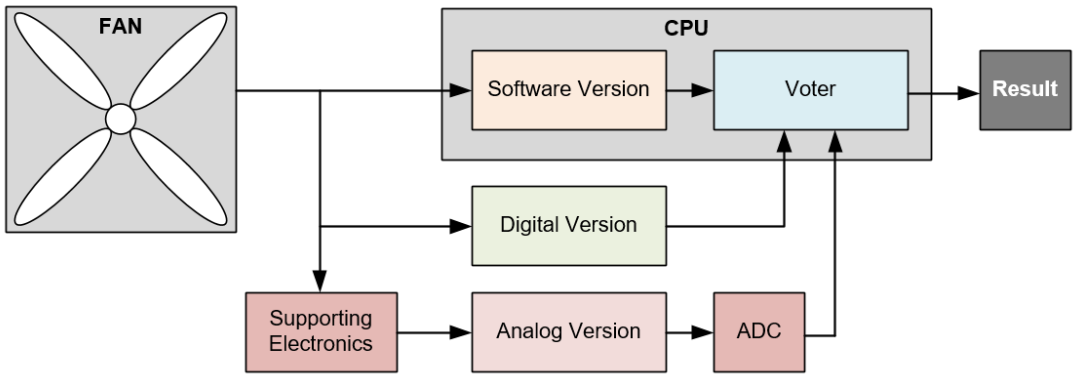

Figure 2-4: Example Triple Modular Redundancy design for fan speed [66].

Others have extended this to N-modular redundancy, with [67] demonstrating a number of alternative arrangements to suit the application, which is reducing hardware power consumption in their case. [68] suggested a quadruple modular redundancy approach using three hardware redundancies and an additional adaptive software approach to be resistant to dual-point faults. Quadruple redundancy is also needed to truly be tolerant to Byzantine faults, a subset of faults that are based on the Byzantine generals problem [69]. Independence is attained through isolating electricals, physically separating and dividing communication between each subsystem. With multiple redundancy can come complexity, however, so the IAEA recommends purchasing pre-qualified and trusted commercial-off-the-shelf hardware and software components to reduce the possibility of risk with new system design.

With the high level of information available by the various sensors and resources available, and given the tightly controlled environments, statistical methods are popular in industrial plants; these are discussed further in Section 3.

2-2.2.1 – Analytical Redundancy

All fault detection processes require system measurements in order to understand the current state of the plant. These measurements are made using sensors within the plant. If the sensor was to fail, the fault detection would not be able to measure how the system was performing, and thus could not know if the system develops a safety critical fault. Often, the same sensors are used for functional and safety software, but this is not always the case. In order to overcome this, some level of redundancy needs to be implemented such that the safety software can still operate in the event a sensor fails. The simplest answer would be to introduce multiple sensors (or actuators); this is called hardware redundancy. Usually the most robust option, it is a necessity in some applications such as throttle pedal position, but brings with it extra hardware costs as well the difficulty of packaging extra components.

The other option is that of analytical redundancy, the process of estimating the sensor output by relying on algebraic relationships between that particular measurement, and a number of other dependent process variables. Analytical redundancy is typically described by two categories [70-72]:

- **Direct Redundancy**: A 'virtual sensor' value is computed by combining the values of other sensors with an algebraic model. The virtual sensor value is then cross-checked against the measured value from a real sensor. A sensor fault would show a discrepancy between actual and computed values.
- **Temporal Redundancy**: Differential relations between a physical plant (or actuator) and sensor value – using *a priori* models of the physical plant – can be used to check the plausibility of a single sensor. For example, if a wheel speed sensor estimates vehicle speed has increased from 10 m/s to 40 m/s within a time step of 10 ms, though both values are plausible in isolation, it is likely impossible that the vehicle had accelerated at a rate of 3000 m/s^2, indicating a sensor fault.

Using analytical redundancy is also beneficial in isolating a fault source in some cases [73] [74]. For example, Tabache et al. [75] used analytical redundancy to reconstruct a virtual sensor from other variables in an EV powertrain. Tabache et al. highlights that safe operation of electric motors in a road vehicle operate on quality sensor measurements from motor current, voltage and speed sensors. These sensors are subject to faults originating from disconnections, as well as signal noise, drift and offset. They therefore developed a fault tolerant control system, whereupon, if a fault was detected in the actual speed sensor, the analytical redundancy produced 'virtual speed sensor' would substitute in place of the faulty

hardware by way of a voter scheme. It is therefore possible that some ‘virtual torque sensor’ could be derived for use in a safety software model. [76] used a mathematical engine model to generate the expected engine output under healthy conditions, from which they could derive residuals for fault detection.

2-2.3 – Aerospace

Much like both the automotive and locomotive industries, aeroplanes are self-propelled vehicles that carry passengers and goods in a highly dynamic environment. Of particular interest is the fly-by-wire technology that is used in most modern aircraft and jets, where a pilot (or autopilot) request a certain action that the control system then uses to determine actuator actions. Planes experience very many noise factors during operation that aren’t as significant in both the automotive and locomotive fields, such as altitude change (which brings with it greatly varying temperatures and air density), cross wind, and mass changes due to fuel burn. Tactical aerospace vehicles are particularly vulnerable to faults as many of them are aerodynamically unstable [77]. Military aerospace contractors purposefully designed them for agility, with the ability to perform manoeuvres that are impossible in a commercial passenger aircraft. In order to keep the tactical aircraft flying, then, a huge effort is placed on the control systems engineers to keep the aircraft stable because the training and mental strain for a pilot to fly the vehicle would make it unfeasible.

Since aircraft operate off the ground there is the inherent risk that, if a component should fail, the worst-case scenario is a crash landing which is usually much more severe in an aircraft than a road vehicle due to collision speed and crash structures. The easiest way to ensure safety in case of a failure is hardware redundancy, but aside from the inevitable added cost, this is usually not even a viable option due to weight and packaging constraints. Safety monitoring is therefore doubly important, as the safe state of the aircraft may not allow an engine to simply be shut off since the aircraft might not be able to land safely; naturally, this is even more of a risk with helicopters due to their relative inability to glide without power. The requirements for aircraft airworthiness is described by an inverted relationship between failure severity and probability, shown in Table 2-1 [78].

Table 2-1: Classification of aerospace failures based on fault severity and likelihood [78]

Fault Class	Fault Severity	Permissible Likelihood	Prob. per flight hour
Minor	Slight functional capability reduction. Physical discomfort.	Probable	10^{-3}
Major	Slight functional capability reduction. Physical distress.	Remote	10^{-5}
Hazardous	Large functional capability reduction. Few serious or fatal injuries.	Extremely Remote	10^{-7}
Catastrophic	Normally occurs with hull loss. Multiple fatalities to occupants.	Extremely Improbable	10^{-9}

From Table 2-1, parallels can be drawn with ISO 26262 in how failure classification is determined, as both use a severity and probability (related in many ways to 'exposure' in ISO 26262). 'Controllability' is a key criteria that was added to ISO 26262 [35], as the writers wanted to capture the drivers' ability to avoid an accident when assessing risk. In some areas, system monitoring goes hand-in-hand with safety monitoring, as the airline wants to check that the aircraft is functioning as efficiently as possible. Health management and fuel burn are arguably the two greatest operating costs to commercial airlines, so the airlines want to ensure that their jets are operating at their best performance and catch any faults that could be detrimental to achieving that performance. Safety monitoring, in a way, is an inherent by-product to system health monitoring, but it should be noted that health monitoring does not cover all safety functions.

As computing electronics made their way into aircraft, digital computers took over many fly-by-wire functions, and subsequently became responsible for flight-critical functions [79]. Initially, ensuring the safety of a flight-critical system was achieved through hardware redundancy, adding multiple sensors, actuators, and computers to ensure that a fault was extremely improbable. A voting system was devised such that the correct signal could be identified. Some traditional analysis techniques included:

- **Limit Checking**: Fault presence if pre-set plant limits were exceeded.
- **Frequency Spectrum Analysis**: Some plant measurements produce a certain normal frequency spectrum. Deviation could indicate a fault presence. Some faults may even produce a characteristic frequency, which could aid in fault isolation.

- **Fault Dictionary Approach**: Compare system behaviour with known fault characteristics, trained *a priori*.

However, with the sheer number of components that require redundancy in an aircraft, the added cost was astronomical both in buying the components but also weight and added maintenance. As such, with more powerful computer processor hardware being developed, analytical redundancy began to emerge to meet safety goals with less hardware components needed, with the model-based approach used most readily. The model utilises analytical redundancy to reduce hardware-redundancy components (e.g. triplex hardware redundancy reduced to duplex [66]) and a knowledge-based approach that makes use of human system knowledge to form the expert system approach. The expert system is useful for a complex system such as an aircraft or road vehicle, in that it combines quantitative reasoning (analytical and hardware), and qualitative reasoning (knowledge-based approach) to expertly detect faults in the system in addition to using some process history. The aerospace industry has used what is called fault-tolerant control architecture. Figure 2-5 shows an example of fault-tolerant control system design.

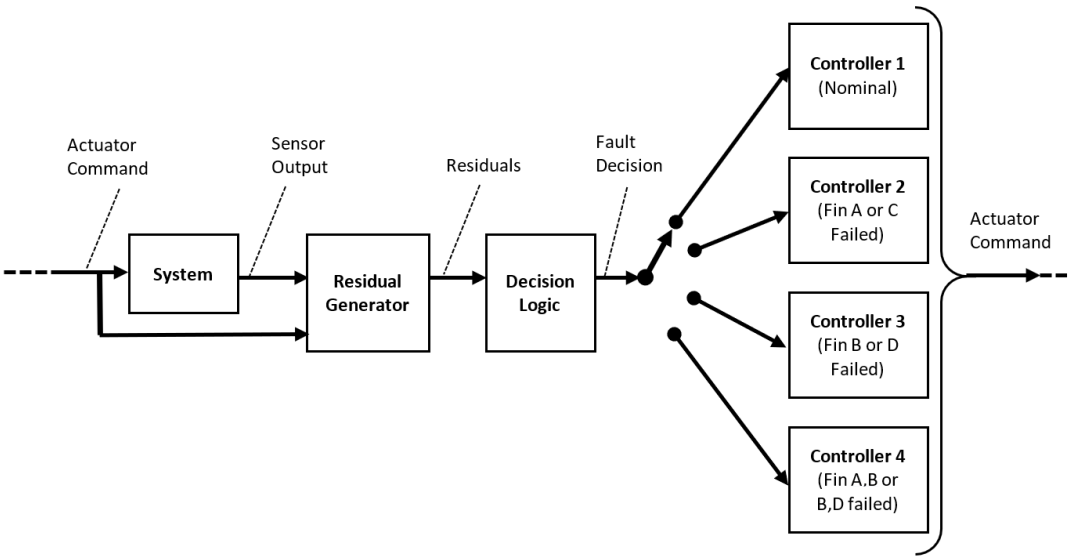

Figure 2-5: Fault-Tolerant Control system design [77]

Figure 2-5 shows an example of fault-tolerant control where multiple controllers are designed with a 'decision logic' block that switches between them based on a perceived fault. The fault is determined within what is called the 'residual generator block' [77]. For the residual generation code, parity space is used which is based on the state-space model. A set of relations, called parity relations, results to zero unless there is a fault, whereby the result will deviate significantly from zero. Using diagnosis equations, the decision logic block then

selects the optimum subsystem controller based on the remaining healthy actuation channels. With simulation on real data, Mukhopadhyay et al. [77] found that the controllers could effectively diagnose and find the fault in real time. [80] used a Kalman Filter bank to detect faulty sensors in an aircraft gas turbine system. The method showed promise in being able to identify a fault sensor by generating residuals in one of the Kalman filters when it had failed. This could be used in a ground vehicle powertrain to detect a faulty sensor where multiple redundant ones are available. Figure 2-5 is actually very similar to the process used currently in the automotive industry. The residual generator is essentially the error between the actuator command and the sensed system state, so it is comparable to the error calculation block of the continuous torque demand monitor in Figure 2-2. The decision logic block is similar to the fault-check block, and the different controllers and selection would be the fault-reaction mechanism. Parity relations are further explored in Section 3.

Mukhopadhyay et al. [77] opens up the topic of Reconfigurable Flight Control - or more generally reconfigurable controllers - reprograms a particular functions controller based on a fault occurrence, in an attempt to reallocate actuator responsibility and maintain a safe level of performance. Typically, a fault-tolerant control system requires:

1) A sufficiently robust, reconfigurable controller.
2) A sufficiently robust fault detection and diagnosis scheme.
3) A reconfiguration mechanism.

Reconfigurable control is seen in the automotive industry where Wang & Wang [81] used fault-tolerant control in a four wheel independently driven electric vehicle. Should one of the motors fail, their control effort redistribution would ensure a safe and controllable vehicle for the driver. Zhang & Jiang [82] classifies Reconfigurable Flight Control methods as linear quadratic regulator, eigenstructure assignment, pseudo-inverse, model following, multiple model, fuzzy logic, adaptive control and neural network; some of these will be discussed later in this section. Zhang & Jiang does note, importantly, that most of these methods assume a perfect fault diagnosis is already available, which is sometimes not the case. Ultimately, then, while the actual reconfiguration mechanism is part of the overall safety concept, it is more part of the fault-reaction mechanism than the fault-detection strategy [83], and thus falls just outside the scope of this project but is very closely linked to the topic.

2-3 – Fault Detection and Diagnostic Methods

From analysing the various industries, the topic of fault detection software concepts can be broken up into two major categories: model-based [74] and process history based [84]. A hierarchy of fault detection and diagnostic methods are mapped out in Figure 2-6.

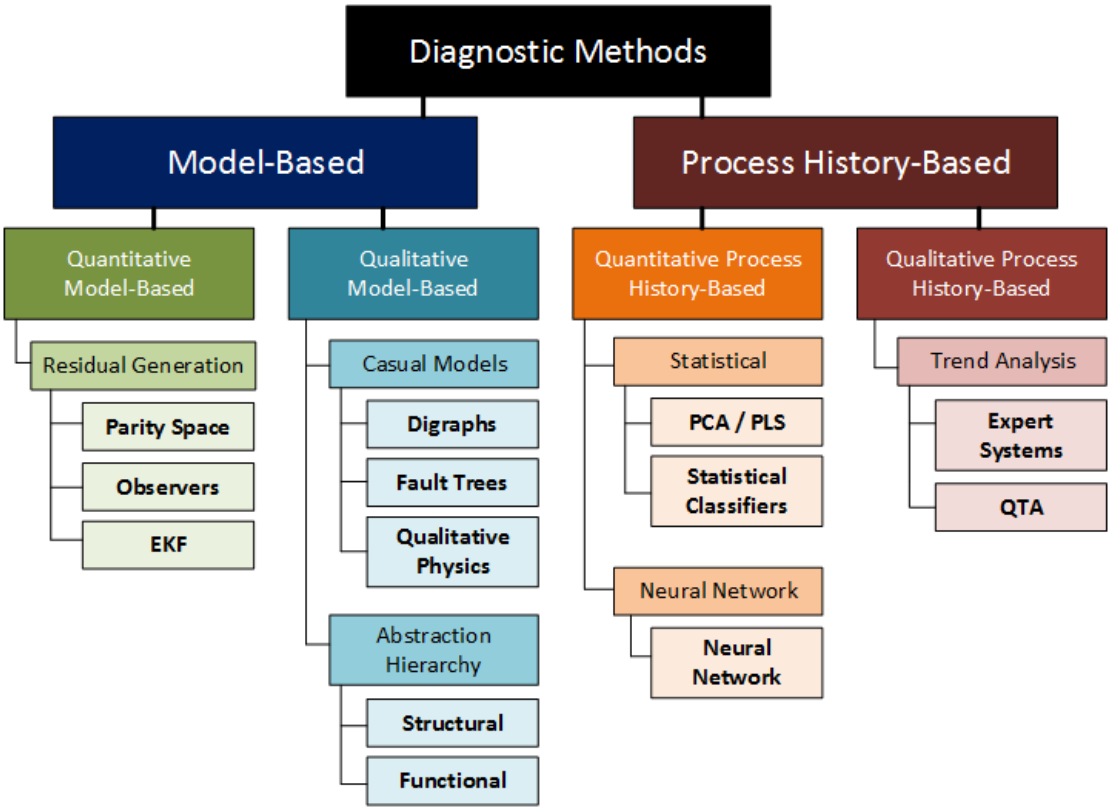

Figure 2-6: Classification of a number of different diagnostic techniques [84]

Figure 2-6 shows the classification and relations between a number of different diagnostic methods. Model-based approaches are what have been described in the E-Gas concept, where a mathematical safety software model is used to determine if the system is behaving as expected, or if a fault has occurred such that performance has deviated. The mathematics in the model is an interpretation of the physics that happen within the vehicle. The interpretation of the physics (i.e. the *a priori* knowledge) can therefore be classified into two types of expressions: quantitative model-based and qualitative model-based. Quantitative model-based methods find a mathematical functional relationship between system inputs and outputs through the use of transfer functions or filters. Qualitative model-based approaches on the other hand uses equations that express the relationships as qualitative functions centred around units in a process.

Process history-based approaches, on the other hand, can only use a large amount of historical input and output process data as it's *a priori* knowledge. The ways in which historical datasets are interpreted, called the 'feature extraction' processes, can also be classified as qualitative and quantitative. Qualitative methods examine qualitative trends, whereas quantitative methods can either be statistical-based extraction methods, or non-statistical such as neural networks. As there are many diagnostic types to consider, only quantitative model-based and quantitative process-history based will be examined in depth in this report, as they seem to be more directly comparable with the current benchmark.

All fault detection systems need some level of *a priori* knowledge to determine if a fault has occurred; without knowing how the system should behave, the safety software cannot know if the system is misbehaving. Venkatasubramanian [74] examines the underlying commonality between all diagnostic methods in how the measurement data is handled. In all methods, the following process is followed:

1) **Measurement space**: The inputs into the diagnostic system, this is where the sensor measurements (including processed but not altered signals) exist, with no *a priori* system knowledge affecting their input.
2) **Feature Space**: From the measurement space, *a priori* knowledge is used to extract the features of the measurements. Developing a feature space is useful as it lowers the complexity of information going into the decision space. This is because features (i.e. faults) tend to 'cluster' more distinctly once *a priori* filtering is applied, and as such classification and discrimination of features become easier. Thus, having more *a priori* knowledge benefits the transformation from measurement space to feature space into a more usable input for the decision space.
3) **Decision Space**: Features from the feature space are mapped into the decision space using some form of objective meeting function, such as a threshold seen in the fault check block of Figure 2-2 (top or a discriminant function as part of an implemented search or learning algorithm. If *a priori* knowledge is extensively used in the mapping from measurement space to feature space, the burden of the search / learning algorithm is greatly reduced.
4) **Class Space**: This is where the results of the decision space are classified into different failure classes, typically through the use of simple thresholds. From here, the diagnosis system will decide which fault reaction to undertake, based on the class of fault identified in the class space.

Therefore, fault detection is the combination of prior knowledge with fault searching and system learning algorithms to process measurement data into types of faults. Prior knowledge can be produced in a number of ways, depending on the engineer's knowledge of the system. It could include invariant relationships between sensor outputs and actuator inputs, based on energy balances and material properties in a process [74]. Alternatively, it could take form in an appropriate system identification transform model, resulting in a bank of filters being used. In process-history based systems, *a priori* knowledge could also use statistical Gaussian distribution expectations of measured data. Fault detection can therefore be divided into two basic steps [70]:

1) **Generation of Error**: This incorporates the Measurement Space and Feature Space from above, and benefits from more *a priori* knowledge.
2) **Decision and isolation of faults**: Decision Space and Classification Space from above.

With this in mind, each of the quantitative-type concepts highlighted in Figure 2-6 will be talked about in more detail.

2-3.1 – Model-Based Methods

2-3.1.1 – Parity Relations

Parity space is one of the simplest forms of model-based fault detection methods. Frank [70] explains that the main idea is to monitor the parity (consistency) between the mathematical equations of the system (analytical redundancy relations) and the actual sensor measurements, and a fault is thus declared if the error between the two exceed the preassigned error bounds. The parity equations are obtained by rearranging and transforming variants of input-output and state-space safety software models, with the model structure rearranged to get the best coverage for fault isolation [74]. Venkatasubramanian et al. performs a review of parity relations in the chemical and industrial processing field, and found that while parity relations were easy to generate using on-line process data and can effectively isolate faults, none of the sources they cited could handle gross process parameter drifts without using multiple models for different dynamical cases, and even then, they could not address significant multiplicative parametric faults.

In many respects, the E-Gas concept [25] used in the automotive industry is a version of this, using a more simply-programmable rule-based model of the functional software, and comparing its output with the measured output of the functional software itself. It is less equations-based than parity relations, owing to it being a software application with developer-defined functions as opposed to purely relying on physical equations.

2-3.1.2 – Observers and Filters

In this approach, observers are designed by reconstructing the system outputs from the sensor measurements, and then using the estimation error to develop observers and innovation to develop the Kalman filters [74]. Referring to the parity relations method, Frank [70] explains that a closed-loop parity space approach can lead to the concept of state estimation. Observers are a type of state estimator that are popular in this area of control theory [82], particularly in chemical engineering, industrial engineering [74], aerospace [79, 82, 85, 86], and has grown in popularity in the automotive industry [75, 87, 88].

Patton [79] examines the use of a 'state estimation filter' (or observer), a fault detection method whereby outputs are estimated based on one or more accepted sensor signals. This is used to isolate both the occurrence and the source of the fault, with the aim to better understand what has caused the fault [79]. Depending on the type of analytical redundancy in place, different forms of estimation filters are available for use. If the sensor measurements are related by algebraic equations (called static analytical redundancy), a classical least-squares method is used. If they are linked by differential equations (dynamic analytical redundancy), and provided they are deterministic the estimation filter used is a Luenberger Observer - also called a simple blender - which is a dynamical combination of the measurements. The Luenberger Observer is typically less time consuming than other estimator techniques, but since it doesn't take into account parameter variations and system disturbances, it is has some drawbacks in particularly noisy plants [75]. Finally, in the general stochastic case where the sensor patterns can be analysed statistically but not necessarily linked and predicted mathematically, a Kalman Filter is used, designed in respect to the measurement noises and disturbance characteristics [89]. Additionally, robustness is improved with an 'unknown input observer', where the fault detection filter is decoupled from unknown inputs, such that the fault detection system becomes insensitive to unmodelled system disturbances whilst having specified sensitivity to sensor and actuator faults.

Kalman filters are of particular interest to this topic, as their strength lies in the fact that they are designed for the stochastic case where noise factors need to be accounted for [70]. The

Kalman filter is a recursive algorithm that aims to minimize estimation error. Robustness to other noise factors can be improved by using a bank of Kalman filters that consider all the available possible system models under all possible dynamical changes [90]. Each Kalman filter or state estimator in the bank is tuned to be sensitive to a different possible fault hypothesis, and then in turn each hypothesis' probability is tested using some decision function such as Bayesian decision theory [91]. One possible setup for a bank of observers is to assign one observer per sensor or actuator, in what is called a dedicated observer scheme, tuned to detect faults in the sensors or actuators [92]. Figure 2-7 shows a bank of Kalman filters used with likelihood functions as part of a multiple hypothesis test:

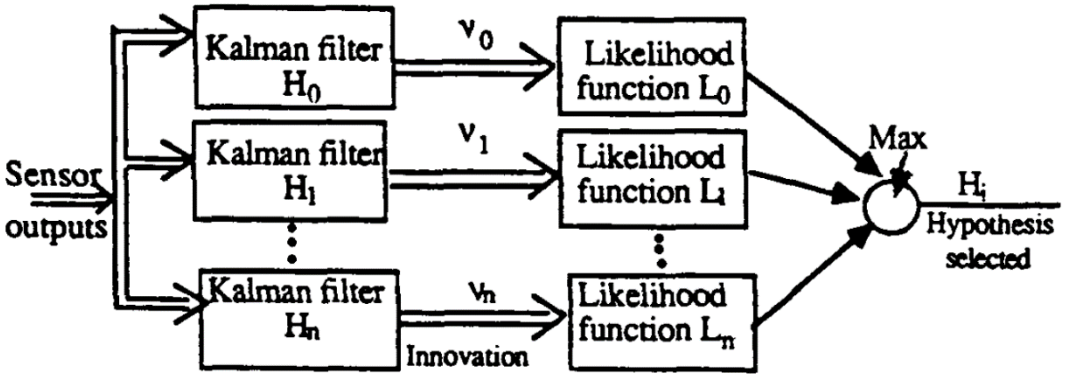

Figure 2-7: Bank of Kalman filters as part of Multiple Hypothesis Test [79].

Kalman filters are excellent at estimating linear states in a system, but have some trouble with non-linearity. Dynamic systems are usually very non-linear, so an Extended Kalman Filter (EKF) is commonly used in practice. The EKF is a modification of the normal Kalman filter in that a non-linear model is linearized at each process iteration by calculating the Jacobian, and subsequently ignoring higher order approximation terms [87]. Patra [85] used an EKF as a state estimator for an attitude controller in a missile. It was found that the EKF was effective in stabilizing the angle of attack by effectively estimating the state of the system. In Zhang & Jiang's paper [82], a two-stage adaptive EKF was used as part of a fault-tolerant control scheme in an aircraft, with fault parameter estimation. The EKF stage 1 first estimates the reduction of control effectiveness, and as such a historical sample size that is smoothed for the fault decision scheme is decided in stage 2. The size of the moving average window to be smoothed is changed so as to best reduce model uncertainties (this is the adaptive aspect of the filter). Compared to a PID controller, the fault-tolerant control actuator fault detection system used in this paper offers both faster and smoother fault reaction, and the ability to reject

any known noise factor. The fast and smooth fault reaction helps to achieve the principle of ‘graceful degradation’.

Ghodbane et al. [86] uses a fault diagnosis method called Extended Multiple Model Adaptive Estimation, which is a technique based on a bank of EKFs. Each EKF is responsible for monitoring the health of an actuator, and assigning a conditional probability to each fault scenario. Consequently, the estimated state vector of the system as a whole is found by summing all EKF state vectors, each of which is weighted by the corresponding probability. In this case, the Extended Multiple Model Adaptive Estimation fault detection process was very effective in isolating the fault to a single actuator, but did take about three seconds for the fault to fully register after it was injected (fault probability value needed to rise), and five seconds to determine the fault was no longer present. In an automotive application, the delay could take longer than the safety window allows, and would need to be tuned for faster fault detection, perhaps at the expense of robustness against false positives. [93] used Kalman filter as part of fault tolerant control in a network distributed control system, whereby the state observer was able to detect a faulty sensor as well as identify and compensate for random network transport delays. While signal transport delays are not likely to be a problem in this application, their approach to handling signal disorder or loss (corruption) may be of interest.

A useful variation of the Kalman filter is the unscented Kalman filter, designed specifically for discrete time cases in non-linear system [87]. It uses the unscented transform to address the problem of propagating Gaussian random variables through non-linear model functions; this eliminates the need for linearization altogether, improving estimation accuracy and lowering computational complexity [87]. Vasu et al. [87] developed an extended mean value engine model that averages the dynamics of an ICE for the purpose of engine control in a vehicle ECU. They used an unscented Kalman filter with the extended mean value engine model and measurement signal post-processing to describe the engine in the model while estimating fewer states of interest, allowing flexibility in the balance between robustly obtaining system information through a noisy signal and computational efficiency. Their estimator method was also shown to be sufficiently modifiable to be sensitive to particular faults. Such an application in an EV or HEV would be useful as being able to isolate and distinguish between ICE or electric motor torque sources, helping the vehicle enact an appropriate fault reaction. Tabbache et al. [75] used an EKF and Luenberger Observer as part of an analytically redundant voter scheme, whereby an EV electric motor speed sensor was supplemented by a ‘virtual sensor’, whose signal was derived from appropriately tuned EKF and adaptive Luenberger Observer using other system inputs. The EKF used stator current and voltage components to reconstruct rotor speed and rotor flux. A voter scheme would then

determine the signal most likely to be true. They found that the Luenberger Observer had better performance at high speed during faulty conditions, whereas the EKF had a higher reliability coefficient at low and medium speed faulty conditions, and the actual speed sensor was naturally most reliable during healthy conditions at all motor speeds. The most reliable signal would then be selected for use in the motor control model based on the conditions.

2-3.2 – Process History Based Methods

2-3.2.1 – Principal Component Analysis

A method commonly used in process engineering is called Principal Component Analysis (PCA), which is sometimes called Proper Orthogonal Decomposition in other engineering disciplines. It is closely related to Factor Analysis, where variables can sometimes be grouped together and represented by a lesser number of variables [94]. Hussein et al. [95] describes PCA as a statistical procedure working by "(decomposing) the variance and covariance structure of a data matrix by defining linear combinations of the columns in the original matrix." Essentially, then, PCA is a data-driven modelling approach that takes a large amount of multivariate data, and seeks to find sets that contain correlated data within them, but such that the sets do not correlate with each other. It focusses on the processes occurring based on the variance and correlation of the outputs it measures. Because it can group data into less sets than there are data points, the lower dimensionality can improve fault detection and diagnosis using outlier statistical methods. The sets can then be used to create predictive models - if captured through a supervised learner environment - for use in other controllers as a knowledge base. Figure 2-8, from Mazzoleni et al., [96] shows how data can be processed into groups using a PCA variant called Principal Direction Divisive Partitioning, which was applied to an aerospace electro-mechanical actuator:

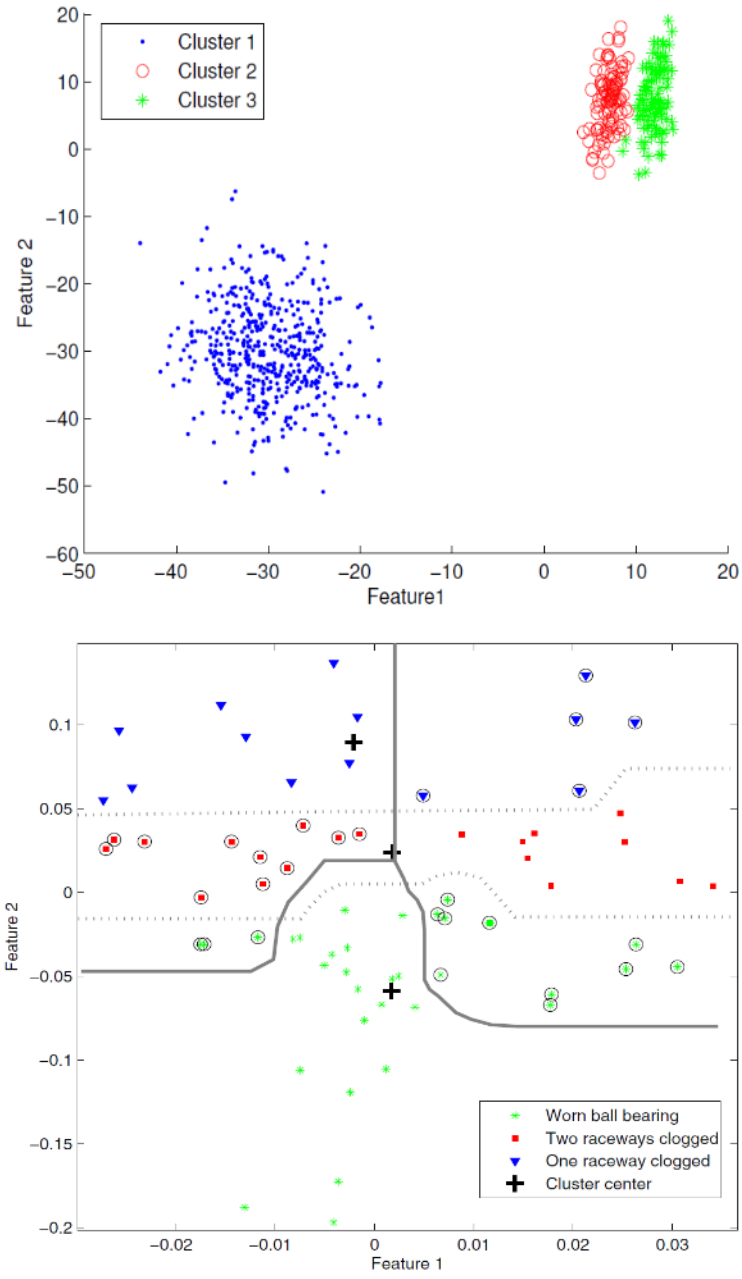

Figure 2-8: Principal Direction Divisive Partitioning results, showing three data clusters in (left), being processed into three failure mode classes (right) [96]

Figure 2-8 shows how data samples between two features can be statistically divided. The two features in this case were arbitrary as they were each a linear combination of 21 variables, including actuator displacement and torque generating current. The solid and dotted lines are possible boundaries for class designation. This method shows a lot of promise, as it both identifies a fault has occurred and the cause of a fault (with *a priori* knowledge). In theory, this could be applied to monitoring torque to an electric motor in a vehicle if the right variables are available.

Hussein et al. [95, 97] used PCA in conjunction with Projection to Latent Structure Discriminant Analysis to treat an electro-pneumatic system as a black box and robustly determine system health information. This is a method that looks to find a line or plane that can differentiate between classes depending on their values. It is based on building classes (visualised as clusters) using a knowledge base from which the model is trained. Hussein et al. found that their setup on a printing system was effective in finding faults such as high throttle and leakage, using signals from pressure, displacement, and vibration sensors. Their approach could be used in multiple automotive powertrains because it can treat the powertrain as a black box. This is because the system does not need to necessarily know what is happening in the system, but simply needs to measure what the outputs are and determine which class or cluster the current performance state is in based on the inputs. The drawback here is that with each measured variable the state becomes increasingly complex. Visualisation of data beyond the third dimension (i.e. more than three measured variables) requires more effort to analyse, but also more processing power to use as an online processor. This will heavily depend on how few classes the outputs can be reduced to, but it is likely that the number of noise factors - and the statistical overlap of recorded data – will make analysis difficult. Additionally, development tests of virtually all possible faults may be needed to be conducted to build the knowledge base necessary for the PCA fault detection; however, since the application in this project is a software function, gathering and recording training data could be automated, and thus performed very quickly at an accelerated rate (i.e. faster than real-time) during development.

Zheng et al. [98] tried to apply PCA to an NPP, but identified that the process varies between "stable – transition – stable" states that aren't as significant in a typical chemical processing plant. Additionally, since some variables in an NPP are sensitive to external parameters such as feed water flow, main steam pressure etc., PCA can't accurately classify the data it records. This is a similar problem faced by the highly dynamic environment and external factors that a vehicle powertrain operates within. Zheng et al. [98] overcame this by developing a Reformative PCA-based Fault Detection Method. The method starts by using K-mean cluster analysis to analyse historical data and obtain datasets for various stable operating conditions. A 'principal component model' group is produced for the whole process using the K-mean cluster datasets. The process state is then determined, with fault detection disabled if the process is in a transitional state. This is done through a 'stability factor'. Finally, the detecting sample's membership to the principal component model group datasets is determined, and a new principal component model is produced for the current stable operating condition. This method was then applied to 26 variables in data taken from a real 600MW NPP, and it was

found that the reformative PCA method can effectively and quickly detect faults on variables that vary with external factors, when the traditional PCA method would not. Figure 2-9 supports their progress [98] with the PCA process running online:

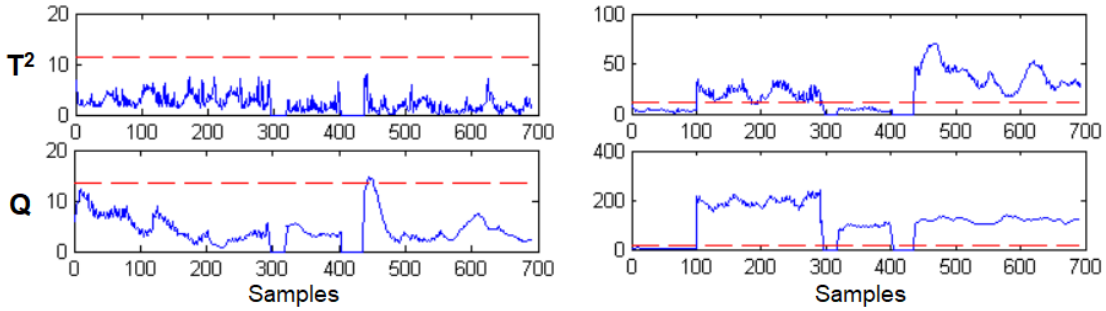

Figure 2-9: Comparison of T^2 and Q in conventional PCA (left) and Reformative PCA (right), showing improvement in fault detection with Reformative PCA method [98].

A drawback to this particular concept by Zheng et al is the need to turn off fault detection during 'transition states', which is a concern for an automotive safety concept. This is simply because it means that fault detection could be switched off during any time where there are external variables affecting powertrain performance, such as a changing road gradient, temperature and weather, and vehicle mass changes. The time in which a fault can become a hazard is very small, and having any lapse in fault detection capability will make the vehicle vulnerable to missing a fault. If this could be overcome with another adaptive controller in conjunction with the reformative PCA method such that it offers full and adequate coverage, then this safety concept could have some merit. Otherwise, another application of PCA could yield similar results regardless of noise factor.

Canonical Correlation Analysis (CCA) is a fusion of statistical process like PCA and expert systems, whereby known statistical correlations are used to set a baseline [99] uses CCA in an industrial processing plant-wide distributed monitoring of variables, where local nodes use independent CCA analyses that are cross checked with others, and can be used to diagnose faulty nodes as well as train new nodes. The system showed improved fault detection rates, improved false positive rates, and greatly improved detection speed. The detection speed is primarily due to reworking with a lower dimensionality at the local level, which allows data to be processed and transferred more quickly between the network. This may be of interest in a distributed control system within an ECU, but may not see as much benefit on a powertrain ECU for powertrain fault monitoring, as bandwidth-limited communication channels are not used for monitoring powertrain software on the same chip. [100] also uses CCA to detect

sensor precision degradation by generating residual signals in sensors when taking into account the canonical correlation into account. The drawback of this particular statistical method is the effort it takes to derive those canonical correlations in the first place, transform them from engineering understanding into the CCA format, and then validate that these relations are correct.

2-3.2.2 – Neural Networks

Neural networks fall into the branch of artificial intelligence. The aim of a neural network is to create relations between inputs and outputs in a system by evolving its internal structure [101]. They can be used to learn and create transfer functions where analytical mathematical equations are difficult to form, working by pattern recognition between input and output vectors [102]. The role of a neural network in the case of fault detection and isolation would be to use training data to develop pattern recognition for what a correctly functioning vehicle would operate as. The training data would be developed offline. In an online learning application, the neural network could possibly adapt to small parameter variations and drifts if the patterns aren't suddenly or significantly different. The same neural network would then be able to recognise if the pattern is suddenly vastly different to normal operation, i.e. deviance from the training data, indicating a fault occurrence. The difficulty would be quickly and correctly diagnosing the source of the fault if prior data (i.e. a pattern for that specific fault) is not available for the operating conditions. Conversely, developing a fully comprehensive training dataset for the neural network could take significant effort, as virtually all driving conditions would need to be tested, and most – if not all – faults be artificially induced.

There are many different types of neural network in use in fault diagnosis and data analysis, but they are typically classified by their learning strategy: supervised and unsupervised. Neural networks with supervised learning strategies have specific learning topology constraints within which they learn [84]. This reduces the task to an estimation of connection weights between data samples, where the connection weights are learned by using the difference between desired and measured parameter values. Cybenko [103] states that this method is useful in classifying arbitrary regions of the learning topology. One of the more widely used supervised learning neural network in chemical engineering is the back-propagation algorithm. Learning performance for neural networks is a key characteristic to try and optimise. To do so, some researchers have performed feature extraction by pre-filtering and data processing the input data [104]. Their results showed greatly reduced learning time for the neural network, and improved clarity in recognising patterns for fault detection. Moussavi et al. [105] used an Adaptive Neuro-Fuzzy Inference System to intelligently control a permanent magnet DC

motor by using an adjustment mechanism with an adaptive controller (something called Model Reference Adaptive Control) where together they found that a hybrid learning technique would improve reaction speed to change in motor dynamics, and maintain control.

Unsupervised learning strategies, also known as self-organising neural networks, restructure their framework based on the inputs it receives, as it uses unsupervised estimation techniques. The objective is pattern recognition, whereby similar patterns are grouped together to form a class. If there is a pattern far enough away from all current classes a new class is created [105, 106]. In this method, therefore, the threshold within which patterns are grouped together is a key calibration term, leading to the clustering technique, which aims to group data samples such that there is the greatest amount of separation between classes; with an objective defined, this essentially becomes an optimisation problem. K-means clustering is a commonly used self-organising analysis technique [107]. It assumes a set number of classes exist, and distributes the data such that each one has at least one data sample is unique to it. It then looks to self-organise to optimise separation between the classes. Puskorius & Feldkamp [108] showed that using a Dynamic Decoupled EKF can greatly improve learning efficiency in neural networks in a highly nonlinear application.

Li [109] used an ANN as a vehicle state estimator for autonomous driving. They improved on the usual practice of relying on curve theory and the Serret-Frenet equation of vehicle kinematics modelling, instead using a Kalman filter to estimate lateral vehicle motion states. The tuning of the Kalman filter itself was done using an ANN, which combines with the Kalman filter to form an ANN observer. While generally favourable results were presented in the paper, there were significant vehicle motions that were missed or damped out by the method, as well as significant deviations from the true values. These deviations are concerning, as it needs to be verified that their characteristics will never result in a missed fault in safety-critical software - something that does not seem feasible in the present time. [110] used artificial neural networks to learn the characteristics of supervised autonomous vehicle behaviour, which was then used to detect deviations from the intended path of the autonomous system, due to system faults. They were able to suggest a form of verification framework, but the sheer scale of the task that high ASIL safety-critical software requires for verification, and reliance on other technologies such as vehicle-to-grid monitoring that could be used to aid in reliability.

Horton [102] used a simple artificial neural network (ANN) for the purpose of parameter identification in mapping aerodynamics in a tactical ground-to-air missile, essentially acting as a state estimator. He compares ANNs with Kalman filter estimators. The Kalman filter can

also ‘learn’ and be ‘taught’, but uses *a priori* knowledge of the system structure along with initial state estimates in order to better estimate the plant state. The Kalman filter then begins to adapt to the system as it is used online. The effectiveness is determined by the basic algorithm capabilities, and the quality of the *a priori* knowledge it is based on. An ANN, by contrast is trained offline first, over a range of operating conditions. It is more tolerant to system structural changes, whereas a Kalman filter may struggle if not properly programmed. An ANN also does not need initial parameter estimates, and does not necessarily require any additional online training regime to continue performing state estimates. The quality of an ANNs state estimates are determined by the capabilities of the chosen network learning architecture, and the size of its knowledge base. Horton found limitations in the neural network performance under new and uncertain operating environments (conditions that it had not trained for). He argued that these could be overcome by further online training. However, he combined the Kalman filter and the ANN into a hybrid structure, where the ANN acts as a basis for the self-adaptive linear Kalman filter estimator. The hybrid state-estimator was very effective, outperforming both counterparts respectively.

2-3.3 – Concept Candidate Selection

Now that the literature review has been completed, the concepts that have been discussed at length in the previous section will form the initial group of concept candidates. In addition to these, the benchmark continuous torque demand monitor will be revisited. As stated previously, a simplified safety software model would result in more noise factors being left unaccounted for, leading to more torque error.

In summary, the preliminary concept candidate shortlist is as follows:

1) The Adaptive Safety Monitor.
2) The Kalman Filter.
3) Principal Component Analysis.
4) Neural Networks.

2-4 – Conclusions

Many fault detection methods are used throughout both the automotive industry and beyond. This literature review has identified some of the methods used in these industries, including the benchmark E-Gas three-level concept widely used in the automotive industry, and which

future concept candidates will be tested against in this investigation. Four preliminary concept candidates have been identified for investigation, two quantitative model-based concepts: the adaptive safety monitor and the Kalman filter, and two quantitative process-history-based concepts: principal component analysis and neural networks. The next chapter will outline the context of the ISO 26262 lifecycle in concept phase to establish key terminology, and understand the structure within which these concepts will need to be developed in (Chapter 3). The subsequent chapter will identify ideal concept attributes to select the concepts that show the most promise before in-depth concept investigations can begin (Chapter 4 and 5).

Chapter 3

Safety Requirements, Ideal Monitoring Attributes, and Test Environment

3 – Summary

The first part of this chapter provides the purpose for the novel safety concepts being developed, through discussing how risk is mitigated using the ISO 26262 engineering framework through the derivation of functional safety requirements. The second part of this chapter derives additional ideal monitoring attributes. These are identified based on ISO 26262, ISO 25010, and OEM expert opinion, and are used to select the most promising concept candidates for detailed investigation via a Pugh Matrix. Finally, the testing methodology will be established for detailed concept development and investigation in the following chapters.

- **Objective 2:** Identify ideal concept attributes and score candidates against them.
- **Objective 3:** Establish a testing method for testing performance of the concepts.

3-1 – Obtaining Safety Requirements for Concept Development

Functional safety is a vehicle-level attribute, whereby the vehicle is shown to be free from unreasonable risk. For that unreasonable risk to be mitigated, it must be identified and quantified at the vehicle level, such that top level safety requirements can be defined to mitigate the risk appropriately. A portion of these safety requirements are then implemented by the safety concept. Therefore, to design an appropriate safety concept, the safety-criticality of that safety concept must first be understood in light of the risk it is supposed to mitigate. This section explores the risk relating to one potential vehicle-level hazard, such that appropriate safety requirements are defined for the safety concept to implement. ISO 26262 provides an engineering framework in Part 3 to identify sources of risk in the powertrain, quantify what risk is unreasonable, and provide a subsequent process for developing safety concepts.

3-1.1 – ISO 26262 Engineering Framework

A familiar V-diagram is at the core of the ISO 26262 engineering framework, emphasised by the fact that it appears in the opening pages of each part of the standard; this can be found in the Appendix [27], with a condensed diagram shown in Figure 3-1:

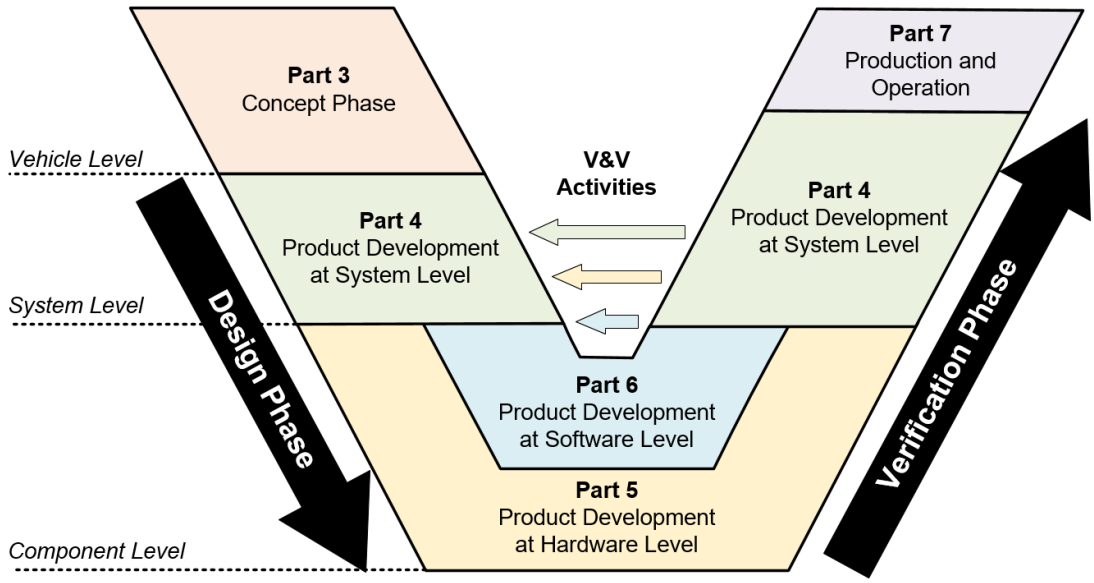

Figure 3-1: Simplified ISO 26262 Process V-Diagram

The standard is conveniently separated into ten parts, but the technical aspects of the safety lifecycle takes place within Parts 3 to 7, with the remaining parts consisting of helpful documentation and supporting management processes. On the left-hand side of the V-diagram – the design phase - the parts roughly contain the following processes:

- **Part 3 – Concept Phase [27]:** Item Definition, Hazard Analysis and Risk Assessment, creation of Safety Goals, conception of Functional Safety Concepts (FSC) and their Functional Safety Requirements to meet the Safety Goals.
- **Part 4 – Product Development at System Level [29]:** specifying a Technical Safety Concept (TSC) and Technical Safety Requirements (TSRs) of the specific system in the vehicle, such that the FSC is effectively implemented, system is designed, and system-level verification and validation activities can be undertaken.
- **Part 5 and 6 – Product Development at Hardware and Software Level [30, 31]:** component-level development to implement TSCs and TSRs, hardware / software integration, and component-level verification activities.
- **Part 7 – Production and Operation [32]:** handling the commissioning of the item in the vehicle.

Each of these parts have relevant validation and verification (V&V) activities on the right-hand side (upward path) of the V-diagram that verify the design phase of each product development level, with some preliminary V&V activities also included in the design phase itself. Consideration for these V&V activities will be made in Chapter 6 of this project, but the focus will remain primarily in concept phase and system design. Full component-level development (Parts 5 and 6) are out of scope for the most part, but again some of the considerations will be touched upon throughout development in this project.

To understand the context within which this project seeks to operate, it is important to understand the flow of requirements in order to achieve functional safety of an item at the vehicle level, and how they drive the need for functional safety concepts in powertrain software; this process begins in Part 3, the Concept Phase. Figure 3-2 shows the overview of this summary and the primary work products as the lifecycle travels down the left-hand side of the V-diagram, starting in the Concept Phase, with the corresponding ISO 26262 Part and Clause shown in the chevrons.

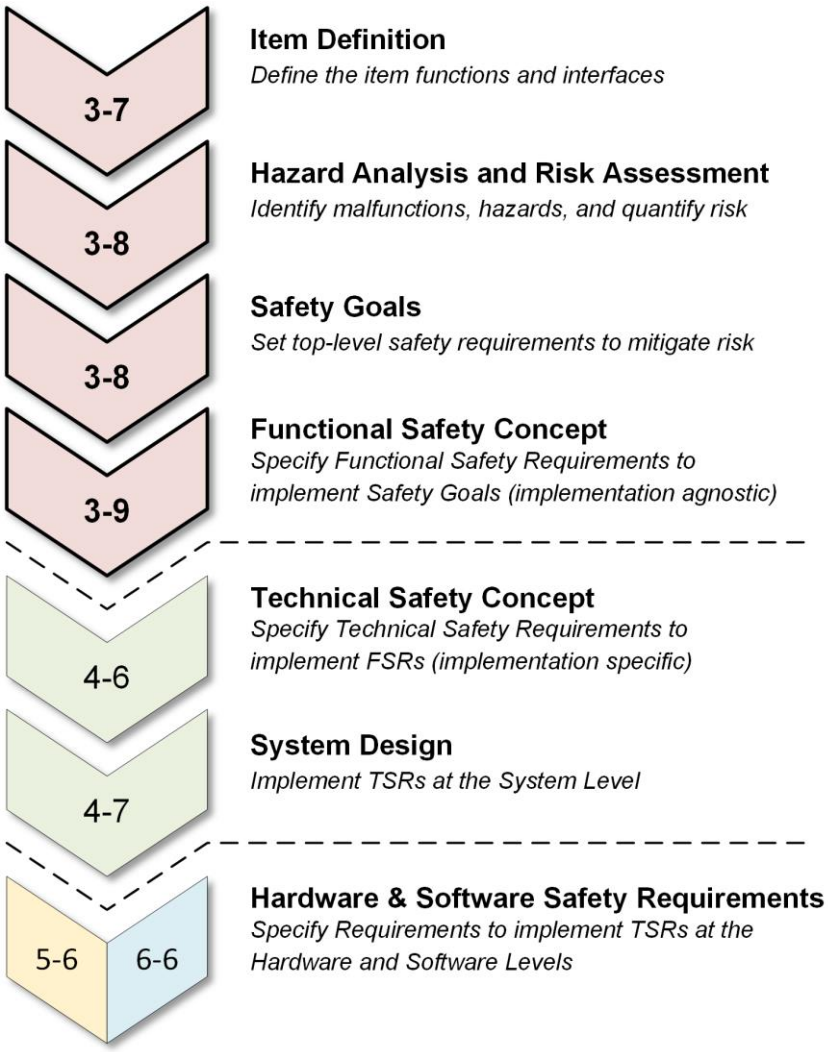

Figure 3-2: Hierarchy of safety requirements [27].

Developing a system through a full safety lifecycle is a significant task and a full safety report is out of scope for a thesis, however, a focused demonstration through the Concept Phase process will be undertaken in this chapter in order to arrive at the starting point for developing novel function safety monitoring concepts.

3-1.2 – Item Definition

The starting point of the Concept phase is the Item Definition. An 'item' in ISO 26262 is the electronic/electrical (E/E) "system or array of systems that implement a function at the vehicle level" (ISO 26262 Part 1-1.69 [26]). The aim of defining the item is to establish its purpose, functionality, and interaction with other vehicle systems: its operational boundary. The E/E item focused on in this thesis is the powertrain system. The functional safety boundary diagram for this item is shown in Figure 3-3.

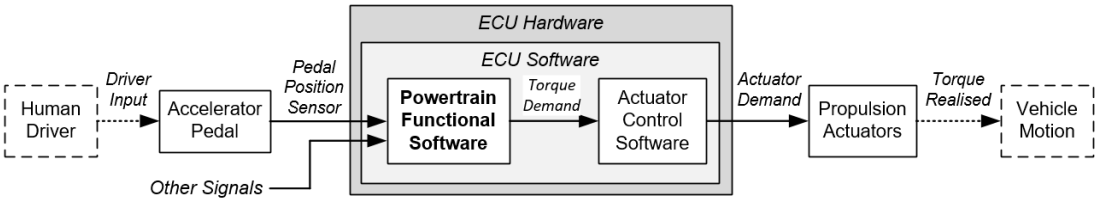

Figure 3-3: Powertrain System Boundary Diagram at the vehicle level.

A primary function of the powertrain system is that of achieving vehicle acceleration. Using the boundary diagram of Figure 3-3, the functions delivered by this item is listed and decomposed according to the system elements. For the sake of brevity, only the function of vehicle acceleration will be examined.

Table 3-1: Decomposed acceleration function and associated elements of the powertrain.

Function	System Elements	Function (decomposed)
Provide vehicle acceleration commensurate with driver demand	Accelerator Pedal	Transmit measured position to Powertrain ECU
	Powertrain ECU	Receive inputs from accelerator pedal, drive mode selector, and other vehicle sensors / systems.
		Calculate desired vehicle direction of travel
		Calculate desired vehicle total torque demand
		Calculate actuator demand for each actuator
		Transmit actuator demand to propulsion actuators
	Propulsion Actuators	Receive actuator demand from Powertrain ECU
		Realise torque according to demand

The vehicle acceleration function in Table 3-1 can be summarised as using a sensor to determine a desired vehicle acceleration, which is delivered by the propulsion actuator(s). The powertrain ECU determines this desired vehicle acceleration by transforming the desired acceleration into a vehicle torque demand via a complex process within the powertrain functional software (hereafter "functional software"). The torque demand is then allocated to the various actuators, which are then controlled via software drivers in the actuator control software, before individual actuator demands are sent to the propulsion actuators. This is a very simplified contextual overview of the system commonly used in the industry (other topologies exist) but is sufficient to describe the safety critical E/E item in its immediate context.

The powertrain system can be regarded as one of the most complex safety critical E/E items within the vehicle, being operational at all times the vehicle is turned on, and active during each and every environmental situation the vehicle is in. It is therefore critical that the safe operation and the mitigation of unreasonable risk is achieved at all times. It must be ensured correct torque demand is delivered in relation to driver pedal demand, as a malfunctioning torque demand could result in unexpected vehicle behaviour, potentially leading to a hazardous situation.

3-1.3 – Hazard Analysis and Risk Assessment

The next step is to conduct a hazard analysis and risk assessment (hereafter, HARA). The goal of the HARA is to identify and categorise the hazardous events that could arise in the event of a malfunction and assign a risk to these hazards. Once these have been identified, safety goals are defined that ensure the prevention and mitigation of identified unreasonable risk in the event of one of these malfunctions occurring in the item. The safety goal includes what is called a 'safe state', a tangible and attainable metric that defines safety for this item with respect to the hazard, typically to be attained within an allotted time once a fault is detected (the 'fault-tolerant time interval', FTTI). These safety goals form the basis for the functional safety concepts, as their purpose is to ensure the safety goals are met.

The Society for Automotive Engineers (SAE) has released a document called J2980 [111], a recommended practice for surface vehicles called 'Considerations for ISO 26262 ASIL Hazard Classification'. This document is useful during the development stage of the safety lifecycle as it provides a more detailed insight into identifying hazards and classifying the risk they pose than ISO 26262 does. J2980 is limited in scope as it focuses on the HARA stage without the prerequisite item definition, a point it stresses in part 1.2 [111]. The document compiles and marries many of the principles found in and recommended by a number of other applicable documents and standards, and reframes them in the context of ISO 26262. An overview of the HARA inputs, processes, and outputs are shown in Figure 3-4.

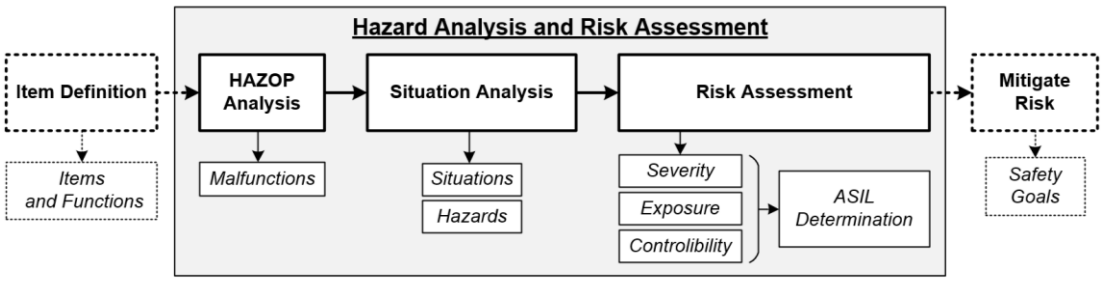

Figure 3-4: Overview of HARA process outlined by J2980 and ISO 26262.

The three main processes of the HARA will be examined more closely in this section, as it provides a traceable context for the reason a functional software malfunction would be considered hazardous, and the level of risk such a malfunction poses.

3-1.3.1 – Hazard and Operability Analysis

A Hazard and Operability (HAZOP) analysis is an explorative analysis method for identifying faults in the various function of an item. It does so by composing sentences with applicable guidewords and phrases to identify the malfunctional behaviours that could lead to hazards to humans at the vehicle level. The procedure is to take a function that the vehicle seeks to provide and apply these guide-phrases to the function to find the resultant malfunction. Consider the following function provided by the powertrain: "provide vehicle acceleration commensurate with driver demand". Applying the HAZOP guide-phrases yields the item malfunctions shown in Table 3-2.

Table 3-2: HAZOP for Acceleration Function

Function vs Guidewords	**Loss of Function when intended**	**Incorrect Function – more**	**Incorrect Function – less**	**Incorrect Function – wrong direction**	**Unintended activation of function**	**Output stuck at a value**
Vehicle Acceleration	*Loss of Acceleration*	*Excessive acceleration*	*Insufficient acceleration*	*Acceleration in reverse direction*	*Unintended acceleration*	*Acceleration stuck at value*

For the purpose of this thesis, the primary malfunction that shall be considered will be "unintended acceleration".

3-1.3.2 – Situational Analysis

In ISO 262626 Part 3-7.4.2 [27], the situations where a certain malfunction could lead to a hazardous behaviour are considered, in that the S, E and C parameters assigned to each hazard are determined when the situation in which the malfunction occurs is taken into account. J2980 provides a useful graphic in Section 4.2.1.4 [111] to aid in considering the possible operational situations of the vehicle, though it does stress that Figure 3-5 is not exhaustive.

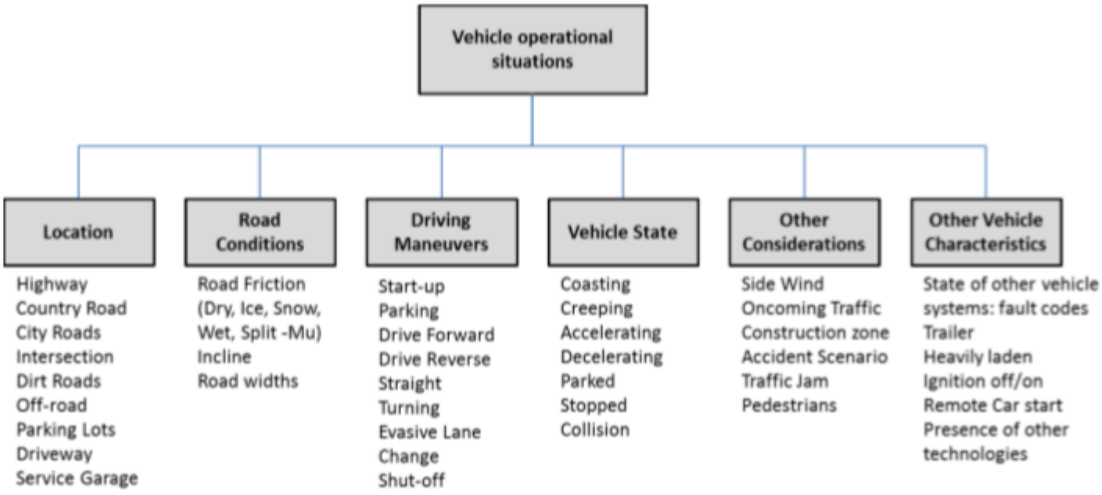

Figure 3-5: Possible Vehicle Operational Situations [111]

The situational analysis has a great impact on each of the different ASIL determination parameters, and should be done carefully within the appropriate operational context the vehicle would be used within. Three situations are considered moving forward:

- Vehicle driving in city or on country roads behind another car
- Vehicle driving at low speed (first gear) in pedestrianised area
- Vehicle is cruising on motorway with moderate-heavy traffic

These will be considered alongside “unintended acceleration” to form the hazardous situations in the hazard analysis part of the HARA in Table 3-7.

3-1.3.3 – Risk Assessment

With the functions and hazards identified, the risk assessment can now take place to determine the risk each hazard poses to human life and property. In line with recommendation from IEC 61508 [53] of assigning a Safety Integrity Level (SIL) to evaluate and categorise the risk of each hazard, ISO 26262 uses the Automotive Safety Integrity Level (ASIL) to determine the level of effort required to mitigate the assessed level of risk each hazard poses. Whereas SIL categories are assessed on the severity (level of potential injury, S) and exposure (probability of occurrence, E), an additional parameter – control, C – is included in determining the ASIL, by assessing the amount of control the driver has in the hazardous situation. While ISO 26262 does provide some guidelines in determining the appropriate S, E and C parameter values, J2980 gives a deeper insight in how these parameters would be justifiably evaluated.

3-1.3.3.1 – Exposure

Exposure is the measure of how often it is expected that a vehicle would be in the situation being assessed. The exposure class in ISO 26262 comprises of five categories, ranging from E0 to E4, shown below with the relevant descriptions in Table 3-3 below.

Table 3-3: Exposure class parameter values with quantitative descriptions in terms of frequency and duration from ISO 26262 Part 3 Annex B.

	Exposure Class				
	E0	**E1**	**E2**	**E3**	**E4**
Description	Incredible	Very low probability	Low probability	Medium probability	High probability
Frequency Method	Never expected	Occurs less often than once a year for great majority of drivers	Occurs a few times a year for great majority of drivers	Occurs once a month or more often for an average driver	Occurs during almost every drive on average
Duration Method	Never Expected	Not Specified	<1% of average operating time	1% to 10% of average operating time	>10% of average operating time

Historical data and evidence should be used to determine a justifiable exposure parameter. Two methods of assessing exposure are presented in ISO 26262 – frequency and duration – which need to be used with common sense applied in order to ascertain the correct value. For

example, crossing a junction only accounts for perhaps 5% of average operating time (E3), but occurs multiple times every drive cycle (E4). The frequency method will be chosen for the risk assessment.

3-1.3.3.2 – Severity

The next step in the risk assessment is to consider how severe the injuries would be from a particular hazardous event. The severity class, S, is given four possible parameters, show in Table 3-4 with their corresponding qualifying descriptions from ISO 26262 [27].

Table 3-4: Severity Class with correlation to planar collision speeds, from various analyses of global accident databases [111]

	Severity Class			
	S0	**S1**	**S2**	**S3**
Description	No Injuries	Light and moderate injuries	Severe and life-threatening injuries (survival probable)	Life-threatening injuries (survival uncertain), fatal injuries
Frontal collision delta *v*	< 4 kph	4 – 20 kph	20 – 40 kph	> 40 kph
Rear collision delta *v*	< 4 kph	4 – 20 kph	20 – 40 kph	> 40 kph
Side collision delta *v*	< 2 kph	2 – 10 kph	10 – 20 kph	> 20 kph

The descriptions provided in Table 3-4 are general guidelines as the actual injury experienced by occupants is virtually non-deterministic due to the many factors involved, and the inherent chaotic nature of an accident. Within each accident situation, the severity is a distribution from minimal risk to human life to human fatality. The shape of that distribution curve and the centre of that distribution is what drives the severity classification. The primary factor affecting the shape of that distribution is the difference in speed at the point of collision, commonly referred to as "delta *v*", with higher delta *v* values increasing the severity of injury. SAE J2980 provides a useful table for various planar vehicle collisions, and the minimum/maximum delta *v* ranges found to correlate with the severity classifications after analysing a number of global accident databases; the ranges chosen for the risk assessment are shown at the bottom of Table 3-4.

3-1.3.3.3 – Controllability

The third classification in the risk assessment is that of controllability by participants involved in the hazardous situation to avoid harm, which includes the host vehicle driver, other road users, and pedestrians. This is done either by achieving a safe state themselves (e.g. applying the brakes to slow car down), or by being able to perform an evasive manoeuvre (e.g. controlling steering to avoid hazards, moving out of the way of a malfunctioning vehicle). Table 3-5 correlates the four controllability classifications with the real-world possibility of harm avoidance by the collision participants.

Table 3-5: Controllability Class with quantification

Controllability Class	Description	Quantification
C0	Controllable in general	If dedicated regulations exist for a particular hazard, Controllability may be rated C0 when it is consistent with the corresponding existing experience concerning sufficient Controllability.
C1	Simply controllable	**≥ 99%** of all drivers or other traffic participants are usually able to avoid specified harm
C2	Normally controllable	**≥ 90%** of all drivers or other traffic participants are usually able to avoid specified harm
C3	Difficult to control or uncontrollable	**< 90%** of all drivers or other traffic participants are usually able to avoid specified harm

Controllability is largely dependent on the human factor in a collision, which can include how easily the hazard is identified, and an appropriate mitigating action taken. Some hazards have an intuitive reaction for the majority of drivers. For example, unintended vehicle acceleration can be intuitively counteracted by depressing the brake pedal [111]. ISO 26262 Part 3-8.4.2.6 [27] considers how a driver reaction may be a key part of achieving the safety goal, and if so, which actions and controls are needed by the driver to ensure that controllability is achieved.

3-1.3.3.4 – ASIL Determination and Completing the HARA

Combining the E, S, and C parameters, the ASIL rating for each hazardous situation can be determined in the risk assessment. The ASIL is used to determine the risk each hazardous situation carries with respect to human harm, and once derived in the risk assessment, is carried

throughout the safety development lifecycle, from Safety Goal to Software Safety Requirement. It also specifies the level of development rigour required to ensure confidence that risk has been mitigated commensurate with the level of that risk. Table 3-6 shows how the ASIL ratings are calculated for each hazardous situation.

Table 3-6: ASIL Rating based on E, S and C classes [27].

Severity Class	Probability Class	Controllability Class		
		C1	C2	C3
S1	E1	QM	QM	QM
	E2	QM	QM	QM
	E3	QM	QM	A
	E4	QM	A	B
S2	E1	QM	QM	QM
	E2	QM	QM	A
	E3	QM	A	B
	E4	A	B	C
S3	E1	QM	QM	A
	E2	QM	A	B
	E3	A	B	C
	E4	B	C	D

Note that S0, E0 or C0 have an ASIL QM rating assigned to them, as each one indicates an acceptable level of risk is present. Note also that only the most likely (E4), most severe (S3), and least controllable (C3) classifications lead to the most stringent ASIL rating of ASIL D.

Compiling the malfunctions, the hazardous situations, the risk assessment into one table for analysis completes the HARA. The HARA is an iterative task that can lead to reassessment as new information is discovered and new hazardous situations considered. Assessing all possible malfunctions and situations is a significant task, so an example HARA table for the malfunction focused on in this thesis – 'unintended acceleration' – is presented in Table 3-7. Here, three driving scenarios are examined, one at low speed, one at moderate speed, and one at high speed, with situations 1 and 2 being adapted from J2980 Table D-3 [111].

Table 3-7: Example HARA for unintended acceleration under three driving scenarios

	ID		Situation 1	Situation 2	Situation 3
HAZARD ANALYSIS	**Function**		Provide acceleration commensurate with driver demand	Provide acceleration commensurate with driver demand	Provide acceleration commensurate with driver demand
	Malfunction		Unintended acceleration	Unintended acceleration	Unintended acceleration
	Situation Description		Vehicle driving in city or on country roads behind another car	Vehicle driving at low speed (first gear) in pedestrianised area	Vehicle is cruising on motorway with moderate-heavy traffic
	Potential Vehicle Hazard		Front/rear collision with the vehicle in front	Frontal collision with pedestrian at low speed, no run over assumed.	Front/rear collision with the vehicle in front
RISK ASSESSMENT	**Automotive Safety Integrity Level (ASIL)**	**E**	E4	E3	E4
		Reason	Very common situation. >10% of operation time	Large proportion of drive cycles take place in pedestrianised area (parking lots, pedestrian crossings etc), likely <10% of operation time, but a smaller subset of areas could lead to run-over.	Very common situation >10% of operating time.
		S	S2	S2	S3
		Reason	Frontal crash of vehicle into rear end of another vehicle at intermediate speed (delta v = 20 kph)	Low collision speed as initial speed is low, and pedestrian is regarded as being close to vehicle.	High speeds increases possibility of higher front/rear delta v (> 40 kph)
		C	C2	C3	C1
		Reason	Situation can be controlled by brake actuation, intuitive driver reaction. Sufficient reaction time in majority of situations	In low gear, acceleration for torque will be great, startling drivers with the unintended acceleration. Proximity to pedestrians lowers reaction time such that <90% of drivers are able to avoid	(similar to Sit. 1) Since at motorway speeds, vehicle is in higher gear. This lessens wheel torque and resulting vehicle acceleration, which would be less startling to driver.
		ASIL	ASIL B	ASIL B	ASIL B

3-1.4 – Safety Goals

With the HARA completed, a safety goal at the vehicle level can be defined for each hazard (ISO 26262 Part 3-7.4.4.3 [27]). The safety goal is a "vehicle-level requirement" (ISO 26262 Part 1-1.108 [26]) that – when achieved – prevents or mitigates any unreasonable risk identified in the relevant hazardous events in the HARA. A safety goal must therefore be quantified, meaning that there must be some quantitative requirement that can be met such that it can be shown that safety has been achieved. A single safety goal can be applied to multiple hazards, inheriting the ASIL of the highest hazard relying upon it.

3-1.4.1 – Choosing the Safety Goal

In the context of ISO 26262, and specifically pertaining to the risk assessment, the safety goal addresses the risk presented in a hazard such that by achieving the safe state, one or more of the ASIL parameters (E, S or C) is reduced. The aim is by arguing that if the vehicle were transitioned to the defined safe state when a malfunction occurred, the risk assessment would produce a new ASIL for the particular hazard. If that new ASIL rating is QM, the residual risk is now small enough to be reasonable with no further risk reduction is necessary: safety has been achieved. The safe state has to be reached before a fault can become a hazard: this is the FTTI, and is defined in ISO 26262 Part 1-1.45 [26].

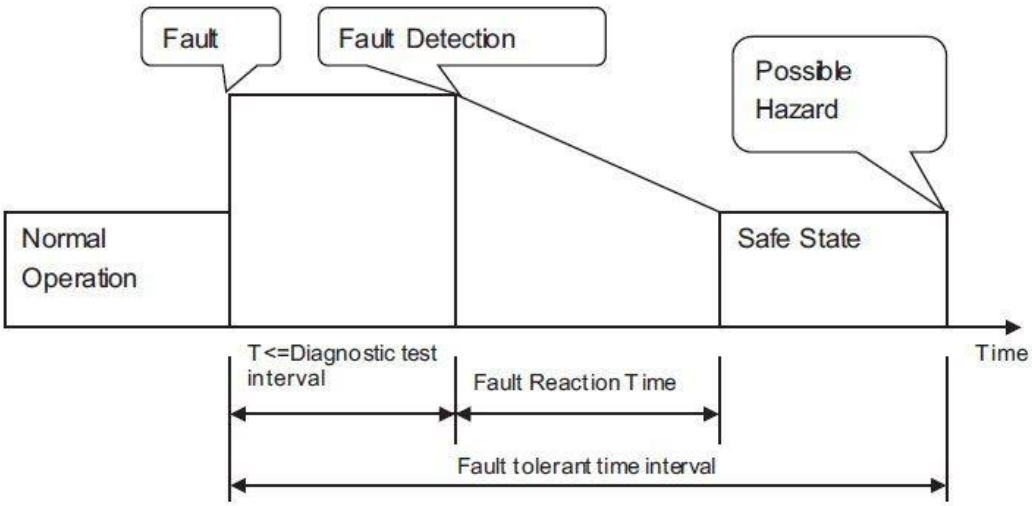

Figure 3-6: Fault Tolerant Time Interval [26]

One common approach in industry is by selecting the safety goal such that the driver is able to regain and maintain control of the situation. For example, taking any of the situations in the malfunction of 'excess torque demand' leads to the hazard 'unintended acceleration'.

Controllability in each of the situations in Table 3-7 is not C0, but if the safe state was to ensure unintended acceleration remained within a manageable (i.e. controllable) magnitude, the controllability parameter could then be reduced to C0, and consequentially the ASIL would reduce to QM. The threshold would need to be manageable in such a way that the driver would easily be able to react to, and compensate for, the hazard, thereby mitigating the hazard.

Birch et al [112] identified the hazard of 'unintended acceleration at low speed among pedestrians' in their paper about safety cases and the functional safety assessment in ISO 26262. In an industrial case study, the hazard could be caused by the torque request and delivery within an electric vehicle, just as in this chapter. They prescribed the accompanying safety goal as 'Vehicle positive longitudinal acceleration shall not exceed driver demand by over 1.5m/s^2 for longer than 1 s', with the rationale being that such a situation is controllable in general (C0) by the driver slowing and stopping the vehicle with application of the brake pedal. The FTTI for this safety goal is 1 second [112]. This is the safety goal chosen to mitigate unintended acceleration in all three situations of the HARA in Table 3-7, inheriting ASIL B.

3-1.5 – Functional Safety Concepts

In order to implement the safety goal, a method and mechanism concept is required, called a functional safety concept (FSC). The FSC is defined in ISO 26262 Part 1-1.52 [26] as "the specification of the functional safety requirements (1.53), with associated information, their allocation (1.1) to architectural elements (1.32), and their interaction necessary to achieve the safety goals (1.108)". The FSC is where functional safety requirements (FSRs) are defined to functionally implement the safety goals, with at least one FSR for each safety goal. They are then allocated to the available architectural elements within the item. A FSR should be technical-implementation agnostic, meaning it should only be defined in terms of functionality. These FSRs will inherit the highest ASIL rating of the safety goals it is assigned to, which in this case is ASIL B.

3-1.5.1 – Defining the Functional Safety Requirements

E-Gas will serve as a strong basis for developing an FSC for the safety goal defined in Section 3-1.4.1, especially seeing as this monitoring concept is considered industry standard [25]. E-Gas Version 5.5 is specifically aimed at addressing the hazard of unintended acceleration in control units (which, according to their own HARA, yielded an ASIL-B rating), leading to its own safety goal of 'prevent unintended acceleration'. To meet this safety goal, the FSRs in E-Gas are allocated to sensors, actuators, and the ECU. The ECU is tasked with all of the

functional monitoring activities in Level 2. Table 3-8 shows each of the FSRs that are allocated to the ECU specifically, and subsequently the monitoring software; a summary is also provided to aid understanding.

Table 3-8: Summary example of ECU-allocated FSRs for E-Gas FSC

Safety Goal	*Vehicle positive longitudinal acceleration shall not exceed driver demand by over 1.5m/s^2 for longer than 1 s*	*ASIL B*

	Task Summary	**E-Gas ECU Functional Safety Requirement**	**ASIL**
FSR 1	Secure input signals	The engine control unit detects sensor faults (e.g. accelerator pedal, throttle-valve, brake, cruise control), additional torque relevant sensors / components) by using appropriate plausibility checks.	B
FSR 2	Calculate actual torque demand	Torque requests of other control devices that are transmitted by networks shall be checked for plausibility (e.g. accelerator pedal, throttle-valve, brake, [cruise control]) in the engine controller.	B
FSR 3	Calculate permissible torque request	In the powertrain control unit a safety concept shall be implemented for validation and confirmation of a not permissible exceeding driving torque or an unintended acceleration.	B
FSR 4	Switch to safe state if a fault is detected	In case of a fault the engine control unit shall switch to a safe state.	B

The derivation and definition of the FSC is the final step in the Concept Phase of ISO 26262 Part 3 [27]. Entering into Part 4 considers the specific system that the FSC is applied to, generating a system-specific Technical Safety Concept and Technical Safety Requirements; the following thesis chapters of concept development occur chiefly at this boundary. The focus of this thesis will be on FSR 3, highlighted in blue in Table 3-8, whereby novel safety monitoring concepts will be explored to ensure the correct torque demand is being provided by the powertrain ECU functional software, fulfilling FSR 3.

3-1.6 – Safety Lifecycle Summary

In the first part of this chapter, the concept phase of ISO 26262 has provided the context and the purpose of the safety concept. In line with Figure 3-2, the following list summarises the outcomes of this section:

- **Function:** Provide acceleration commensurate with driver demand.
- **Hazardous Situation**: Unintended acceleration near pedestrians.
- **Safety Goal:** Vehicle positive longitudinal acceleration shall not exceed driver demand by over 1.5 m/s^2 for longer than 1 second.
- **Functional Safety Requirement**: In the powertrain control unit a safety concept shall be implemented for validation and confirmation of a not permissible exceeding driving torque or an unintended acceleration.

3-2 – Ideal Safety Concept Attributes

The previous section used the concept phase framework of the ISO 26262 safety lifecycle to define the core task that the safety concept needs to accomplish: its functional safety requirement. In this section, the additional other desirable attributes of the safety concept will be derived and discussed. These attributes are not necessarily required by ISO 26262, but are desirable to the automotive OEM as it may make it easier to meet the ISO 26262 requirements, easier to develop, and/or easier to implement. These attributes will be used to identify the most promising concept candidates from the preliminary shortlist produced in the literature review.

3-2.1 – Primary Motivations

To identify the ideal safety monitor attributes, a general understanding of the motivations that go into selection of a safety monitor need to be highlighted. These main motivations were brought from the functional safety team working at the sponsoring OEM.

3-2.1.1 – Meeting Functional Safety Requirements and Performance Criteria

First, the safety monitor must be able to fulfil the task it is designed to do, namely, be able to always ensure safety goals are met. This is captured by conforming to the requirements of ISO 26262, as the standard provides the framework to develop requirements against which the safety monitor is evaluated. Along with this is the matter of how reliable the system is in

operation, which is to say, its ability to provide availability to the driver and to not unnecessarily limit availability due to a non-hazardous fault.

3-2.1.2 – Development Effort

The second motivation is that of development effort on the part of the OEM. In Chapter 1, this was highlighted as the main motivation for this project. The development effort of the safety monitors have increased as the software complexity of the functional software also increased. Consider two different FSCs/TSCs for the same item and safety goal, each meeting the FSRs/TSRs and yielding the same reliability etc. The first concept has a higher initial design effort than the second, so it would make sense to choose the second concept as this would be less costly for the manufacturer. However, if the first concept can be easily transferred between applications – the net effort required over all applications may be less for the first concept than the second, yielding a lower net cost for the manufacturer. Furthermore, the first concept may simply build on existing software used in industry (i.e. the E-Gas continuous demand monitor [25]), so the actual starting point for initial design may be higher than for the brand new second concept. Many engineering groups may work on the safety software across numerous companies, so a highly complex concept may make calibration and modification difficult. Related to this, and of very high importance, is the effort required to verify the concept as per ISO 26262. A concept would benefit from being easier to verify, as the verification effort is a major part of increasing the development costs of the concept. A new concept that meets the requirements and is reliable, but requires a significantly larger verification effort will ultimately cost more, particularly if any minor modification or calibration requires extensive reverification.

3-2.1.3 – Implementation Effort

Lastly, the consideration of actually implementing the concept on the vehicle and releasing it to the consumer. A key component in this is the hardware cost related to the storage and computational requirements that the concept brings. For example, a concept that requires high performance computing to run large amounts of data, or the storage of many variables, would require more expensive hardware per vehicle. In special, low volume vehicle productions, this may be a sensible trade off if it leads to lowered development effort, but in mass market vehicles, the hardware cost is multiplied many times over, so a small per-vehicle saving in hardware costs may make economic sense.

All of these main motivators can be summarised by minimizing OEM cost as much as possible while still meeting the safety criteria.

3-2.2 – Software Quality and ISO 25010

In this software environment, ISO 26262 does not address many of the attributes associated with the quality of software as this is not the focus of the standard. However, many of the motivations that drive safety criteria are common with developing and evaluating quality software. For example, software that is poorly designed can lead to difficulty in full verification, or could lead to unnecessary resource utilisation, both undesirable attributes in quality software and with respect to a functional safety monitor. The quality of software can arguably be in relation to the low-level programming implementation and coding – which is largely dependent on the skill of the programmer – but can also be inherent to the architecture and framework of the software concept and how it is structured to meet its goals. In addition to qualities and attributes that can be gleaned from ISO 26262, it therefore also makes sense to look at an ISO standard that more directly addresses the quality of the software used in such systems for more possible quality characteristics.

ISO 25010 is an international standard on systems and software engineering [113]. Based on ISO 9126:1991, it forms the second part of a series of standards governing software quality, SQuaRE, which stands for Software Quality Requirements and Evaluation. ISO 25010 focuses specifically on the quality aspect of the software itself, with other parts in the series looking at themes such as management (ISO 25000 [114]), measurement (ISO 25020 [115]), requirements (ISO 25030 [116]), and evaluation (ISO 25040 [117]).

The standard separates the quality of the software into two main sections. First, the “quality in use” is examined, which highlights five key characteristics related to the interaction with the system by a human operator, either an end user or maintenance by a developer. These characteristics are:

- Effectiveness
- Efficiency
- Satisfaction
- Freedom from Risk
- Context Coverage

The application of safety software is different to a computer program used by a consumer, in that there is no direct usage: it is a hidden function from the consumer. Effectiveness in this context is defined in ISO 25010 Section 4.1 as specifically relating to how effective the software is in enabling the user to achieve their goal. Efficiency addresses the resources required to achieve the goal of the user. Satisfaction is with regards to user experience. Freedom from Risk is inherent to the safety software, as that is indeed its primary purpose, so it is not a new attribute. Context coverage addresses the number of different scenarios in which a user can use the software, so would not relate to the ability of the safety software to cover all possible scenarios.

The ‘quality in use’ model is therefore not relevant. Instead, a product quality model is provided by ISO 25010 Section 3.3 that addresses the quality of the software itself, not the usage experience. Eight key characteristics and thirty-one sub-characteristics are identified and shown in Table 3-9; a detailed explanation of each of these characteristics can be found in ISO 25010 Section 4.2.

Table 3-9: Product Quality Model - characteristics and sub-characteristics from ISO 25010 [113]

ISO 25010 Characteristic	IS 25010 Sub-Characteristic
Functional Suitability	Functional Completeness Functional Correctness Functional Appropriateness
Performance Efficiency	Time-behaviour Resource Utilisation Capacity
Compatibility	Co-existence Interoperability
Usability	Appropriateness recognisability Learnability Operability User Error protection User interface aesthetics Accessibility
Reliability	Maturity Availability Fault Tolerance Recoverability
Security	Confidentiality Integrity Non-repudiation Accountability Authenticity
Maintainability	Modularity Reusability Analysability Modifiability Testability
Portability	Adaptability Installability Replaceability

3-2.3 – Ideal Monitoring Attributes

With guidance from ISO 26262 [26], a product quality model from ISO 25010 [113], and industry experience from Jaguar Land Rover, ideal attributes of the functional safety monitor have been identified. Each attribute will be briefly described in the context of a functional

safety monitor, any relationships with other attributes noted, and any associated (sub)characteristics from ISO 25010 stated.

3-2.3.1 – Functional Suitability

The safety concept must comply with the FSRs and TSRs from the ISO 26262 safety lifecycle. It must be able to reliably, quickly and accurately detect true faults that will violate the safety goal every time they occur, including latent and multiple faults as necessary. All true faults must be classified and isolated for correct fault reaction to take place. It is important to quickly be able to start fault reaction within the FTTI such that a "graceful transition to a safe state" [26] can occur, but could also lead to sensitivity from high frequency noise factors, resulting in false positives (affecting reliability).

Related ideal attributes are Reliability and Verifiability. Related ISO 25010 sub-characteristics include Functional Suitability, Functional Appropriateness, Functional Completeness, Functional Correctness (Accuracy), and Time-Behaviour.

3-2.3.2 – Reliability

In this context, reliability is a subset of robustness, ensuring that maximum availability of the vehicles performance is given to the driver. A true fault leads only to the necessary reduction of availability in order to maintain safety. There is a low chance of false positives occurring with high reliability, due to a low sensitivity towards noise factors. Low reliability will unnecessarily limit vehicle performance availability with false positive occurrence. It also considers recoverability, as it should not unnecessarily limit performance after an ECU reset. There becomes a trade-off, whereby the appropriate level of engineering effort should be dedicated to improving reliability. In order to select the appropriate trade-off, the project and safety requirements should be examined. The project requirements will set the minimum level of availability the system should have, and the safety requirements will set the level of safety the system should have. Engineering effort for reliability will be complete once both sets of requirements have been verified to have been met during the verification phase of the project, i.e. limitation of unnecessary vehicle performance availability is acceptably low, and safety has been achieved.

Related ideal attributes are Functional Suitability, Initial Design Effort, and Simplicity. Related ISO 25010 sub-characteristics include Reliability, Availability, Fault-Tolerance, Recoverability, and Maturity.

3-2.3.3 – Verifiability

A safety concept must be verifiable, in the sense that it can be verified to ensure it will always meet the FSRs over the lifetime of the vehicle, proving that the software complies with ISO 26262. If the software cannot be verified to maintain safety objectives at all times, it is not suitable for a safety-critical application. This attribute also covers ease of verifiability, as a concept may be verifiable but could require great effort, leading to increased costs for the OEM and ultimately the consumer. On the other hand, verification could be fairly straight-forward with easily identifiable and testable systematic testing, which in a software environment could be automated, reducing costs. Additionally, should the functional software need to be recalibrated, the amount of reverification required would also impact costs. ISO 26262 Part 4-7.4.3.6 [29] states: 'a decision not to re-use well-trusted design principles should be justified'. This is relevant here as some of the methods being evaluated are novel in this application (i.e. not well-trusted) and thus must be verified to comply with the ISO 26262 standard at all times. With a process-history based concept, any stored data that is used to drive concept operation (such as neural network training data) needs to also be both verifiable and verified.

Related ideal attributes are Functional Suitability, Initial Design Effort, Modifiability, and Simplicity. Related ISO 25010 sub-characteristics include Functional Suitability Compliance, Analysability and Testability.

3-2.3.4 – Compatibility

Compatibility covers how well the concept can operate with other items in the greater vehicle plant, including the functional software and other safety software modules. It also considers if the new concept can simply replace the existing one, or whether additional, novel development work is needed, and if hardware changes are required. Additionally, the safety concept should be able to coexist with sufficient independence from modules that are not safety critical, specifically, ASIL QM level functional software.

Related ideal attributes are Simplicity, Hardware and Computational Requirement, and Initial Design Effort. Related ISO 25010 sub-characteristics include Compatibility, Replaceability, Coexistence, Compatibility Compliance, and Interoperability.

3-2.3.5 – Hardware and Computational Requirement

This attribute is directly related to cost of implementation and packaging needs for when the vehicle is released, including the storage requirements and the computational resources the concept would need to operate. Consideration also needs to be given if a certain concept requires special hardware or storage facilities, e.g. an online-learning concept may need hardware redundancies such that if a memory unit fails, all the captured training data isn't also lost. Other, more novel monitoring concepts may require specific sensors or network connectivity of a high minimum hardware or bandwidth specification in order to meet safety goals, all adding to implementation costs. As previously mentioned, the importance of this attribute is dependent on the economics of the target application: a low-volume vehicle model would put lower emphasis on these per-vehicle costs than a mass market production run would.

Related ideal attributes are Compatibility and Commercial Feasibility. Related ISO 25010 sub-characteristics include Performance Efficiency, Maintainability, Installability, and Resource Utilisation.

3-2.3.6 – Initial Design Effort

The initial design effort required to produce the concept is directly related to the cost involved in the development stage, and reducing the initial design effort will lead to greatly reduced costs. Design effort covers the necessary coding expertise for designing and calibrating the concept for a specific vehicle application. A high initial design effort can make commercial sense if it enables easy transferability of the concept to other applications, meaning the net cost is lower over multiple vehicle programs. Verification effort is related to this attribute as both contribute greatly to the cost of development, with initial design effort on the left-hand side of the 'V' cycle, and verification effort on the right-hand side.

Related ideal attributes are Verifiability, Simplicity, Compatibility, Modifiability and Commercial Feasibility. Related ISO 25010 sub-characteristics include Maintainability.

3-2.3.7 – Simplicity

The concept should be simple, or rather, not unnecessarily complex. This is to avoid failures from unconstrained complexity, and implies modularity, encapsulation within the design, and adequate level of granularity [27]. Highly complex concepts may be more difficult to design, verify, modify, and could necessitate higher hardware specification and computational resources. Additionally, it could be more difficult to trace changes if multiple teams work on the software. Therefore, keeping the concept as simple as it needs to be is usually desirable, unless added complexity could enable easier modification or better resource utilisation.

Related ideal attributes are Initial Design Effort, Verifiability, Modifiability, Hardware and Computational Requirement, and Compatibility.

3-2.3.8 – Modifiability

Suitability for configurable architectural design, and maintainability of the software architectural design is desirable. With the functional software in the process of development, it is desirable to be able to quickly and easily calibrate or modify the concept without great effort. Modifiability is also related to how easily the concept could be transferred to a new application (powertrain or vehicle) with only some calibration work required, potentially reducing the initial design effort required in the new application, and thus the cost involved. The option to modify could be related to having that capability programmed in during initial design, though at the expense of simplicity.

Related ideal attributes are Initial Design Effort, Verifiability, and Simplicity. Related ISO 25010 sub-characteristics include Modifiable, Modularity, Changeability, Stability, Reusability, Transferability, Portability, and Adaptability.

3-2.3.9 – Commercial Feasibility

Feasibility for the design and implementation of software units', the technique is only worth exploring if it could possibly be commercially deployed, provided it meets all other requirements. Ensures that development time, development costs and implementation costs are jointly justifiable within the business case.

This ideal attribute relates to all others.

3-2.3.10 – Evolvable

As the vehicle ages, or changes are made throughout its lifetime, the mathematical model becomes less representative of the actual plant. The ability of the concept to adapt to these changes over time could prove highly beneficial, as it could minimise effort spent on calibrations during the development stage as the functional software is being changed, but potentially even enable the safety concept to maintain reliability over a longer period of time and with new, unaccounted-for noise factors interacting with the system. The biggest concern with an evolvable safety concept is that of verifiability, as now all possible future evolution permutations of the concept would need to be ensured to be functionally suitable. Other concerns relate to how complex the system could become, and the additional hardware and computational requirements to process the evolution of the concept.

Related ideal attributes are Reliability, Verifiability, Initial Design Effort, Hardware and Computational Requirement, Simplicity, Modifiability, and Compatibility. Related ISO 25010 sub-characteristics include Adaptability.

3-2.3.11 – Security

Ideally, the OEM should ensure that the software is secure, such that it cannot be maliciously altered after commission in a way that would cause the safety concept to fail to meet its safety objective, ultimately endangering occupants. Currently, security analysis is not required by ISO 26262, but will likely be introduced in future legislation.

Related ideal attributes are Compatibility, Hardware and Computational Requirement, Initial Design Effort, Simplicity, and Modifiability. Related ISO 25010 sub-characteristics include Security, Confidentiality, Integrity, Non-Repudiation, Accountability, Authenticity.

3-2.4 – Importance Weighting and Pugh Matrix Score

Each of the ideal attributes shown in the previous section are important selection criteria to help the OEM evaluate current and potential safety concepts. However, the importance of each attribute varies according to the needs of the OEM. These could be relative to technical factors, with considerations for the safety goals and associated ASILs the concept seeks to address, the range powertrain variations, available hardware and storage resources available, expected noise factors, software complexity and features etc. In addition to this, management and

commercial factors also play a significant role in impacting attribute importance, such as the vehicle production cycle plans (current and future), the inter-company working groups (and intra-company engineering departments), budget etc.

The manufacturer sponsoring this research project is looking to use the research outputs in a mass-market vehicle application. They are concerned with the cost of having to recalibrate and re-verify safety software every time there is an update in the functional software. Certainly, the concepts must perform the task required (i.e. be functionally suitable), and be able to be verified, to comply with ISO 26262 requirements. They hope that the new concepts are compatible with the current software and hardware, such that minimal changes are necessary and very little to no additional cost is added for new hardware, since the software could find use in thousands of new vehicles. Initial design effort is of lower importance than it being modifiable to allow easy transfer between vehicle powertrains, and straightforward re-calibration. While a simple concept would be ideal, it is of lower importance, along with commercial feasibility, so as not to limit the consideration of possible new concepts for this project. An evolvable concept presents an interesting and challenging avenue of research, but with many questions surrounding verifiability it is of lower priority. Finally, product security is worth mentioning due to possible future legislation, but is not of major focus in this project.

With this in mind, a weighting has been applied to each attribute according to the needs of the OEM funding this research project, shown in Table 3-10.

Table 3-10: Importance weighting of each attribute for this application

Attribute	Weight
Functional Suitability	10
Reliability	7
Verifiability	10
Compatibility	8
Hardware and Computational Requirement	8
Initial Design Effort	6
Simplicity	6
Modifiability	7
Commercial Feasibility	6
Evolvable	3
Security	0

The weightings above are used to evaluate and score the current and candidate concepts using a Pugh Matrix, with an overall Pugh Matrix score for each concept calculated by multiplying the importance weight by the concept score for each attribute, and summing these results together. Each attribute can have a maximum score of 10 and a minimum score of 0.

3-2.5 – Concept Selection

Taking the concepts that were highlighted in the literature review, and knowledge of their strengths and any use in industry, the attribute scores could be estimated to narrow down which concepts have potential for investigation. In doing so, efforts could be directed towards the most promising concepts, and concepts that are clearly unsuitable (i.e. concepts that score less favourably versus the benchmark will lose their candidacy). The benchmark for determining concept suitability is – as identified in the literature review – the continuous torque demand monitor. Table 3-11 shows the initial attribute scores of each concept candidate, and the benchmark, as of the first year of research; the scores of the continuous torque demand monitor benchmark concept were derived with expert input from Jaguar Land Rover functional safety engineers, as they use this concept in practice.

Table 3-11: Initial, pre-investigation attributes scores of concept candidates identified in the literature review scored against the benchmark concept. Total score difference to benchmark is shown in parentheses in the rightmost column.

	Functional Suitability	**Reliability**	**Verifiability**	**Compatibility**	**Hardware and Computational Requirement**	**Initial Design Effort**	**Simplicity**	**Modifiability**	**Commercial Feasibility**	**Evolvable**	**Total** *(and difference versus benchmark)*
Weight	*10*	*7*	*10*	*8*	*8*	*6*	*6*	*7*	*6*	*3*	***710***
Continuous Torque Demand Monitor	**10**	**7**	**10**	**6**	**8**	**5**	**8**	**7**	**7**	**2**	**536**
Observer	4	8	8	2	8	2	3	3	2	3	**328*** *(-208)*
Neural Network	8	6	2	3	6	5	6	3	4	8	**343** *(-187)*
PCA	9	7	7	8	9	7	8	6	8	5	**540** *(+4)*
Adaptive Safety Monitor	10	8	9	8	9	7	9	8	8	5	**597** *(+61)*

* score shown follows initial investigation, yielding no usable results.

3-2.5.1 – Observers

An initial investigation into observers – specifically, the Kalman filter – found that while they could be used in a functional software fault detection and diagnosis environment, such a concept would essentially be based on the current method for fault detection. Like the current method, a mathematical model of the monitored system is compared to the measured outputs of said system in order to produce an error value (residuals); additional mathematical functions would also need to be applied to measured inputs y and/or predicted states $\hat{x}$.

The strength of Kalman filters in a traditional sense is that they can "see through" noisy data using known dynamics, but in a software environment where the system is a user-defined software function, those dynamics are often highly complex, and discrete (as opposed to continuous states). Noise factors in the functional software would manifest themselves as process noise (w_k), but it is more likely that certain 'dynamics' would be missed during

modelling. It seems the Kalman filter can only account for zero-mean Gaussian white noise, so noise factors that cause a signal offset, for example, will affect residuals and thus possibly register false readings. On the other hand, the residuals and observer banks could be arranged to be sensitive to certain faults and not to others, which would then feed into a decision block to evaluate fault magnitude and isolate it using the residuals. The decision block could theoretically look for combinations of residuals to determine if a true fault has occurred, ignoring false positive spikes in residuals, but this will require extensive testing and will be likely be made bespoke to each powertrain setup, limiting transferability. Extended State Observers were found to be a relatively new type of observer that again relies on system dynamics, but its application potential to monitor the functional software is likely low due to the nature of the software interactions found there, the rate at which it needs to detect a fault, and the safety critical nature of the application.

Investigation was halted ultimately due to incompatibility between the model form of safety software and the form required for Kalman filter implementation.

3-2.5.2 – Neural Networks

From initial learning in the literature review, it was determined the primary barrier facing the neural network as a safety software concept is verification. The weightings are very abstract, making full formal verification a near impossible task. Furthermore, the more successfully implemented ANNs found in literature continue to evolve and adapt after deployment. Verifying this would be very difficult due to the abstract nature of the inner workings of an ANNs, and though reliability and performance may be improved through evolution and adaptation, verifying safety due to these post-deployment adaptations has been determined unfeasible at this point. Neural networks are a significant change from the current continuous torque demand monitor method used, as it is not clear which signals should be included in the input layer, and what the output layer would resemble, making the concept less compatible over the benchmark concept. Each modification would require full retraining of the neural network, as the weightings may or may not be related to the sections being modified. While the neural network training does not require much expertise (since it is learning for itself), the setup up and interpretation of the model would need to take place for each powertrain variant and modification, impacting initial design effort. Neural networks may require significant storage and computational requirements depending on the complexity, and the number of neuron layers; this complexity makes the concept less simple. Overall, verification is the primary concern, making this concept commercially infeasible for this application.

3-2.5.3 – PCA

From the literature, it seems that PCA has been successfully used in safety-critical fault detection applications, so there is reason to believe that it can be implemented successfully as a safety concept on a vehicle. Until the investigation is conducted, it is unknown if it is functionally suitable, particularly as it is a novel concept; the same can be said about reliability. As PCA is a process-history based method, verifiability can be dependent on the quality of the training data it is developed on (Verifiability) but should be easier to verify than neural networks owing to being able to see the components of the PCA models once derived, though may need some expert knowledge to interpret them appropriately. It is expected that PCA can be compatible with the current system, as it performs essentially the same function as the safety software fault detection, with the same inputs producing the torque error (compatibility).

PCA is excellent at describing very large amounts of data using just a few parameters, and could therefore reduce the storage capacity needed; once the PCA model is derived, the training data does not also need to be stored on a production vehicle, so minimal storage space is required. Additionally, it seems that PCA only needs to perform some multiplications to produce a torque error from the inputs, so little computational resources are used (hardware and computational requirements). Attaining sufficient training data for PCA to derive the model during development can be a difficult – therefore costly – task for most applications, but since it is monitoring a software plant, it is expected that the training data can quickly be attained, provided full coverage can be made (Initial Design Effort).

Once derived, a PCA model is self-contained and can be a simple concept to implement (Simplicity). However, any changes made to the functional software may require re-derivation of the PCA model, impacting the ease of modifiability; if the necessary training data is easy to obtain through automated scripts – seeing as the application is software-based – modification may be relatively straightforward. There is scope also to include some evolving automated re-derivation process, improving the PCA model after the vehicle has been sold, but this naturally brings with it verifiability concerns. Overall, there is potential for PCA as a commercially feasible safety concept.

3-2.5.4 – Adaptive Safety Monitor

The highest attributes score is held by the theorised adaptive safety monitor. As it is essentially an adaptive version of the continuous demand monitor, it should be similarly functionally

suitable. A simpler safety software model means a reduced initial design effort, reduced hardware (and possibly computational) requirements, makes modifications easier, and provides a simpler concept overall.

The safety software now can provide greater reliability to the driver, with long term changes to the vehicle less likely to affect false positives in the safety software due to how the concept changes on the vehicle according to slow moving torque errors (Evolvable). However, this adaptation could impact verifiability effort, as it needs to be ensured that the concept does not adapt in a way that could make it functionally unsuitable. Despite this, the initial attributes points towards the adaptive safety monitor possessing strong commercial feasibility, and therefore is a prime candidate for investigation.

3-2.5.5 – Selected Concept Candidates

Of the four potential concepts, only the PCA safety monitor and the adaptive safety monitor have initial scores greater than the continuous torque demand monitor, and therefore will be carried forwards for investigation.

3-3 – Concept Development and Testing Environment

To test the PCA and adaptive safety monitor concepts, a testing environment needs to be established. The methodology for evaluating these concepts will be to use the testing environment to determine if the concepts are indeed able to detect faults in a timely manner, versus the benchmark – i.e. they meet functional suitability requirements. The experience gained throughout the development of these new concepts will then been used to determine a post-investigation score of each attribute in Chapter 6.

A development and testing environment has been created using the Matlab and Simulink workspace to fairly evaluate and test the selected concepts. Data collection was conducted in a Range Rover Hybrid test vehicle, and that data was used – along with vehicle parameters – to produce a vehicle model in Simulink, using the Simscape modelling toolbox to validate the model against the collected data; details to this modelling and validation process can be found in the Appendix.

3-3.1 – Twin-EV Powertrain and Vehicle Model

The sponsoring manufacturer was able to provide powertrain control software for a pre-production twin-EV powertrain. The powertrain comprises of two reduction-geared 300 Nm electric motors with a maximum angular speed of 7000 rpm, one on each axle driving the differential. It was decided that such a powertrain would suit safety concept development much better than the hybrid variant, as complex unmodelled transient dynamics in the ICE would be eliminated, as would the 8-speed transmission, reducing simulation times to ~1% of the previous powertrain model. The only drawback was that a test vehicle housing this powertrain was not available for data collection and re-validation, so the decision was made to modify the validated Range Rover Hybrid model to match the twin-EV powertrain [17] shown in Figure 3-7.

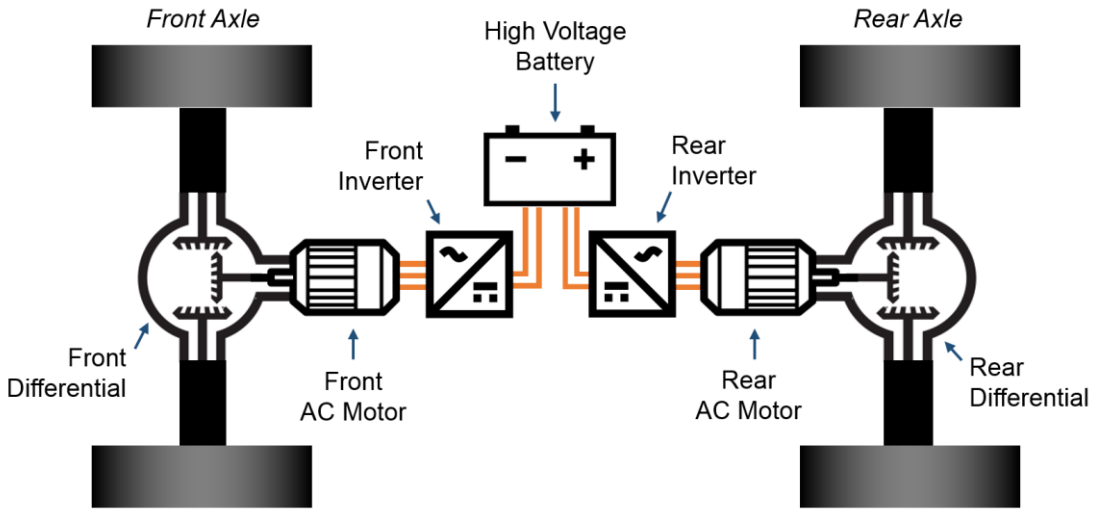

Figure 3-7: Twin-EV Powertrain. Black connections indicate mechanical coupling, and orange indicates HV electrical connections.

The drawback was deemed acceptable for this investigation, as it serves the purpose of being a suitable environment for testing the operation and effectiveness of the novel safety monitoring concept candidates. The vehicle parameters for the Twin-EV powertrain are shown in Table 3-12. Some of the parameter values have been slightly altered from the Range Rover Hybrid due to commercial reasons.

Table 3-12: List of key vehicle parameters used in Twin-EV powertrain model

Parameter	Value	Unit
Vehicle Mass	2000	kg
Drag coefficient	0.37	-
Frontal Area	3.07	m^2
CG Height	0.5	m
CG from front axle	1.4	m
Wheelbase	3	m
Trackwidth	2.2	m
Wheel Radius	0.35	m
Tyre Friction Coefficient	0.02	-
E-machine Inertia	0.06	kgm^2
Electrical Efficiency Constant	0.9	-
Mechanical Efficiency Constant	0.95	-
Max E-machine Torque (per motor)	300	Nm
Max E-machine speed	7000	rpm
Reduction Gear Ratio	9.04:1	-
Rear Differential Gear Ratio	1:1	-
Rear Differential Inertia	3.2	kgm^2
Front Differential Gear Ratio	1:1	-
Front Differential Inertia	3.2	kgm^2

Many of the same vehicle parameter values were carried over from the Range Rover Hybrid as it was expected a similar class vehicle would eventually carry this powertrain. The arrangement of the new twin-EV Simscape model, centralised control structure, and driver model is shown in Figure 3-8.

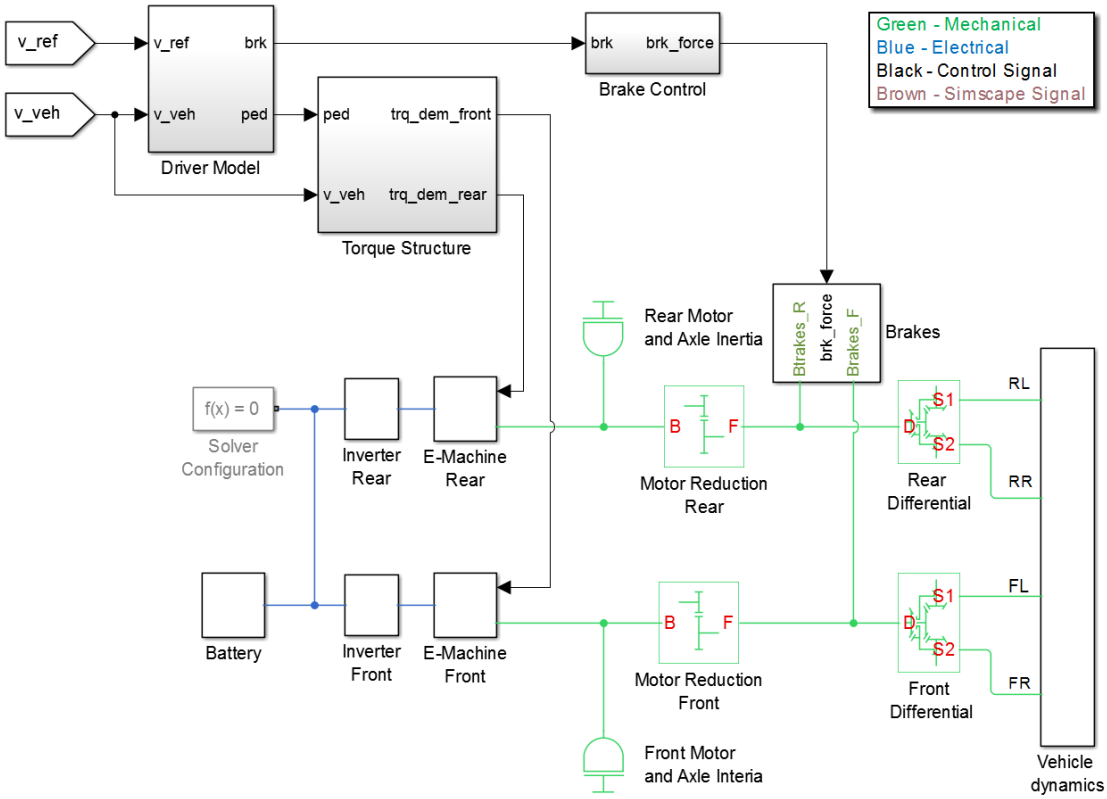

Figure 3-8: Twin-EV Simscape vehicle model with centralised control system

The vehicle model components are unchanged, with exception to the electric motors which can now produce 300 Nm at 0 rpm. A reduction gear of 9.04:1 is placed on each axle before the differential, stepping up the torque and stepping down the speed; the differentials both have a 1:1 final drive ratio. The total torque demand from the torque structure is the sum of torque demand to the front and rear motors.

3-3.2 – Safety Software Test Method

The powertrain control software provided by the OEM was calibrated for the twin-EV development vehicle. Due to IP sensitivity, the complex software was contained mostly within an S-function that was not a searchable Simulink model. However, a list of variables was also provided which included a torque map, whereby desired total wheel torque was calculated by accelerator pedal position and vehicle speed: this is used as part of the complex calculation of torque demand.

Using just the vehicle map, and accounting for the total gear ratio from each e-machine to its axle (9.04:1), the basic drive functionality of the complex powertrain control software S-

function could be achieved without unintended interactions with – and interventions from – unidentifiable software components hidden in the powertrain control software. Seeing as the actuators are identical, the torque demand is simply split evenly for delivery. This comparatively simple set of software, in effect, formed the functional software that new safety software concepts will seek to monitor, and is shown in Figure 3-9.

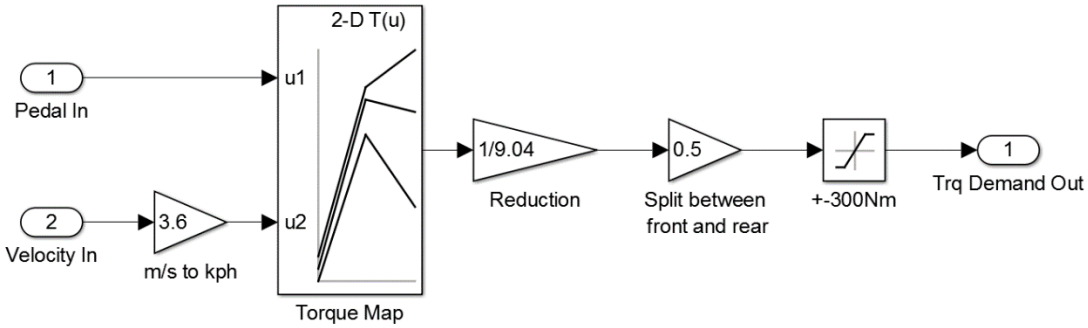

Figure 3-9: Simplified Torque Structure from complex VSC provided by the OEM

While the vehicle velocity input is calculated directly from the vehicle dynamics body Simscape block, control of the pedal percentage input needs to come from the driver. A driver model was created that would ultimately modulate the accelerator pedal to match the vehicle velocity with a desired target vehicle velocity. The driver model is simply a PI controller with an output range of 0% to 100%, though there is scope to dynamically adjust the PI gains through scheduling and reset integral terms as necessary.

In order for concept testing and performance assessment to be made, a test method needs to be established. The assumption is made that if a concept is functionally suitable, it should be able to detect faults in the functional software due to the comparison of actual torque demand from the functional software, and permitted torque demand as calculated by the safety software. Therefore, in the absence of faults, it can be assumed the torque error would be zero.

The most straightforward way to achieve this in the test environment would be to simply duplicate the torque structure module as the safety software model. When no fault is present, the torque error would be zero. If a fault were to be then injected in the torque structure (but not the safety software) a torque error of the difference between permitted torque demand and torque demand – the same profile of the fault injection – would be produced. Figure 3-10 shows the Simulink test structure for benchmarking new concepts, with p indicating pedal position (%), v indicating vehicle velocity, and τ indicating torque.

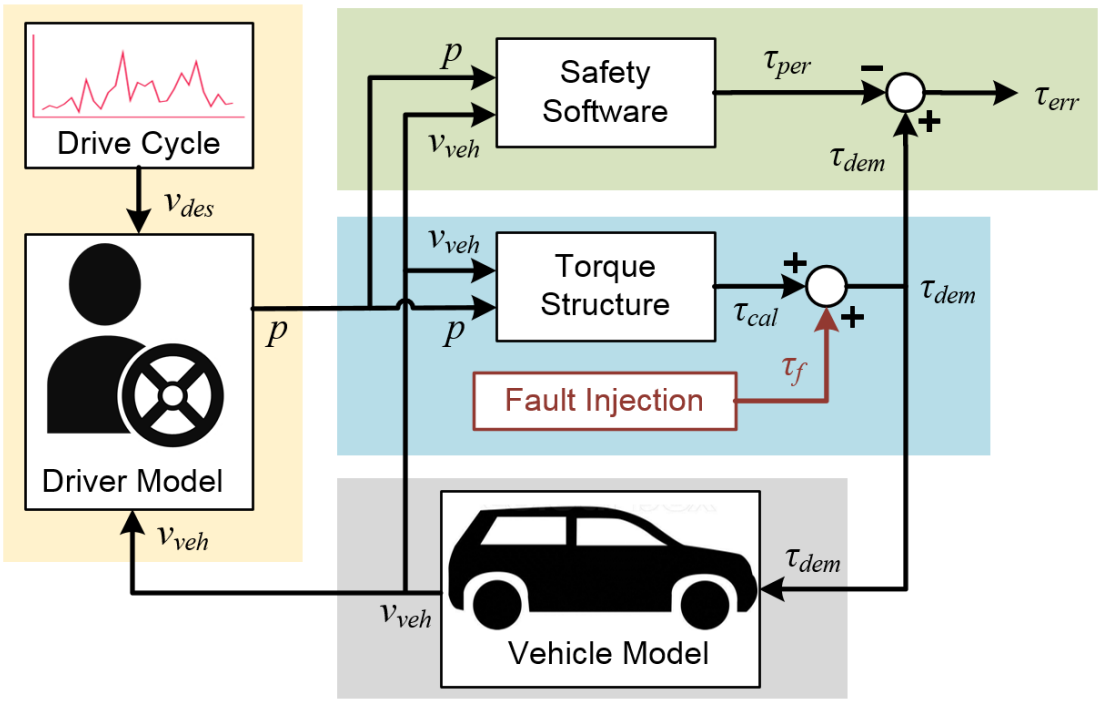

Figure 3-10: Simulink Test Structure with driver model (yellow), functional software (blue), safety software (green), and Simscape twin-EV vehicle model (grey).

The aim of this test is to determine whether a new concept is functionally suitable, i.e. it can detect the fault in the functional software accurately and quickly. The fault reaction mechanism and intervention system will be excluded from analysis, as fault detection is the primary focus of this project. The fault reaction mechanism serves a separate function to detection, as it simply acts when the fault detection concept determines a fault has occurred. Most of the other attributes will be assessed through the experience of development and anticipated challenges of each new concept with the expert knowledge from the literature review and from Jaguar Land Rover, seeing as any new concepts cannot feasibly be put through the full safety lifecycle; where possible like-for-like testing will be conducted.

3-4 – Conclusions

With the concept phase of ISO 26262 explored through the lens of powertrain safety software, the core task of the safety monitoring concept has been defined. The limited scope of addressing a safety goal preventing unintended acceleration means a focused goal of monitoring malfunctions of torque demand can be made, guiding future investigation. Based on the principles of ISO 26262, ISO 25010, and expert opinion from the sponsoring OEM,

eleven ideal monitoring attributes were identified to winnow down the concept candidates to those that show the most promise against the benchmark continuous torque demand monitoring concept, shortening the preliminary concept list to two concept candidates: the adaptive safety monitor and the PCA safety monitor. In the following two chapters, the development of the adaptive safety monitor and PCA safety monitor will be detailed using the testing environment established in the second half of this chapter.

Chapter 4

Adaptive Safety Monitor

4 – Summary

A novel adaptive safety monitor is proposed as an innovative software fault-detection concept, aiming to enable transferability between powertrains without modification and minimal recalibration effort. This chapter will outline specific challenges of development burden faced by current fault-detection methods, and how an adaptive safety monitor concept can overcome them. Development of the adaptive safety monitor concept is discussed, with the introduction of a two-stage algorithm, and a performance analysis is conducted through model simulation to demonstrate improved robustness against false faults over the continuous torque demand monitor. A parameter calibration and optimisation process is then demonstrated through design-of-experiments, concluding with further work and an outlook into future commercial applications.

- **Objective 4:** Conduct detailed investigation into candidate concept.

4-1 – Adaptive Safety Monitor Concept

4-1.1 – Safety Software Model Fidelity

In the introduction (Chapter 1), the problem of rising software complexity was identified as putting increased burden on automotive manufacturers. The continuous torque demand monitor (alluded to in Chapter 1 and introduced with more detail in Chapter 2) monitors the functional software using a set of high-integrity safety software. Typically, this safety software contains a simplified version of the functional software in a safety software model; a perfect copy of the functional software is not used for the model, as its complexity makes the effort of verification too great. The safety software model used in the continuous torque demand monitor is, however, of a high-fidelity to minimize the modelling discrepancies between the safety software and the functional software, and to provide accurate torque error estimations.

One way to reduce the development burden is to use a further simplified, low-fidelity model in the safety software. Provided the functional suitability is not sacrificed (ability to always meet safety goals), a simpler safety software model will be less costly to develop, validate and verify. Additionally, the transferability between powertrain variants could be increased due to the relative simplicity. However, the simplified model could consequentially at times differ more from the functional software it monitors in such a way that it falsely identifies a torque error to be an unsafe fault, leading to an unnecessary limitation of powertrain availability to the driver in reaction to this false fault flag. While safety is still achieved with this simplified safety software model, the driver has the inconvenience of reduced availability when there was no fault present. In this context, the strengths of a high-fidelity model can be summarised as:

- Meets safety goals.
- More reliable against false faults, leading to increased functional availability.

while the benefits of a low-fidelity model can be summarised as:

- Meets safety goals.
- Reduced complexity.
- Reduced development and verification effort.
- Reduced cost.
- Increased transferability between powertrain variants.
- Possible reduced storage and computational requirement.

In practice, functional safety engineers seek to strike a balance between development cost and robustness against false positives in the safety monitors they develop, permitting a certain amount of torque error to be present, provided safety goals are still met. Typically, this should not cause a false fault to be flagged under normal operating conditions (which in this case implies no malfunction has occurred and any torque error is due only to noise factors, primarily unstructured uncertainties), but it leaves less tolerance for any additional factors in the torque error to occur before a fault is flagged. The tolerance is called the 'minimum headroom' of the safety monitor, practically being considered the amount of additional torque error from a fault that could cause a fault to be flagged; a high-fidelity model will have a larger minimum headroom, and a perfect model would have a minimum headroom equal to the 'nominal headroom' – that is, the torque error permissible before the safety goal is violated. To illustrate this, consider Figure 4-1, which compares a representative histogram distribution of nominal torque error over some period of time in both a high-fidelity and low-fidelity model in the safety software under normal operating conditions [37].

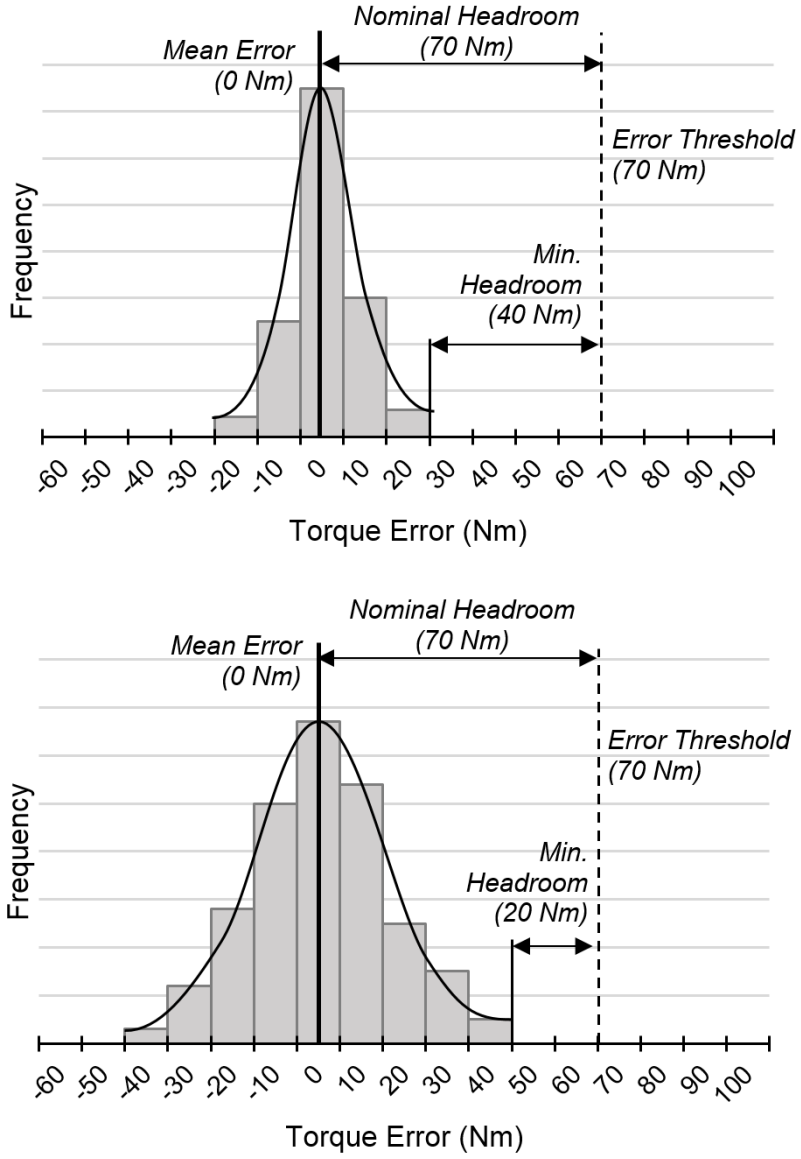

Figure 4-1: Representative torque error histogram and distribution under normal operating conditions for high-fidelity (top) and low-fidelity (bottom) safety software models

In the nominal torque error calculation of Figure 4-1, both models are assumed to exhibit a torque error distribution centre around zero under normal operating conditions. Comparing the variance of the torque error, however, a high-fidelity model would exhibit a tighter distribution under normal operating conditions compared to that of the low-fidelity model, leaving a sizable minimum headroom to allow for manageable faults to occur without a false fault flag; The low-fidelity model, meanwhile, leaves much less headroom for manageable faults to occur before a fault is flagged by the safety monitor, unnecessarily limiting availability to the driver since the fault would not, in fact, violate a safety goal.

Now consider a real malfunction occurring in the functional software that is being monitored by the safety software model. The malfunction yields a small fault of 30 Nm which, in itself, does not violate a safety goal for unintended acceleration. The distribution of torque error seen in the safety models from Figure 4-1 will tend to shift, such that the centre of torque error distribution surrounds the 30 Nm fault. The higher-fidelity model, with its tighter torque error distribution and larger minimum headroom, can tolerate the 30 Nm fault and not cause a fault to be flagged. However, the lower-fidelity model, with its wider torque error distribution, cannot tolerate such a fault as one or more of the torque error measurements exceed the error threshold. Therefore, a fault is incorrectly flagged – a false positive, since the 30 Nm fault does not actually violate the safety goal – due to the noise factors present in such a low-fidelity model. Figure 4-2 illustrates this scenario.

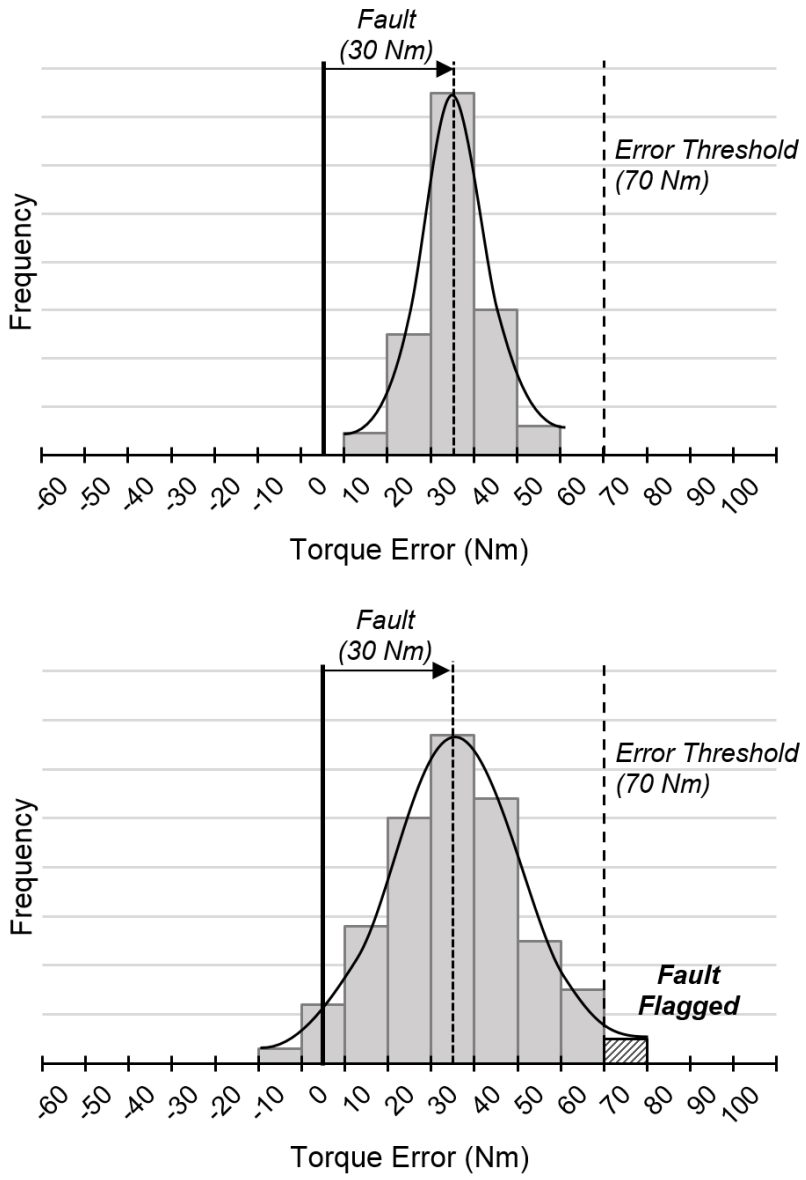

Figure 4-2: Representative torque error histogram and distribution with 30 Nm fault for high-fidelity (top) and low-fidelity (bottom) safety software models

It is also possible that unmodelled changes in operating conditions such as aggressive driver behavior, high altitude, and high temperatures can result in a wider distribution of torque error during normal operating conditions, leading to false fault flags with no real fault present in functional software.

4-1.2 – Adaptive Safety Monitor

When torque error distribution was used to visualize the effect of high and low fidelity safety software models, it was theorised that the safety monitor could be allowed to slowly adapt to manageable changes in mean error since these do not pose immediate safety violations to the driver, regardless of the source of the slow-moving torque error. In doing so, there is a reduced likelihood that a slow-moving change in torque error distribution, would cause torque error values at the edges of the distribution to exceed the error threshold and flag a false fault. A fast and significant change in torque error – one that should indicate a safety risk to the driver – should always cause the distribution to exceed the error threshold. A fast change of a manageable magnitude – such that the driver retains control and would not be put at risk – could still cause an adaptive low-fidelity model to exceed the error threshold since it is only permitted to adapt at a prescribed safe rate, potentially causing a false fault flag. The small fault magnitude does mean, however, that the distribution is quickly brought back below the error threshold, and the amount of availability lost to the driver would be minimal.

The adaptive safety monitor could therefore allow a low-fidelity safety software model to be used as part of the functional safety concept, while still achieving safety, but with the added benefit of increased robustness against false faults, thereby increasing functional availability to the driver. The adaptive safety monitor is compatible with the existing functional safety concept architecture, becoming implementable between the nominal torque error calculation and the fault decision function. Birch et al [37] developed the first instance of the adaptive safety monitor, whereby the mean error is calculated and stored. For the assumed Gaussian distribution, the mean error is simply the mean of all the sampled torque error values within the torque error sampling window. Then the error offset is calculated, though this is not simply the most recent mean error value as it is limited to a degree based on allowable rate of change and maximum or minimum allowable offset value. This prevents the adaptive safety monitor from adapting too quickly, such that an unsafe fault would be adapted to by the adaptive safety monitor faster than the driver could and consequently be missed. Once the error offset is calculated, it is subtracted from the nominal torque error to produce the ‘offset torque error’.

4-1.3 – Two-Stage Adaptive Safety Monitor

Noise factors that are inherently present in non-perfect-fidelity safety software models yield different torque error behaviors in magnitude and rate of propagation. Certain noise factors propagate very quickly, such as proportional gains in a controller or measurements in the plant itself such as electric motor current. Others noise factors, meanwhile, propagate quite slowly, such as sensor drift over time, changes in controller behavior due to a small integral gain, or measurements in the plant itself such as compensating for actuator temperature. In a case where multiple noise factors occur simultaneously (as is often the case) it can be beneficial to be able to adapt slowly to slow-propagating noise factors, while having a different approach to fast-propagating noise factors.

During development, it was found that performing just one offset calculation step greatly limited tunability of the way the adaptive safety monitor performs in this respect. If a sensor drift was simulated with another faster-propagating noise factor included, the adaptive safety monitor with only one adaption calculation would be tuned between ignoring all fast-propagating noise factors or having to re-adapt to slower propagating noise factors when a fast-propagating noise factor occurred; each tune of the adaptive safety monitor parameters became a compromise. Therefore, a two-stage adaptive algorithm was introduced that allowed separately tuned responses to slower and faster torque error propagation, which improved performance overall. The two-stage adaptive algorithm is illustrated in Figure 4-3.

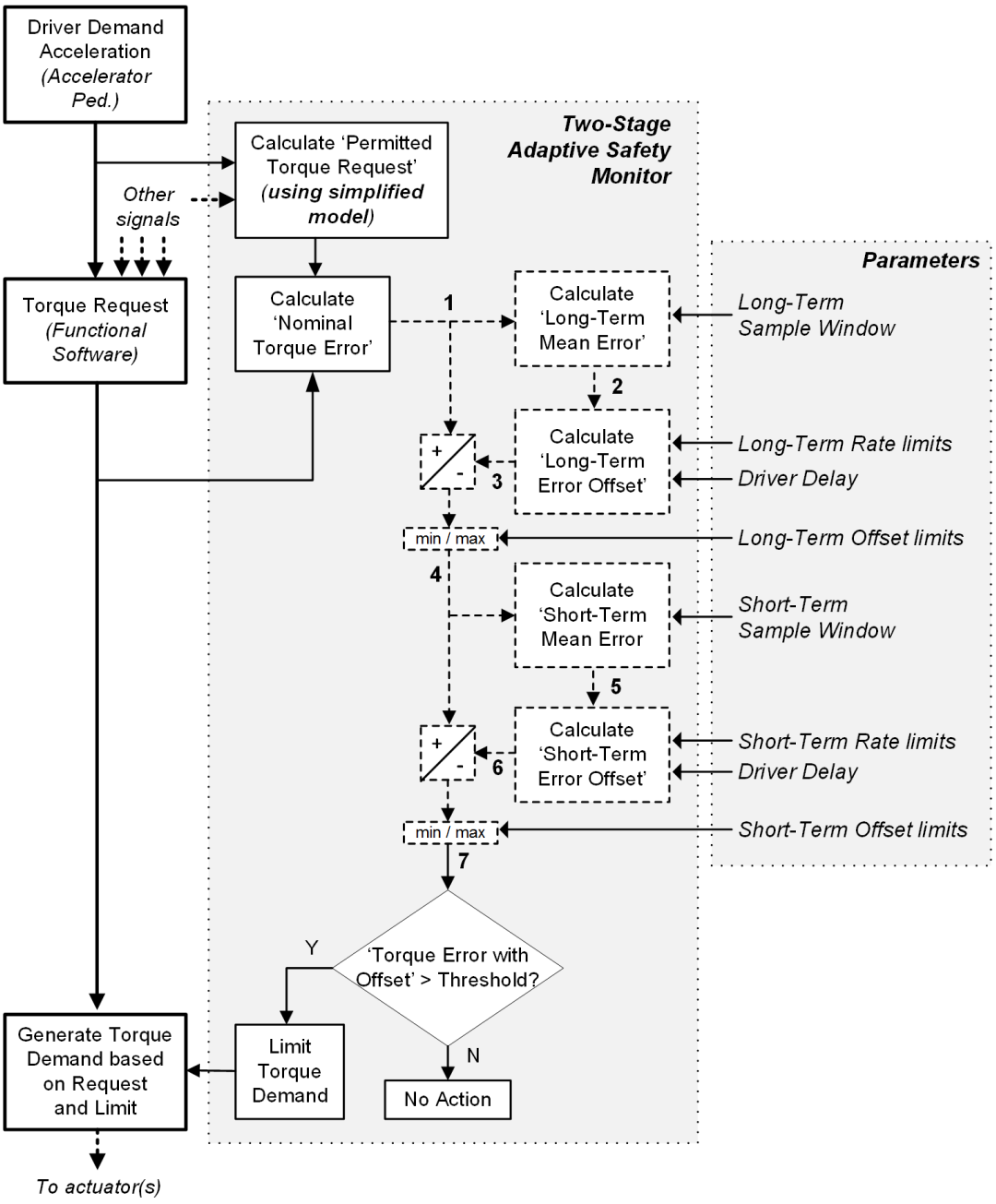

Figure 4-3: Two-Stage Adaptive Safety Monitor Algorithm theoretical architecture with simplified safety software model, parameters, and annotated calculation steps.

The Two-Stage Adaptive Safety Monitor function (highlighted by dashed outlines in Figure 4-3) is coupled with a simplified safety software model, which is less costly to develop when compared to the high-fidelity model used in the continuous torque demand monitor, and comprises of two adaptation calculation stages tuned for different noise factor propagation rates; these will be discussed in further detail in the next section.

4-1.4 – Equations

The equations governing the calculation of the output of the two-stage adaptive safety monitor are outlined in this section. Figure 4-3 is used to describe the calculation steps and how the tunable adaptive safety monitor parameters are used.

4-1.4.1 – Long-Term Adaptation Stage

First, the long-term adaptation stage deals with slow-propagating noise factors that typically exhibit slow changes compensated for in the functional software, such as ambient air pressure or battery temperature. It seeks to adapt slowly over time in the same way a driver slowly begins to experience any long-term deviations from the original expected baseline behaviour. After obtaining the current nominal torque error, $\boldsymbol{\varepsilon_{\tau,t}}$, in step 1 of Figure 4-3, the long-term mean error for current time *t*, $\varepsilon_{\tau l,t}$, is calculated as the average of all nominal torque errors over the long-term sample window, t_l, using nominal torque errors from time t-t_l to time t:

$$\varepsilon_{\tau l,t} = \left(\frac{t_l}{t_{sample}}\right) \times \sum_{i=t-t_l}^{t} \varepsilon_{\tau,i} \tag{4-1}$$

where t_{sample} is the timestep size. Next, in step 3 of Figure 4-3, the long-term offset for current time *t*, $\tau_{lo,t}$, is the previous long term offset $\tau_{lo,t-1}$ plus the change, where r_l is the long-term rate of change limit. The driver fault delay for time *t*, ($K_{dd,t}$) can prevent any adaptation taking place if equal to zero; the reasoning behind the driver delay is later discussed in Section 5-2.2.2.

$$\tau_{lo,t} = \tau_{lo,t-1} + \left(\min\left(\left|\varepsilon_{\tau l,t}\right|, r_l\right) \times \operatorname{sgn}\left(\varepsilon_{\tau l,t}\right) \times t_{sample} \times K_{dd,t}\right) \tag{4-2}$$

The long-term offset is used in step 4 of Figure 4-3 to calculate the current long-term offset torque error, $\varepsilon_{\tau lo,t}$, which is the current nominal torque error minus the long-term offset, bounded by the minimum and maximum long-term offset limits, min_l and max_l.

$$\varepsilon_{\tau lo,t} = min_l \leq \varepsilon_{\tau,t} - \tau_{lo,t} \leq max_l \tag{4-3}$$

The minimum and maximum long-term offset limits can be static or dynamic.

4-1.4.2 – Short-Term Adaptation Stage

The short-term adaptive stage deals with more dynamic changes in torque error that could arise through some active control systems like cruise control. Through using a smaller torque error sampling window to calculate the 'short-term mean error', immediate changes can be better estimated, being less weighted by historic data. This is then used to calculate the 'short-term offset torque error' with a greater allowable rate-of-change and a separate set of allowable offset limits. Note that the 'short-term mean error' is calculated using the average 'long-term offset torque error', as opposed to the nominal torque error, because it is assumed that long-term effects would at worst compound with short-term ones to give the overall nominal torque error, and by offsetting long-term effects first a more accurate estimation of short-term effects can be obtained by the short-term adaptation stage of the safety monitor.

The short-term mean error for current time t, $\boldsymbol{\varepsilon_{\tau s,t}}$, is the average of all long-term offset torque error $\boldsymbol{\varepsilon_{\tau lo,i}}$ over the short-term sampling period, t-t_s to t; this relates to step 5 of Figure 4-3.

$$\varepsilon_{\tau s,t} = \left(\frac{t_s}{t_{sample}}\right) \times \sum_{i=t-t_s}^{t} \varepsilon_{\tau lo,i} \tag{4-4}$$

Next, the short-term offset for current time *t*, $\tau_{so,t}$, can be calculate (step 6 of Figure 4-3). It is the previous short-term offset $\tau_{so,t}$ plus the change for the sample unit, where r_s is the short-term rate of change limit, and t_{sample} the timestep size in seconds. The driver fault delay for time *t*, ($K_{dd,t}$) can again prevent change taking place, if equal to zero.

$$\tau_{so,t} = \tau_{so,t-1} + \left(\min\left(\left|\varepsilon_{\tau s,t}\right|, r_s\right) \times \operatorname{sgn}\left(\varepsilon_{\tau s,t}\right) \times t_{sample} \times K_{dd,t}\right) \tag{4-5}$$

The short-term offset torque error, $\varepsilon_{\tau so}$, is the current long-term offset torque error minus the short-term offset, bounded by the minimum and maximum total offset limits, min_{total} and max_{total}.

$$\varepsilon_{\tau so,t} = min_{total} \leq \varepsilon_{\tau lo,t} - \tau_{so,t} \leq max_{total} \tag{4-6}$$

This short-term offset torque error, step 7 in Figure 4-3, is the output of the two-stage adaptive safety monitor, and can also be referred to as simply the 'offset torque error', since it is the cumulation of both long-term and short-term offset stages.

4-2 – Performance Analysis

Consider a passenger car offered by an automotive OEM with a range of powertrain options with automatic transmissions, including at least one ICE variant and one full-EV variant. A common drivability function in automatic transmission passenger cars is ‘creep’, where the car will very slowly move forwards from standstill upon the release of the brake pedal, up to a speed of approximately 6 kph. While the outcome in both powertrain variants are the same, the method in which creep is achieved is very different. For the ICE variant with a torque convertor, creep is achieved simply as a by-product of the engine idle-speed governor acting through the torque converter, and not directly controlled by an active controller [118].

A full-EV variant, on the other hand, has to actively control the torque demand to the electric motors such that creep behaviour is emulated, meaning an additional functional software component needs to be developed in torque structure to add this capability. Consequently, a safety monitor likely will also need to be developed alongside in order to account for the additional torque demand supplied by the creep controller. Not doing so will cause a torque error to occur at low speed, since a torque is being demanded despite no input from the driver. Of course, if a fault is flagged by the safety software due to this torque error, it would be a false fault, and will unnecessarily limit powertrain availability. Torque demand cannot simply be capped either, since the creep controller will need to be able to creep up an incline, potentially demanding very large torque values. However, due to the slow dynamic nature of vehicle creep functionality, an opportunity is presented to implement an adaptive safety monitor on a simplified safety software instead of developing and verifying additional safety software.

4-2.1 – Vehicle Model and Methodology

A Matlab/Simulink Simscape full-EV vehicle model has been developed as a test bed for the adaptive safety monitor, based on a powertrain with one electric motor on each drive axle. The two electric motors are controlled by a torque demand supplied by the functional software torque structure, which itself is controlled by a driver model trying to achieve a target vehicle speed. Figure 4-4 shows the test architecture used.

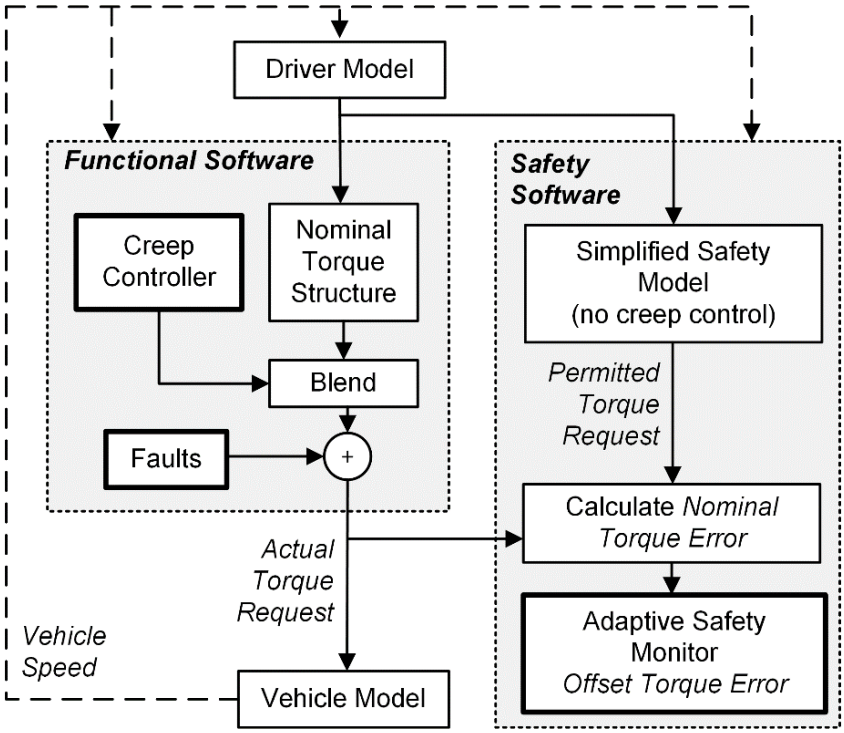

Figure 4-4: Test model architecture for fault detection performance using an adaptive safety monitor, which does not include a fault reaction mechanism.

4-2.1.1 – Safety Goals

Recall that the safety goal Birch et al. [112] defines to address the hazardous event of 'unintended acceleration during low-speed manoeuvre in a pedestrianised area' is to ensure 'vehicle positive longitudinal acceleration shall not exceed driver demand by more than 1.5 m/s^2 for longer than 1 s'. Their fault reaction mechanism for this hazard is to limit excess torque demand to 150 Nm, which they justify by stating that the maximum vehicle acceleration that can be achieved by 150 Nm is 1.5 m/s^2. Through simulation of this vehicle model, it was found that on a -5.5 degree downhill slope (considered the limit of designed creep functionality in this simulation before mechanical brake intervention is needed), an excess torque fault of 90 Nm will give a maximum possible acceleration of 1.5 m/s^2. Therefore, the torque error threshold will be set at ±90 Nm, with safety being achieved once torque error is contained within these limits.

4-2.1.2 – Functional Software

The creep controller for an EV (or a hybrid in EV-mode) will most likely see it incorporated into functional software, different from a conventional powertrain where the idle governor is

included in the engine control software, post torque demand calculation. To emulate the functionality of creep found in a conventional powertrain, the EV creep controller a torque request to match a target creep speed. From a standstill with brake pedal applied the target speed will be zero, and releasing the brake pedal with no accelerator pedal input will result in target speed ramping up to 6 kph over a period of four seconds. When the accelerator pedal is then actuated, the nominal torque structure will take over from the creep controller to deliver torque demand. During a coast-down event, the nominal torque structure will also emulate engine braking through the application of brake torque through the motors, which in turn has an added benefit of electric regeneration. As vehicle velocity approaches target creep velocity, a feed-forward element in the creep controller ensures a smooth transition from nominal torque structure to creep control. A blending function ensures a smooth transfer of torque demand supply between creep control and nominal torque structure.

4-2.1.3 – Safety Software

A simplified model is used for the safety software, which comprises only of an accurate representation of the nominal torque structure and the blend function – a signal smoothing function that both ensures a smooth transition between the nominal torque structure and creep controller, but also generally smooths the response of nominal torque demand. The blending function could exist within the nominal torque structure, but for ease of tuning during development, it existed outside of the nominal torque structure. The functional software does not include any information about the creep controller in the calculation of permitted torque request, and as such it will generate some torque error whenever the creep controller is actively contributing to the torque demand; it is this nominal torque error that will be sent on to the adaptive safety monitor. A list of parameter settings for this set of simulations can be found later in Table 4-2.

4-2.1.4 – Fault Injection

In order to meet the primary safety goal of detecting and preventing unintended acceleration, sudden and significant erroneous torque demand must be detectable by the fault detection module when such a fault occurs, meaning the adaptive safety monitor must not alter the torque error to the point that a dangerous fault would be missed. Meanwhile, slow and/or small faults would not violate the safety goal and can be ignored so as not to cause a false fault flag and limit functional availability. To test this with the adaptive safety monitor, faults can be injected

on the actual torque request in the functional software to simulate such a phenomenon, with the assumption that the safety software will not suffer the same fault.

4-2.1.5 – Fault Detection and Reaction

For this investigation, torque errors of ±90 Nm for any length of time in the will result in a fault being flagged by the fault detection module, owing to it violating the safety goal. A fault reaction mechanism has not yet been implemented in this simulation, but could be expected to be similar to Birch et al [112] by limiting torque demand to 90 Nm when a fault occurs

4-2.1.6 – Test Case

A representative simulation is used where the vehicle first enters into creep from a standstill with no driver pedal input, causing the target creep velocity to begin to ramp and reach 6 kph. Then, the driver demands a ramped speed increase at 12 s with the accelerator pedal to 36 kph where the velocity is then held from 17 s to 22 s. A coast-down event then takes place at 22 s, easing the vehicle back into creep velocity; only simulated overrun braking by the nominal torque structure will be used, as the driver will not use the mechanical brake pedal. Four step-change fault events will take place in simulations where faults are injected; their start time, end time, and direction are shown in Table 4-1.

Table 4-1: Fault event schedule for simulations

Event	Start	End	Duration	Direction
Fault 1	7 s	8 s	1s	Positive
Fault 2	16 s	17.5 s	1.5s	Positive
Fault 3	19 s	20 s	1s	Negative
Fault 4	39 s	40 s	1s	Negative

4-2.2 – Results and Discussion

4-2.2.1 – Simulation 1: No fault

Figure 4-5 compares the nominal torque error from the functional software with the offset torque error calculated in the adaptive safety monitor over the fifty second simulation. Since there is no fault occurrence in this simulation, the error threshold should not be exceeded.

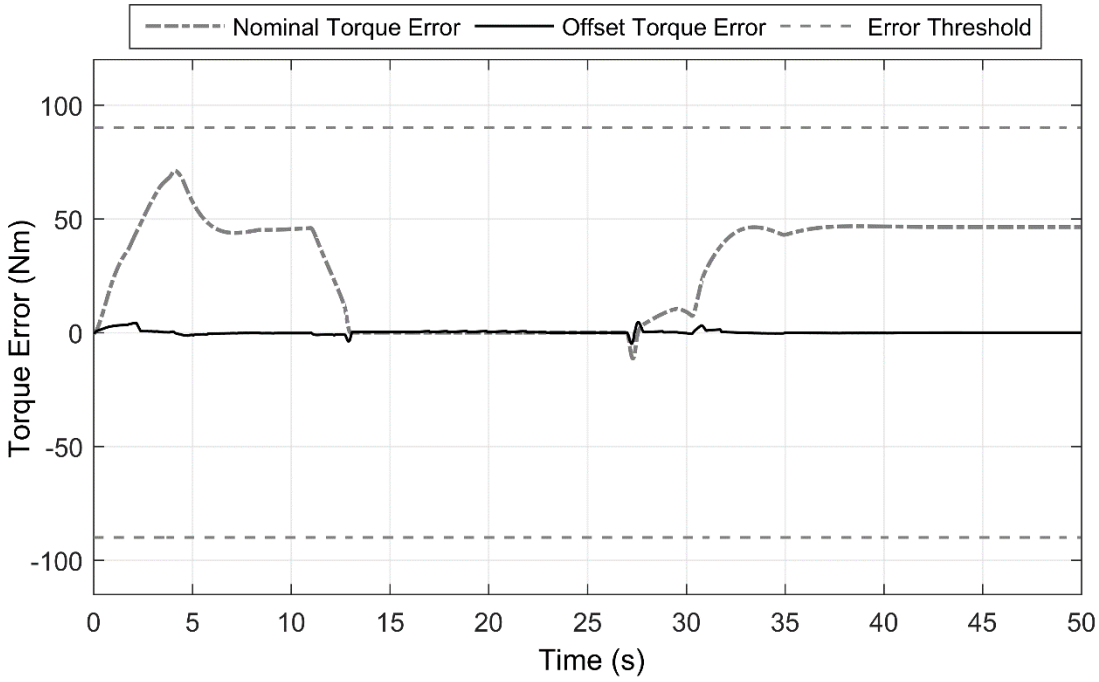

Figure 4-5: Nominal and offset torque error with no fault injection

Figure 4-5 shows that the nominal torque error varies by about 70 Nm due to the torque error introduced by the creep controller. With the adaptive safety monitor, however, the offset torque error varies less than 10 Nm from zero for the whole duration of the simulation. The most significant variation occurs during transition into and out of creep control functionality from the nominal torque structure at 13 s and at 27 s.

4-2.2.2 – Simulation 2: 70 Nm Fault

Four manageable faults of 70 Nm are injected into the torque request; these should ideally not cause the error threshold to be exceeded at any point by the torque error, since they do not violate the safety goals. The results are shown in Figure 4-6.

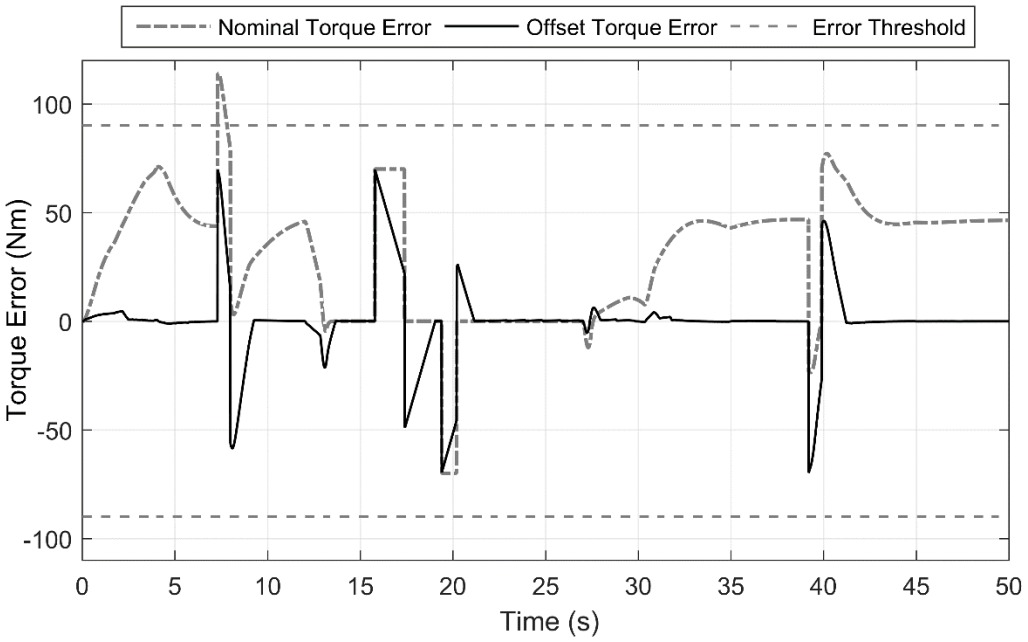

Figure 4-6: Nominal and offset torque error with 70 Nm fault injections

For the nominal torque error shown in Figure 4-6, a false fault flag takes place during Fault 1 and almost including Fault 4 which nearly reaches the error threshold. The adaptive safety monitor, meanwhile, successfully prevents any false fault flag from taking place, always keeping torque error within the error threshold.

Each fault causes an initial 70 Nm torque error magnitude in the offset torque error, to which the adaptive safety monitor then begins to adapt. An important observation to note is an overshoot in the opposite direction immediately after each fault event. This can be attributed, in part, to the creep controller actively counteracting the unexpected deviation from target creep velocity due to the fault when torque demand is sourced from the creep controller. The other factor is the adaptive safety monitor compensates for the fault over its duration after the initial step of 70 Nm, only for the nominal torque error to suddenly step 70 Nm back towards zero, resulting in the overshoot in offset torque error.

A key reasoning for the adaptive safety monitor concept is that since the driver is effectively acts as a controller in the powertrain system, it takes into account the driver's cognitive model of the powertrain, whereby a constant fault – after the initial step-change in torque error – would not violate the safety goal as the driver will adapt to it over time. Therefore, initially after a sudden change in torque error, the driver may need some time to realise a fault had occurred relative to their mental model, rather than immediately adapting. Following the initial reaction time, the driver could then begin to adapt to the constant fault. In terms of the adaptive

safety monitor, this aspect could be realised as limiting the adaption rate for a short period after a sudden fault of a prescribed – and perhaps variable – magnitude is noticed in the nominal torque error; this function is included for the next simulation, where the delay is set to 1 second.

4-2.2.3 – Simulation 3: 120 Nm Fault

In this simulation, the capability of detecting unsafe faults is tested, with each 120 Nm fault event ideally being allowed to exceed the error threshold. This is the primary objective of the safety monitor, as dangerous faults cannot be missed, but over time they can be safely adapted to in a way similar to how the driver adapts. Crucially, it is the initial fault occurrence that needs to be detected and mitigated within the FTTI [26]. Figure 5-7 shows the results of this simulation.

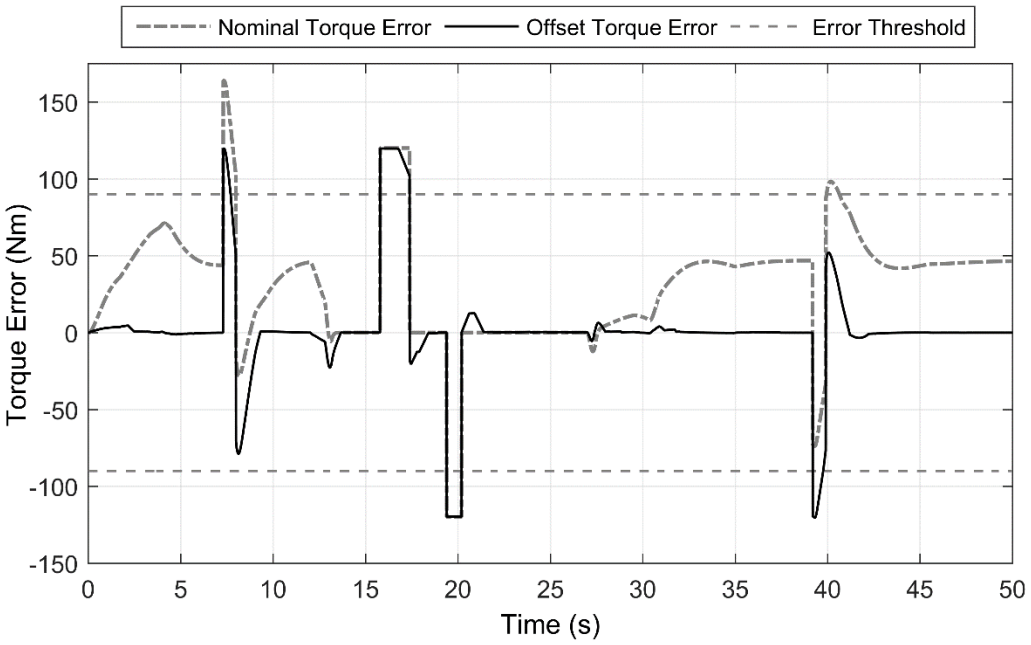

Figure 4-7: Nominal and offset torque error with 120 Nm fault injections.

From Figure 4-7, it is clear that for each fault event the adaptive safety monitor accurately allows all unsafe faults to exceed the error threshold to the correct amount at the start of each event. The additional driver delay rules prevent initial adaption by the driver to account for that reaction time, which is set at one second in this simulation. An overshoot is noticeable at the end of Fault 1, but since the adaption offset is very limited due to the new functionality, this overshoot is mostly due to the nominal torque error as the creep controller actively attempts to counteract the deviation from target creep velocity.

A very important observation and comparison is the behaviour of the nominal and offset torque errors during and following Fault 4. Here, due to the positive nominal torque error immediately preceding the negative 120 Nm fault event, the net nominal torque error at the start of Fault 4 only reaches -70 Nm, which incorrectly does not exceed the -90 Nm error threshold. In this case, an unsafe fault would have been missed altogether, a critical failing of solely using the simplified model in the safety software. However, since the adaptive safety monitor has adapted to that preceding nominal torque error, it forces the offset torque error to correctly exceed that lower error threshold. Furthermore, the overshoot in nominal torque error following Fault 4 actually exceeds the positive error threshold, resulting in a false positive in the wrong direction; the adaptive safety monitor, meanwhile, does not suffer the same problem.

4-2.2.4 – Simulation 4: 120 Nm fault, uphill on incline

In this simulation, shown in Figure 4-8, the same four 120 Nm faults are added into the torque demand calculation, but now the car is on a 5.5 degree uphill incline. On an incline, torque demands from the creep controller are higher since more torque is required to overcome acceleration due to gravity, which in turn lead to larger and fast-changing torque errors.

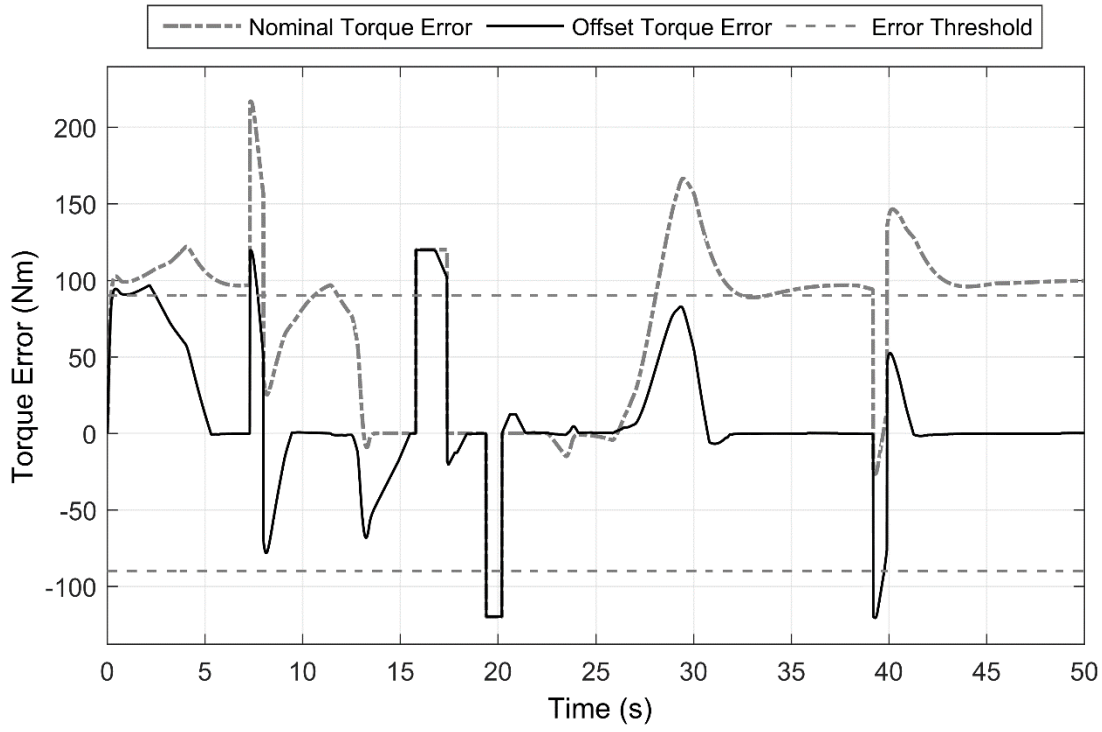

Figure 4-8: Nominal and offset torque error with 120 Nm fault injections for a 5.5 degree uphill incline.

Figure 4-8 show the nominal and offset torque errors in this scenario. Nominal torque error exceeds the error threshold at creep speed, leading to many false fault flags throughout the simulation. The adaptive safety monitor is able to prevent most false flags from taking place, with the exception of the very first few seconds as the creep controller must quickly increase torque demand to prevent rollback and match the increasing target creep velocity. The error threshold is only exceeded by a small amount, however, and if the fault reaction mechanism is to limit torque to 90 Nm, this would only result in a very small loss of availability to the driver. All the faults, meanwhile, are promptly and correctly handled by the adaptive safety monitor.

This phenomenon leads to questions about the ability of the adaptive safety monitor to detect real faults. Consider a scenario where a real fault of 120 Nm occurs at 13 s, when offset torque error is -70 Nm. To the driver, this is indeed a real fault of 120 Nm since the torque error is only present due to differences between the creep controller and nominal torque structure. This fault would be missed by the adaptive safety monitor, since the net offset torque error of only 50 Nm does not exceed any error threshold. However, on this uphill incline, it was found that an excess torque of over 305 Nm is required to cause unintended longitudinal acceleration of 1.5 m/s^2. Therefore, this 120 Nm fault would not, in fact, be a missed fault in terms of violating the primary safety goal. Even if the offset torque error is at the lower error threshold of -90 Nm, a 180 Nm real fault will be captured by the adaptive safety monitor.

4-2.2.5 – Simulation 5: 120 Nm fault, downhill on an incline

The last simulation considers adaptive safety monitor performance on a 5.5 degree downhill slope. Due to the long coast-down time on the hill, an axis break is necessary in Figure 4-9 to concisely show the full simulation; also, Fault 4 is moved to 60 s.

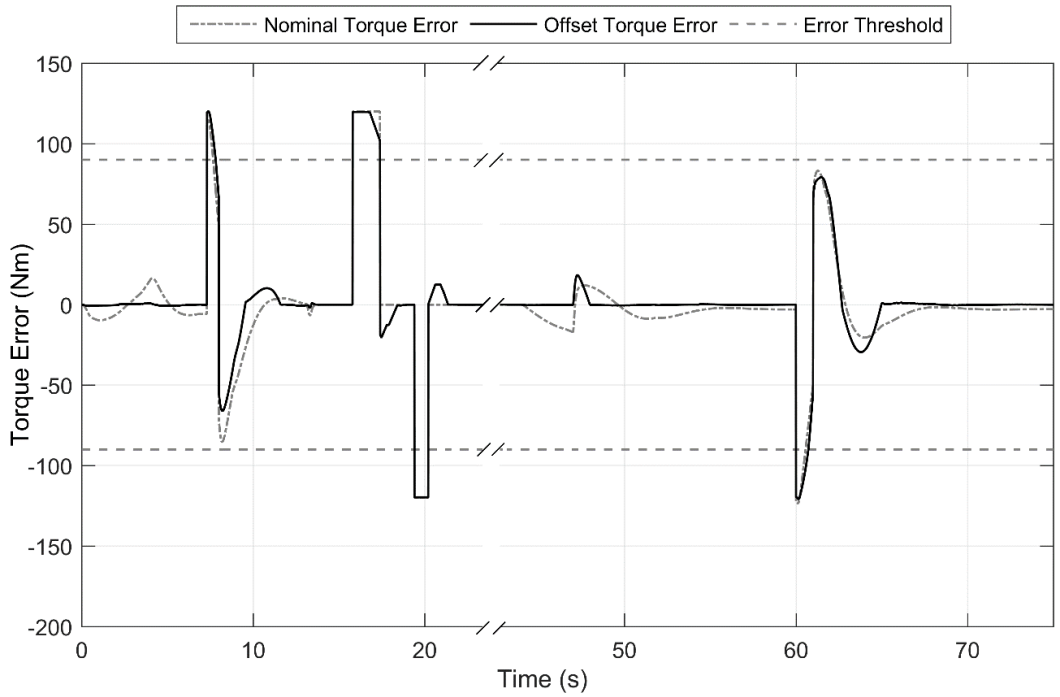

Figure 4-9: Nominal and offset torque error with 120 Nm fault injections for a 5.5 degree downhill incline, with axis break.

Because of the downhill gradient, torque output by the electric motors are much less when compared to that on an uphill incline. Again, each true fault is successfully captured by the adaptive safety monitors offset torque error output, though the nominal torque error does not vary by much outside of fault events. Large overshoots are experienced following the Fault 1 and Fault 4: these will be discussed in the following section.

4-2.3 – Accuracy of the Adaptive Safety Monitor

Accuracy is an attribute quality that is captured by the ideal monitoring attribute of Functional Suitability in Chapter 3. It is the measure of how well the safety monitor performs in terms of the primary safety requirements it must meet, which is to say, how often the safety monitor will detect and classify a safety goal violation when one occurs. From each simulation shown, the adaptive safety monitor has been able to detect and classify a safety goal violation at every occurrence. Though a truly significant improvement over the nominal method, this alone does not guarantee complete accuracy of the adaptive safety monitor. The purpose of the adaptive safety monitor is to eliminate nominal torque error due to noise factors, while ensuring nominal torque error due to real faults are allowed to propagate. Throughout the simulations, however, there are some instances where this nominal torque error is non-zero when a real fault is not present. While these deviations are permissible as – at worst – they result in a false positive, should a real fault occur in the opposite direction to the torque error due to noise, it is possible that the resulting offset torque error would not exceed the torque error threshold. In the preceding experiments for this application, there have been two sources of non-zero offset torque error *not* due to a real fault, which will be referred to generally as 'overshoots.' A brief discussion is provided with respect to how these overshoots could affect the ability of the adaptive safety monitor to detect a safety goal violation: its accuracy.

4-2.3.1 – Accuracy during Creep Control Intervention

The first source of overshoot occurs when the creep controller is active, including transitioning into and out of creep control. Consider Figure 4-9 of Simulation 5. Seemingly, the main criticism is the performance of the adaptive safety monitor immediately following Faults 1 and 4, where significant overshoots are exhibited once each real fault ends. The main contribution to these overshoots is the inherent reaction by the creep controller to counteract against the affect the fault has on vehicle speed. At first, this would seem a vulnerability: should another -120 Nm fault occur at 63 s (where offset torque error is 80 Nm, resulting in a net torque error of -40 Nm), the lower error threshold would not be exceeded by the offset torque error, and a true fault seemingly missed. However, a consideration must be made for what the driver actually expects so soon after Fault 4. Since the torque error counteraction is carried out by the creep controller and not the driver, it could be argued that the driver is still expecting no forward acceleration during this period. By examining such a case where a second larger fault takes place almost immediately after the first smaller fault, an indication can be made on whether a fault of that magnitude would indeed be unsafe, and whether the adaptive safety monitor had missed an unsafe fault. Figure 4-10 compares the torque errors

and the corresponding vehicle acceleration of an initial 95 Nm fault at 79 s, followed quickly by a second 170 Nm fault at 80.3 s, starting from steady creep velocity of 6 kph on a 5.5 degree downhill incline.

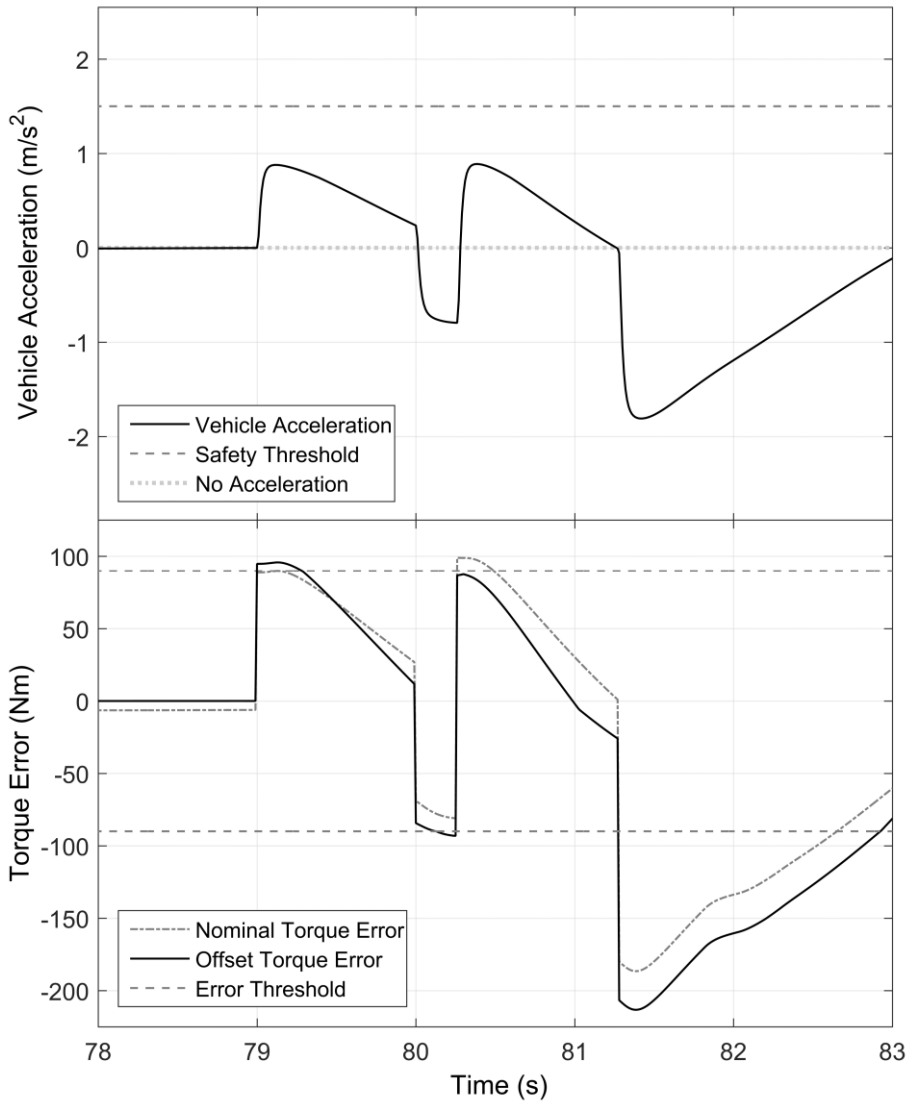

Figure 4-10: Vehicle acceleration and torque error compared on a -5.5 degree slope, with a 95 Nm fault followed by a 170 Nm fault.

Ignoring the fact that the safety goal is not actually violated, peak vehicle acceleration for both the first 95 Nm fault and subsequent 170 Nm fault are actually roughly the same at 0.9 m/s^2, due to the same reason that leads to the 'overshoot' in torque error in the first place: creep controller intervention. This shows that, though the total acceleration as the second fault takes place approaches 1.7 m/s^2, the actual acceleration first 'cancels out' the -0.8 m/s^2 unintended deceleration felt by the driver by the creep controller, leading to just 0.9 m/s^2 of unintended

acceleration. Following both faults, the creep controller provides brake torque in order to reduce vehicle speed back to creep velocity, which would violate a -1.5 m/s^2 threshold (if the driver still expects no deceleration); this is, however, still captured by the offset torque error, and therefore the adaptive safety monitor maintains accuracy at the safety goal level. Better calibration of the adaptive safety monitor parameters (or refinement of the creep control software) could improve the reliability by reducing unnecessary overshoots when no fault is present.

4-2.3.2 – Accuracy during Driver Control

The second source of overshoot occurs when the driver is in control and cannot be explained by the creep control intervention. Overshoots in this category can be found immediately following Fault 2 and Fault 3 in Figure 4-9. In these cases, only the adaptive safety monitor can be to blame, as the nominal torque error does not exhibit the same overshoots. The overshoots in these situations are generally due to the calibration of the adaptive safety monitor parameters with respect to its application. The overshoot immediately following Fault 2 is due to the driver delay parameter, as it allows the adaptive safety monitor to begin adapting after a set period of time. Should this reflect driver perception as intended, the resultant offset torque error would be an accurate representation of how the driver actually experienced the end of Fault 2, as they had started to adapt to the fault. The driver delay parameter value needs to be validated, possibly by a driver task analysis, with respect to the controllability parameter corresponding to the hazard that the safety goal addresses (this is discussed in Chapter 6). The overshoot that occurs after Fault 3 is similarly most likely due to the calibration of the adaptive safety monitor parameters, and for this reason investigating the calibration of the adaptive safety monitor parameters can prove beneficial in improving accuracy in this area, an investigation which is performed in the next section.

4-3 – Calibration and Parameter Optimisation

An anticipated benefit of using an adaptive safety monitor is that it could be transferred and used for many different applications within the functional software and between multiple powertrains. These applications vary in requirement, so the adaptive safety monitor needs to be calibrated somewhat for each application to ensure adequate performance. There are a number of (currently) fixed parameters that can be tuned to achieve optimum desired performance from the monitor, shown in Table 4-2.

Table 4-2: Adaptive safety monitor tuneable parameters, limits, and current values.

Parameter	Current Value	Lower Limit	Upper Limit
Long-Term Sample Window	20 s	10 s	50 s
Short-Term Sample Window	0.1 s	0.1 s	10 s
Driver Fault Delay	1 s	1 s	3 s
Long-Term Adaption Rate	5 Nm/s	0.5 Nm/s	10 Nm/s
Short-Term Adaption Rate	25 Nm/s	5 Nm/s	25 Nm/s
Maximum Long-Term Offset	200 Nm	50 Nm	300 Nm
Maximum Total Offset	400 Nm	50 Nm	500 Nm
Minimum Long-Term Offset	-200 Nm	-50 Nm	-300 Nm
Minimum Total Offset	-450 Nm	-50 Nm	-500 Nm
Upper Error Threshold	90 Nm	90 Nm	90 Nm
Lower Error Threshold	-90 Nm	-90 Nm	-90 Nm

An automated design-of-experiments process has been conceived to improve the performance of the adaptive safety monitor for a particular nominal torque error dataset. A controlled experiment takes place where a representative nominal torque error input is captured with known faults injected. The ideal offset torque error output of the adaptive safety monitor is then defined *a priori*. The ideal torque error output is typically what an expected response from a perfectly tuned adaptive safety monitor would yield, which is the elimination of all noise factors, capturing just the known faults at the correct magnitude, but with the inclusion of a valid driver delay. In this example, using the nominal torque error input taken from Figure 4-7, the accompanying ideal offset torque error is overlaid in Figure 4-11.

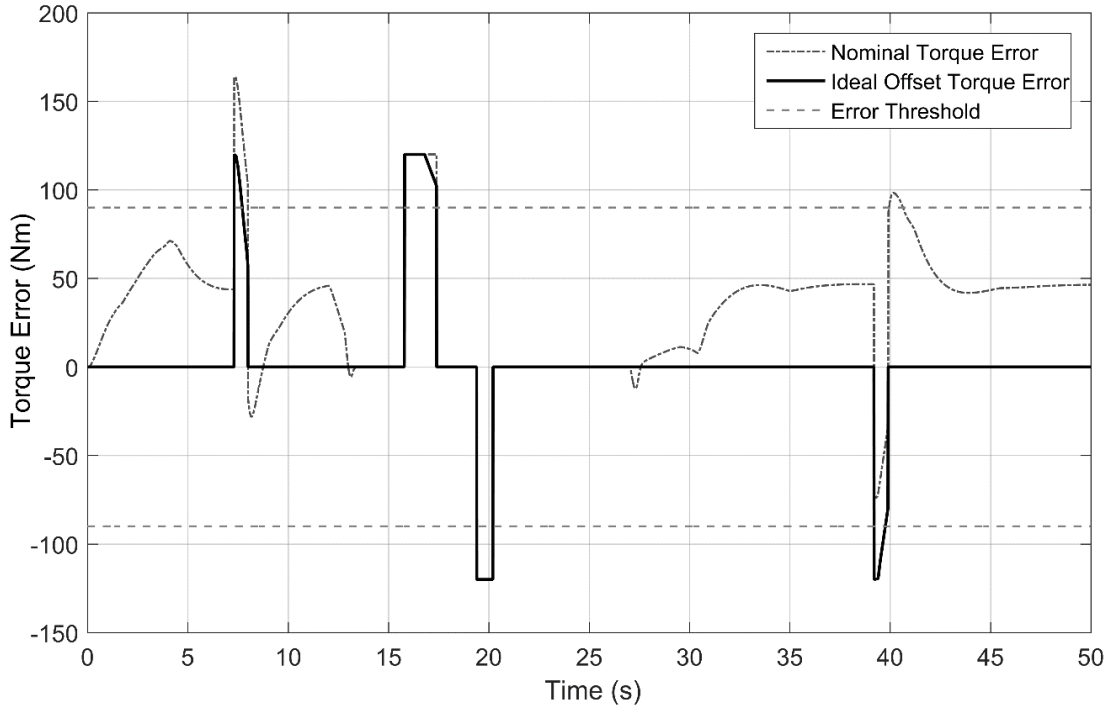

Figure 4-11: Ideal offset torque error output for the nominal torque error input, used for parameter calibration and optimisation testing.

The design-of-experiments consists of a space-filling Hobol sequence of parameter combinations cases (using parameter limits from Table 4-2), which an automated routine will use to individually test and score each case with. Some rules are used to filter out nonsensical cases, for example where short-term sample window is larger than long-term sample window. Each case is tested using the same nominal torque error, and the output offset torque error ($\varepsilon_{\tau so}$) is compared to the ideal output ($\varepsilon_{\tau io}$). A simple metric, called the torque error difference integral, *S*, is used to score the performance of each test case, shown in Equation (4-7).

$$S = t_{sample} \times \sum |\varepsilon_{\tau so} - \varepsilon_{\tau io}| \tag{4-7}$$

A lower *S* for a particular case indicates the offset torque error better matches the ideal torque error output, and multiplying the sum by the sample time t_{sample} ensures a normalised output into terms of torque error seconds (Nms); with the current parameters, an *S* score of 157.5 Nms is attained. From 8000 initial cases, 4160 were valid for testing. The cases were then sorted by ascending *S* score, yielding *S* scores ranging from 1278 Nms to 142.8 Nms in Figure 4-12.

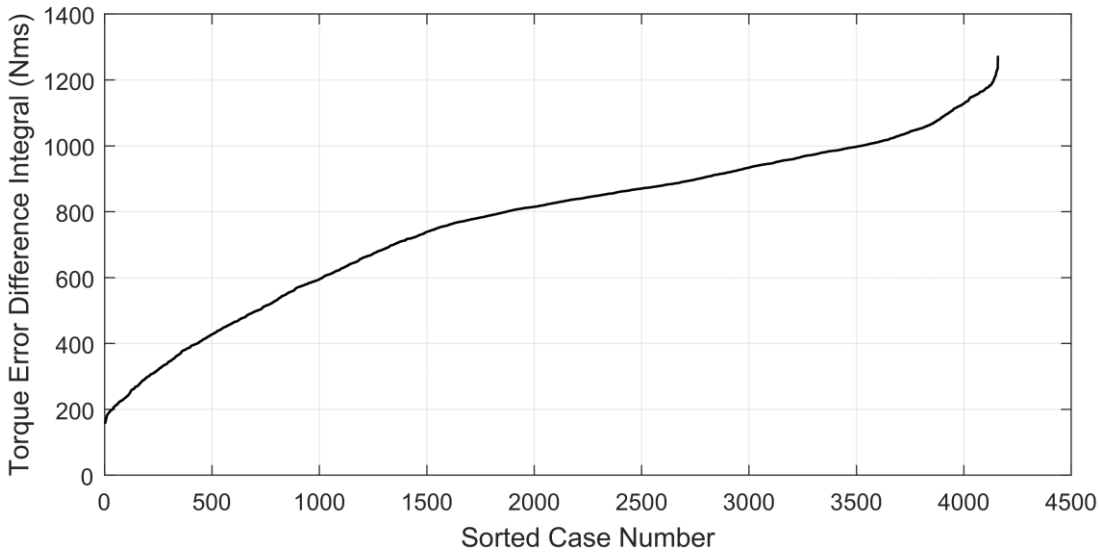

Figure 4-12: Parameter combination cases, sorted by ascending S value

The best performing parameter combination improves the performance of the adaptive safety monitor by 9.3% on this particular torque error dataset. The results of this test are shown in Figure 4-13, using the updated optimal parameter values seen in Table 4-3.

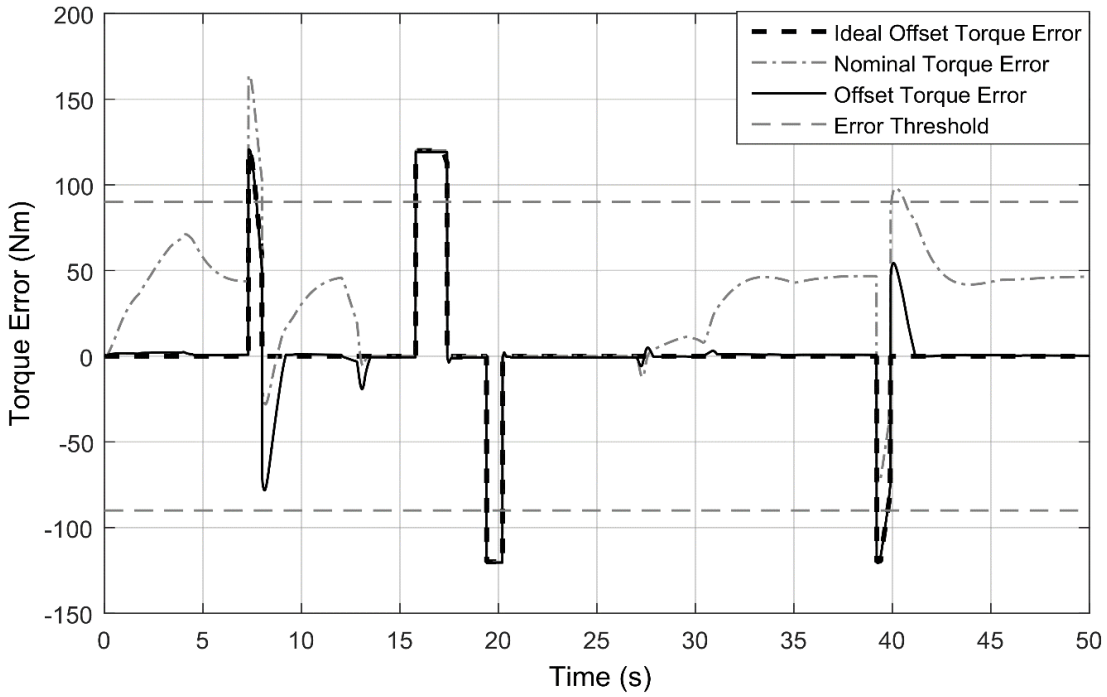

Figure 4-13: Offset torque error output using parameters from best-performing case, compared to nominal torque error and ideal offset torque error.

The updated optimal parameter values are shown in Table 4-3.

Table 4-3: Updated parameter values and change over previous.

Parameter	Old	New	Change
Long-Term Sample Window	20 s	26.3 s	+ 6.3
Short-Term Sample Window	0.1 s	0.1 s	0
Driver Fault Delay	1 s	1.37 s	+ 0.37
Long-Term Adaption Rate	5 Nm/s	9.99 Nm/s	+ 4.99
Short-Term Adaption Rate	25 Nm/s	22.89 Nm/s	- 2.11
Maximum Long-Term Offset	200 Nm	211 Nm	+ 11
Maximum Total Offset	400 Nm	290 Nm	- 110
Minimum Long-Term Offset	-200 Nm	-224 Nm	- 24
Minimum Total Offset	-450 Nm	-402 Nm	+ 48

The main areas of improvement are immediately following Fault 2 and Fault 3, where the offset torque error overshoots seen in Figure 4-7 are reduced. Most of this can be attributed to the larger driver fault delay values as the adaptive safety monitor does not begin adapting until just before the end of Fault 2. Faults 1 and 4 still experience an overshoot after the fault finishes due to the fact that the short-term offset stops updating at the start of the fault to allow the fault to exceed the error threshold; when the fault finishes, however, the creep controller had already started to counteract the fault while the short-term offset doesn't adapt, leading to the overshoots in both nominal and offset torque error This is clear when examining the difference between nominal offset torque error immediately preceding and immediately following each fault, as the difference is largely the same, the small difference being attributed to the long-term offset adaption (which still slowly adapts during faults). Marginal gains are also realised with reduced variation as the creep controller and nominal torque structure are switched between (around 13 s and 27 s). With the sorted cases, the trends and effect of each parameter can be examined to give a better understanding of how best to select values in the future. Due to the highly noisy data produced this way, the sorted parameters values are filtered using Savitzky-Golay filtering for qualitative analysis. Savitzky-Golay filtering was chosen for its ability to preserve low frequency attributes while removing noise and high frequency effects [119]. Figure 4-14, Figure 4-15, and Figure 4-16 show the results of this exercise, grouping parameters with like units together.

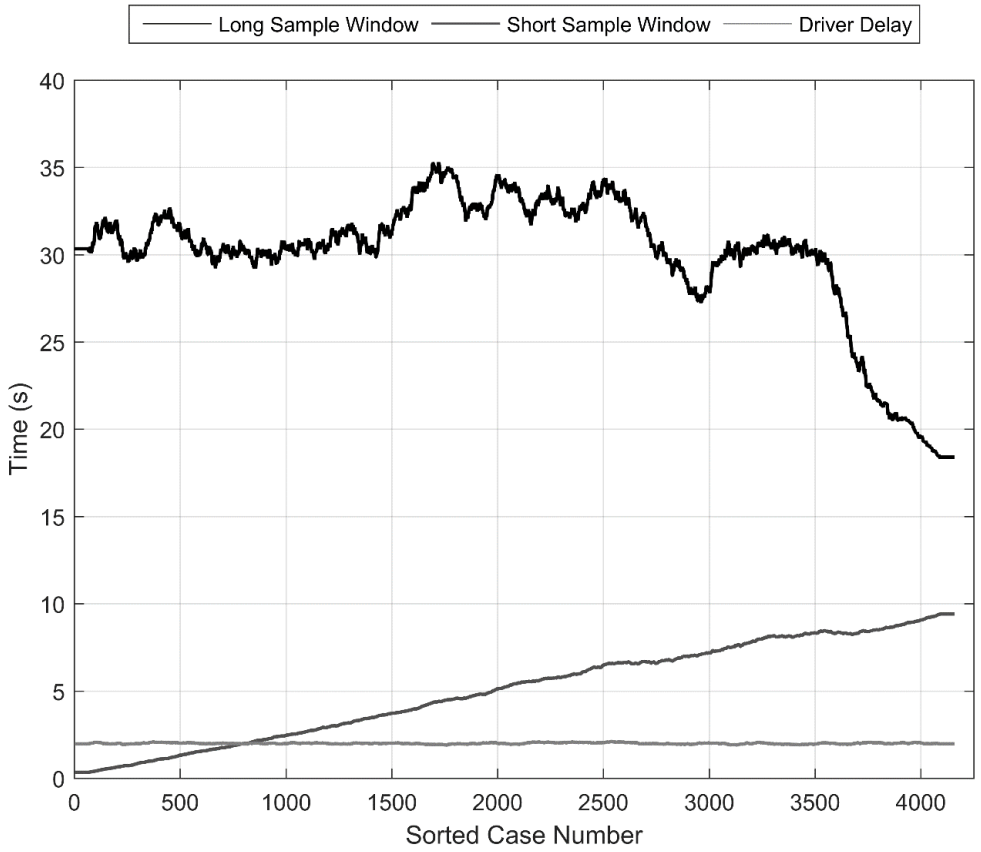

Figure 4-14: Savitzky-Golay filtered 'time' based parameter values, sorted by ascending S value.

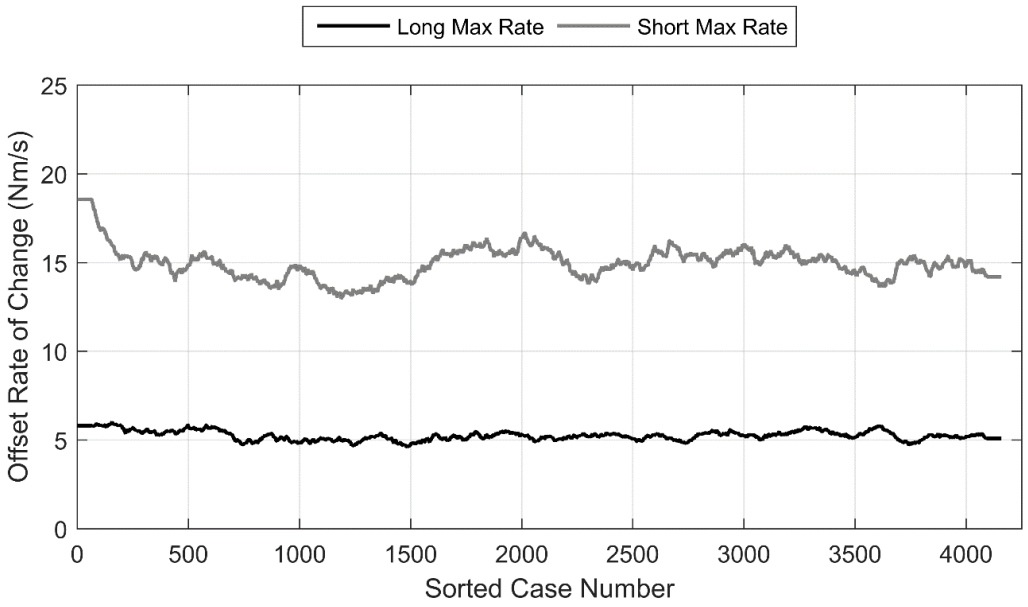

Figure 4-15: Savitzky-Golay filtered 'torque error rate-of-change' based parameter values, sorted by ascending S value.

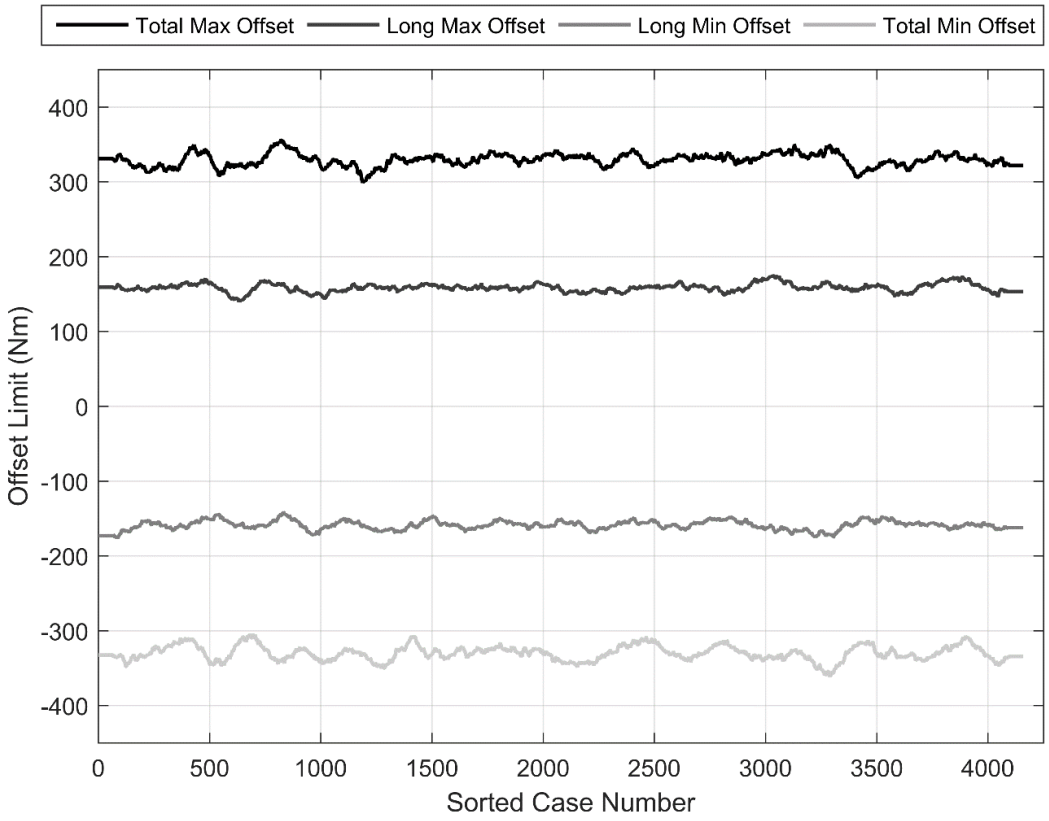

Figure 4-16: Savitzky-Golay filtered 'torque' based parameter values, sorted by ascending S value.

The parameter with the clearest trend is the near-linear relationship between a very low Short-Term Sample Window and the lowest *S* scores in Figure 4-14. Meanwhile, the lowest Long-Term Sample Window values tend to show the opposite effect by corresponding to high *S* scores, which relates to the increase seen with the optimised parameter value over the previous one. Driver-delay does not seem to make much difference in this test, but may yield different results when faced with a different dataset. A minor trend is noticeable initially in Figure 4-15 when using higher Short-Term Adaption Rate values to improve *S* score, as does Long-Term Adaption Rate, though not to the same degree; it may be the case that when presented with a larger fault over a longer period, higher Long-Term Adaption Rate has a more significant effect on performance. This highlights the importance of choosing a representative dataset from the application target to perform these tests on. Finally, none of the offset limits in Figure 4-16 seemed to show any clear trend, but again this is likely due to the fact none of the offset limits were really reached in this set of experiments, and a different application could find these parameters have a stronger effect on the outcome.

4-4 – Conclusions

A novel functional safety concept called the two-stage adaptive safety monitor was introduced in this chapter, seeking to address the issue of safety software complexity by enabling a simpler safety software model to be used with an adaptive element in the safety monitor, overcoming many of the drawbacks that would otherwise prevail, primarily robustness against false faults.

A performance analysis of this concept was carried out on an electric vehicle powertrain simulation, where an active creep controller found in the functional software was omitted in the safety software for reduced cost and increased transferability. The adaptive safety monitor yielded very good results when the functional software encountered large faults, while also preventing many false fault flags from limiting powertrain availability when small faults were encountered. This was tested on both a flat surface, and on uphill and downhill inclines, and showed to be a large improvement over using the simplified safety software model alone. A parameter calibration and optimisation method was introduced, using an automated design-of-experiments process to tune the parameter values in the adaptive safety monitor for a given drive cycle. Parameter performance trends were also identified, and further developments were suggested for a more comprehensive automated parameter optimisation process, such that performance is improved over all operating conditions, and parameter calibration costs are further reduced.

The adaptive safety monitor shows great promise in the early stages of the safety lifecycle, but will still need full validation and verification ahead of product delivery. Furthermore, a full assessment of possible applications the adaptive safety monitor should be done to explore the full feasible simplification of the safety software. Part of the reasoning behind the use of the adaptive safety monitor is attributed to having the driver in-the-loop as a controller of the powertrain. Rapidly developing autonomous vehicle technology will see the driver largely taken out of the greater powertrain control, but the simplicity of the adaptive safety monitor means that it could likely be utilised in many other safety-related applications, even outside automotive applications.

In the next chapter, the novel safety monitoring concept based on PCA is explored.

Chapter 5

Principal Component Analysis Safety Monitor

5 – Summary

In this chapter, Principal Component Analysis (PCA) is investigated as a potential functional safety concept for software safety monitoring in a passenger vehicle. PCA, using Hotelling's T^2 statistic is a well-established fault detection method employed in other industries. Here, a Local PCA concept with an automated PCA model derivation process is developed. PCA models of a nonlinear control software function are generated using only its inputs and outputs. These models are then stored and used in real-time as a safety monitor to estimate torque error in the control software, demonstrated through a vehicle simulation. This concept is shown to have potential as a functional safety monitoring tool within its current limitations.

- **Objective 4:** Conduct detailed investigation into candidate concept.

5-1 – Principal Component Analysis

With the computational and telemetric ability to capture large amounts of data from a system during development stages, the area of Statistical Process Control (SPC) has expanded as a viable control technique [120]. One of the most well-known and widely utilised SPC methods is that of Principal Component Analysis (PCA). PCA is a statistical technique based on variation between variables within a dataset. Traditionally, PCA is used to analyse a set of data of variables with possible underlying correlations, and rotate the data such that they become orthogonally uncorrelated (or at least less correlated) [120]. It seeks to fit a linear model to the training data, centred on the mean of all the recorded data points and with new orthogonal axes. The first principal component (PC) is found along the semi-axis of greatest variation in this data, the second PC along a remaining orthogonal semi-axis with the next greatest variation, and so on. The magnitudes of variation along the semi-axes are described by the eigenvalues of the data, and the semi-axes themselves are described by the corresponding eigenvector coefficients. The eigenvectors and eigenvalues are calculated using Singular Value Decomposition (SVD) [121]. The eigenvalues are arranged in decreasing order in the diagonal eigenvalue matrix Λ, and the corresponding eigenvector coefficients arranged in column format in the eigenvector matrix **V**; it is these matrices that describe the PCA model.

Using a training dataset **X**, represented as an n x m matrix, where n is the number of data samples, and m the number of variables; $\mathbf{X}_{i,j}$ would be the value associated with the j^{th} variable of the i^{th} data sample x. The first step of PCA is to find the mean of each column (variable) of **X**, denoted by $\overline{\mathbf{X}}$ [122].

$$\overline{\mathbf{X}} = \frac{\sum_{i=1}^{n} \mathbf{X}_i}{n} \tag{5-1}$$

Next, the covariance of each dimension needs to be determined:

$$cov(\mathbf{X}, \mathbf{Y}) = \frac{\sum_{i=1}^{n} (\mathbf{X}_i - \overline{\mathbf{X}})(\mathbf{Y}_i - \overline{\mathbf{Y}})}{(n-1)} \tag{5-2}$$

and the covariances assembled into the covariance matrix, **C**:

$$\mathbf{C}^{nxn} = \left(c_{i,j}, c_{i,j} = cov\left(Dim_i, Dim_j\right)\right) \tag{5-3}$$

An example of **C** is shown below for a 3-variable system (i.e. $m = 3$), with original variables x, y and z:

$$\mathbf{C} = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \tag{5-4}$$

The next step in the PCA analysis is obtaining the eigenvectors and eigenvalues from the covariance matrix using SVD, whose theory follows as below [121]:

$$\mathbf{A} = \mathbf{USV}^T \quad \text{and} \quad \mathbf{A}^T = \mathbf{VSU}^T \tag{5-5}$$

Where **A** is some n x m matrix, **U** is the eigenvector matrix of $\mathbf{AA}^T$, **V** the eigenvector matrix of $\mathbf{A}^T\mathbf{A}$, and **S** the matrix with the square roots of their eigenvalues along its diagonal (though it is not square). Then,

$$\mathbf{A}^T\mathbf{A} = \mathbf{VSU}^T\mathbf{USV}^T \tag{5-6}$$

Since **U** and **V** are eigenvector matrices, by definition their columns consist of orthonormal vectors, meaning they are square orthogonal matrices, and therefore satisfy the following condition:

$$\mathbf{U}^T = \mathbf{U}^{-1} \tag{5-7}$$

Thus, (5-7) can be re-written:

$$\begin{aligned} \mathbf{A}^T\mathbf{A} &= \mathbf{VSU}^{-1}\mathbf{USV}^T \\ \mathbf{A}^T\mathbf{A} &= \mathbf{VSISV}^T \\ \mathbf{A}^T\mathbf{A} &= \mathbf{VS}^2\mathbf{V}^T \\ \mathbf{A}^T\mathbf{A} &= \mathbf{V}\Lambda\mathbf{V}^T \end{aligned} \tag{5-8}$$

If we equate $\mathbf{A}^T\mathbf{A}$ to the covariance matrix **C**:

$$\mathbf{A}^T\mathbf{A} = \mathbf{C} \tag{5-9}$$

Then the eigenvalues λ and eigenvectors v are obtained using the following equation:

$$\mathbf{C}v = \lambda v$$
$$(\mathbf{C} - \lambda\mathbf{I})v = 0$$
$$|\mathbf{C} - \lambda\mathbf{I}| = 0 \quad (5\text{-}10)$$

Once the relevant λ and v are obtained, **V** and Λ can be derived. **V** is the matrix consisting of eigenvectors corresponding to **C** in the columns, and Λ the matrix of eigenvalues corresponding to the eigenvectors of **C**, a square matrix with the eigenvalues λ along the diagonal. The columns of **V** and Λ are then ordered in descending order of eigenvalue magnitude. The i^{th} PC is described by the eigenvector in the i^{th} column of **V**, and the i^{th} eigenvalue in the i^{th} column of Λ.

5-1.1 – PCA as a Fault Detection Concept

Jolliffe [120] discusses two key methods typically used with SPC to identify whether a particular data sample *x* would fall within the known limits of variation in the PCA model, or if it is instead classified as an outlier (an indicator of faulty behaviour). The first is Squared Prediction Error (SPE, hereafter *Q*) which was introduced by Jackson and Mudholkar [123] as a means to approximate control limits, based on the distribution of *Q*. Jackson and Hearne [124] offer a variant of this process to detect groups of outlying new datapoints, rather than single outliers. The other key method is called Hotelling's T^2 statistic, introduced by H. Hotelling in 1931 [125]. It is a multivariate statistical method that is the generalized counterpart of Student's t-test [120]. The equations for T^2 and *Q* are shown below [126]:

$$T^2 = x\mathbf{V}(\Lambda)^{-1}\mathbf{V}^T x^T \quad (5\text{-}11)$$
$$Q = x(\mathbf{I} - \mathbf{V}\mathbf{V}^{\mathbf{T}})x^T \quad (5\text{-}12)$$

If a new data sample *x* does not closely match the PCA model – i.e. it exhibits unusual variation relative to the model – the T^2 and *Q* statistic values will increase. In short, T^2 provides information on unusual variation *within* the model-space, whereas *Q* provides information about unusual variation *outside* the model-space [127]. For example, if the original dataset **X** is three-dimensional, and after PCA analysis it is decided that only two PCs can describe 99% of the data, the model-space can be reduced to two dimensions within a 3-D dataspace; *x* will still be described by the three coordinates in the dataspace. If a new sample *x* falls within the 2-D plane that PC1 and PC2 describe in the 3-D dataspace, but falls far away from the normal data, then there will be a large T^2 value. Compare this to a data sample that falls outside the

PC1-PC2 plane, in ‘unmodelled’ space, a variation which would instead be captured by a Q value. Figure 5-1 illustrates the difference between T^2 and Q.

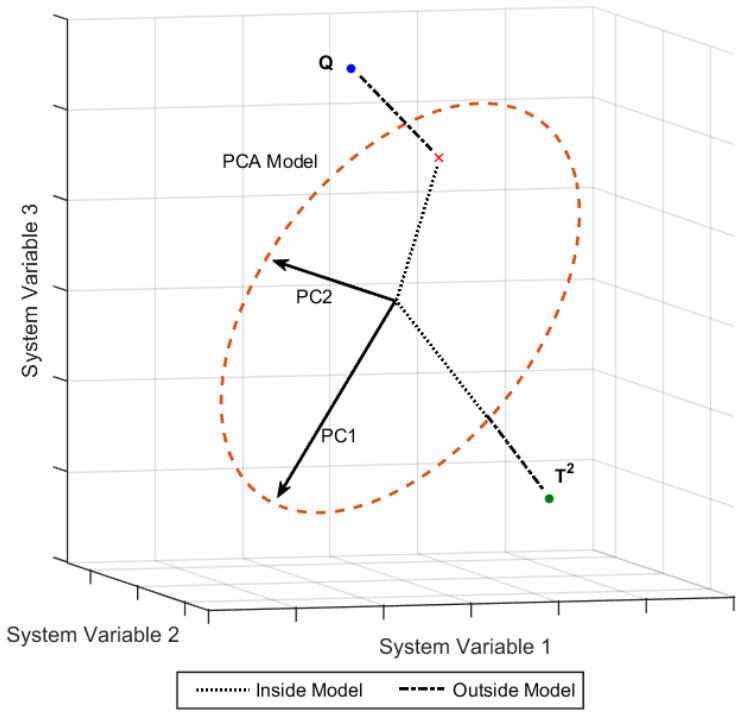

Figure 5-1: T^2 and Q statistics on 2D PCA model in 3D data space.

PCA-based fault detection concepts have been used in many industrial processing applications. Penha et al. [127] discusses how PCA was used to model the normal operation of five water temperature sensors in a nuclear reactor. The model was then used to robustly identify faulty temperature sensors, through both the T^2 and Q residuals. Hao et al. [128] looked at multiple variants of PCA-based fault detection strategies as part of a concept for detecting faults during non-steady transient operational periods in a manufacturing plant, and found their particular Longitudinal-Standardization PCA-based Adaptive Confidence Limit method to be very effective in both capturing faults and being robust against false alarms.

For application as a functional safety monitor, large amounts of data will need to be captured to derive a representative PCA model for all operating conditions of the system. Eggett and Pulsipher [129] conducted a simulation in order to compare the performance of T^2, Q, and the Jackson and Hearne [130] variation of Q. They found that T^2 was preferable for large datasets, whereas Q performed better with small samples. Therefore, only T^2 will be considered within the scope of this investigation.

5-2 – PCA Safety Monitoring Concept Development

This investigation seeks to investigate whether PCA is a functionally suitable safety monitor concept and a viable alternative for use in a vehicular torque structure, such that the fault reaction mechanism is effective in its mitigation of a fault. The key objectives pertaining to functional suitability of the PCA concept in this chapter are:

1) Quickly detect that a fault has occurred.
2) Determine direction of the fault (positive or negative).
3) Determine the magnitude of the fault.

In this section, the stages of development of the PCA safety monitoring concept are outlined, leading to a concept that meets these objectives.

5-2.1 – PCA Safety Monitoring of Powertrain Software

For application in an online vehicle safety monitoring system, PCA could be used to rapidly develop a model of the vehicle torque structure software, or a component thereof. The process can be described by four major phases:

1) Define Scope of PCA Safety Monitor.
2) Gather training data from normal operation.
3) Derive PCA models during development.
4) Store PCA models for online fault detection.

5-2.1.1 – Define Scope of PCA Safety Monitor Concept

A modern vehicle torque structure (functional software) contains many software components and functions that are used to calculate the appropriate torque demand from accelerator pedal and other inputs. Some of these include external factors such as vehicle speed and ambient air temperature; others compensate for powertrain dynamics, such as actuator or battery temperature performance characteristics; other examples include active control and/or deliberate intervention functions such as stability and traction control, driver-selectable pedal modes, high-voltage recharging control etc. The scope of the PCA Safety Monitor needs to be set such that inputs and outputs and their operating ranges are defined, such that comprehensive testing can take place.

5-2.1.2 – Gather Training Data

The PCA Safety Monitor Concept needs to know the normal operation of the nominal torque structure, or a particular section thereof, in order to create the appropriate PCA models. The functional software acts as a black box for the PCA concept, as only the inputs and output are recorded for training data. The training data should be comprehensive in that the full range of possible input values are tested to fully capture the functional software operating space. A second set of training data is also captured with an artificial fault injected into the output signal, which is used to aid derivation and test performance of the PCA fault detection during development. The input and output signals will become the original variables during the PCA model generation process, where each data point x will exist in the m-dimensional dataspace.

5-2.1.3 – Derive PCA models in development

This stage in the concept is the focus of this chapter. Here, the normal and faulty training data is used to derive PCA models based on the original variables. The PCA models seek to fit the normal data with little variation under normal operating conditions, but must be able to detect a fault if it causes unusual variation in the output based on the inputs. The PCA models must be tested, validated and verified in this stage.

5-2.1.4 – Store PCA models for online fault detection

With the PCA models defined, they are stored and used as reference for online execution when the vehicle is deployed. The PCA concept will compare newly measured data samples to the stored PCA models to monitor the behaviour of the inputs and output of the functional software components, and flag a fault if unusual variation occurs along with information regarding that fault.

5-2.2 – Linear Software Function

The first stage of development is to test the PCA and T^2 fault detection capability, and demonstrate the method is viable on a simple case. The test will capture training data for two inputs and one output. This can be thought of as a torque demand pedal map, where the two inputs are accelerator pedal position percentage, p, and vehicle forward velocity, v, and the output torque demand, τ. Note that any equations describing this torque demand pedal map is not necessary *a priori* knowledge for the PCA concept, as such equations merely describe the

component that will be treated as a black box during a PCA model derivation process. Training data for this black-box component is generated using input parameter sweeps, where uniformly-distributed combinations of values for the inputs are used, the resultant outputs captured, and all the datapoints stored in **X** for PCA analysis to take place. PCA analysis produces the PCA model in the form of the eigenvector and eigenvalue matrices, **V** and Λ, which are then stored for online fault detection using T^2. Figure 5-2 shows the result of a PCA development stage on a linear torque pedal map.

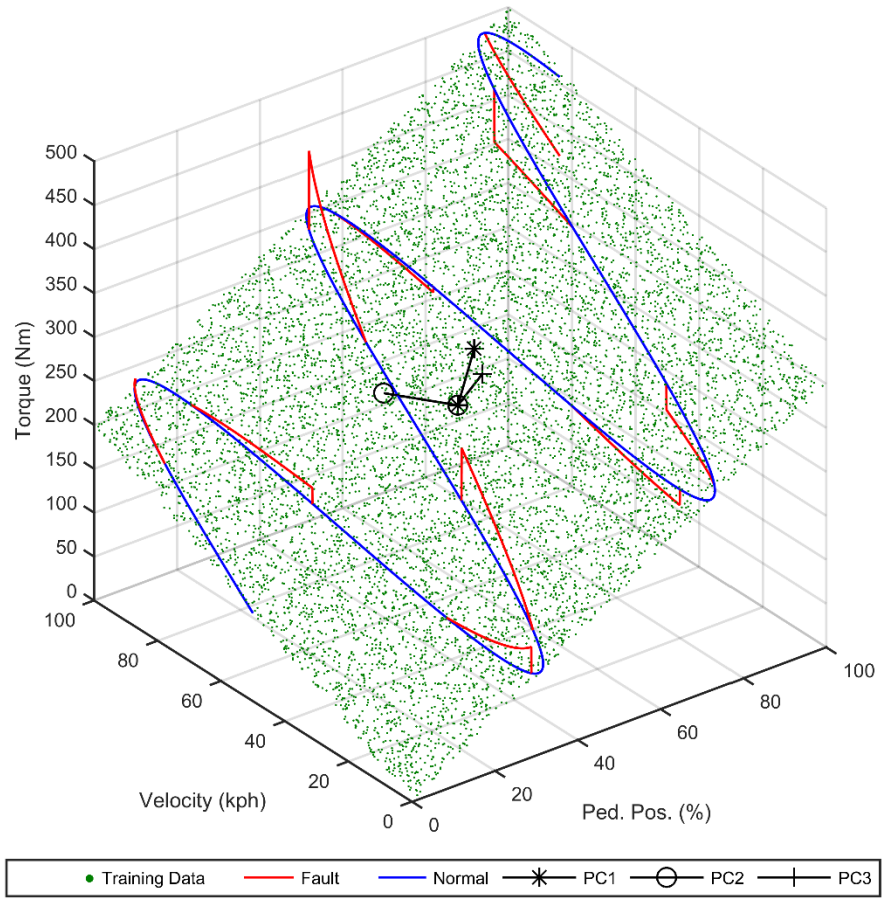

Figure 5-2: Linear PCA Training and Test Data, and Model.

In this test, pedal position percentage *p* has a range of 0-100%, and vehicle velocity *v* a range of 0-160 kph. 10000 unique input value combinations were used to generate the training data set. With this data, the PCA model is derived, and the three PCs are shown originating from the model centre in Figure 5-2. To test the online fault detection capability using this PCA model, a set of test data comprising of inputs and outputs of the software component is generated over a 100 s period, called 'normal test data'. This tests the PCA model's ability to prevent undue variation under normal operating conditions, which would otherwise cause a false fault to be flagged by the fault reaction mechanism. A second set of test data modifies

the first through the inclusion of a fault signal, a simulated 'torque error' effect additively applied to the torque output. Figure 5-3 shows the torque error signal used in these tests.

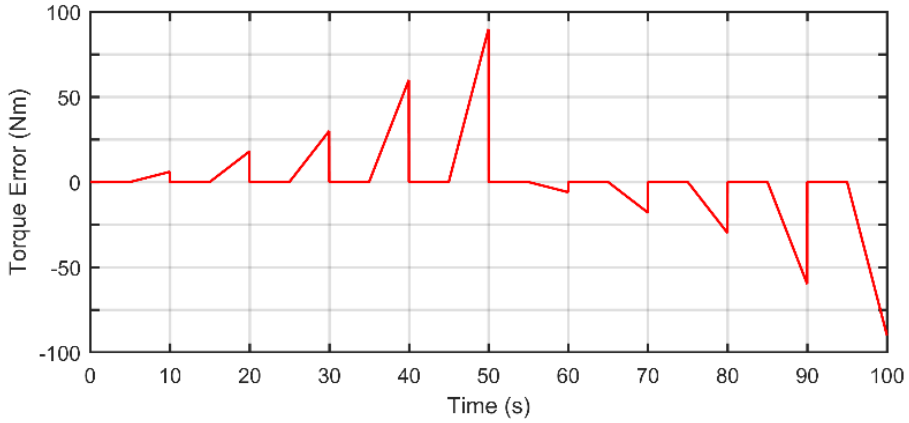

Figure 5-3: Torque Error signal

The set of normal data with this fault signal applied is called 'fault test data'. When the normal and fault test data are used for their respective T^2 online tests, the resulting T^2 outputs are generated and shown in Figure 5-4.

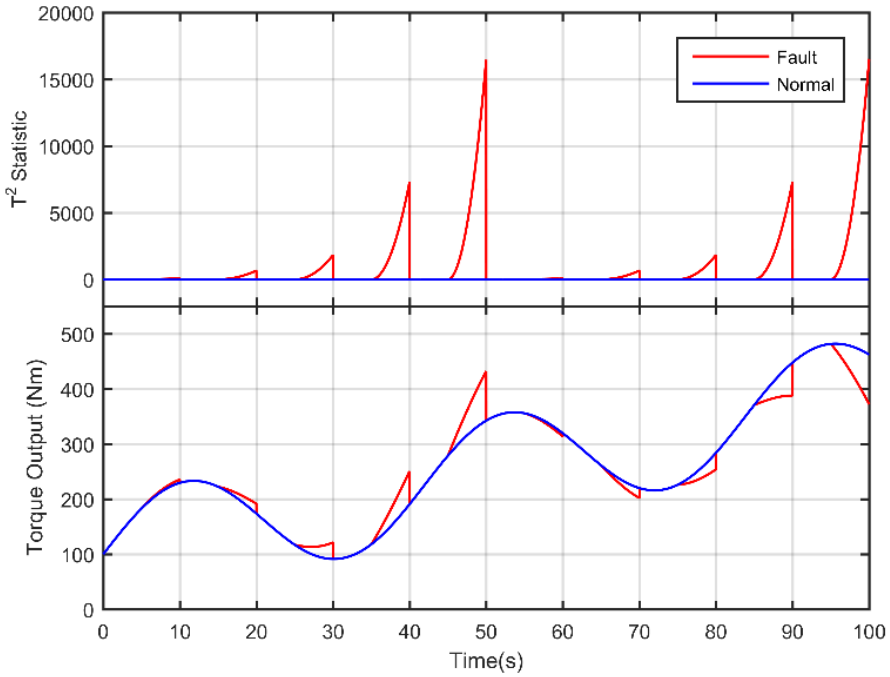

Figure 5-4: Linear PCA Online test, showing T^2 output for normal and fault conditions (top) for given torque outputs (bottom).

From Figure 5-5, it is clear that T^2 is able to reliably identify fault occurrences, and with information regarding the severity of the fault in terms of absolute magnitude, when scaled

appropriately. However, the fault T^2 tends to curve during each ramp, rather than increasing linearly as the torque error signal does, meaning that a single scalar will not accurately scale the T^2 at all points to correspond with the torque error. Another drawback here is that information relating to fault direction is not available since $T^2 \geq 0$.

5-2.2.1 – Output Data Offsetting

Currently, there is no way to know whether a given T^2 value signifies a positive or negative torque error, a metric which is required to appropriately mitigate an unintended longitudinal acceleration safety hazard, and which is the second objective of this investigation. It was found during investigation that when a positive constant a is added to the output of the T^2 online test data – but not the training data – both normal and faulty T^2 test signals have their T^2 values increased. While this seems to worsen performance due to far greater T^2 values under normal conditions, it reveals qualitative information pertaining to fault direction. Figure 5-5 shows the resulting T^2 with a constant offset, a, of 1000 Nm applied to the output (torque) of the same normal and fault test datasets as previously shown in Figure 5-2.

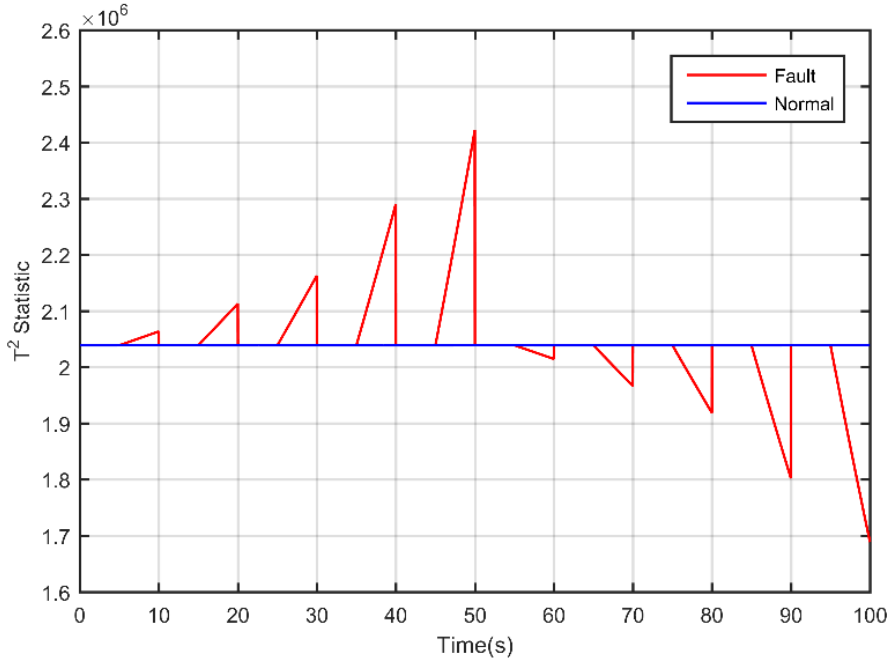

Figure 5-5: Conventional PCA Linear T^2, with output offset a = 1000 Nm applied.

Figure 5-5 now shows that qualitative information about torque direction is clearly visible, as now all positive torque faults are above the normal T^2 line at a given time, and all negative faults lie below. Additionally, there seems to be a much more linear ramp, qualitatively matching the torque error signal of Figure 5-3 more accurately and simplifying the

interpretation effort. During this output offsetting investigation, it seemed that a sufficiently large a was needed to improve this aspect, as an a that was too small would yield similar erroneous qualitative fault T^2 behaviour as seen in Figure 5-4. This is likely due to the fact that the offset needs to be greater than the expected largest absolute magnitude in Nm of the faulty behaviour, such that it doesn't cross the plane created by a, i.e. the new baseline from which T^2 calculations take place. An output offset of $a > 1000$ would work just as well.

5-2.3 – Nonlinear Software Function

In practice, most software components in the functional software are not perfectly linear. For a multiple input single output system, a curved and often non-smooth, non-linear surface is typically formed. Since PCA results in linear models centred at the training data mean, there will always be some difference between the model and the non-linear dataset in some areas. Therefore, if a single PCA model is used for the entire non-linear dataset, there will be some variation in the normal T^2 test, particularly data points further away from the centre of the dataset. Consider now some non-linear relationship in the equation describing the torque demand pedal map: with the same input data trace and fault signals used in the linear PCA T^2 tests, the training data and T^2 test data output yielded by such a non-linear equation could be as shown in Figure 5-6.

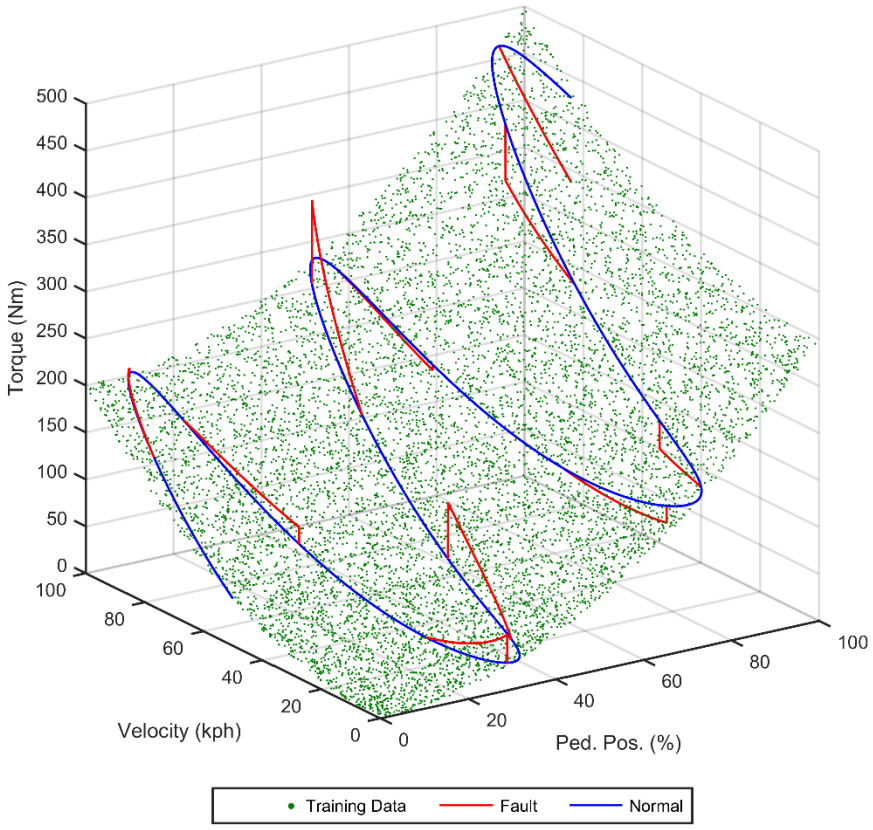

Figure 5-6: Nonlinear PCA Training and Test Data.

Figure 5-6 shows that a curved surface dataset has been attained from the new function, with the same 500 Nm range in the output axis. The curved surface will likely lead to reduced performance from the linear PCA model, as normal operating data varies with respect to the third eigenvector (the smallest one), leading to variation in normal T^2. Using the same output offset as before (a = 1000 Nm), the same T^2 online test is conducted, and the results are exhibited in Figure 5-7.

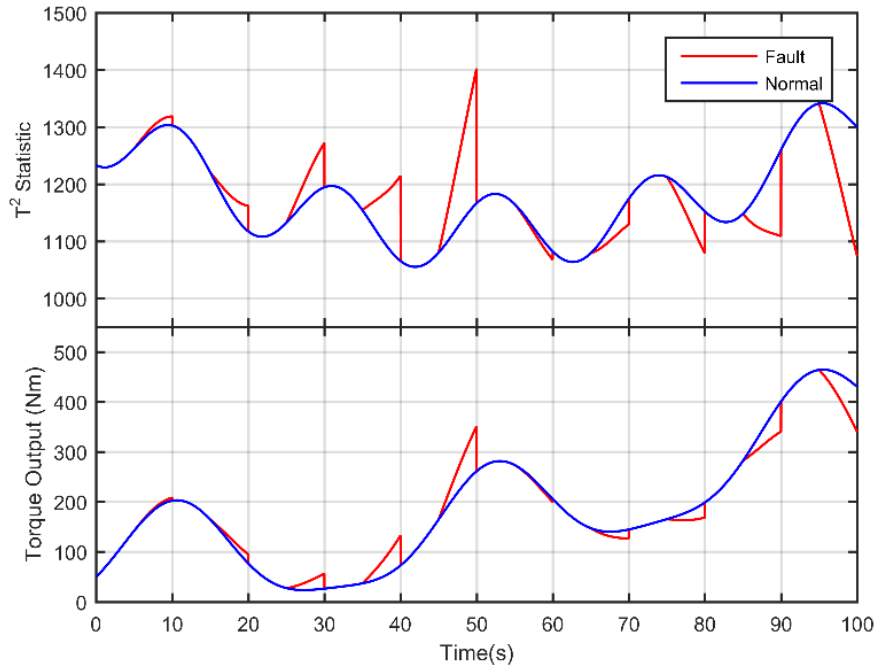

Figure 5-7: Curved training dataset, and T^2 offset with linear PCA model.

Reduced performance on this dataset is clear in Figure 5-7 as normal T^2 now varies considerably. As expected, greater normal T^2 variation will make it difficult to detect and distinguish abnormal faulty T^2 variation, leading to both many false faults being flagged, and some true faults may be missed altogether. This is all due to the fact that a single linear PCA model cannot accurately describe the nonlinear training data.

5-2.3.1 – Local PCA Concept

To remedy this, a simple concept has been devised that places multiple linear PCA models in the dataspace to fit the non-linear training dataset. The training data is segmented into cells by creating boundaries on the axes of original input variables; an individual PCA model is derived for each cell *k*. In theory, this means that each individual PCA model will be a better representation of the local training data than a single model covering the whole dataset would. This concept will be called Local Principal Component Analysis (Local PCA), and the basic theory of the concept is illustrated using a simple two-variable system in Figure 5-8.

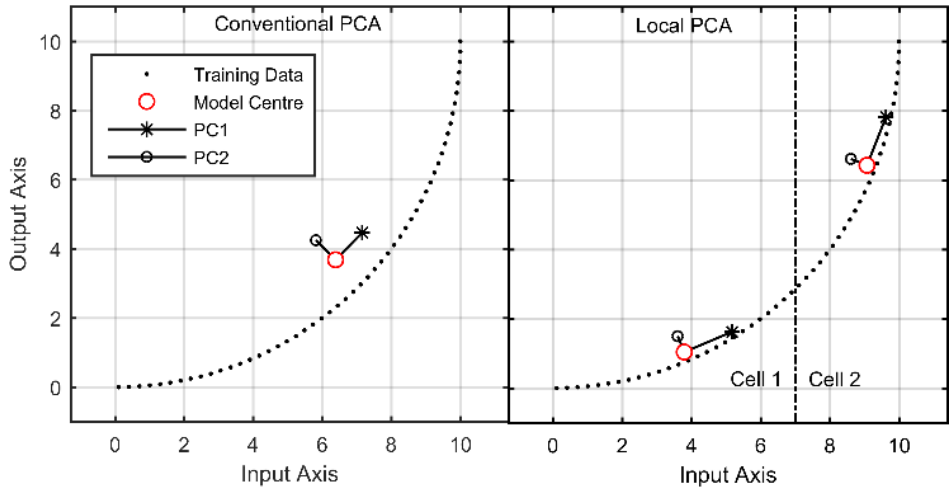

Figure 5-8: Comparing Conventional PCA (left) to Local PCA of two cells (right).

In Figure 5-8, the model centres in the Local PCA concept is found much closer to their local training data, and the individual Local PCA eigenvectors more closely resemble the data it is estimating than the regular PCA. The second eigenvalue is also much smaller, meaning that there is less variation in the 'fault direction' (output axis), and would theoretically mean better distinction between fault and normal data. Only the input axes are divided, because only faults are being checked on the outputs; if the output axis was also divided, then a fault could move the data sample x into a different cell, and could lead to a missed fault. Therefore, only the input axes are divided to create cells. Using the same non-linear data as before, and with the same a =1000 Nm applied, results of the T^2 online test are shown in Figure 5-9.

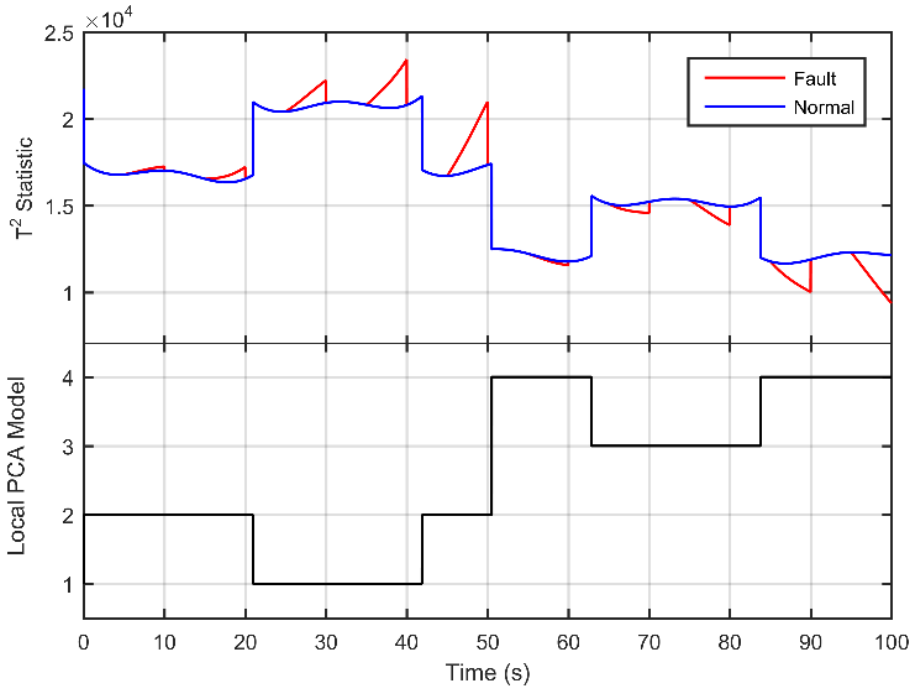

Figure 5-9: Local PCA T^2, with an offset applied.

From Figure 5-9, the Local PCA concept now causes T^2 to experience sharp discontinuities as the current data sample x passes from one cell to another; this, naturally, is due to differences in the PCA models. These discontinuities are significant enough that even large faults cannot be robustly identified. However, within each cell k, normal variation stays within a narrower relative threshold, and using the average value of normal variation as a baseline better distinguishes faulty from normal T^2 behaviour within each cell. Therefore, if the average normal T^2 value, b, during an online test was captured, it could be used offset all future T^2 values for that cell, such that each cell has the same average baseline (i.e. zero). Then a common 'normal threshold' can be used to identify faulty behaviour, regardless of which cell it occurs within. This 'average T^2 offset' b can be calculated purely from normal training data, without knowledge of fault response.

The average T^2 offset calculation is demonstrated in Figure 5-10, taking the previous result of Local PCA online test from Figure 5-9, considering only the normal T^2 values of cell 2, and finding the average of all values in that cell.

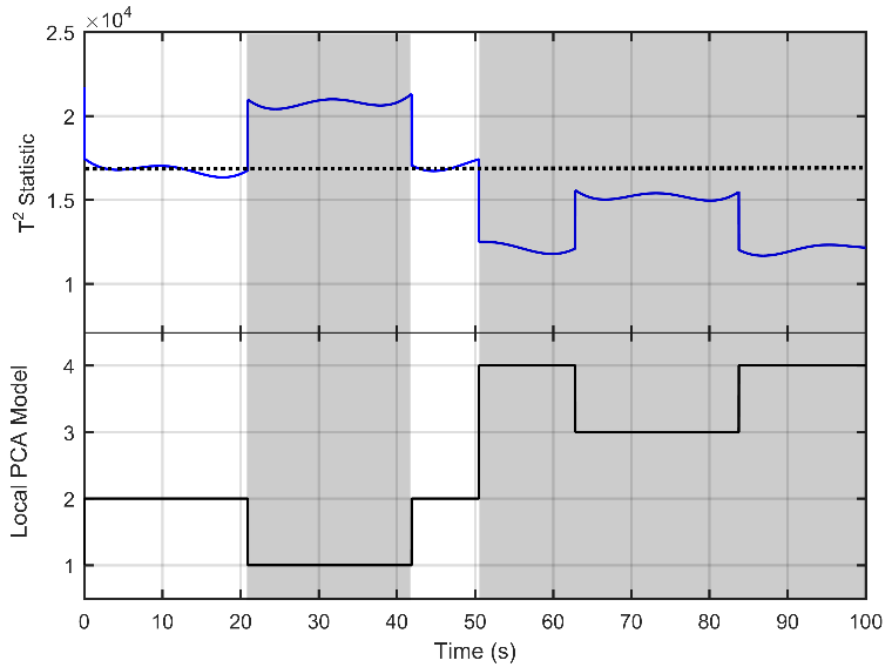

Figure 5-10: Example demonstrating how average offsets are derived a posteriori from normal T^2 online test for each cell's Local PCA model.

The average T^2 offset for cell 2, b_2, would roughly be 1.7e4 from Figure 5-10, and is shown by the dotted black line. The T^2 online test is repeated for the same normal and fault test data, but this time using the cell membership already calculated for each new data sample x (based on its input axis coordinates) and the corresponding average T^2 offset b_k is subtracted from the nominal normal and faulty T^2 outputs.

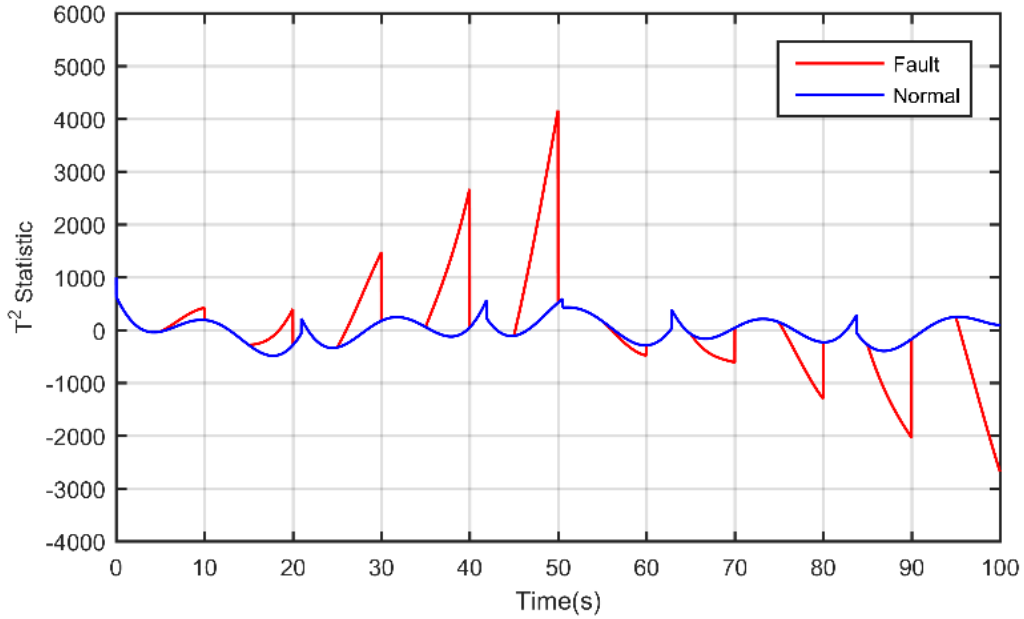

Figure 5-11: T^2 with average normal T^2 offsets b_k applied to each corresponding cell k.

Comparing Figure 5-11 to Figure 5-7, it is apparent that using average normal T^2 offset for the cells means that faults are much more easily distinguishable. Additionally, obtaining information about fault direction and magnitude is simpler, as a negative torque error (i.e. too little torque demanded) now tends to be a negative value. This is not a complete concept yet, though: the normal T^2 still varies by a considerable amount and will benefit from reduced normal variation. Moreover, a fundamental problem persists: T^2 is a qualitative statistic as opposed to a useful, quantitative value for use in fault detection. T^2 needs to somehow be interpreted into a useful metric, akin to the familiar 'torque error' typically used by the fault detection system.

5-2.3.2 – T^2 Normalisation

Although the cells now share a common baseline of zero T^2 under normal conditions, the interpretation of a given fault magnitude is not common. Therefore, there also needs to be some form of T^2 interpretation scalar, a possibility that has now been made possible with the introduction of this average normal T^2 offset. In order to select the correct scalar, a different, more comprehensive T^2 test is introduced

Instead of using a varying additive fault signal from Figure 5-3, a constant 100 Nm fault f is now applied to examine the behaviour between cells as a comprehensive input parameter sweep is conducted; this is shown in Figure 5-12.

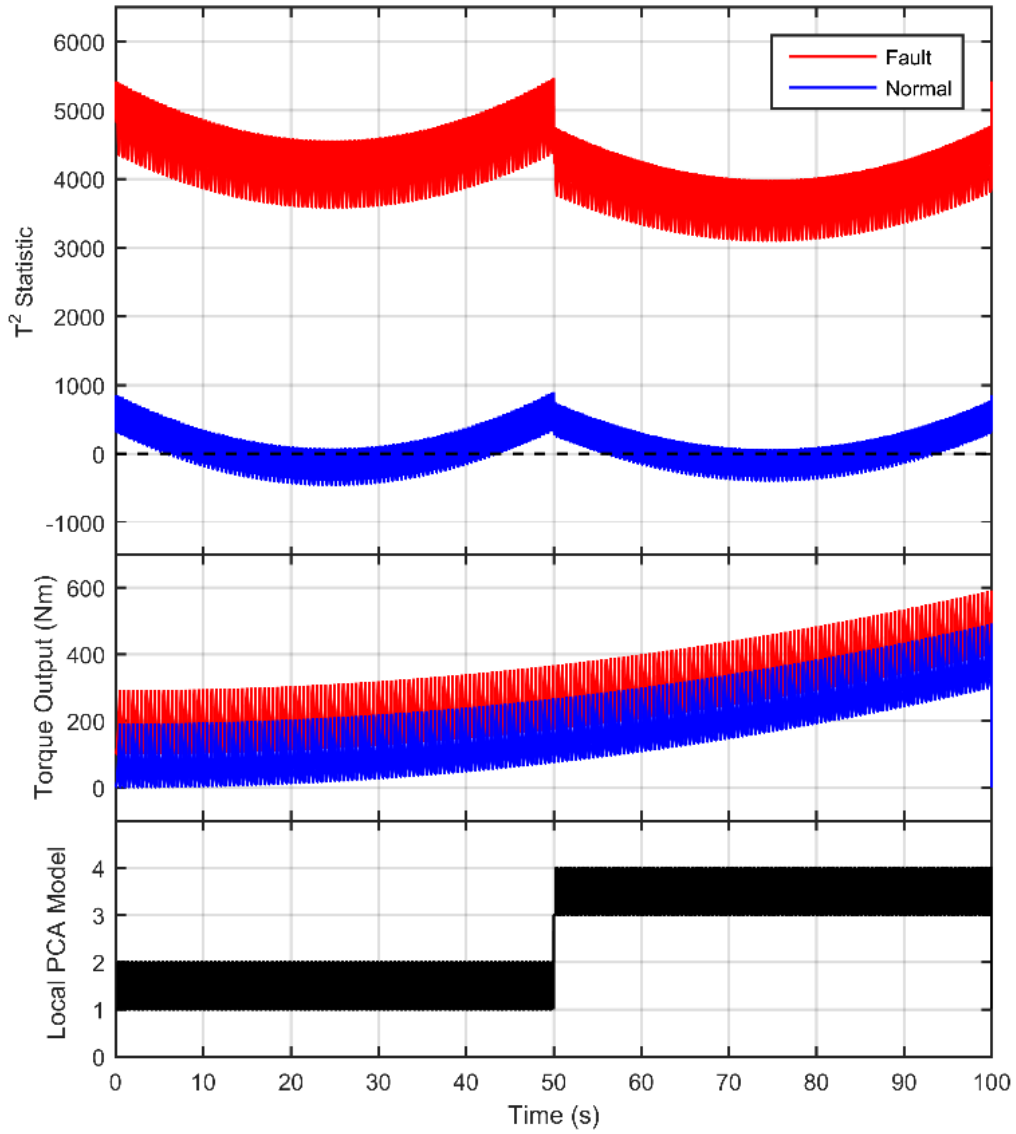

Figure 5-12: Input parameter sweep under normal conditions and with constant 100 Nm fault f on subdivided nonlinear training data.

A constrained view of the x-axis in Figure 5-12 can be found in Figure 5-13, showing more signal detail otherwise lost due to frequency.

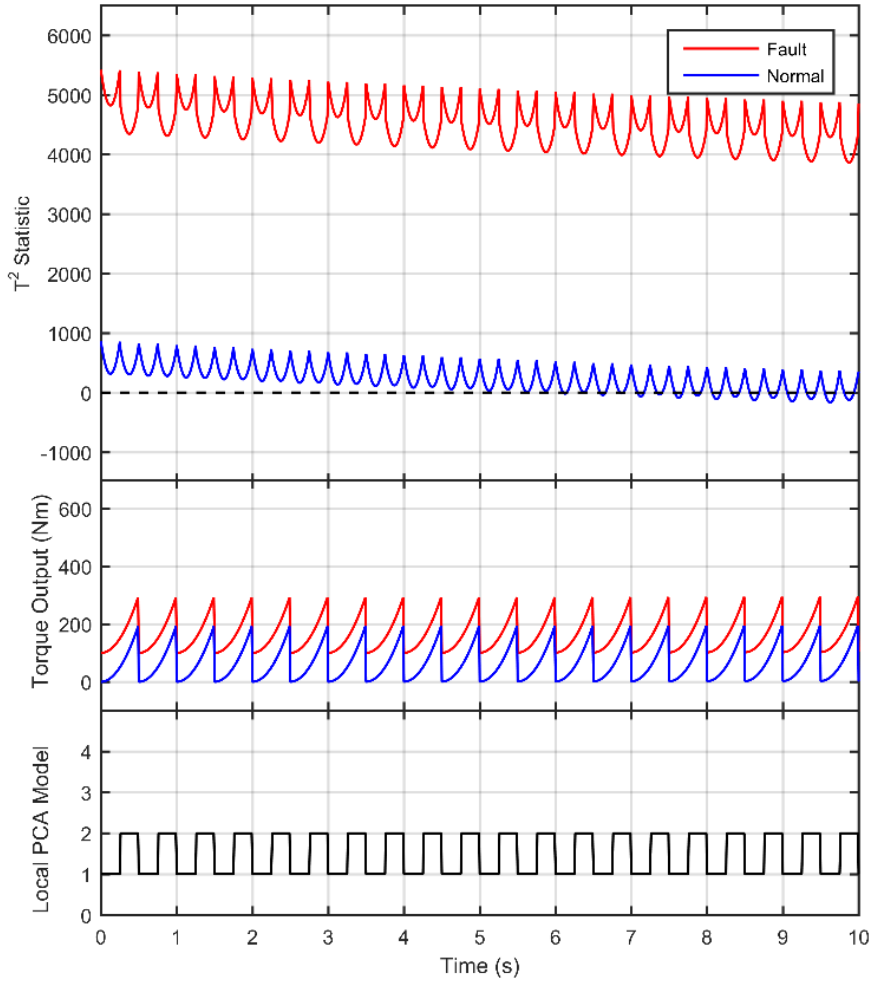

Figure 5-13: Input parameter sweep under normal conditions and with constant 100 Nm fault, f, on subdivided nonlinear training data; first 10 seconds only (see Figure 5-12).

The variation in the fault offset T^2 generally varies around some average value in the same way as normal offset T^2 varies around zero, but it is clear that Local PCA models 1 and 2 tend to produce greater T^2 outputs for the same fault compared to models 3 and 4, indicating that a single scalar for the all cells will not correctly estimate torque error. With that in mind, the average fault offset T^2 value has to be found for each cell, $\overline{T^2}_{fault,k}$, obtained by averaging all the fault offset T^2 values for a given test that is produced when cell k is active. Then, all of the T^2 values produced in each cell can be divided by the corresponding $\overline{T^2}_{fault,k}$ such as to normalise each cell's response to the same 100 Nm fault, f. Since normal offset T^2 is already centred on zero, there will not be a further offset change in normal offset T^2. The result of this is that normal offset T^2 in each cell will now vary around zero under normal conditions (as before), but now around 1 for fault offset T^2 whenever the fault magnitude equals f. With all of T^2 normalised, the $\left(\overline{T^2}_{fault,k}\right)^{-1}$ scalar for each cell can be scaled by the known fault magnitude f to produce the T^2 torque error estimate, T^2_{ETE}, a useful metric for the fault detection

and reaction mechanism to act upon. Considering all of these steps, then, a single T^2 scalar c_k can be derived for each cell k, which is used to obtain T^2_{ETE} at any given time. Deriving c_k is summarised in Equation (5-13):

$$c_k = f \times \left(\overline{T^2}_{fault,k} \right)^{-1} \tag{5-13}$$

where f is the constant fault used in the faulty T^2 test, and $\overline{T^2}_{fault,k}$ the average fault offset T^2 for cell k captured after the test. Using the same test as before (Figure 5-12), the T^2 scalar c_k for each cell k is derived and stored. When a new set of input parameter sweeps are then conducted, the T^2 output is multiplied by the corresponding T^2 scalar c_k based on which cell is active for the current T^2. This test includes the normal offset T^2 test, the previous faulty offset T^2 test from a constant 100 Nm fault f, and two more faulty offset T^2 tests – constant 50 Nm and -75 Nm faults – to provide indication whether a given derived c_k is valid for any fault injection magnitude. Figure 5-14 compares the T^2 estimated torque error, T^2_{ETE}, outputs of these tests.

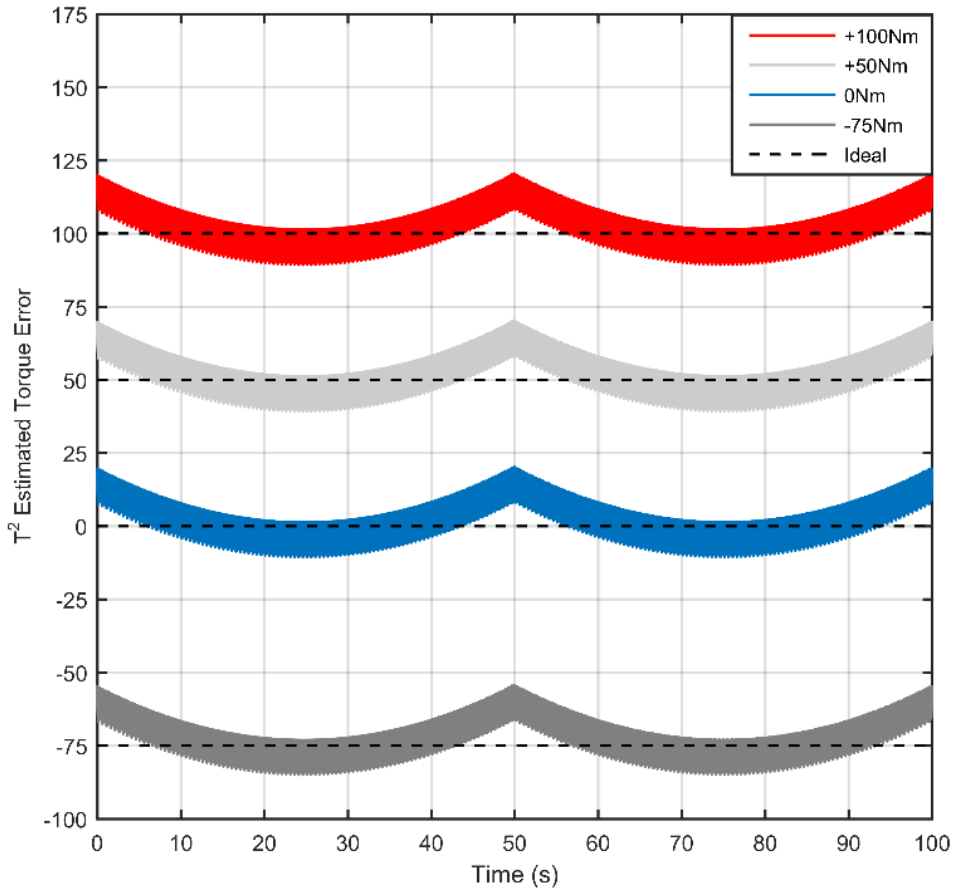

Figure 5-14: T^2 estimated torque error outputs for normal and fault data with subdivided training data for -75 Nm, 0 Nm, 50 Nm, 100 Nm values of f.

From Figure 5-14 it can be seen that the T^2 normalisation process has successfully normalised the cells in the T^2 estimated torque error outputs, such that a fault on the output will be quickly, and with reasonable accuracy, be detected by this concept. The original fault of 100 Nm is easily distinguishable from normal operation, with a variance range of -10 Nm to +20 Nm.

Furthermore, both the 50 Nm and -75 Nm faults are correctly classified (with like variance) during their T^2 online tests using the cell membership k to determine appropriate $\mathbf{V_k}$, $\Lambda_\mathbf{k}$, b_k and c_k. This important finding should indicate that, regardless of which value for f is used to derive b and c during development, these will provide a correct T^2_{ETE} for any encountered fault magnitude. In summary, the methods employed to calculate T^2_{ETE} can be described by the following equations.

$$\mathbf{M_k} = \mathbf{V_k}(\Lambda_k)^{-1}\mathbf{V_k^T} \tag{5-14}$$

$$x_a = x + [0,0,\dots,a] \tag{5-15}$$

$$T^2_{ETE} = \left[[x_a \mathbf{M_k} x_a^T] - b_k\right] \times c_k \tag{5-16}$$

where k is the determined cell membership of datapoint x, and x itself is a vector whose elements are the original dataspace coordinates, with all the inputs first, then the output at the end. Since the PCA models do not change once they are released with the vehicle, (5-14) can be pre-multiplied to produce the single [m x m] PCA model $\mathbf{M_k}$, such that a lesser burden is placed on the computational and storage resources during real-time execution. These are promising results, but variation around the ideal in both fault and normal T^2_{ETE} is still significant enough to potentially inhibit consideration of this concept as a viable fault detection method for this application. Therefore, the next section seeks to reduce this variation through the introduction of an automated refinement process.

5-2.4 – Automated Subdivision PCA

It was demonstrated in the preceding section that through the creation of divisions in the input axes of the training data such that cells of training data are produced, better matching PCA models could be found for the local training data within the cells they are being derived from, leading to reduced variation in T^2. It seems reasonable, then, to assume that further cell divisions could yield improved performance by reducing normal variation, and thereby improving the fault detection robustness of this method. However, simply dividing the cells indiscriminately will quickly lead to impractical amounts of data needing storage, exacerbated further with systems of greater dimensionality.

An automated cell subdivision process has been conceived, whereby cell divisions are tested and refined, such that improved performance can be obtained while not drastically expanding the amount of data storage required; this process has been called Subdivision PCA. The basic principle is straightforward: test if a subdivision improves the fault detection capabilities above a desired amount, and if so create that subdivision. Figure 5-15 illustrates one iteration of the subdivision refinement process.

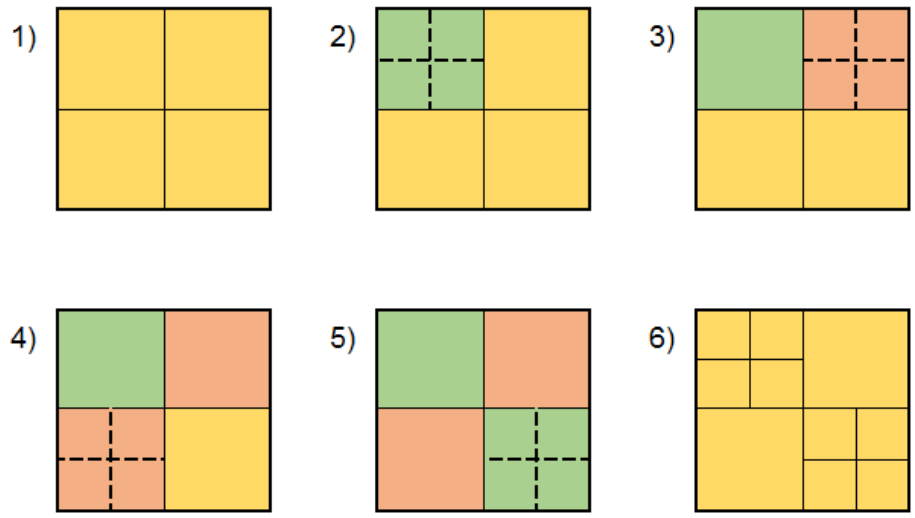

Figure 5-15: One iteration of cell subdivision in subdivision refinement process, with green cells showing earmarked cells for future subdivision.

Figure 5-15 shows the input space of a three-dimensional training data dataspace. The starting cells and their boundaries (solid black lines) for this iteration is shown in 1), and the performance of the T^2 tests stored. Then in 2) through 5), the same tests are repeated, but with new probationary subdivisions in one cell at a time (dashed lines) to determine if these subdivisions bring benefit; green-shaded cells yield improvements whereas red-shaded cells do not. Cells that show improvement then have their subdivisions confirmed, ready for the next iteration 6).

5-2.4.1 – Error Integral, Cell Selection Function and Subdivision Refinement

In order to determine which cells to subdivide, some way to quantify their performance needs to be established. One way to understand the performance of the set of PCA models during the T^2 tests is to determine how distinguishable a fault is from normal operation. Variation in either normal or faulty offset T^2 will tend to reduce this discernibility, and reduction of this variation is the main aim of the subdivision refinement process. A simple metric has been derived to

take this into account, called the 'error integral' e. With a known constant fault magnitude f being used in the T^2 online tests, the error integral e can be calculated using (5-17).

$$e = \sum \left(\left| T^2_{ETE,fault} - f \right| + \left| T^2_{ETE,normal} \right| \right) \tag{5-17}$$

With perfect PCA model matching, faulty offset T^2 will equal f and normal offset T^2 will equal zero, leading to $e = 0$; any deviation from these baselines will increase e. This is calculated for each datapoint during the T^2 online test and cumulatively summed together. The criteria for the subdivision refinement process is therefore to minimize e. With each iteration of subdivision refinement, each cell needs to be tested for improved subdivision performance. This is called the Cell Selection Function, and is comprised of the following steps:

1) Start the subdivision iteration.
2) Take an untested cell k of the current subdivision, and create a probationary subdivision (which includes all other cells of the current subdivision).
3) Derive PCA models of the whole training data using the probationary subdivision.
4) Conduct T^2 test using PCA models from probationary subdivision.
5) Derive average offsets and T^2 scalars, and apply to normal and faulty T^2.
6) Calculate error integral e for probationary subdivision, and compare with e from current subdivision.
7) If the probationary e is greater than the current e, earmark cell k for subdivision refinement in the next iteration. Otherwise, do not subdivide it.
8) Revert back to the current subdivision in step 2 until all cells are tested.
9) Subdivide earmarked cells, end subdivision iteration.

There are a number of criteria that can control the number of subdivision iterations performed, such as a minimum cell size (related to a set number of subdivision iterations), maximum number of PCA models, error integral requirement, the 'knee point' in error integral improvement, or until no improvement could be made to error integral by any further subdivision. The earmarked cells could be sorted based on percentage e improvement and only those above a certain percentage chosen for further subdivision. Figure 5-16 shows the development process flowchart to with this cell selection function.

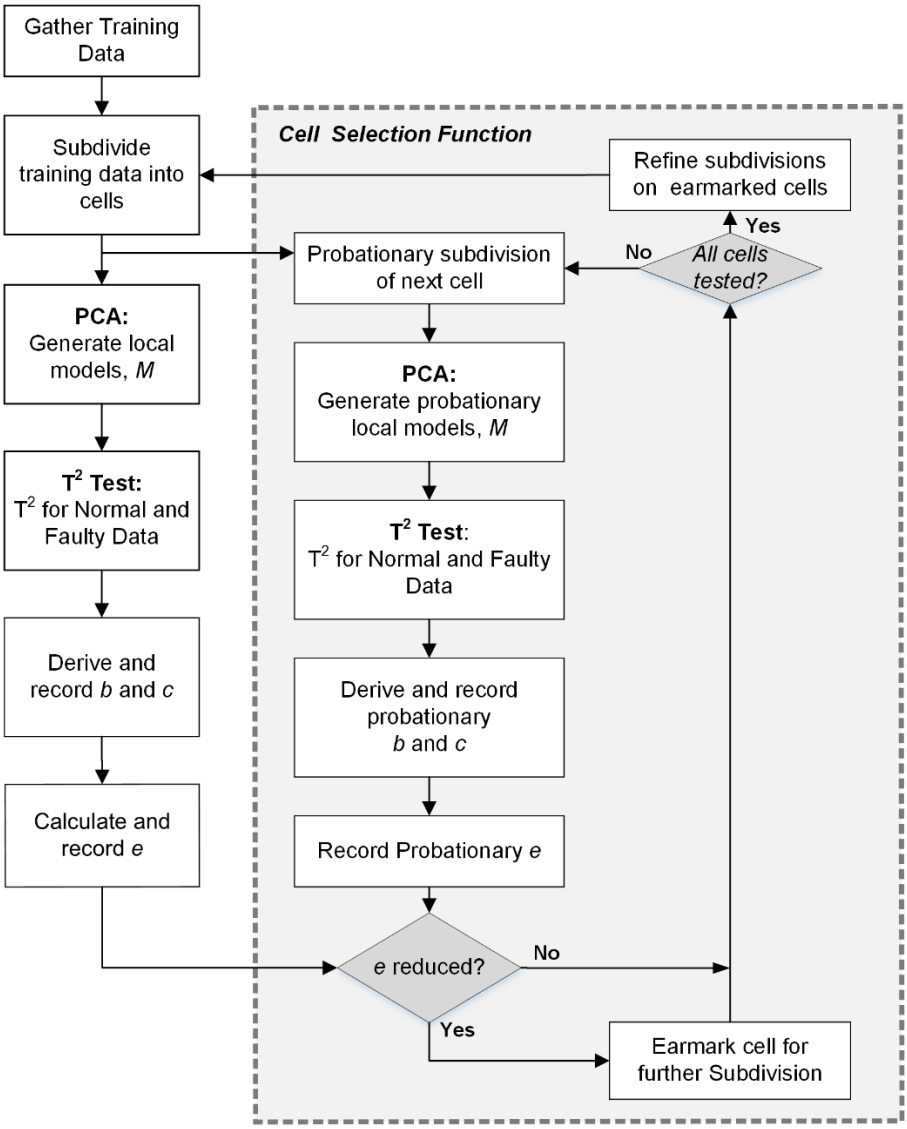

Figure 5-16: Subdivision Refinement Process with Cell Selection Function.

Continuing on with the same data from Figure 5-16, the subdivision refinement process is carried out to see if variation can be reduced. After three subdivision refinement iterations, a set of 64 cells are created with corresponding PCA models, offsets, and T^2 scalars. Figure 5-17 shows the resulting T^2 estimated torque error outputs using these PCA models.

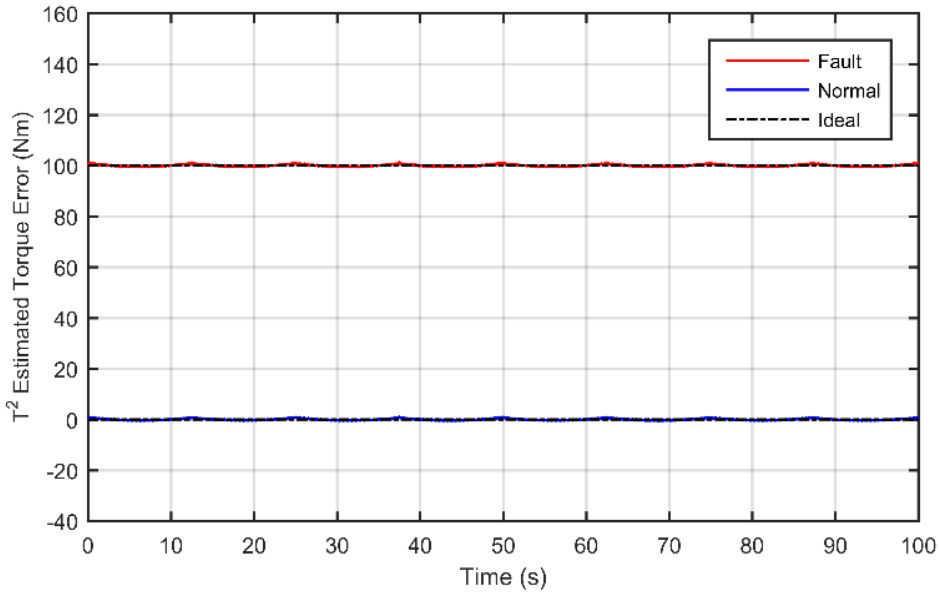

Figure 5-17: Normal and Fault T^2 estimated torque error outputs after three subdivision iterations.

With more PCA models describing the nonlinear training data due to the automated subdivision process, variation in both fault and normal T^2 estimated torque error are dramatically reduced, with a variation of >2 Nm in both signals across the whole input space.

5-2.4.2 – Weighted Subdivision

Up to this point, the only type of cell subdivision considered has been to create a new cell boundary halfway between the existing ones along a particular input axis. There is scope to investigate altering the 'weighting' of these new subdivisions, whereby additional tests are conducted to find if biasing boundaries would yield improved performance at no additional storage cost. Figure 5-18 illustrates the difference between the nominal Subdivision Refinement and Weighted Subdivision Refinement.

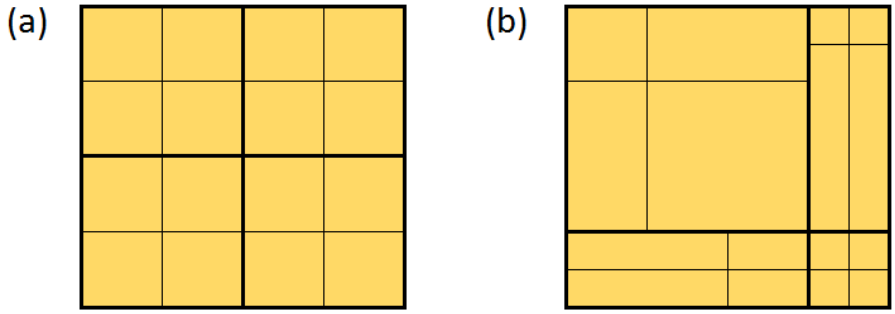

Figure 5-18: Regular Subdivision (a) compared to an example Weighted Subdivision (b) after two subdivision iterations.

The Weighted Subdivision Refinement process is also easily automated, and included as a subroutine within the Cell Selection function of Figure 5-16. When the next cell is tested, all allowed combinations of boundary weighting are tested individually, and the best performing one (with the lowest *e*) is compared to the current subdivisions, and if *e* has been reduced, then that cell is earmarked with the corresponding subdivision weightings. It allows the refinement process to better match non-linear data with fewer subdivision iterations.

5-3 – Implementation of PCA in a simulated vehicle torque structure

To test the Automated Weighted Subdivision PCA concept for implementation, a Simscape vehicle powertrain model has been developed in the Simulink environment. The vehicle modelled is a twin-axle EV from Chapter 3 (first shown in Figure 3-7), with an electric motor on each axle; Figure 5-19 revisits the EV powertrain layout.

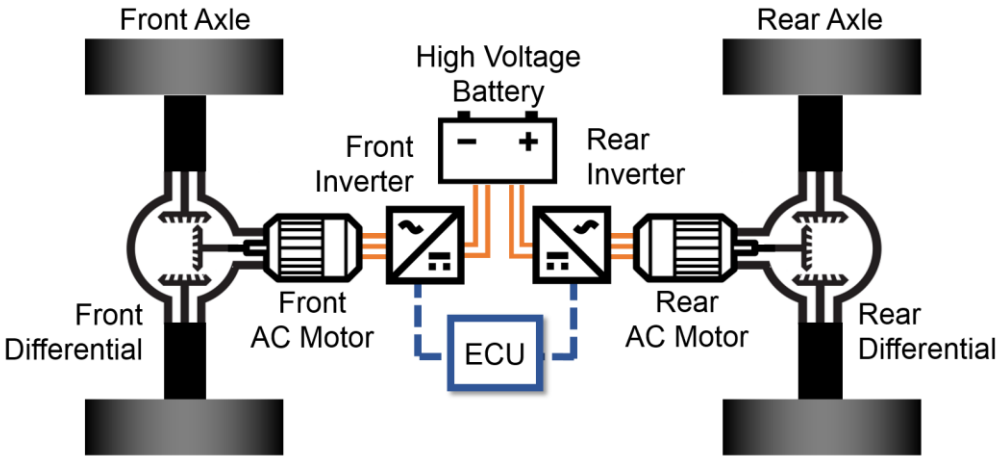

Figure 5-19: EV Powertrain Architecture.

The model consists of two main sections: 1) the physical drivetrain and vehicle model, and 2) the torque structure control software, similar to a typical modern vehicle control setup. A torque demand is sent to the actuators from the torque structure control software, which itself is controlled by the accelerator pedal demand from the human driver model and the measured physical vehicle velocity. The PCA FSC will seek to detect faults in the torque structure. Figure 5-20 shows the intended architecture of the physical model, torque structure, and PCA module.

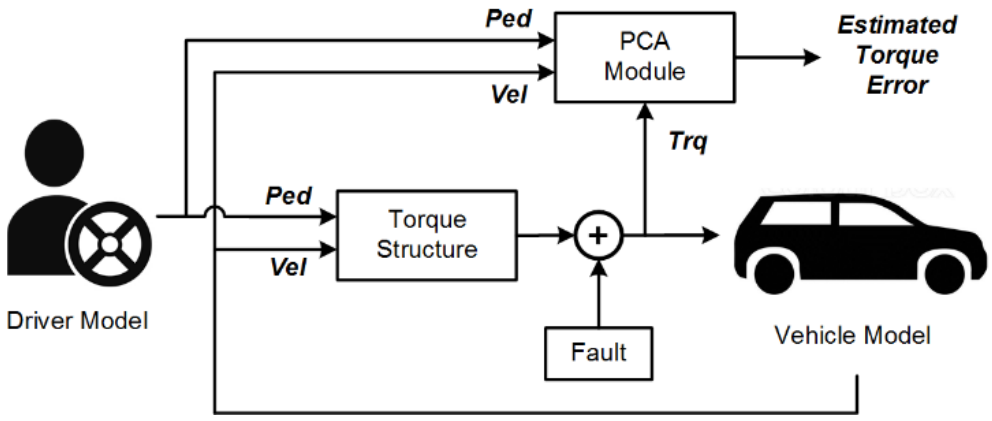

Figure 5-20: System view of simulation model.

Birch et al. [112] defines a safety goal to address the hazard of unintended acceleration: "vehicle positive longitudinal acceleration shall not exceed driver demand by more than 1.5 m/s^2 for longer than 1 s'. They found, for their particular EV vehicle, that the maximum acceleration achievable by a 150 Nm fault is 1.5 m/s^2, and therefore defined the functional

safety requirement and fault reaction mechanism to limit erroneous excess torque to 150 Nm. Considering the same safety goal Birch et al. defines, the same test was conducted on the Simulink vehicle model in this chapter to find that a ~140 Nm fault would result in a maximum 1.5 m/s^2 acceleration. Therefore, with respect to this safety goal, the PCA module must be able to always detect a fault of 140 Nm in the torque software to achieve safety with respect to this hazard.

5-3.1 – PCA Development Stage

The PCA derivation and online implementation process previously outlined will be put into practice for this application.

5-3.1.1 – Define Scope of PCA Safety Monitor Concept

This first step in the development stage is to identify the item being monitored and scope. In this case, the simplified vehicle torque structure is being monitored, with inputs of accelerator pedal position and measured vehicle velocity and an output of torque demand; this is the part that will be extracted for PCA model development.

5-3.1.2 – Gather Training Data

With the scope defined, the software component is extracted for analysis. In this case, two uniformly-distributed random number generators are used as inputs to generate 10000 data samples of pedal position, velocity, and torque for the PCA models to be trained on. Random number generators of different seeds are also used to generate test data for refinement in the next stage.

5-3.1.3 – Derive PCA models

The training data and test data is used to calculate the PCA models in the automated weighted subdivision process. After four subdivision iterations, the final $\mathbf{M}$, b, and c are derived and stored for each 256 cells; for this application, a = 10000. Figure 5-21 shows the performance of these PCA models on the test data:

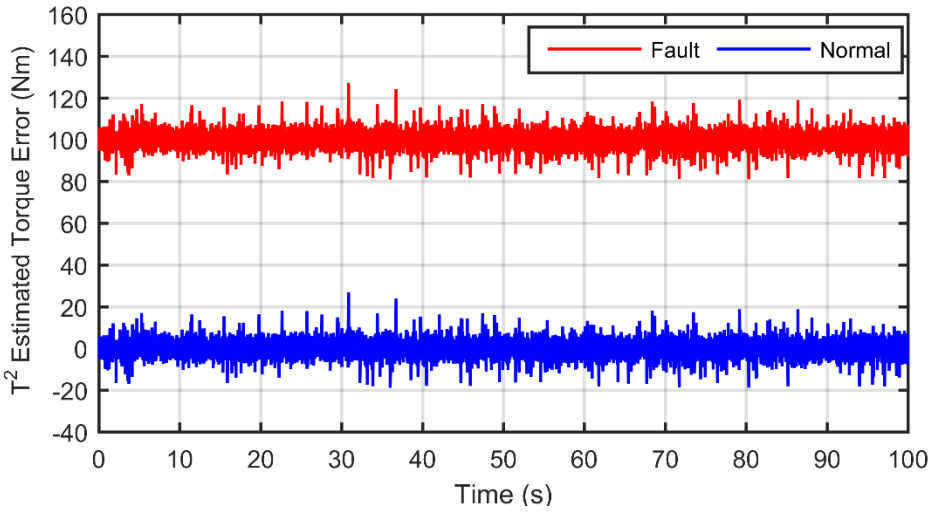

Figure 5-21: T^2 for Normal and 100 Nm Fault

Compared to Figure 5-17, there does exist some greater variation in both normal and fault T^2 here in Figure 5-21, which can be attributed to the greater complexity system being modelled. Furthermore, it was found that the instances of greatest variation are found specifically in just two cells. Depending on design requirements, more iterations and cell refinement could in theory be used to increase fidelity and improve robustness at the expense of increased storage requirement. The ability to choose the number of iterations of the PCA model development for a particular application highlights the capability to tune the PCA module in the development stage to the particular need or constraint of the application.

5-3.2 – PCA Module Online Implementation and Test

With the PCA module derived, the final step is to now implement it into the vehicle. The same input and output signals are routed to the module, and the estimated torque error output from the PCA module would be sent to the fault decision and reaction mechanism. Fault detection is the focus of this thesis, so no active fault reaction will take place. A test can now take place.

The human driver model will attempt to match a simple target vehicle velocity – recorded real-world driving data – by actuating the accelerator pedal and brake pedal. Eight fault events will take place over the course of the 100 s drive cycle, which will be additively injected on the torque output, as per Figure 5-20. Figure 5-22 shows the fault signal injection schedule, which the PCA module has no knowledge of.

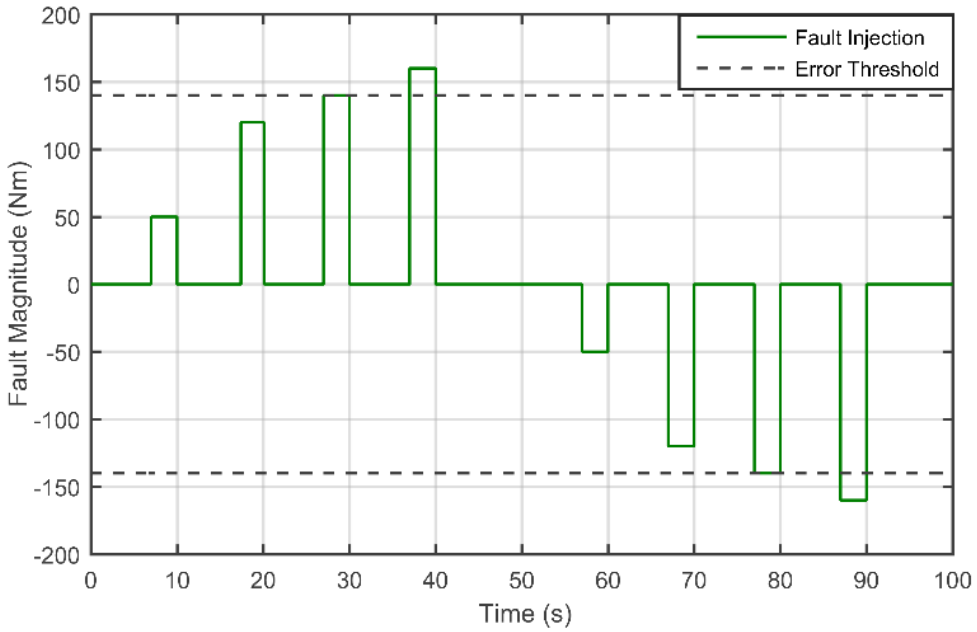

Figure 5-22: Fault injection signal schedule.

The aim of this exercise is to see whether the PCA module will be able to replicate these faults, as they are torque errors. Both positive and negative faults are included to test correct estimation in both directions. The resulting estimated torque error output from the PCA module for this test case is shown in Figure 5-23.

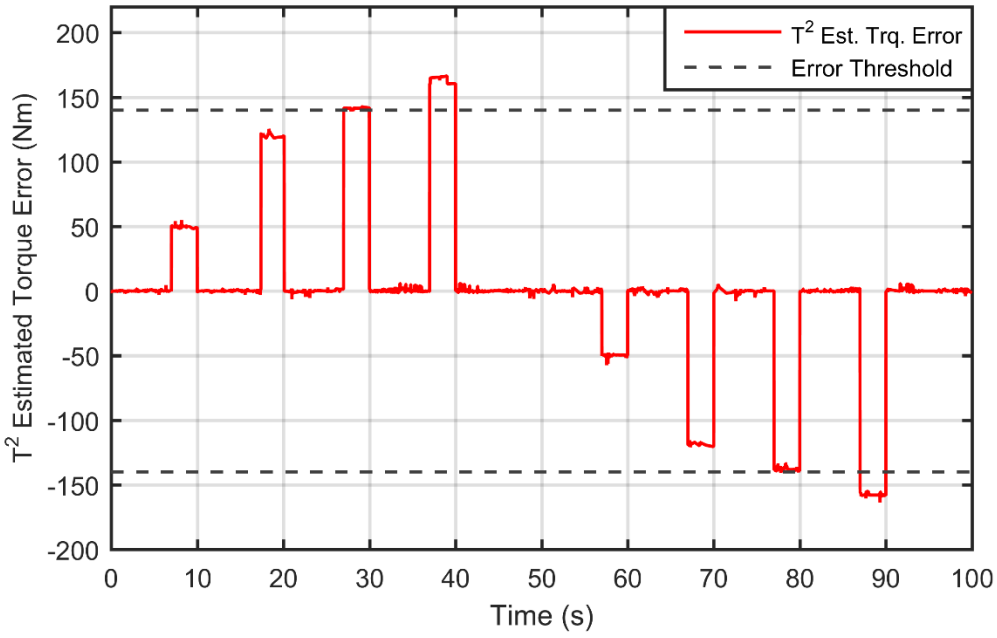

Figure 5-23: T^2 Estimated Torque Error from PCA Module.

The results in Figure 5-23 are very promising. With the exception of a small amount of expected noise, the PCA module manages to capture each fault event in the test case. The largest variation occurs in the last fault event, where a momentary spike of 5 Nm occurs. Other periods of signal noise exist, but generally it is kept low.

With the current PCA module noise, the error threshold for fault detection would need to be reduced slightly to ensure a true hazard isn't missed. For example, if the true hazard was 145 Nm, but due to noise the estimated torque error from the PCA module read 135 Nm, a fault that violates the functional safety requirement would be missed. The amount of reduction would be related to the expected normal variation following the PCA model derivation during development. Even so, the noise in Figure 5-23 is generally low with respect to the signal fidelity required for fault detection in a vehicle.

5-4 – Time-Dependent Functions

The Subdivision PCA concept has been shown to work quite well on multi-variate static MISO systems, where there is no dynamic time-element. In the previous examples, a given set of inputs would always result in the same output irrespective of preceding inputs. However, the functional software for vehicle torque management in modern cars will invariably have some form of dynamic time-dependent behaviour in the function. Some examples would include:

- Torque blending function
- Traction Control function
- ICE dynamics compensation

These cases present a fundamental issue for PCA: a given set of instantaneous inputs may produce different outputs, depending on what the inputs were in the time preceding the current data sample. When the software function being monitored includes a time-dynamic behaviour such as a time-delay or smoothing function, the training data produced resembles a 'cloud' of datapoints as opposed to a surface. The variation in this direction results in a much larger eigenvalue in the last PC, decreasing distinction between a fault and normal behaviour. In the literature review, PCA was shown to have been successfully implemented in industrial engineering applications, such as chemical process and nuclear power plants. Indeed, these are also dynamic systems that have some level of time-dependent behaviour to them, papers by [127] show the typical sample interval to be about one minute, as opposed to the 0.01 s in the torque management software. Additionally, control signals are typically not used as variables,

but rather just system and sensor variables (temperature, pressure, flow rate etc.) which would change less rapidly than a control input, unless a clear fault is present. Therefore, the time-behaviour performance of the Subdivision PCA needs to be examined. The simplest form of this would be to include a time delay on the output of the system, capturing the inputs and the delayed output.

Great consideration for the Subdivision PCA method needs to be given to how the training data is gathered, because to build a comprehensive dataset of inputs and outputs would require a sufficiently representative combination of possible preceding inputs to the current state of the system. The following number of training data points n is thus required:

$$n = \prod_{i=1}^{m_{in}} d_i^{\frac{\omega}{t}} \tag{5-18}$$

where d_i is the chosen representative resolution of the i^{th} input variable (e.g. $d_{pedal} = 101$, where each integer percentage of a possible 0-100% pedal positions is captured), ω is the sampling frequency in Hz, and t is the time constant of the non-linear behaviour that takes the longest historical effect into account (e.g. $t = 1$ for a 1 second delay on the output). Clearly then, this would surmount to an extremely large number of data points to build a representative dataset for the Subdivision PCA to train on, but which is not necessarily impossible to do computationally in a software environment. A major drawback is that each non-linear behaviour would need to be analysed for its time constant, which in itself may actually change depending on some other influence, such as different torque blending transfer functions for different pedal maps.

5-4.1 – Compensating for Time-Dependent Function Dynamics

A possible remedy would be to model these time-dependent characteristics outside of the PCA model, effectively negating the temporal behaviour. Depending on the nature of the time-dependent behaviour, the PCA safety monitoring concept could be arranged in one of two ways:

1) Use other inputs as inputs into an external dynamic model of the time-dependent behaviour, the output of which becomes an additional variable for the PCA module to learn.

2) Use the output of the PCA module as an input into a dynamic model, the output of which is then used for the fault detection.

Both approaches would incur additional initial design costs as it makes each PCA module more application specific, but would also depend on whether or not the dynamic behaviour could even be modelled in the first place. An example layout of the second option is shown in Figure 5-24.

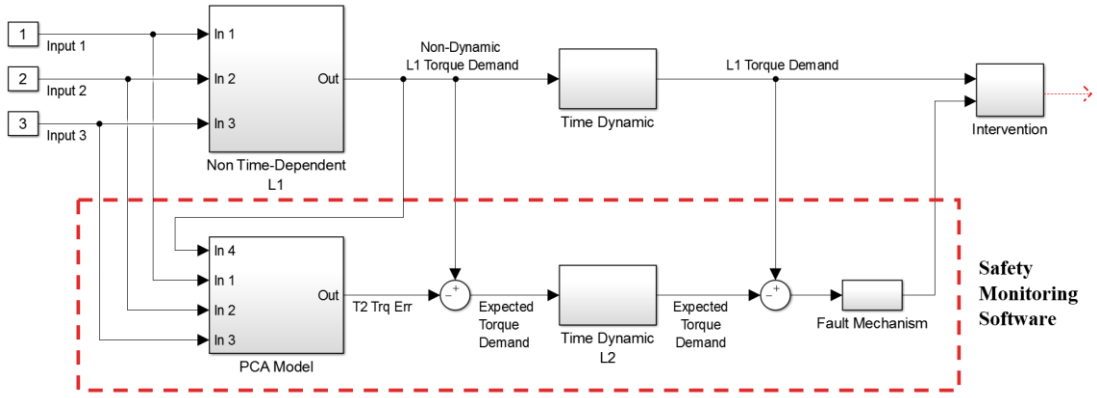

Figure 5-24: Architecture of separating out dynamic behaviour from PCA for Fault Detection.

In the architecture shown in Figure 5-24, PCA is performed on the non-linear, non-time-dependent part of the functional software with T^2 torque error as the output. The T^2 torque error output can then be added to the current torque demand from the functional software to obtain the expected torque demand. This is then sent into a safety software version of the dynamic function (time dynamic), and the outputs of this is compared to the same output in the functional software to get the error for the whole scope of the safety monitor. Separating out the dynamic component of a function is very difficult in practice, especially as there can be multiple dynamics interacting with each other; gathering the required training data becomes exponentially infeasible when additional dynamics are considered.

Another potential remedy that does not explicitly separate out the time dynamic component of the functional software is the inclusion of an inverse of the dynamic function on the output going to the PCA model, or in some cases a duplication of the dynamic model on the inputs going into the PCA model such that the dynamic component is cancelled out; for example, if the time dynamic is a 0.5 s transport delay, the same delay can be applied to the inputs such

that the dynamic effect is cancelled out. Examples of these two layouts are represented in Figure 5-25 and Figure 5-26.

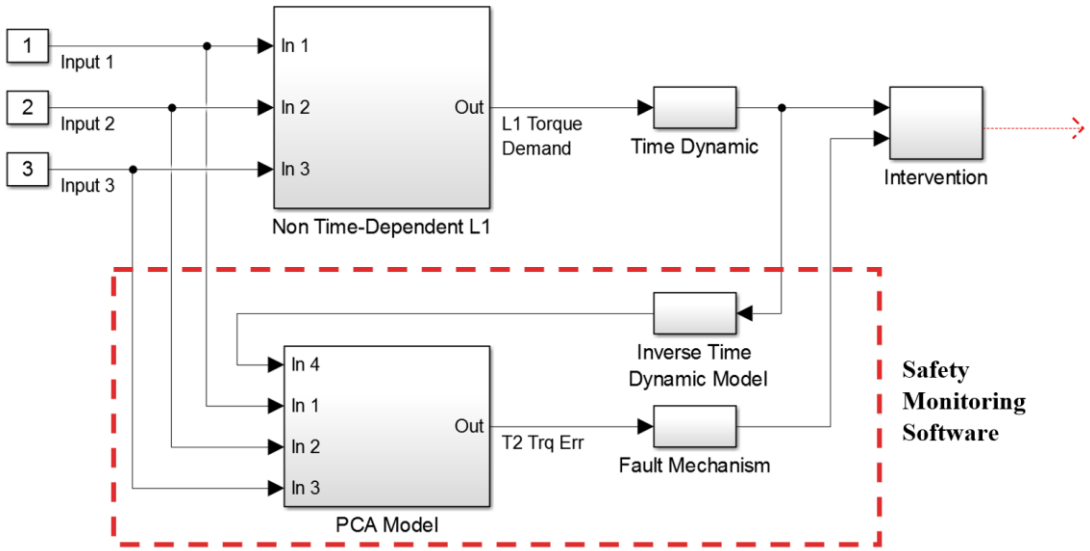

Figure 5-25: Inverse time dynamic model on functional software (L1) Output

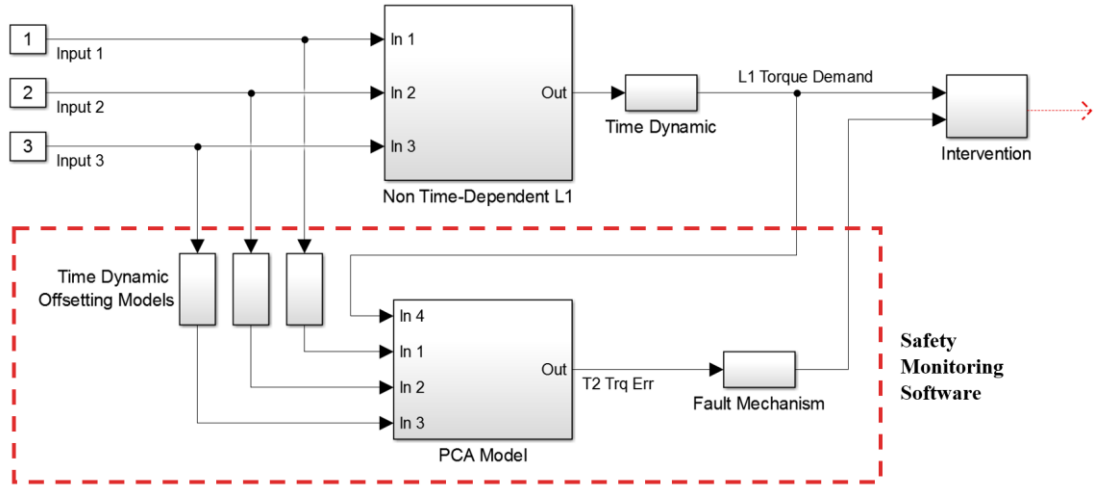

Figure 5-26: Time-dynamic offsetting models on Inputs

In theory, and with a bit of extra work bespoke to each type of dynamic considered, this could negate the time-dynamic between inputs and outputs. Unfortunately, three drawbacks have been identified:

1) Consideration for the Fault Tolerant Time Interval

2) Separating out time dynamic component for modelling still required

3) Complex and sometimes inaccurate inverse models

5-4.1.1 – Fault Tolerant Time Interval

ISO 26262 Part 1, 1.45 [26] defines the Fault Tolerant Time Interval (FTTI) as “time-span in which a fault (1.42) or fault can be present in a system (1.129) before a hazardous (1.57) event occurs”, (International Standards Organization, 2014). With this in mind, ‘undoing’ the time dynamic by adding an inverse model effectively will double the ‘Diagnostic Test Interval’ between ‘Fault’ and ‘Fault Detection, increasing the burden on the ‘Fault Reaction Time’ in order for the system to reach a safe state within the FTTI. This is illustrated in Figure 5-27 with the dotted line, which takes twice as long to detect a fault, and leaves very little time to get the system to a Safe State.

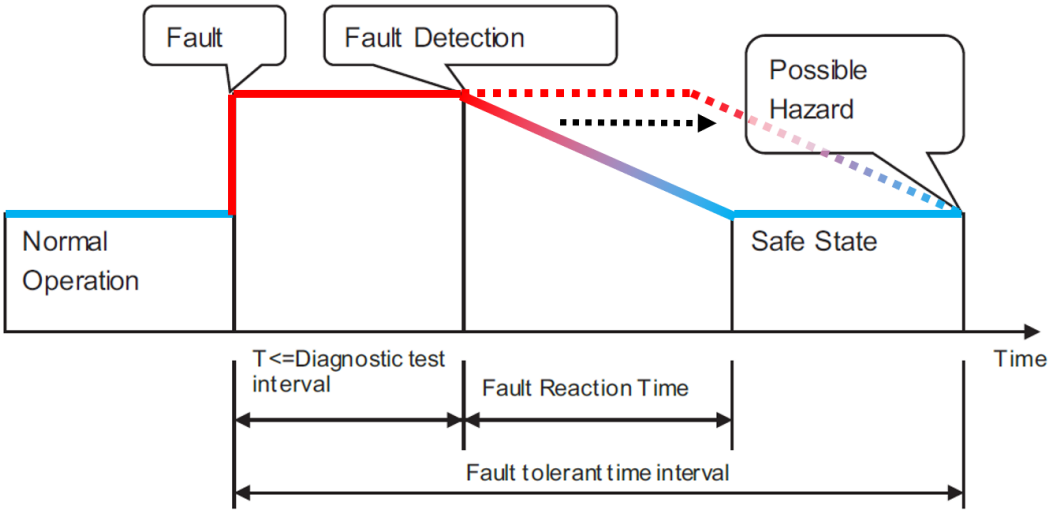

Figure 5-27: Fault reaction time and fault tolerant time interval (ISO 26262 Part 1-1.44, Fig. 4 [26], with modification)

While simpler dynamics with a short time constant may still leave sufficient space for a successful fault reaction, the environment in which passenger cars operate will likely demand a short FTTI for unintended acceleration.

5-4.1.2 – Separating Out Dynamic Components

In order to derive an inverse dynamic model, the dynamic component needs to be known and the inverse function modelled. This second method therefore suffers from the same problem as the architecture in Figure 5-24, in that separating out the dynamic component of a function is not always possible, especially where multiple components interact with each other.

5-4.1.3 – Complex and Inaccurate Inverse Models

Even if the dynamic component was known, the inverse model may not be able to be modelled such that it can accurately counteract the dynamic component or requires significant computational resources to operate. The time dynamics in the torque structure may well be so complex that to attempt to model them would not be a commercially feasible approach, particularly as these models would independently need to be verified, quickly adding to initial design cost reducing simplicity and modifiability. Doing so may well make the concept near-identical to how the Continuous Torque Demand Monitor is developed, negating the benefit of using the PCA concept.

5-5 – Further Constraints and Feasibility

The PCA Safety Monitor is shown to have promise as an FSC, but a number of constraints are present in this concept that are yet to be overcome.

5-5.1 – Validation and Verification

Before this concept can be implemented in a real vehicle, robustness and verification methods will need to be explored further to ensure the PCA safety monitor will fully ensure the safe operation of the software component it is monitoring at all times. Verification will form much of the future work on this concept, with some of the key aspects explored in Chapter 6.

5-5.2 – MIMO Functions

Currently, only multiple-input-single-output (MISO) systems are able to be monitored. This is due to the fact that T^2 varies when there is variation in the direction of the smallest eigenvalue (last PC), and it is not currently known if the error could be reconstructed when two outputs are being monitored by the same PCA module in a multiple-input-multiple-output (MIMO) system. However, it is believed that if a system has multiple inputs and outputs but only one output needs to be monitored, a PCA concept could set up in a way that treats the remaining outputs as inputs and are subdivided accordingly. A fault could, in theory, only be triggered if the monitored output did not match the known input and other output value combinations. By extension, multiple PCA modules can then be individually developed to be sensitive to errors on a particular output signal, with the PCA modules then arranged such that their estimation

outputs are combined to create an overall MIMO monitoring function. Figure 5-28 demonstrates such an arrangement.

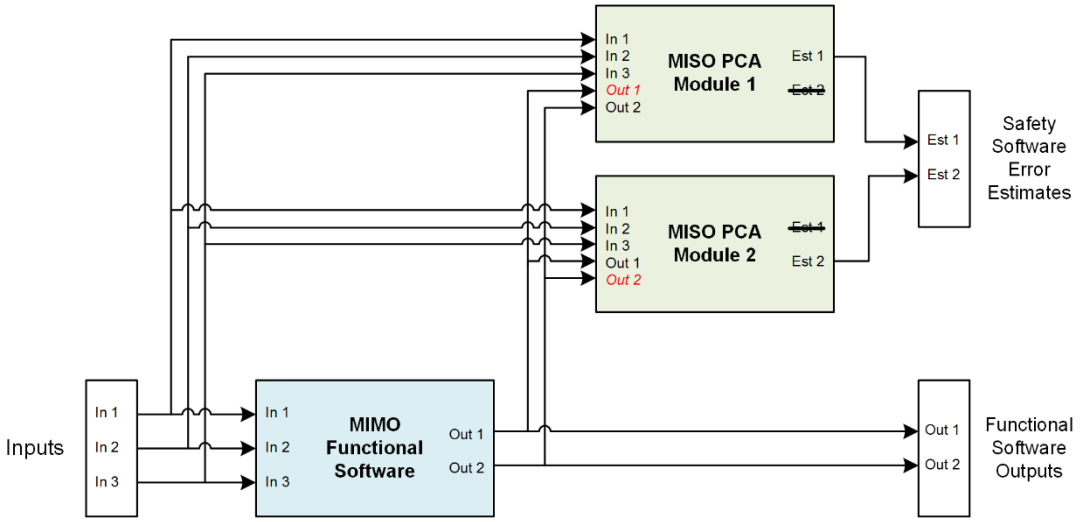

Figure 5-28: Example Architecture for applying PCA Module to MIMO Functional Software. Red labels in the PCA Modules indicate signals that are treated as "outputs" for the purposes of PCA error estimation.

In Figure 5-28 each PCA Safety Software Module is a MISO model that considers the same inputs and outputs in the estimation of errors, but each is derived to be sensitive only to estimate errors with respect to one output from the MIMO Functional Software. This is done by selecting one of the Functional Software outputs as an output for the PCA derivation process (i.e. it does not get subdivided during automated subdivision development), while the other output gets treated as an input. The number of PCA Modules will depend on the number of outputs that need to be monitored. In such an architecture, dynamics between output signals (e.g. where output 1 affects the value of output 2) can also be captured in the relevant MISO PCA Modules. Of course, such an arrangement increases requirements for storage and computation, as well as increases design, implementation, and verification effort.

5-5.3 – Hardware Storage Requirements

While PCA is excellent at describing large amounts of data with just a few matrices, the subdivision process creates a relatively large amount of data to be stored, which is likely to be a constraint on modern vehicle ECUs. Each cell k consists of an [m x m] model matrix $\mathbf{M_k}$, the boundaries matrix $\mathbf{B_k}$ of size [m_{inputs} x 2], the T^2 scalar c_k, and the T^2 offset b_k. Finally, the

single output offset constant a is stored. This means the number of ROM-stored values (n_{val}) for one PCA module (excluding code for execution) is described in Equation (5-19):

$$n_{val} = \left(\left(m^2 + 2m_{inputs} + 2\right) \times n_{cell}\right) + 1 \tag{5-19}$$

where n_{cell} is the number of cells contained in the PCA module. The per-cell storage requirement is squarely dependent on the number of monitored system variables in this concept, so increasing this quickly increases the storage requirement. The other aspect of this to consider is that as the number of monitored system variables is increased more nonlinearities are introduced into the resulting PCA system, and would need more subdivisions to capture these non-linearities. Each subdivision made on a cell increases the number of cells to a total of $2^{m_{in}} - 1$, and subsequently gives more opportunities to further refine in the next subdivision, likely increasing the number of cells exponentially. The PCA safety monitor investigation included test cases with different m, leading to storage requirements differing by nearly two orders of magnitude. For example:

- One of the test cases on a three-variable system (two inputs, one output) yielded 13 cells in the subdivision process, resulting in 780 bytes of data.
- One of the comprehensive test cases on a four-variable system (three-input one-output) yielded 481 cells, resulting in 46176 bytes of data.

There are a number of storage optimisation methods that could drastically reduce the number of cells, such as allowing for a very small amount of increased variation if it meant sharing a PCA model for near-congruent cells, rather than storing separate models. Additionally, more complex cells could be derived, at the cost of increased storage space to describe their more complex boundaries.

5-5.4 – Computational Requirements

Finally, the computational requirement also needs consideration. The T^2 analysis equations are similarly dependent on the number of system variables being monitored. Once the cell membership is determined, calculating the T^2 estimated torque error, T^2_{ETE}, just requires the T^2 analysis, output offset and T^2 normalisation; the number of multiplications, ops_{mult}, to be performed in real-time on the ECU for calculating are:

$$ops_{mult} = m^2 + m + 1 \tag{5-20}$$

and the number of summation calculations, ops_{sum}, is:

$$ops_{sum} = m^2 \tag{5-21}$$

leading to the total number of operations, *ops*, to be:

$$ops = 2m^2 + m + 1 \tag{5-22}$$

Figure 5-29 shows the number of summation and multiplication operations requiring execution on the ECU per timestep, as the number of measured variables, *m*, increases (excluding cell membership calculation and other executable code):

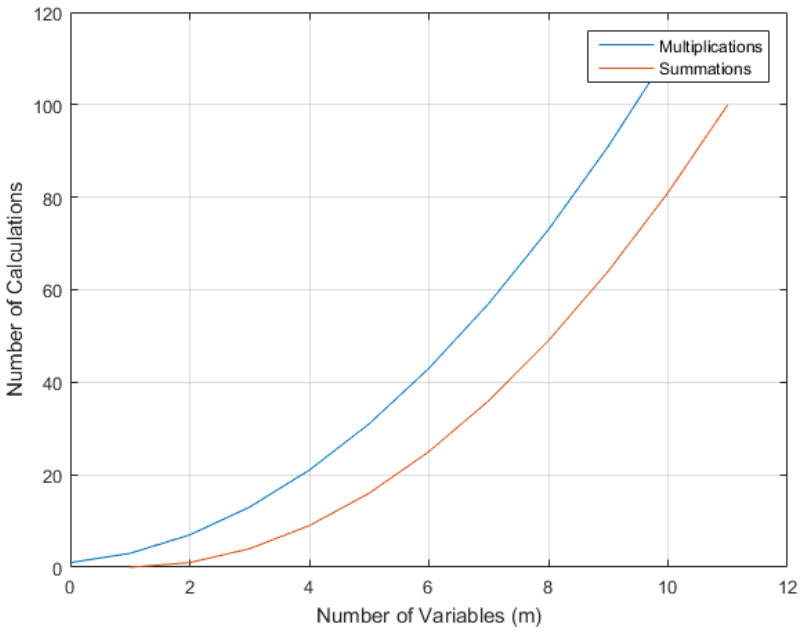

Figure 5-29: Computational requirement per step for increasing numbers of PCA Variables

Increasing the number of monitored system variables will increase the calculations nonlinearly due to the squared *m* term. Many factors will play into the computational requirements feasibility for this concept. Storage of the necessary data for the PCA module is a major limiting factor in modern vehicle ECUs, particularly when the module monitors more than three system variables on very nonlinear functions. However, storage and computational resources are ever increasing as ECU technology continues to develop, and could lead to more complex functions, such as this PCA monitoring concept seeing commercial feasibility in future passenger vehicles, should the other limitations be overcome.

5-6 – Conclusions

This chapter has explored and detailed the stages of development that has led to a viable method for deriving a PCA-based safety monitor as part of a functional safety concept. The PCA safety monitor can meet the goals of being able to quickly detect a fault and determine its direction and magnitude. An automated subdivision refinement process was then detailed, whereby multiple cells of PCA models were created to improve performance and better describe non-linear datasets.

The process was then successfully carried out within a simulated vehicle environment, where a software component was identified (torque structure), training data was generated, the PCA models derived, and the PCA safety monitor implemented. The PCA safety monitor was able to detect all the injected faults during the drive cycle with a small amount of variation, and correctly estimate the magnitude and direction of each fault. Within the constraints and limitations discussed in the final section, the PCA safety monitor holds merit as a viable safety monitoring concept, pending the thorough verification and robustness testing required by ISO 26262.

The next chapter considers the verification and validation considerations for both the PCA safety monitor of this chapter, and the adaptive safety monitor, before revisiting the ideal attribute scores preliminarily assigned in Chapter 3 and updating the attributes with justifications based on the outcomes from the two investigations.

Chapter 6

Validation & Verification Considerations and Concept Evaluation

6 – Summary

The PCA Safety Monitor and the Adaptive Safety Monitor have both been shown to hold value as safety monitor candidates, according to initial testing. In order for these to be progressed commercially, attention needs to be paid toward any unique validation and verification (V&V) considerations that have been identified as a result of these investigations. A full V&V analysis of either concept is out of scope for this project, but key questions are discussed, and are addressed with potential solutions. Finally, the concepts will be each re-evaluated according to the ideal concept attributes identified in Chapter 3 to produce a final Pugh matrix score.

- **Objective 5:** Determine if candidate concepts could be suitable for a future production vehicle, using ideal concept attributes.

6-1 – Validation and Verification Considerations

6-1.1 – Principal Component Analysis Validation & Verification Concerns

As PCA is a process-history-based quantitative analysis method [84], many of the concerns are rooted in ensuring the validity and integrity of the data used for deriving and testing the PCA models, in addition to certain limitations of the current PCA system that pose other V&V concerns.

6-1.1.1 – Training Data: Verification of Training and Test Data

As the PCA models are derived from a set of training data captured from the functional software component to be monitored, the models can only be as good as the quality of the training data. As such, the verification of the training data is a key question, since poor training data could lead to PCA models that are either unrepresentative of the software component, or include any systematic problems in the software component. Similarly, a separate set of the same source of data is used for the T^2 testing and verification process, which checks the performance of the PCA models over the range of inputs and outputs. Should the test data be invalid, the test would also be invalid, and could mean poor performance and loss of functional suitability by the PCA module. Furthermore, for the T^2 testing process, there is a need to ensure that enough test data is captured for the process to verify the PCA models. Using too little test data could mean too sparse a representation of the functional software in the test data, possibly invalidating a test. Of course, with a deterministic software component, a finite number of possible states exist, and each possible input combination could be tested at the significantly increased expense of development resources. Thus, if there were to be some reduced set of test data (as there currently is), it must still be valid and comprehensive enough to satisfy verification of the PCA model performance.

Possible Solution: The Data Safety Initiative Working Group was established by the Safety Critical Systems Club, with the aim of developing clear practices of handling data (as opposed to software and hardware) in safety-critical systems through the publication of the Data Safety Guidance [131]. Structured around ISO 31000 (with the purpose of assisting data safety consideration in future standards), a four-phase "Data Safety Management Process" (DSMP) is outlined in the latest version of the Data Safety Guidance handbook:

- Establish Context
- Identify Risks
- Analyse Risks
- Evaluate and Treat Risks

The DSMP first considers the whole context of system development context, requirements, and design, in order to establish the rigour devoted to addressing the risk appetite. Here, the type of data can be identified from the list provided in Section 6.1.6 and Appendix E of the Data Safety Guidance, with PCA training data possibly falling under "1- Predictive" ("data used to model or predict behaviours and performance") or "9-Machine Learning" ("data used to train the system to enable it to learn from the characteristics of the data"), and PCA test data falling under "verification" ("data used to test and analyse the system"). Sources of risk are then identified along with potential consequences to produce a comprehensive list of possible risks, for which the Data Safety Guidance handbook provides data-related HAZOP guidewords. With a list of risks established, these risks can then be individually analysed and the data-safety related risk level determined by the Data Safety Assurance Level (DSAL) on a scale of lowest assurance (DSAL 0) to highest assurance (DSAL 4). Each risk's DSAL is calculated based on parameters of 'Likelihood' and 'Severity' using Table 6-1; calculation of Likelihood and Severity is found in Section 6.3.1 of [131].

Table 6-1: Data Safety Assurance Level

		Likelihood		
		Low	***Medium***	***High***
Severity	***Negligible***	DSAL 0	DSAL 0	DSAL 1
	Minor	DSAL 0	DSAL 1	DSAL 2
	Moderate	DSAL 1	DSAL 2	DSAL 3
	Major	DSAL 2	DSAL 3	DSAL 4
	Catastrophic	DSAL 3	DSAL 4	DSAL 4

The use of DSALs is reminiscent of ASILs in ISO 26262 [26] and SILs in IEC 61508 [53]. Mitigation of the risks are then either conducted within the software or the data itself, with numerous techniques recommended based on the DSAL of each risk. With the introduction of the Data Safety Guidance, a clear structure to determine the best countermeasures and mitigation techniques against poor training data are provided, and could be used to ensure the proper training of the PCA modules is achieved.

6-1.1.2 – Training Data: Validity of Data Collection Method

The method that is used to collect both the training data and T^2 test data needs to be representative of the full range of possible states the functional software component could take on. Currently, obtaining training and test data is achieved through sweeping the input parameters on the functional software component and measuring the resulting output(s), to create the training data sets $\mathbf{X}_{\mathbf{train}}$ and $\mathbf{X}_{\mathbf{test}}$. For a functional software component with no state dynamics, each data sample tested can be chosen randomly, as testing on one data sample has no impact on the next one tested. However, if there was a software component which contains some dynamics, such that the test outcomes of each data sample are affected by the order in which they're tested, the data collection method would need to be validated in order to ensure the captured data sets being used for training and testing are valid.

Possible Solution: It is possible that a system with state dynamics in the functional software component could need to rely on a vehicle simulation model incorporated into the data collection method to ensure the data is captured in the manner that the functional software component processes it; simulator usage is a highly recommended technique for DSAL 3 and DSAL 4 data applications by the DSMP in Section 6.4.2.8 [131]. Two further issues are, however, introduced: 1) the verification of the vehicle model used, and 2) the completeness concern surrounding a comprehensive data capturing exercise. An automated systematic test schedule is a possible option, but needs to be ensured that both enough data, and the right data, is captured; this could be verified through other processes found in the DSMP, such as Statistics-Based Sampling, a highly recommended technique for DSAL 3 and DSAL 4 data applications in the DSMP Section 6.4.2.5.

6-1.1.3 – PCA Model Derivation: Systematic Errors in Functional Software

If all the training data has been validated and verified to be suitable for model derivation using the methods stated previously, there would still exist concerns relating to the derivation of the PCA models themselves. These models need to be valid representations of the functional software through the training data and need to be verified to show they are functionally suitable. Three key concerns are present with regards to the PCA safety monitor model derivation. If some systematic behaviour is present in the functional software component that leads to a violation of the safety goal – as opposed to a malfunction in the software after release – the training data collected on such a software component would likely also contain such behaviour, leading to a PCA module that would consider potentially hazardous behaviour as 'normal'. Single outliers and discontinuities in the training data are generally ignored or

averaged out when the PCA models are derived, but if a significant part of the training data is captured on flawed operation, the PCA safety concept may allow such functionality to continue without fault detection, as it was trained on this data. Verification of the absence of such behaviour is therefore imperative.

Possible Solution: Possibly the most significant verification concern with the PCA concept, the verification method used in Chapter 5 would not be able to detect such faults, as it tests relative to the training data. There are two possible ways to address this concern, with both testing for violation of safety goals at the simulated vehicle level. Firstly, more rigour could be applied to ensuring the training data itself is free from such behaviour in the first place, which would include some sort of rule-based checking throughout a systematic test schedule as part of a full vehicle model. Testing the training data is essentially testing the functional software for systematic errors, as opposed to malfunctions after deployment. As such, this method would require a good understanding of the safety goal(s), and what conditions would violate them, such that the rules are able to capture any systematic flaws in the functional software. The second method is to go through with PCA model derivation and implement the PCA safety monitor as the safety software with the functional software. The full system is then subjected to the similar tests as the first method, as part of a full vehicle simulation systematic test programme, this time testing whether the fully derived and implemented system violates any vehicle-level safety goals.

The second method has the advantage of being able to be the final sign-off in terms of verification since it is testing the end product (thereby adding assurance), but does mean any necessary modification that might result from the tests will be more costly. This is since finding errors in the late stage could mean reverting all the way back to the first stage of gathering training data, if it is a systematic error. The first method, meanwhile, will allow for more rapid changes to be made in the functional software and/or training data before deriving PCA models. Furthermore, it may not be possible to trace any resulting safety goal violations back to systematic flaws in the training data, as it could be a problem in the PCA model itself, unless this is also verified; even then, further testing would be required to specifically identify the region of training data responsible for the systematic flaw. It is possible that the development effort required to define and create the data/system testing environment yields a situation similar to what the benchmark method would yield, driving up cost for the PCA concept significantly.

6-1.1.4 – PCA Model Derivation: Unmonitored Signal Produces Significant Adverse Effect

Reducing the number of functional software signals to monitor in the PCA safety software module is a key way in which computational resources, storage requirements, and initial development effort can be reduced, especially when considering the trend of increasing functional software complexity. Similarly, it will need to be verified that excluding certain signals will not lead to a point where the PCA models are not representative enough such that the T^2 estimated torque error is too inaccurate, as that could mean reduced reliability of availability, or worse, a missed fault. Testing in Chapter 5 showed that the linear PCA safety monitor is sensitive to non-linear data, leading to the conception of the Local PCA concept and automated subdivision process. If a signal in the functional software component is left unmonitored, an unmodelled non-linearity could easily cause the T^2 torque error estimate to be inaccurate. Therefore, it would need to be verified that such non-linearities do not exist or can only exist within a constrained magnitude.

Possible Solution: This concern is congruent to simplifying the safety software model in the benchmark safety monitoring concept, with the adverse effect essentially being increased torque error under normal conditions due to an unmodelled noise factor. Therefore, established methods such as P-diagrams could be used to examine the effect of each of the noise factors, to determine which should and shouldn't be included in the safety software.

6-1.1.5 – PCA Model Derivation: Unknown State Dynamics in Functional Software Component

Testing in Chapter 5 showed that the PCA safety monitor is sensitive to temporal state dynamics that may exist in the functional software component being monitored. A PCA module could be set up in such a way that those dynamics are accounted for, effectively removing the burden on PCA to monitor time-dynamic behaviour. Such modelling requires expert knowledge by the engineer, and the safety software component would need to be verified as per the current benchmark. However, it must be verified that all temporal functions of the functional software are compensated for, such that the resulting PCA models are able to produce an accurate estimate of torque error through T^2 at all times.

Possible Solution: With a fully comprehensive data collection routine, it is expected that such time-dynamics would surface. These dynamics are effectively unknown noise factors. If the

training data does not capture any such unknown dynamics, at worst the resulting PCA module will flag a false positive and reduce availability to the driver. If the training data does capture such unknown dynamics, a significant normal variation will manifest itself as part of the T^2 test and should be easily identifiable, after which the engineer will be able to take measures to analyse the source of the normal variation, persisting until the dynamic is accounted for.

6-1.1.6 – PCA Model Derivation: Validity of T^2 Normalisation method

The step which maps T^2 values (after output signal offsetting and normal baseline offsetting) measures the average T^2 output in each cell for a known fault magnitude f. T^2 is normalised by dividing by this average, and then multiplying by f to produce the T^2 scalar, c_k. The concern stems from choosing which f to use in this step, and whether this method is sensitive to the selection of f during the cell subdivision process, as a poor choice of f could lead to sub-optimal cell selection.

Solution: A brief test was actually conducted during the development of the PCA concept to test whether choice of f impacts the way the subdivisions are decided during the automated subdivision process of the PCA module. It was expected that if f did affect cell performance, then two different choices of f would result in different subdivision cell boundaries being drawn. A comprehensive set of data was not yet used at this point of the investigation. Three inputs (accelerator pedal position, vehicle speed, and actuator temperature) were subdivided using 100 Nm and 50 Nm for f. The subdivisions on the input spaces are examined and compared in Figure 6-1.

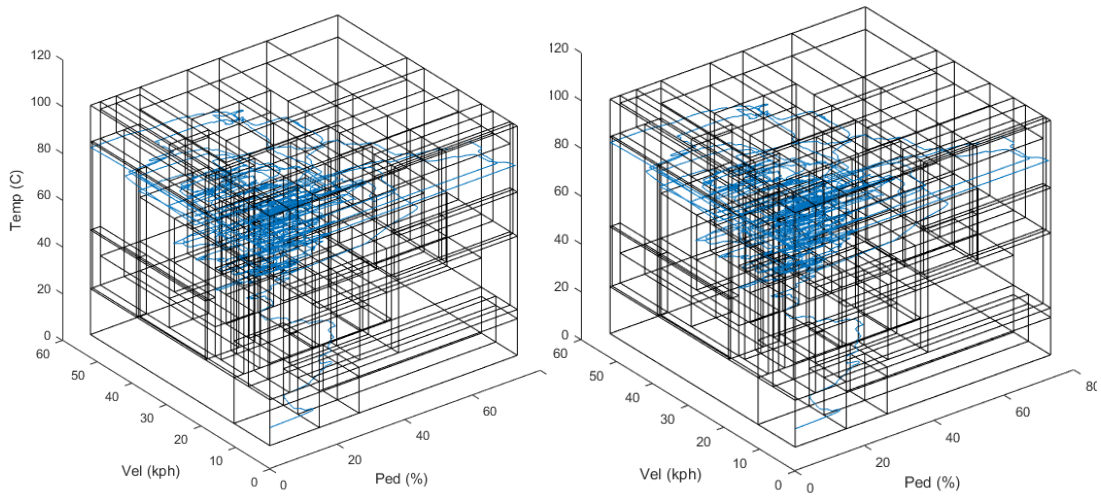

Figure 6-1: Three-dimensional input dataspace with training data (blue) and subdivision cells for 100 Nm f (left) and 50 Nm f (right).

Figure 6-1 shows the 3D input space subdivisions for both choices of *f*, and the training data used. It was confirmed that the exact same subdivisions had been made irrespective of fault magnitude – provided the fault magnitude stays constant throughout the data. This finding was pivotal in enabling the justification of using the T^2 normalisation method, as this indicated that choice of *f* was arbitrary, and therefore valid for finding the T^2 scalar, c_k. For full assurance, this could be retested with a comprehensive set of data across the whole input space.

6-1.2 - Adaptive Safety Monitor Validation &Verification Concerns

The adaptive safety monitor, being an adaptive version of the continuous torque demand monitor, faces most of the same validation and verification challenges as the benchmark system with regards to the overall structure of the fault detection and reaction mechanisms. The focus of this section is the adaptive module, and its interactions with the simplified safety software model.

6-1.2.1 – Adaptive Module Parameters: Validating Maximum Adaption Rate Limit

The maximum adaption rate limit is a very important tuning parameter of the concept, which is calculated using the sum of the maximum adaption rate limits for both long-term and short-term calculations. On the one hand, if the maximum adaption rate limit is set too high, the adaptive safety monitor could adapt more quickly to a fault than a driver is able to, such that

a hazardous fault is deemed acceptable and dismissed. On the other hand, the adaption rate could be set too low, which means it begins to behave more like the continuous torque demand monitor, losing the adaption ability; depending on how the safety model had been simplified, this could lead to an increase in torque error due to noise factors, reducing reliability through unnecessarily limiting availability to the driver. Ideally, the maximum adaption rate limit is set to the most one can expect the driver to adapt to, but a safety margin could be given such that a slightly lower maximum adaption rate limit is chosen; this is to ensure that the adaptive safety monitor doesn't adapt too quickly and miss a fault, as it is preferable to instead lose some robustness of availability for the sake of safety.

Possible Solution: It is important to determine how quickly the driver is able to adapt to such torque errors, a straightforward task in principle, but once the variation of driver ability is considered, finding that answer is not simple as each and every driver has a different ability, often with various abilities in different situations. The controllability parameter in ISO 26262 Part 3 is evaluated on "the probability that the driver … [is] able to gain sufficient control of the hazardous event, such that they are able to avoid the specific harm." [27]. While this refers to determining the controllability of the hazard, it could be argued this statement indicates that some level of driver ability is assumed, and that the driver is given some responsibility for interacting with the system to comply with the safety goals. Indeed, ISO 26262 Part 3-8.4.2.6 supports this by supplying guidelines for specifying the necessary actions of the driver in order to comply with the safety goals. One could argue, then, that there can be an assumption on the driver ability. NOTE 1 of Clause 3-8.4.2.6 b) suggests a driver task analysis could be helpful in determining driver behaviour, such as the prevention of surprise/panic/shock, which could currently be the best approach to validate the maximum adaption rate limit – and other adaptive module parameters such as driver delay and offset limit – that is used.

6-1.2.2 – Adaptive Module Parameters: Verifying Parameters across Operating Range

Chapter 4 showed that the adaptive safety monitor parameters could be optimised for a specific use case, but it is likely that the parameters result in the 'best compromise'; it could be that the chosen set of parameters perform very well in one area, but not very well in all areas. At worst, this could mean the performance is really no different than that of the benchmark, meaning the adaptive capability is eliminated and a more complex safety software is required to ensure the safety goals are met with reasonable robustness of availability.

Possible Solution: To address the issue of parameter verification across the entire operating range, an automated software-based Monte-Carlo simulation with a representative set of scenarios created through design of experiments could be conducted to ensure that the chosen set of parameters meet the functional suitability requirements [132, 133]. It is likely, though, that some robustness of availability is sacrificed in order that the chosen parameters can achieve safety throughout the whole operating range. Therefore, a possible solution for improving robustness of availability while still maintaining safety is to subdivide the operating range, such that the parameters need only be verified for a subset of the operating range; multiple sets of parameters could then be stored and interpolated between as the operating conditions change, for example if the driver selects a different drive mode. Maximum rate limits, sampling time windows, and driver delay parameters are all easily changed on the go, but the total offset limits are not as straightforward to interpolate between subdivisions, and could simply not be included in the parameter scheduling process. Of course, this increases the storage requirement of the adaptive module somewhat, depending on the subdivisions created; investigation of such an operating range subdividing and parameter scheduling concept forms much of the future work for this concept.

6-1.2.3 – Safety Software Simplifications

The simplifications made in the safety software of the adaptive safety monitor concept can affect the ability of the adaptive safety monitor to detect faults. The core reasoning that enables the adaptive safety monitor to work is the idea that the driver is able to adapt to slow moving torque errors. If the nominal torque error produced by the simplified safety software is propagates quickly and significantly, the adaptive safety monitor will not adapt to it fast enough (as per the max adaption rate limit) and will result in some offset torque error. Should the source of such nominal torque error be a fault or malfunction, then that is the right course of action by the adaptive safety monitor. However, if such fast and significant nominal torque error occurs due to noise factors, this could lead to false positives.

Moreover, consider a situation where some noise factor produces a nominal torque error of 50 Nm, to which the adaptive safety monitor has adapted to. Should it suddenly ‘switch off’ (i.e. suddenly produce a nominal torque error of 0 Nm) the adaptive safety monitor will produce a -50 Nm offset torque error, as that is the relative change. However, if there immediately follows a fault of 125 Nm (to an error threshold of 100 Nm, for example) the offset torque error from the adaptive module would be 75 Nm. This could potentially be a missed fault, as initially the nominal torque error was produced due to unmodelled functional software components in the simplified safety software, which the driver simply understands as normal

vehicle behaviour. By switching to a new drive mode, the driver could expect a -50 Nm change in torque, but instead receives 125 Nm more than the driver expects from the new mode, but only a 75 Nm increase when compared to the baseline from the previous mode

The problem here could highlight a potential shortcoming of the adaptive module when trying to mimic the driver behavioural model: the adaptive module itself is a reactive system, responding to historical behaviour from the nominal torque error output, whereas the driver also has a feedforward element; consider that the driver may anticipate a change in the functional software before it happens (e.g. manually selecting a different drive mode) in a way that the adaptive safety monitor could only adapt to after the change takes place and resulting torque error changes.

Possible Solution: To overcome this, though, the function could just be included into the safety software, at the expense of simplicity, such that the nominal torque error is not introduced in the first place. One point to note is that switching between modes while driving is typically achieved through torque blending for drivability and safety, so for the majority of cases no changes need to be made. However, there could be a need to verify that such behaviour is not present in the simplified safety model. Therefore, extra care must be taken when developing the simplified safety model to ensure certain switching modes - or any other noise factors - do not cause a large sudden change in the nominal torque error. As it's a software-based function, an automated full factorial and/or Monte Carlo approach could be conducted with a systematic test schedule [132, 134, 135].

6-2 – Concept Evaluation and Scoring

PCA and the adaptive safety monitor have been shown in the preceding sections as viable safety monitoring concepts. The experience of developing and implementing each concept has identified the strengths and weaknesses of both PCA and the adaptive safety monitor concepts when compared to the continuous torque demand monitor. The ideal monitoring attributes can therefore be revisited and scores adjusted based on these experiences.

6-2.1 – Post-Investigation Scores

In the literature review (Chapter 2), four safety concepts were identified. Chapter 3 used expert knowledge from Jaguar Land Rover, considerations from ISO 26262 [26], and principles of software quality from ISO 25010 [113] to derive a set of ideal safety concept attributes and

individual weightings. The concepts were then scored based on the anticipated experience of investigation to identify PCA safety monitor and the adaptive safety monitor as the concepts with the highest potential of success, excluding the other candidates with little promise.

Following the development and testing investigations of both candidate concepts, considering the validation and verification considerations, and with input from experts at Jaguar Land Rover during development, the concepts have been rescored based on research findings. The final ideal attribute score for each concept is shown in Table 6-2, with the previous, pre-investigation scores in parentheses.

Table 6-2: Post-investigation attribute scores of PCA safety monitor and the adaptive safety monitor; pre-investigation attribute scores in parentheses.

	Functional Suitability	**Reliability**	**Verifiability**	**Compatibility**	**Hardware and Computational Requirement**	**Initial Design Effort**	**Simplicity**	**Modifiability**	**Commercial Feasibility**	**Evolvable**	**Total** *(and difference versus benchmark)*
Weight	*10*	*7*	*10*	*8*	*8*	*6*	*6*	*7*	*6*	*3*	***710***
Continuous Torque Demand Monitor	**10**	**7**	**10**	**6**	**8**	**5**	**8**	**7**	**7**	**2**	**536**
PCA	5 (9)	8 (7)	7 (7)	6 (8)	2 (9)	5 (7)	6 (8)	6 (6)	2 (8)	4 (5)	**376** *(-160)*
Adaptive Safety Monitor	10 (10)	9 (8)	9 (9)	8 (8)	9 (9)	7 (7)	9 (9)	7 (8)	8 (8)	5 (5)	**597** *(+61)*

6-2.2 – Discussion

This section will discuss the justification for each attribute score of both concepts, before a final recommendation is made.

6-2.2.1 – Functional Suitability

6-2.2.1.1 – PCA Safety Monitor

For time-invariant non-linear systems, the subdivision PCA is shown to detect faults as they occur very quickly and provide reasonably accurate information of these faults with regards

to both direction and magnitude, which are important for fault classification and an appropriate and correct fault reaction. While there is some small variation under both normal and faulty conditions due to modelling errors, with further refinement of subdivisions it is believed noise under normal conditions can be reduced further using other subdivision techniques.

The biggest counter to the PCA safety monitor with regards to Functional Suitability, however, is the ability to handle time-variant dynamic behaviour within the system. The current system deals very well with static behaviour, in the sense that certain inputs will give certain outputs every time, but if the output is a function of both the current and previous inputs, then without knowledge of that behaviour PCA is shown to be ineffective in the current form. Some remedies were proposed, but more development would be required to understand how these problems could be overcome for different applications.

Another important consideration is that PCA may only be valid on a MISO system. Since there is only a single T^2 value that determines the current error of the system, and the fact that the subdivisions are only made on input variables, systems with multiple output variables that need to be monitored may not work with the subdivision PCA method. This will mostly be because interpreting a single T^2 value to describe two or more outputs errors will not enable accurate information to be obtained, since the single T^2 value may not be able to distinguish between an error on one output versus the other. Additionally, the necessity of offsetting the output data in the data space to provide information on fault direction may not be as effective, since the data would need to be offset in two axes. A MIMO system would likely require multiple PCA models, where the same inputs variables are used, but with one output signal per PCA module, and the multiple T^2 values interpreted somehow.

In the end, the PCA safety monitor is functionally suitable under certain conditions, such as simplifying complex yet non-temporal functions. More work is needed to overcome problems associated with temporal dynamics.

6-2.2.1.2 – Adaptive Safety Monitor

As the adaptive safety monitor is an adaptive version of the current continuous torque demand monitor, many of the strengths of functional suitability are carried over. With the reasoning explained in Chapter 4, and the tests demonstrating the effectiveness, the adaptive safety monitor holds real promise as a functionally suitable concept. By nature, sudden and significant faults that would violate the safety goal are able to be detected just as quickly as

the continuous torque demand monitor, as in these times it behaves just as the conventional variant would.
Accuracy is often improved; as was shown in the experiments, noise factors can yield a torque error that is offset from the true error. In these instances, and depending on the nature of those torque errors, the adaptive safety monitor can provide a more accurate torque error estimation than the continuous torque demand monitor could.

6-2.2.2 – Reliability

6-2.2.2.1 – PCA Safety Monitor

A motivation behind the subdivision PCA concept was to improve robustness in identifying a fault using the T^2 statistic. Using a single PCA model on a non-linear dataset caused significant variation (noise) to occur in T^2 under normal conditions, such that there was little improvement over simply using the raw torque output since faulty T^2 in one area would overlap with normal variation in another. Noise reduces the reliability of the concept, as a true safe condition calculated with noise may yield a false positive.

With the introduction of the subdivision PCA, however, it was shown that variation in both normal and faulty T^2 could be made to be more easily discernible by reducing variation within each subdivision cell. The PCA models were able to reduce noise to the point that the T^2 values were considered sufficiently low for non-linear time-invariant systems. Still, some noise on the T^2 output exists which; further subdivision refinements could be conducted, motivated through attempting to reduce this noise. The criteria for normal variation could even be varied across the data space according to the derived technical safety requirements. For example, a 25 Nm fault at low-speed low pedal position could be an unsafe condition, whereas even a 50 Nm fault could be tolerable at high speed in sixth gear, and would not need as high robustness, meaning normal data could be tolerated to vary more.

Unfortunately, testing on time-variant dynamic plants showed that the training data came to resemble more of a cloud of training data, as opposed to the surface seen previously. This causes a greater amount of variation in each cell, and in turn causes large variation in normal and fault T^2, such that they again overlap by significant amounts, and mean that faults cannot reliably be identified: in fact, typical faults aren't at all identifiable, with the exception only of the largest positive ones.

6-2.2.2.2 – Adaptive Safety Monitor

The main strength of the adaptive safety monitor, and indeed the motivation behind the concept, is to improve the reliability of the safety software concept when its safety software model is simplified. The increased torque error from the uncompensated noise factors in simplifying the safety software is offset by the adaptive safety monitor, leading to more availability for the driver as slow propagating or constant torque errors are deemed safe for the driver, and are therefore adapted to, correcting the minimum headroom. This was shown through the tests conducted in Chapter 4, with most of the false positives were compensated for in a way that would prevent a fault reaction.

Equally, if a fault in the system occurs slowly, or is reasonably small, the adaptive safety monitor will compensate for this such that a second small fault that occurs later in the drive cycle would not unnecessarily reduce availability to the driver, as the driver would have already adapted to the first fault.

In a couple of instances during the creep control simulations, the torque error changed faster than the adaptive safety monitor was allowed to adapt to it, typically on uphill or downhill sections where brake torque or driving torques are generally greater to overcome gradients. The adaptive safety monitor still represented a significant improvement of reliability over the conventional monitor in this aspect, but the safety engineer would still need to be mindful of where these torque errors originate from as before, and could still be required to compensate or account for the noise factors causing them, where possible.

6-2.2.3 – Verifiability

6-2.2.3.1 – PCA Safety Monitor

ISO 26262 requires that the safety software can be proven to comply with the standard, meaning that the software will maintain safety throughout the lifetime of the vehicle. Subdivision PCA consists of many steps in deriving the PCA models due to subdivisions and optimisations, but once the set of models have been created they would not change. A number of verification concerns exist with the subdivision PCA method, as discussed earlier in this chapter, and generally split into two subcategories: training data, and the PCA models themselves.

The training data used to derive and test the PCA models needs to be validated and verified, as errant data will lead to poor models. Additionally, the validity of the data collection method itself needs to be ensured, so that the data is representative of the system being modelled, such that functionally suitable PCA models are derived. With the recent publication of the Data Safety Guidance [131], a framework has been provided to address these concerns about data safety through a Data Safety Management Process to identify, analyse, and treat risks associated with data, and give useful techniques for risk mitigation strategies.

Even with good data, the PCA models need to be verified to make sure they are functionally suitable. The T^2 testing process can be used to comprehensively test the nominal performance of the PCA models, with high relative normal T^2 variation tending to indicate poor matching of the training data, when no real fault is present; this can be automated as part of a design of experiments or Monte Carlo systematic testing strategy, with the use of P-diagrams to identify noise factors in the system. Concerns relating to model fidelity are addressed this way, and could indicate whether certain signals need to be included in the PCA model. However, concerns regarding the verification of absence of systematic hazards is of a higher concern, as it may be possible that such hazards would be hidden in the training data itself. Any additional software needed to package the PCA model needs to be verified too, and while this is expected to be a familiar exercise, it could negate the benefits that this concept brings.

A predicted issue with using the subdivided PCA method was that Part 4 Section 7.4.3.6 of ISO 26262 states: '…a decision not to re-use well-trusted design principles should be justified' [29]. Addressing these verification concerns to a satisfactory degree could be the justification needed to satisfy this clause for higher ASIL-rated applications.

6-2.2.3.2 – Adaptive Safety Monitor

With the adaptive safety monitor being an adaptive version of the continuous torque demand monitor, the verification techniques used to verify the simplified safety model would – in general – remain the same. In fact, the burden of verification is likely to be lessened considerably, depending on the amount of simplification made to the safety software model versus the benchmark. This is due to less software components being included in the safety model, and less signals that need to be verified to be correct.

With the introduction of the adaptive element to the software, new verification efforts are needed to both validate the concept approach, and to verify that the concept will meet the safety goal(s). As discussed earlier in this chapter, a primary challenge of the adaptive safety

monitoring concept relates to validation of the adaption rate of the concept, to ensure it is representative of the human driver; this concern can be addressed with a driver task analysis (suggested by ISO 26262 [27]) to quantify the boundaries of adaption rate of a driver, and to choose the expected driver adaption rate baseline relative to the ASIL of the safety goal. Secondly, verification of the other adaptive parameter values needs to be conducted due to the concern that reliability performance may be better in certain operating ranges than in others. This could be remedied with a parameter scheduling concept, whereby the full operating range is subdivided into cells, each with its own parameters that perform best within that cell's operating range; derivation of these cells and parameters is expected to be automated, as is the verification of the parameters within these ranges, through a Monte-Carlo simulation. Finally, the verification of the simplified safety software – when coupled with the adaptive module – is a concern, as sudden changes in nominal torque error under normal operating conditions could cause the adaptive safety monitor to be vulnerable to false positives or missed faults. The solution to this could be some software-based full factorial and/or Monte Carlo approach to see if any such behaviour is present, and account for this behaviour in simplified safety software at the cost of complexity [135].

In general, it is expected that the adaptive safety monitor concept is verifiable, with each of the verifiability concerns addressed with a possible course of action, being just short of a perfect score due to the novel work that is needed to validate and verify.

6-2.2.4 – Compatibility

6-2.2.4.1 – PCA Safety Monitor

It is believed that if PCA is indeed a functionally suitable, verifiable, and reliable fault detection concept, then it can be incorporated into an existing torque structure with a few small changes. A fault monitor basically calculates an expected torque demand using the same inputs as the functional software, then compares the expected torque demand to what is actually being demanded, flagging a fault if there is an issue. The difference with the PCA in the current form is that the module output is not expected torque, but a T^2 estimation of torque error based on the inputs and outputs of the functional software. Ultimately, however, the PCA T^2 fault detection function is essentially a lookup table and some straightforward mathematical operations to obtain an output value.

Compatibility becomes less clear when multiple sets of PCA models are needed for a defined functional software module scope, arranged in a unique architecture: for example, an

arrangement which may be required for discrete signals such as selectable drive modes. The architecture becomes more complex and dissimilar to the safety software module it would replace, but as long as the inputs and outputs remain the same it should not be considered as less desirable.

Another aspect of compatibility, however, is how deriving the PCA models is substantially different to the development process of the current safety software. Engineers would need to be trained on how to develop the PCA correctly, including training data gathering, setting up the test and derivation environment, and integrating it within the torque structure, and the verification of the modules derived. It should be noted that much of the PCA derivation and optimisation work could be automated.

6-2.2.4.2 – Adaptive Safety Monitor

The adaptive safety monitor is, as stated, an adaptive version of the continuous torque demand monitor, with an adaptive module in the safety software. The strengths that the adaptive safety monitor has over the conventional method are accounting for the fact that the safety software model can be simplified. First, this potentially reduces the hardware and computational requirements, so less additional resources would be required to run the safety software. Secondly, since less needs to be compensated for due to the simplified safety model, there is potential for coexistence with further independence from QM-rated functional software. Thirdly, the new concept can quite simply replace the continuous torque demand monitor with little-to-no architectural changes to the way data is handled, other than changes relating to which signals are needed for the simplified safety software model.

However, one change in the process the safety engineer follows to derive the simplified safety model is perhaps deciding which signals can be left for the adaptive safety monitor to compensate and which ones still need an explicit signal from the functional software. Understanding the adaptive nature of the safety monitor may require some additional training for engineers seeking to implement the adaptive module into the safety software.

6-2.2.5 – Hardware and Computational Requirement

6-2.2.5.1 – PCA Safety Monitor

PCA is excellent at describing large amounts of data with just a few matrices. Linear models in particular are very simple to estimate using a single PCA, virtually regardless of the number

of data points. With non-linear data sets, however, multiple PCA models are needed to describe the dataset, which – while still vastly more efficient than storing the raw data – can significantly increase the storage requirements. This is related to the number of measured variables in the PCA module, as it was derived that both the storage of PCA module-related data, and number of computations required per timestep, had a second-order relationship with the number of measured variables. Significant improvements were made by pre-multiplying the eigenvector and inverse-eigenvalue matrices $\mathbf{V}$ and Λ^{-1}, but despite these improvements the storage requirement is nevertheless still an inhibiting factor with Subdivision PCA, due to the number of models needed for accurate operation. For this reason, this attribute score dropped significantly following the investigation.

6-2.2.5.2 – Adaptive Safety Monitor

In general, it is expected that an adaptive safety monitor with a simplified safety software model is expected to have a lower data storage and computational requirement than the continuous torque demand monitor with a typical safety software model. Naturally, this is due to – and depends on – the reduced number of components that are needed to calculate the nominal torque error, but the possible reduction is difficult to quantify.

However, the added adaptive module adds a small amount to the storage and computational requirements. The length of the long-term and short-term sampling windows add to the memory requirement, as a moving average for both long-term and short-term error offsets are calculated using the most recent values for nominal torque error. Not accounting for the driver delay function and other rules relating to rate limits, the base functionality of the adaptive safety monitor requires additional summation, multiplication, and storage of previous variables as follows:

$$ops_{sum} = n_{long} + n_{short} + 2 \tag{6-1}$$

$$ops_{mult} = 2 \tag{6-2}$$

$$storage = n_{long} + n_{short} \tag{6-3}$$

where n_{long} is the number of previous values of the nominal torque error needed, and n_{short} the number of long-term torque error values needed to calculate the short-term error offset. For a 100 Hz operating frequency (a timestep of 0.01 s), the ideal n_{short} from the Chapter 4 optimisation exercise would be 10, and the n_{long} ideal of 26.3 s would be 2630. Unlike PCA, however, the computational and storage requirements for the adaptive safety monitor do not

scale exponentially as these values are increased, as in general only the torque error is monitored. Possible future concepts that include parameter scheduling could use other ASIL-rated signals to aid in the calculation, increasing the computational requirement to some extent, but much of this computational requirement is offset, by the simpler safety model that can be used.

6-2.2.6 – Initial Design Effort

6-2.2.6.1 – PCA Safety Monitor

Reducing the initial design effort was a key driver to investigating PCA as a potential safety monitor, as the method would not need to have detailed expert knowledge of the system to derive PCA models, just the data. In reality, initial design effort is difficult to quantify given the novelty of this concept. Much of initial design work consists of defining the functional software scope within which the PCA module would be applied, which signals are necessary to capture for training, and how the PCA architecture would need to be designed. Also, initial design cost would include any additional modelling that could be required where PCA couldn't be used, such as in time-variant systems. The verification of the derived PCA module(s) would also need to be undertaken.

However, given that this is a software environment, and the fact that many of the development processes could be automated, once the expert design setup has taken place and the code environment for automated processes defined, there would not be much effort left other than verification, which itself could be automated to some extent. In fact, once the initial design of the automated environment had been completed it could be used for future PCA module developments, perhaps with some minor tweaks; in addition, lessons learnt through design setup can be carried forwards to future developments to decrease future design costs. There are three points for consideration:

1. If there is a small change in the functional software, PCA would likely only need to be retrained on new training data, which can be automated and therefore would not require much effort. If a substantial change takes place in functional software that requires a redesign of the safety software, however, expert design setup would again be required; it should be kept in mind, though, that the current system does require expert design for any change in functional software.
2. The second point is whether onboard vehicle testing is required for training data gathering, as to capture every operating condition would be a time-consuming and

expensive undertaking, especially if changes in functional software are continuously being made during development.

3. The cost of verifying training data is not currently known, and likely has many variables affecting the effort required by the manufacturer.

6-2.2.6.2 – Adaptive Safety Monitor

By enabling a simpler safety model to be used, the adaptive safety monitor concept reduces the initial effort of design required by the manufacturer in this area. The simplified safety model is, however, still architecturally the same as the benchmark model and not fundamentally different (such as the PCA concept). As such, the simplified safety model lends itself to some of the same fundamental drawbacks faced by the current benchmark, but at a reduced magnitude. A source of additional initial design effort with the adaptive safety monitor over the benchmark is that more analysis may be required to know which software components that are left unaccounted for may affect the adaptive safety monitor negatively, as a somewhat large and fast change in the nominal torque error could negatively impact performance. The adaptive module itself, however, is easily transferrable and can be automatically calibrated for a given set of functional software and safety model through a systematic design of experiments.

It is expected that development would become an iterative process of starting with an initial safety model, optimising the adaptive safety monitor for it and testing it. Then, the necessary additional functional software components would be included into the safety model, the adaptive module re-calibrated, and the concept tested again; this cycle is repeated until satisfactory results are achieved. The iterative process is similar to that of the benchmark, with the small added effort of including the adaptive safety monitor, but with less components that need including in the safety model.

6-2.2.7 – Simplicity

6-2.2.7.1 – PCA Safety Monitor

There are many steps necessary to derive the PCA models during the initial design stage, and the investigation shows that some expert understanding of the PCA process will be required to perform the initial setup. PCA can quickly become quite an abstract method, particularly with regards to multiple-dimensionality of the eigenvectors, which in theory could be n-dimensional. However, much of most complex work in deriving the PCA models can be

automated. The most complex aspect of this concept is the expert setup design, choosing which signals are necessary for the PCA to be trained on, and modelling and verifying non-PCA components that may be required within the scope, the last of which is necessary for the current safety software design to a greater extent.

Once the PCA models have been derived in the initial design with the corresponding boundaries, T^2 scalars and average normal offsets for each cell, the online process is a fairly simple one, since it is essentially a lookup table to find the correct PCA model given the current operating point, multiplying it by the current data sample, subtracting a stored number, and multiplying by stored scalar.

6-2.2.7.2 – Adaptive Safety Monitor

The adaptive safety monitor directly addresses the safety software complexity problem by enabling a simpler safety model to be used, with just the addition of a single module within the safety software. The solution is elegant, easy to understand by current engineers, and does not require a major change in the way the safety software is developed. Once the reasoning is understood, the adaptive safety monitor is intuitive, and uses familiar terms and units unlike the level of abstraction introduced through the PCA safety monitor.

The adaptive safety monitor falls short of a perfect score, however, as some extra thought does need to be given as to *which* software components can be left out in the simpler safety software model. Additionally, the automated optimisation of the adaptive parameters require some expert knowledge of the system in order to set up the design of experiments programme properly. Overall, the creation of safety software package as a whole is understood to be a simpler task with the simpler safety model and adaptive safety monitor than the current benchmark demands.

6-2.2.8 – Modifiability

6-2.2.8.1 – PCA Safety Monitor

If a PCA module needs to be modified, be it for a change in functional software or for application in a new powertrain, it is theorised that there would be four levels of modification; in ascending order of work required:

1) **Partial Retrain:** this would be the case where a distinct and containable part of the functional software was changed, for example a change in just the sand-based selectable drive mode pedal map. Since a change here would not affect the pedal maps in other selectable drive modes, only the PCA subdivision models that are affected here need to be retrained on new data. Depending on the architecture of the online PCA module, however, a full reverification may be required. Verification would only be necessary on the part of the PCA that was changed.

2) **Full Retrain:** same as above, but where parameters are changed in the functional software that affect the whole operation for a PCA module. Reverification will likely be required for that PCA model.

3) **Partial Redesign:** where a component within the safety software scope needs to be changed and re-verified that doesn't also require retraining of a PCA module; typically this would be a module that accepts the PCA output as an input, whose output does not affect PCA operation. Verification would only be needed on the part that was changed, and not of the PCA module itself.

4) **Full Redesign:** where a non-PCA component inside the scope has been changed, which would affect the training data and would thus also require reanalysis of PCA module signals, any required changes to non-PCA components, and naturally a full retrain of any PCA modules. The whole safety software scope would need to be rederived and reverified.

The main assumption here is that the majority of PCA retraining would be automated in a software environment, making retraining straightforward and inexpensive. Otherwise, any retraining would be difficult, and would likely make Partial Redesign the least effortful modification task. Transferring the module between powertrains could be possible, if the same architectural design is used, requiring only retraining of the PCA module and possibly some parameter changes in non-PCA components.

6-2.2.8.2 – Adaptive Safety Monitor

Modification of the adaptive safety monitor takes place in the two major components: the simplified safety model and the adaptive module. The simplified safety model is expected to contain less components in the first place, so a change at the functional level is less likely to require a corresponding modification at the safety software level. When a component that is

accounted for in the safety model is changed, however, the modification process in the safety model will be largely the same as the benchmark, though it is possible that such modification is less effortful as the number of interdependencies that could require reverification is less in the simpler safety software. The impact of all changes on the ability of the adaptive safety monitor to appropriately adapt to them would need to be assessed, however, so some reverification and re-tuning of the adaptive module would possibly be required with each change. A modification to a component in the functional software that is not accounted for in the simpler safety model could change the nature of the torque error under normal conditions in a way that the adaptive safety monitor cannot reliably handle, requiring that component to be included in the simplified safety model, making it slightly more complex. On the other hand, such a change could require no modification of the safety software, as the adaptive safety monitor would be able to account for it, significantly reducing the effort required when modifying the functional software, though general re-verification may still be required.

Modifications to the adaptive module, on the other hand, are currently just a matter of tuning the parameters, which has been shown can be automated. Changes to the driver-delay function rules, and any other such components in the adaptive module, would need more careful consideration when modifying, but testing and verification would be carried out in the same way unless major changes are made.

6-2.2.9 – Commercial Feasibility

6-2.2.9.1 – PCA Safety Monitor

PCA has shown a lot of promise as a fault detection concept in an automotive application, with the lure of automated model derivation and lessened expert knowledge requirement. It appears, however, that the PCA in its most basic form cannot handle non-linear systems, and while Subdivision PCA solves this shortcoming, it consequently now requires significantly more storage capacity, and some additional processing power. Despite overcoming the issue of non-linear data handling, time-dynamics has shown to cause large variance in normal T^2 torque error estimation, leading to faults are not reliably identifiable. In order to get Subdivision PCA to work, much architectural work is needed to the point that initial design effort becomes too great, and becomes more complex. While it is expected that Subdivision PCA can be verified with some more work, and that it could in fact be automated to reduce costs, it is ultimately the storage requirement that makes Subdivision PCA infeasible in a commercial application with the current and near-future hardware capabilities.

6-2.2.9.2 – Adaptive Safety Monitor

The adaptive safety monitor sets out to address the problem of safety software complexity by directly enabling the use of a simplified safety model through the reframing of the term ‘unexpected’ for the driver. The concept behaves almost identically to the benchmark when fast and significant faults are experienced, but crucially adds reliability of availability to the driver when slow, and less-hazardous faults propagate. In doing so, an adaptive safety monitor is more tolerant to the use of a simplified safety model, provided it is simplified in the right way.

The adaptive safety monitor is, however, very compatible with the current benchmark, as it simply adds a tuneable adaptive module to the nominal torque error of the benchmark concept. The hardware requirement is lessened somewhat by the simplified safety model, though attention should be paid to the storage and computational requirements of the adaptive module, even though they are expected to be have a net reduction over the benchmark and would scale well. The initial design effort is lessened with a simpler safety model and the possibility of automated calibration of the adaptive safety monitor. Modifying the functional software should be less effortful in general, but the efforts required could be larger than expected if certain components are changed that require them to be included in the safety model; this is not unique to the adaptive safety monitor, but with less functional software components included in the simplified safety model than in the benchmark, it is more likely to occur. Finally, the adaptive safety monitor provides some level of evolution as it can adapt to changes in the powertrain over time. More active evolution of the adaptive module parameters themselves presents a difficult verification problem. Overall, the adaptive safety monitor provides a positive case for commercial feasibility.

6-2.2.10 – Evolvable

6-2.2.10.1 – PCA Safety Monitor

The PCA concept is not evolvable currently, because the PCA models are trained offline and the calculated models stored on the vehicle. In order for the PCA models to evolve, they would need to be retrained online. Retraining would require again testing a range of different subdivision combinations, each time performing calculating the offset b_k and T^2 Scalar for each cell and verifying the subdivisions within a testing environment. The demand on the ECU in both processing power and memory is currently infeasible, but advances in future ECU

hardware could make online evolution possible. The main question remains as to whether online PCA evolution is verifiable, as the current verification method requires known fault data to compare and be discernible from known normal data.

6-2.2.10.2 – Adaptive Safety Monitor

As the vehicle ages, the performance of the vehicle may not match that of the functional software. Depending on the functional safety concept, and the assumptions made in the safety model (for simplicity) could lead to a great amount of nominal torque error produced under non-malfunction conditions; this could be due to sensor drift, mechanical wear, increased friction losses etc. As such, the reliability – and possibly even the functional suitability – of the safety software could be compromised. However, the adaptive safety monitor is able to adapt to these very slow-moving changes over time if they result in some sort of an offset.

Active evolution could be realised through dynamically changing the adaptive module parameters, such that the adaptive module behaves optimally. Doing so presents an interesting yet challenging verification problem, as these changes must always ensure compliance with the FSRs and TSRs. This would require significantly more initial design effort, and greater computational requirements for active analysis and change. Most significantly, the verification effort would be significantly larger as it would be required not only the initial set of adaptive module parameters, but all possible states of tune. To include active evolution would require significant effort and appropriate constraints but could mean changes to the functional software could lead to reduced effort of modification to the safety software. As of now, however, the verification challenge is great and is not being considered as feasible.

6-3 – Conclusions

The investigation has shown that the PCA concept, while having many strengths, is not a suitable safety concept in the current form. Primarily, the poor functional suitability when faced with system dynamics over time diminishes the range of functional software components the PCA concept can monitor. Furthermore, the high hardware and computational requirement is a great limitation. While Moore's Law [4] could be cited, the ACEs trends will similarly increase software complexity, resulting in an exponentially more complex PCA module. With regards to the adaptive safety monitor, however, the scores show the concept provides a feasible improvement to the benchmark continuous torque demand monitor, matching or exceeding its score in all areas except verifiability. The reason for that is a wider range of

verification techniques will be needed due to the adaptive module, though verification effort of the simplified safety model will be reduced. It is, however, expected to be the case that as software complexity increases further, the total verification effort is expected to be less than the continuous torque demand monitor. The investigation concludes with the recommendation of further investigation of the adaptive safety monitor as a replacement for the continuous torque demand monitor.

Chapter 7

Conclusions

7 – Summary

The original research question is revisited following the completion of the research objectives, and an overall conclusion is made. The novel research outcomes are then discussed, before intended future work is stated.

7-1 – Overall Conclusion

The prospect of increased software complexity is a concern of modern automotive OEMs. It has been brought on by the propagation of the ACEs trends: autonomy, connectivity, and electrification. Of the three trends, electrification is the most immediate with EV or electrified vehicle already widely available from most major OEMs; autonomy and connected vehicle are in development with some degree of each trend already finding itself on the market; they are, however, less pressing in the present than electrification. In this context, software complexity is problematic for the OEM who is responsible for still meeting the development, verification, and implementation requirements as before, but with a much more complex software product.

In light of the concerns that increased software complexity raise for the OEM, novel functional safety monitoring software concepts were investigated to answer the following research question:

- *Could a new approach or method of implementing safety software reduce the cost burden on the manufacturer, while still ensuring the safe operation of the functional software in vehicle powertrains?*

In summary, the answer to the original research question is "yes". After identifying ideal safety monitoring attributes and investigating two novel safety concepts, an adaptive safety monitor has shown great promise as a feasible alternative functional safety monitoring concept through enabling simpler safety software to be used to meet safety requirements, whilst maintaining sufficient reliability. A PCA safety monitor has also shown promise as a method of reducing the burden of initial design effort through automated model derivation, perform sufficiently within a limited scope.

7-2 – Summary and Discussion of Research Outcomes

7-2.1 – Ideal Safety Monitoring Attributes

The identification of the ideal monitoring attributes was an important step in selecting suitable and viable concept candidates for detailed investigation. ISO 26262 is an excellent tool and engineering framework, but is broad enough to cover both safety-critical software and hardware. That is why ISO 25010 was explored: while functional safety and verifiability are primarily derived from ISO 26262, the extra attributes that impact software design are captured

in this software quality standard instead. The weightings given to each attribute for scoring was chosen through discussion with safety experts at Jaguar Land Rover. It is possible that a different application of safety software could look to alter the weightings to be more in line with constraints. For example, if a safety monitor is needed for a relatively simple power assisted braking – as opposed to a full set of powertrain control functional software – the attribute of simple may be more important. Or, for example, a bespoke one-off sports car may heavily weight importance to modifiability to aid in setup and calibration, and less importance on hardware and computational requirements, because adding a more expensive/capable ECU hardware for one car is much more feasible than a mass-market production run. The aim with the ideal monitoring attributes was to distil the desirable attributes for safety from ISO 25010 and ISO 26262, and allow the engineering safety manager to decide which attribute weighting best captures their application, and be able to make informed decisions on safety concept suitability.

7.2.1.1 – Novel Research Outcomes

- **Ideal Safety Monitoring Attributes:** This was the first time a set of ideal safety monitoring attributes were identified beyond the basic requirements of ISO 26262, where additional desirable traits are also considered to this extent. Combining elements from both ISO 26262 and ISO 25010, and distilling them into a set of distinct attributes that include crossovers from both standards is the novel highlight in this chapter. These are: Functional Suitability, Reliability, Verifiability, Compatibility, Hardware and Computational Requirement, Initial Design Effort, Simplicity, Modifiability, Commercial Feasibility, Evolvable, and Security.

7-2.2 – Principal Component Analysis Safety Monitor

During literature review, a safety concept based on PCA seemed promising. The prospect of being able to let the safety software learn for itself the complexities of the functional software as opposed to the safety engineer needing to understand, interpret, and design, was desirable and novel. The investigation proved successful, with it being successfully applied to a vehicle torque structure, and it yielding favourable results in detecting torque errors. The initial investigation has shown that using a PCA safety monitor requires a rethink of how to derive safety software, as the ecosystem surrounding the PCA concept derivation had to be conceived. It was also found that for something as complex as functional software in a torque structure, the nonlinearities are such that an exponential number of PCA models would likely be required to meet the functional suitability requirements in the first place. However, a

handful of alternative architectures were also identified, using numerous PCA modules arranged in such a way that each set is responsible for a smaller part of the full system. Any linear subsystem would benefit from the low storage requirement of a single PCA model, but there are still a few limitations that need to be overcome by PCA, primarily, the handling of temporal-dynamics. Overall, there are certainly benefits in using PCA as a safety concept in vehicle software in certain situations, but its current limitations make it unsuitable for this application.

7-2.2.1 – Novel Research Outcomes

These are the set of novel research outcomes relating to the PCA safety monitor:

- **Implementation of PCA module as a vehicle software safety monitor:** This was the first time PCA has been investigated and shown to hold some merit as a real-time powertrain control software safety monitor, including compatibility with the existing fault reaction mechanism. Despite the current limitations, there is scope that certain applications could benefit from this concept, not just a torque structure. It is abstract enough that any software where an error is quantifiable, a PCA module may be able to monitor the output.

- **PCA safety monitor derivation process:** A PCA-based safety monitor of any kind has not been seen in the automotive sector before. The investigation established a full workflow process for producing a PCA module for real-time fault detection application, starting with appropriate training data collection, derivation of PCA models for a torque structure, refining the models for better performance, testing/verification of these models, and implementation into a software monitoring application. The whole process has not been seen before in any industry, and indeed, many of the smaller elements of the process are believed to be completely novel across all industries.

- **Output signal offset:** This is the first time, as far as the author is aware, that a method was applied to extract qualitative information about fault directionality, through the simple process of adding a constant offset to the output signal of the plant training data – in this case, the torque demand. It also has the added benefit of resulting in a more linear mapping of T^2 to estimated torque error. While it does not result directly in the T^2 estimated torque error, it is a crucial step towards it.

- **Local PCA Concept**: The local PCA concept is a novel method of estimating torque error in the functional software based on comparing training data with current software states. Similar concepts have been seen before and certainly linearizing a non-linear surface is a well-established practice, but has not been seen in the application PCA to software monitoring. Being initially conceived as a solution to modelling nonlinear data, local PCA uses known information about the training data to correctly adjust and scale the measured T^2 signal to produce a reasonable torque error estimation. Translating and scaling data is itself not a new method, but combined with using known artificially injected faults into the system training data to correctly calibrate the T^2 scale normalisation is novel, as is the investigation indicating the validity of using a single scalar as opposed to having to map the whole.

- **Automated Subdivision process:** The automated derivation and refinement of Local PCA models through subdivision, and the inception of the performance metric *e* used for refinement of PCA models in the automated routine, is a creative way to ease the manual calibration effort required by the development engineer in deriving PCA models for fault detection. Additionally, the idea of a weighted subdivision for improving PCA module performance per model stored is also a creative solution to reducing the storage and computational requirement.

7-2.3 – Adaptive Safety Monitor

A new functional safety concept called the adaptive safety monitor was investigated, seeking to address the issue of safety software complexity by enabling a simpler safety software model to be used with an adaptive element in the safety monitor, overcoming many of the drawbacks that would otherwise prevail. The novel thought process behind this concept has led to an investigation and development of a viable functional safety concept to ease the development, implementation, and verification burden on vehicle manufacturers. It was shown in the investigation that the adaptive safety monitor can successfully improve the robustness of a less-complex variant of safety software by adapting to slow-propagating torque errors as a human would be able to adapt, while still capturing real faults with more accurate quantification. It also enables more accurate indication of safety goal violation, as it provides a better depiction of what the driver is experiencing at the vehicle level, as opposed to simply measuring how the system is behaving or malfunctioning. It was also shown how the adaptive safety monitor parameters could be analysed for sensitivity using a design of experiments approach, which can also be used to calibrate the adaptive safety monitor for optimal

performance. Future research will focus much on this aspect, seeking to derive a similar automated calibration process from lessons learnt through the PCA derivation process.

While the adaptive safety monitor is a commercially feasible answer to the software complexity problem for powertrain safety software, a possible exception could be in the case of autonomous vehicles. The adaptive safety monitor relies on the fact that a driver is still driving the vehicle. Though autonomous vehicles are actively being developed, the current market is still very much non-autonomous, and as such most cars currently sold and in foreseeable future will still require a driver in the loop. The adaptive safety monitor could still be used in an autonomous context, but possibly with a calibration suited to the autonomous driver. More investigation needs to be taken on this matter, but it can be argued that vehicle autonomy will not make the adaptive safety monitor redundant. The added benefits of being simpler, less resource intensive, and demanding less initial design effort than the benchmark continuous torque demand monitor ensures the viability of the adaptive safety monitor as a successful concept candidate in this research into novel safety monitoring concepts.

7-2.3.1 – Novel Research Outcomes

These are the set of novel research outcomes relating to the adaptive safety monitor:

- **Adaptive safety monitor reasoning:** The reasoning behind the adaptive safety monitor is the core of its innovation. By redefining the nature of ‘safety’ by examining the word ‘unintended’ (as in ‘prevent unintended acceleration’), and including the human driver into the equation, a better-defined understanding of safety can be used to justify the possibility of the adaptive safety monitor. Whereas before just the fault magnitude was examined, now it can be justified that all slow-propagating faults can be adapted to by the driver, and do not cause a threat. If this is permissible, it opens the door to making adaptive safety software, that itself can adapt to unmodelled noise factors in a less complex safety software model. All of this stems from the innovative reasoning that precedes the adaptive safety monitor.

- **Two-stage adaptive algorithm:** Reasoning aside, the way the adaptive safety monitor fundamentally operates is not exactly new, as it is simply a statistical moving average being subtracted from the current value. However, the two-stage algorithm that has been developed leverages the long-term and short-term memory of the driver in its design to better imitate expected driver behaviour, and allow for more tuning options. Additional adaption parameters also expand the capabilities of the adaptive safety

monitor beyond its fundamental diagram, and are essential in achieving the calibration to match driver behaviour; indeed, these were developed and tested specifically to that end.

- **Automated optimisation procedure:** The automated optimisation procedure was conceived to improve the performance of the adaptive safety monitor. It is a design of experiments approach, which itself is not novel, but the performance metric and trend analysis conducted to improve adaptive safety monitor performance is the foundation for possibly an innovative parameter scheduling derivation process; this will be discussed more in Future Research.

7-3 – Future Research

The investigations in this project were completed insofar as meeting the objectives set out in the introduction. Throughout the investigation, it has been alluded to that plans have been made for future work. The overall aim of the future research activities is different for the two concepts: for the adaptive safety monitor, the aim is to deploy it on a production vehicle, which means addressing the verification and validation concerns; for the PCA safety monitor, it is continuing the investigation to overcome the limitations of the concept. Verification and validation concerns will not be looked at for the PCA safety monitor until the functional suitability limitations are overcome. For the sake of brevity, the topics will just be noted here, with detail and initial work on these topics found in Appendix C.

For the PCA safety monitor, the following topics are to be investigated in the future:

- Using alternative PCA module architectures to improve performance.
- Using last eigenvalue for determining T^2 scalar c_k.

For the adaptive safety monitor, the following topics are to be investigated in the future:

- Using a Driver Task Analysis to validate the driver delay parameter.
- Development of a systematic test environment to verify the concept.
- A parameter scheduling concept based on varying operating scenarios.

References

1. Juncker, J.-C., *Commission Implementing Regulation (EU) 2018/1003 of 16 July 2018 amending Implementing Regulation (EU) 2017/1152 to clarify and simplify the correlation procedure and to adapt it to changes to Regulation (EU) 2017/1151*, in *2018/1003*, E. Union, Editor. 2018, Official Journal of the European Union: Brussels. p. 16-21.

2. Dinsdale, A. and A. Berdichevskiy, *The Future is Now: Transforming the Automotive Customer Experience*, in *Great Expectations: Insights exploring new automotive business models and consumer preferences*. 2017, Deloitte Insights.

3. Underwood, S., *Expert Forecast and Roadmap for Sustainable Transportation*, in *Automated, Connected, and Electric Vehicle Systems*. 2014, Institute for Advanced Vehicle Systems: Dearborn.

4. Moore, G.E., *Cramming more Components onto integrated circuits.* Proceedings of the IEEE, 1965. **86**(1): p. 82-86.

5. Berndt, E.R., E.R. Dulberger, and N.J. Rappaport, *Price and Quality of Desktop and Mobile Personal Computers: A Quarter Century of History*. Proceedings of the Hundred Thirteenth Annual Meeting of the American Economic Association, 2001. **91**(2).

6. Kurzwell, R., *The Singularity Is Near: When Humans Transcend Biology*. 2006: Penguin Books. 672.

7. *Self-Driving Cars: Tesla and NVidia*. 2018 [cited 2018 July 2018]; Available from: https://www.nvidia.com/en-us/self-driving-cars/partners/tesla/.

8. Nielsen, J. *Nielsen's Law of Internet Bandwidth*. Web Usability 1998 [cited 2018 July 2018]; Available from: https://www.nngroup.com/articles/law-of-bandwidth/.

9. Greenbaum, E., *5G, Standard-Setting, and National Security*, in *National Security Journal*, A. Whiting, Editor. 2018, Harvard Law School: Cambridge, MA.

10. *Global Energy Storage*, in *Energy Storage Service*. 2012, Wood Mackenzie: Edinburgh.

11. Hellemans, A. *How Long Before Sodium Batteries Are Worth Their Salt?* Energywise, 2017.

12. Mohr, D., *Automotive revolution – perspective towards 2030*, in *Advanced Industries*. 2016, McKinsey & Company: Stuttgart.

13. SAE, *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. 2018, SAE International: USA.

14. *European Commission set to announce major new road safety package*. News, 2018.

15. NTSB, *Preliminary Report Released for Crash Involving Pedestrian, Uber Technologies, Inc., Test Vehicle*, in *NTSB News Releases*. 2018, National Transport Safety Board.

16. *Ann Arbor Connected Vehicle Test Environment (AACVTE)*. Research Archive 2012 [cited 2018 July]; Available from: https://www.its.dot.gov/research_archives/safety/aacvte.htm.

17. Burton, N., *A History of Electric Cars*. 2013, Marlborough, UK: The Crowood Press Ltd.

18. Chapman, K., *The State of Innovation Report 2017*. 2017, Clarivate Analytics: UK.

19. Petrof, A. *These countries want to ban gas and diesel cars*. Powering Your World, 2017.

20. *Coalition Agreement 'Confidence in the Future'*. 2017, Government of the Netherlands: Amsterdam. p. 60.

21. EAFO, *Size of electric vehicle market share in the European Union in 2017, by country, in percent.* 2017, EAFO.

22. NTSB, *Preliminary Report*. 2018, National Transport Safety Board: California, USA. p. 4.

23. Chestney, N. *Electric vehicles could lift UK peak power demand by 5-8 GW by 2030 - National Grid*. Business News, 2018.

24. *Symbioz Demo Car*. 2017, Groupe Renault: France.

25. EGAS Workgroup, *Standardized E-Gas Monitoring Concept for Gasoline and Diesel Engine Control Units*. 2018, IAV.

26. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 1: Vocabulary*. 2011, International Organization for Standardization: Switzerland.

27. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 3: Concept Phase*. 2011, International Organization for Standardization: Switzerland.

28. ISO, *ISO 26262 2nd Edition: Functional Safety - Road Vehicles*, in *Part 1: Vocabulary*. 2018, International Organization for Standardization: Switzerland.

29. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 4: Product Development at the System Level*. 2011, International Organization for Standardization: Switzerland.

30. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 5: Product Development at the Hardware Level*. 2011, International Organization for Standardization: Switzerland.

31. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 6: Product Development at the Software Level*. 2011, International Organization for Standardization: Switzerland.

32. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 7: Production and Operation*. 2011, International Organization for Standardization: Switzerland.

33. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 2: Management of Functional Safety*. 2011, International Organization for Standardization: Switzerland.

34. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 8: Supporting Processes*. 2011, International Organization for Standardization: Switzerland.

35. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 9: Automotive Safety Integrity Level (ASIL)-orientated nd safety-orientated analysis*. 2011, International Organization for Standardization: Switzerland.

36. ISO, *ISO 26262: Functional Safety - Road Vehicles*, in *Part 10: Guide to ISO 26262*. 2011, International Organization for Standardization: Switzerland.

37. Birch, J., et al. *Development of an Adaptive Safety Monitoring Function*. in *SSS '16*. 2016. Brighton: Safety-Critical Systems Club.

38. Basseville, M. and A. Benveniste, *Detection of abrupt changes in signals and dynamic systems*. 1986, Springer-Verlag.

39. Azimi, V. and P.A. Vela. *Robust Adaptive Quadratic Programming and Safety Performance of Nonlinear Systems with Unstructured Uncertainties*. in *2018 IEEE Conference on Decision and Control (CDC)*. 2018.

40. Darnell, P., J. Birch, and R. Sabbella, *A verification module for verifying accuracy of a controller*, W.I.P. Organization., Editor. 2016, Jaguar Land Rover: UK.

41. Dingel, O., et al., *Model-Based Assessment of Hybrid Powertrain Solutions.* 2011.

42. Monticello, M. *2017 Acura NSX Hybrid is the Friendly Supercar*. Buying Guide, 2016.

43. Li, C., G. Chen, and C. Zong, *Fault-Tolerant Control for 4WID/4WIS Electric Vehicles*, in *SAE 2014 International Powertrain, Fuels & Lubricants Meeting*. 2014, SAE International.

44. Rongrong, W. and W. Junmin, *Fault-Tolerant Control With Active Fault Diagnosis for Four-Wheel Independently Driven Electric Ground Vehicles.* Vehicular Technology, IEEE Transactions on, 2011. **60**(9): p. 4276-4287.

45. Connection, T.C. *2016 Acura NSX*. 2015 [cited 2015; Available from: http://images.thecarconnection.com/med/2016-acura-nsx_100496891_m.jpg.

46. Moure, C. and K. Kersting, *Development of Functional Safety in a Multi-Motor Control System for Electric Vehicles*, in *SAE World Congress*. 2012, SAE International: Detroit.

47. Chen, L., *Integrated Vehicle Safety Monitoring System for Running Trains*, U.P. Office, Editor. 2014: United States.

48. Efanov, D. *New Architecture of Monitoring Systems of Train Traffic Control Devices at Wayside Stations*. in *2018 IEEE East-West Design & Test Symposium (EWDTS)*. 2018.

49. O'Connell, K., M. O'Connell, and P. Lowry, *A rail train diagnostics system*, E.P. Office, Editor. 2014: Ireland.

50. Themsche, S.v., *Risk Adverse Society*, in *The advent of unmanned electric vehicles: The choices between e-mobility and immobility*. 2016, Springer International. p. 49-155.

51. Welankiwar, A., et al. *Fault Detection in Railway Tracks Using Artificial Neural Networks*. in *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*. 2018.

52. *Train Control and Monitoring System TCMS in Rail Vehicles*, Selectron Systems AG, Editor. 2015, Selectron: Switzerland. p. 3.

53. IEC, *IEC 61508: Functional Safety*. 2010, IEC: Switzerland.

54. Guo, L., et al., *Electric power locomotive system fault detecting method, involves judging whether fault information or abnormal information is matched with fault interpretation information, and installing fault knowledge library in expert diagnosis system*. 2014, Shuohuang Railway Dev Co Ltd (Shgr-C) China Shenhua Energy Co Ltd (Shgr-C).

55. Sunder, R., et al., *Operational Experiences with Onboard Diagnosis System for High Speed Trains*. 2002, Institute for Safety Technology (ISTec) GmbH: Germany.

56. Chao, Z., X. Jing, and S. Lijuan, *Fault Diagnosis System of Locomotive Axle's Track Based on Virtual Instrument*, in *International Forum on Information Technology and Applications*. 2010, I2TS: Brazil.

57. Mukherjee, R., et al., *Method for monitoring bearing cage condition in locomotive induction motors*, I.P. Office, Editor. 2015, General Electric Co(Gene-C): India.

58. IEC/IEEE, *IEC 62582: Nuclear Power Plants - Instrumentation and control important to safety - Electrical equipment condition monitoring methods.*, in *Part 1: General*. 2011, IEC: Geneva.

59. Amano, Y., *The Fukushima Daiicha Accident*, in *Report by the Director General and Technical Volumes*. 2015, International Atomic Energy Agency: Vienna.

60. *Frequently Asked Chernobyl Questions*. Feature Stories, 2013.

61. IEC/IEEE, *IEC 62582: Nuclear Power Plants - Instrumentation and control important to safety - Electrical equipment condition monitoring methods.*, in *Part 2: Indenter Modulus*. 2011, IEC: Geneva.

62. IEC/IEEE, *IEC 62582: Nuclear Power Plants - Instrumentation and control important to safety - Electrical equipment condition monitoring methods.*, in *Part 3: Elongation at Break*. 2011, IEC: Geneva.

63. IEC/IEEE, *IEC 62582: Nuclear Power Plants - Instrumentation and control important to safety - Electrical equipment condition monitoring methods.*, in *Part 4: Oxidation Induction Techniques*. 2011, IEC: Geneva.

64. IEC/IEEE, *IEC 62582: Nuclear Power Plants - Instrumentation and control important to safety - Electrical equipment condition monitoring methods.*, in *Part 5: Optical Time Domain Reflectometry*. 2011, IEC: Geneva.

65. IAEA. *Instrumentation and Control (I&C) Systems in Nuclear Power Plants: A Time of Transition*. in *52nd IAEA General Conference*. 2008. Vienna: International Atomic Energy Agency.

66. Hiari, O., W. Sadeh, and O. Rawashdeh, *Towards single-chip diversity TMR for automotive applications*, in *IEEE International Conference on Electro/Information Technology*. 2012, IEEE: Indianapolis, USA.

67. Mallavarapu, P., et al. *Fault-tolerant digital filters on FPGA using hardware redundancy techniques*. in *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*. 2017.

68. Kim, D. and R. Voyles. *Quadruple adaptive redundancy with fault detection estimator*. in *2017 13th IEEE Conference on Automation Science and Engineering (CASE)*. 2017.

69. Tseng, L. *Voting in the Presence of Byzantine Faults*. in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*. 2017.

70. Frank, P., *Fault Diagnosis in Dynamic Systems using Analytical and Knowledge-based Redundancy a Survey and some New Results.* Automatica, 1990. **26**(3): p. 459-474.

71. Chow, E.Y. and A.S. Willsky, *Analytical Redundancy and the Design of Robust Failure-Detection Systems.* IEEE Transactions on Automatic Control, 1984. **29**(7): p. 603-614.

72. Basseville, M., *Detecting Changes in Signals and Systems - A Survey*. Automatica, 1988. **24**(3): p. 309-326.

73. Forslund, A., et al., *Evaluating How Functional Performance in Aerospace Components Is Affected by Geometric Variation*, in *SAE International Journal of Aerospace*. 2018, SAE International. p. 5-26.

74. Venkatasubramanian, V., et al., *A review of process fault detection and diagnosis: Part I: Quantitative model-based methods.* Computers & Chemical Engineering, 2003. **27**(3): p. 293-311.

75. Tabbache, B., et al., *Virtual-Sensor-Based Maximum-Likelihood Voting Approach for Fault-Tolerant Control of Electric Vehicle Powertrains.* IEEE Transactions on Vehicular Technology, 2013. **62**(3): p. 1075-1083.

76. Yar, A., et al., *A Framework for Model Based Detection of Misfire in a Gasoline Engine with Dynamic Skip Fire*. 2019, SAE International.

77. Halder, P., S.K. Chaudhuri, and S. Mukhopadhyay, *Fault detection, diagnosis and control in a tactical aerospace vehicle*, in *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region*. 2003, IEEE: Bangalore, India.

78. Florio, F.D., *Airworthiness: An Introduction to Aircraft Certification*. 2nd ed. 2010, UK: Butterworth-Heinemann.

79. Patton, R.J., *Fault detection and diagnosis in aerospace systems using analytical redundancy*. Computing & Control Engineering Journal, 1991. **2**(3): p. 127-136.

80. Kuznetsova, T.A. *Kalman-Filtering Based Algorithm for Sensor's Channel Fault Detection and Isolation*. in *2018 International Multi-Conference on Industrial Engineering and Modern Technologies (FarEastCon)*. 2018.

81. Rongrong, W. and W. Junmin, *Fault-Tolerant Control With Active Fault Diagnosis for Four-Wheel Independently Driven Electric Ground Vehicles.* IEEE Vehicular Technologies, 2011. **60**(9): p. 4276-4287.

82. Zhang, Y.M. and J. Jiang, *Active fault-tolerant control system against partial actuator failures.* IEE Proceedings - Control Theory and Applications, 2002. **149**(1): p. 95-104.

83. Deosthale, E.V., et al., *Sensor Selection for Selective Clutch Fault Isolation in Automatic Transmissions Based on Degree of Fault Tolerance*. 2019, SAE International.

84. Venkatasubramanian, V., et al., *A review of process fault detection and diagnosis: Part III: Process history based methods.* Computers & Chemical Engineering, 2003. **27**(3): p. 327-346.

85. Patra, N., et al., *Kalman filter based optimal control approach for attitude control of a missile*, in *2013 International Conference on Computer Communication and Informatics*. 2013, IEEE: Coimbatore, India.

86. Ghodbane, A., et al., *Design of an Actuator Fault Tolerant Flight Control System Using Fault Detection and Diagnosis*, in *SAE 2013 AeroTech Congress & Exhibition*. 2013, SAE International: USA.

87. Vasu, J., A.K. Deb, and S. Mukhopadhyay. *Development of Extended MVEM based UKF estimators*. in *2011 Annual IEEE India Conference*. 2011. Hyderabad, India: IEEE.

88. Turin, R.C. and H.P. Geering. *Model-reference adaptive A/F-ratio control in an SI engine based on Kalman-filtering techniques*. in *Proceedings of 1995 American Control Conference - ACC'95*. 1995. Seattle, USA: IEEE.

89. Heng-Zhan, Y., Y. Kang, and Q. Fu-Cai. *Fault diagnosis method for stochastic systems with unknown parameters*. in *2018 Chinese Control And Decision Conference (CCDC)*. 2018.

90. Willsky, A.S., *Detection of Abrupt Changes in Signals and Dynamical Systems*. Lecture Notes in Control and Information Sciences, ed. A.B. Michèle Basseville. Vol. 77. 2005, Berlin: Springer Verlang. 22.

91. Wilbers, D.M. and J.L. Speyer, *Detection filters for aircraft sensor and actuator faults*, in *Proceedings. ICCON IEEE International Conference on Control and Applications*. 1989, IEEE: Jerusalem, Israel.

92. Clark, R.N., *Instrument Fault Detection.* IEEE Transactions on Aerospace and Electronic Systems, 1978. **14**(3): p. 456-465.

93. Pang, Z., et al. *Active fault tolerant control of networked systems with sensor fault*. in *2017 29th Chinese Control And Decision Conference (CCDC)*. 2017.

94. Naik;, G.R., *Advances in Principal Component Analysis*. 2018: Springer Singapore.

95. Hussein, W., A. Onsy, and I. El Sherif, *Health Monitoring of Electro-Pneumatic Controlled Systems Using Multivariate Latent Methods: An Experimental Validation.* SAE International Journal of MAterials and Manufacturing, 2014. **7**(1): p. 162-172.

96. Mazzoleni, M., et al., *Fault Detection via modified Principal Direction Divisive Partitioning and application to aerospace electro-mechanical actuators*, in *53rd IEEE Conference on Decision and Control*. 2014, IEEE: Los Angeles, USA.

97. Hussein, W.M., A. Onsy, and I. El Sherif, *Health Monitoring of Electro-Pneumatic Controlled Systems Using Multivariate Latent Methods: An Experimental Validation.* 2013. **7**(1): p. 162-172.

98. Zheng, N., et al., *A reformative PCA-based fault detection method suitable for power plant process*, in *Machine Learning and Cybernetics*. 2005, IEEE: Guangzhou. p. 5.

99. Chen, Z., et al., *A Distributed Canonical Correlation Analysis-Based Fault Detection Method for Plant-Wide Process Monitoring.* IEEE Transactions on Industrial Informatics, 2019. **15**(5): p. 2710-2720.

100. Chen, Z., et al., *A Cumulative Canonical Correlation Analysis-Based Sensor Precision Degradation Detection Method.* IEEE Transactions on Industrial Electronics, 2019. **66**(8): p. 6321-6330.

101. Aggarwal, C.C., *Neural Networks and Deep Learning*. 2018, New York, USA: Springer International Publishing.

102. Horton, M.P. *Real-time identification of missile aerodynamics using a linearised Kalman filter aided by an artificial neural network*. in *IEE Proceedings - Control Theory and Applications*. 1997. UK: IET.

103. Cybenko, G., *Approximation by superpositions of a signoidal function.* Mathematics of Control, Signals and Systems, 1989. **2**(4): p. 303-314.

104. Chii-Shang, T. and C. Chuei-Tin, *Dynamic process diagnosis via integrated neural networks.* Computers & Chemical Engineering, 1995. **19**(1): p. 747-752.

105. Moussavi, S.Z., et al., *PMDC Motor Speed Control Optimization by Implementing ANFIS and MRAC.* International Journal of Control Science and Engineering, 2014. **4**(1): p. 1-8.

106. Venkatasubramanian, V., et al., *A review of process fault detection and diagnosis Part III: Process history based methods.* Computers & Chemical Engineering, 2003. **27**(3): p. 327-346.

107. Duda, R.O., *Pattern classification and scene analysis*. 1st ed. 1973, New York: Wiley.

108. Puskorius, G.V. and L.A. Feldkamp. *Model reference adaptive control with recurrent networks trained by the dynamic DEKF algorithm*. in *IJCNN International Joint Conference on Neural Networks*. 1992. Baltimore, USA: IEEE.

109. Li, C., et al., *Vehicle States Estimation Based on Recurrent Neural Network and Road Constraints in Automated Driving*. 2018, SAE International.

110. Appel, M.A. and Q. Ahmed, *Intelligent Vehicle Monitoring for Safety and Security*. 2019, SAE International.

111. SAE, *Considerations for ISO 26262 ASIL Hazard Classification*. 2018, SAE International.

112. Birch, J., et al. *Safety Cases and their role in ISO 26262 Functional Safety Assessment*. in *SAFECOMP 2013*. 2013. Toulouse, France: Springer.

113. ISO, *ISO 25010: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)*, in *System and software quality models*. 2011, International Organization for Standardization: Switzerland.

114. ISO, *ISO 25000: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)*, in *Guide to SQuaRE*. 2014, ISO: Switzerland.

115. ISO, *ISO 25020: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)*, in *Measurement reference model and guide*. 2007, International Organization for Standardization: Switzerland.

116. ISO, *ISO 25030: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)*, in *Quality Requirements*. 2007, International Organization for Standardization: Switzerland.

117. ISO, *ISO 25040: Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE)*, in *Evaluation Process*. 2011, International Organization for Standardization: Switzerland.

118. Inoue, G. and Y. Ishida, *Development of Crawl Control*, in *SAE World Congress and Exhibition*. 2008, SAE International: Detroit.

119. Savitzky, A. and M.J. Golay, *Smoothing and differentiation of data by simplified least squares procedures.* Analytical Chemistry, 1964. **36**(8): p. 1627-1639.

120. Jolliffe, I.T., *Principal Component Analysis*. 2ed ed, ed. P.D. P Bickel, S Fienberg, K Krickeberg, I Olkin, N Wermuth, S Zeger. 2013, New York: Springer-Verlag New York Inc. 518.

121. Fraenkel, E. and F.M. White, *Singular Value Decomposition Tutorial* in *BE400*. 2009, Massachusetts Institute of Technology: Cambridge, MA, USA.

122. Smith, L.I., *A Tutorial on Principal Component Analysis*. 2002, University of Otago: Dunedin, New Zealand.

123. Jackson, J.E. and G.S. Mudholkar, *Control Procedures for Residuals Associated with Principal Component Analysis.* Technometrics, 1979. **21**(3): p. 341-349.

124. Jackson, J.E. and F.T. Hearne, *Hotellings T_M^2 for Principal Components - What About Absolute Values?* Technometrics, 1979. **21**(2): p. 235-255.

125. Hotelling, H., *The generalization of Student's Ratio.* Annals of Mathematical Statistics, 1931. **2**(3): p. 360-378.

126. Li, Y., X. Quin, and J. Guo, *Fault Diagnosis in Industrial Process Based on Locality Preserving Projections*, in *International Conference on Intelligent System Design and Engineering Application*. 2010, IEEE: Hunan, China.

127. Penha, R.M.L. and J.W. Hines, *Using Principal Component Analysis Modelling to Monitor Temperature Sensors in a Nuclear Research Reactor*. 2001, Instituto de Pesquisas Energeticas e Nucleares - IPEN: Sao Paulo. p. 1-11.

128. Hao, W., W. Tianzhen, and N. Mengqi, *A longitudinal-standardization multi-period PCA fault detection strategy based-on adaptive confidence limit*, in *IECON 2014*. 2014, IEEE: Dallas, TX, USA.

129. Eggett, D.L. and B.A. Pulsipher, *Principal Components in Multivariate Control Charts*, in *American Statistical Association Annual Meeting*. 1989, American Statistical Association: Washington D.C.

130. Jackson, J.E. and F.T. Hearne, *Relationships among coefficients of vectors used in principal components.* Technometrics, 1973. **15**(3): p. 601-610.

131. Data Safety Initiative Working Group, *Data Safety Guidance Version 3.0*. 2018, Safety Critical Systems Club: UK.

132. Metropolis, N. and S. Ulam, *The Monte Carlo Method.* Journal of the American Statistical Association, 1949. **44**(247): p. 335-341.

133. Zhang, Q., et al., *Development of robust interconnect model based on design of experiments and multiobjective optimization.* IEEE Transactions on Electron Devices, 2001. **48**(9): p. 1885-1891.

134. Raychaudhuri, S., *Introduction to Monte Carlo simulation*, in *2008 Winter Simulation Conference*. 2008, IEEE: Miami, FL, USA.

135. Furbringer, J.M. and C.A. Roulet, *Comparison and Combination of Factorial and Monte-Carlo Design in Sensitivity Analysis* Building and Environment, 1995. **30**(4): p. 505-519.

136. *INCA-MCE V2 Measurement and Calibration Embedded*, ETAS, Editor. 2017, ETAS: UK.

137. Expert, P., *How does the Land Rover Terrain Response system work?*, in *Official Blog*. 2017, Land Rover Fairfield: UK.

138. *Gaydon Proving Grounds*. 2018, Google. p. 52.182324, -1.496584.

139. *Model and simulate rotational and translational mechanical systems*. 2018; Available from: https://uk.mathworks.com/products/simdrive.html.

140. Fastenmeier, W. and H. Gstalter, *Driving task analysis as a tool in traffc safety research and practice.* Safety Science, 2006. **45**(9): p. 952-979.

141. Rasmussen, J., *Information Processing & Human-Machine Interaction*. System Science and Engineering. Vol. 12. 1986, Amsterdam: North-Holland.

Appendix A

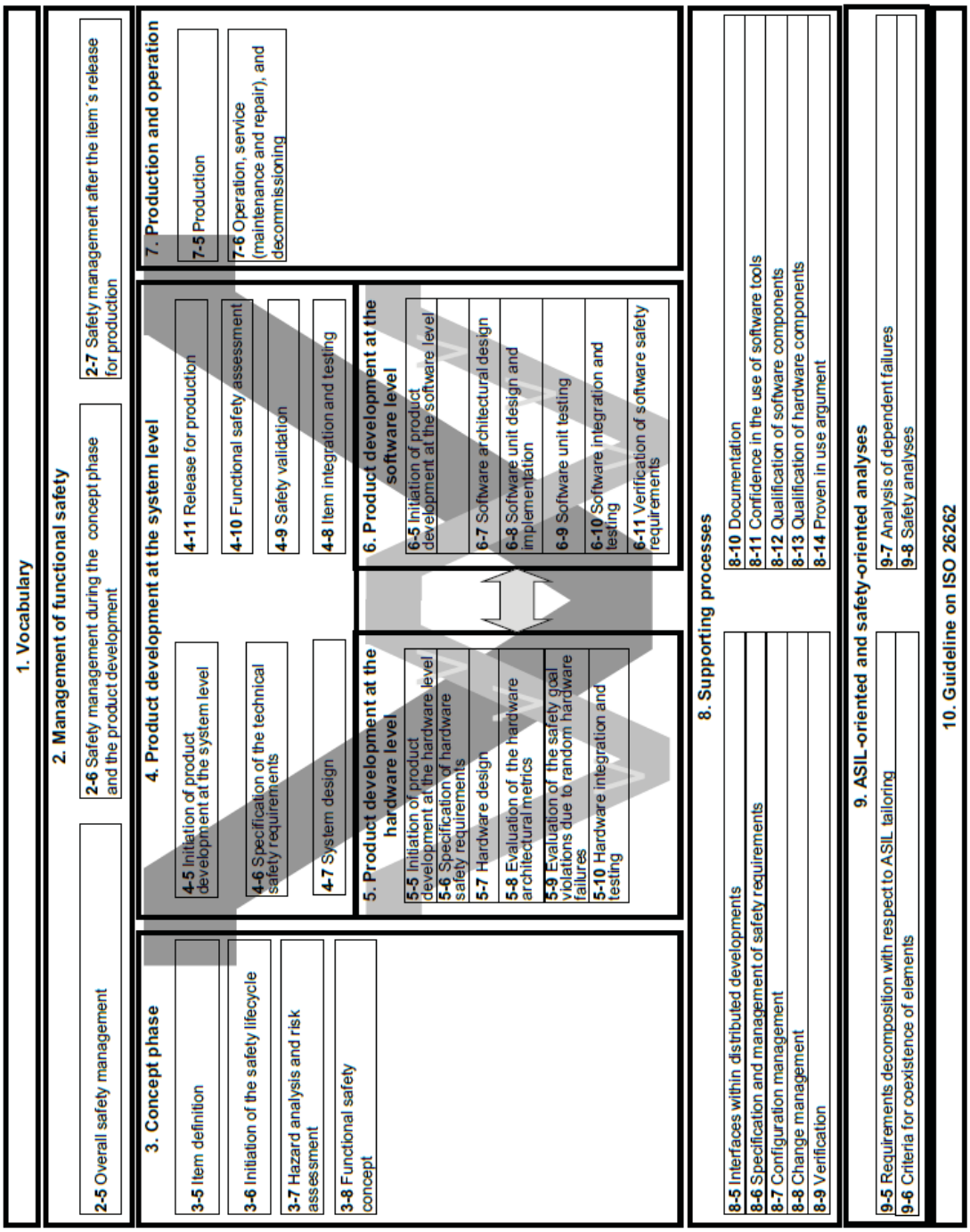

Figure A-1: ISO 26262 Safety Lifecycle [26]

Appendix B

B-1 – Data Collection

At the start of the project, seven sets of vehicle data were collected using a test vehicle at the Gaydon Proving Ground in Warwickshire, UK. The Proving Ground is home to a range of different driving circuits used to simulate various operational conditions that could be encountered by a production vehicle; Figure B-1 highlights these circuits. The data was collected in a 2014 Range Rover Hybrid development vehicle, captured using INCA software [136] as the vehicle was being driven on four different circuits of the Proving Grounds, with appropriate Terrain Response mode [137] selected. A description of the conditions under which each dataset was produced is shown in Table B-1

Table B-1: Vehicle Data driving conditions

Data	Circuit	TR mode	Description
1	Emissions	General	Tarmac, general drive; high speed.
2	Emissions	General	Tarmac Special programs off; high speed
3	Developing World	General	Light off-road; low speed
4	Developing World	Grass-Gravel-Snow	Light off-road; low speed
5	Cross Country and 4x4	Mud Ruts	Cross country, including boulders; low speed
6	Cross Country and 4x4	Sand	Sand and muddy; low speed, though much wheel spin.
7	Emissions	General	Tarmac, general drive; low speed.

Figure B-1: Map of Gaydon Proving Grounds with test circuits highlighted [138]**.**

B-2 – Range Rover Hybrid Powertrain

The vehicle model is based on the vehicle in which the data was collected, the 2014 Range Rover Hybrid. It utilises a parallel-hybrid powertrain with a 250 kW 3.0L Diesel V6 ICE coupled to a ZF 8-speed automatic transmission. Along this connection, and between the K0 clutch and automatic transmission, a 35 kW e-machine is located. The transmission is connected to a transfer case that distributes the torque to the front and rear axles, via their differentials. An inverter connects the high-voltage battery with the e-machine. Figure B-2 shows the schematic for this HEV powertrain [17].

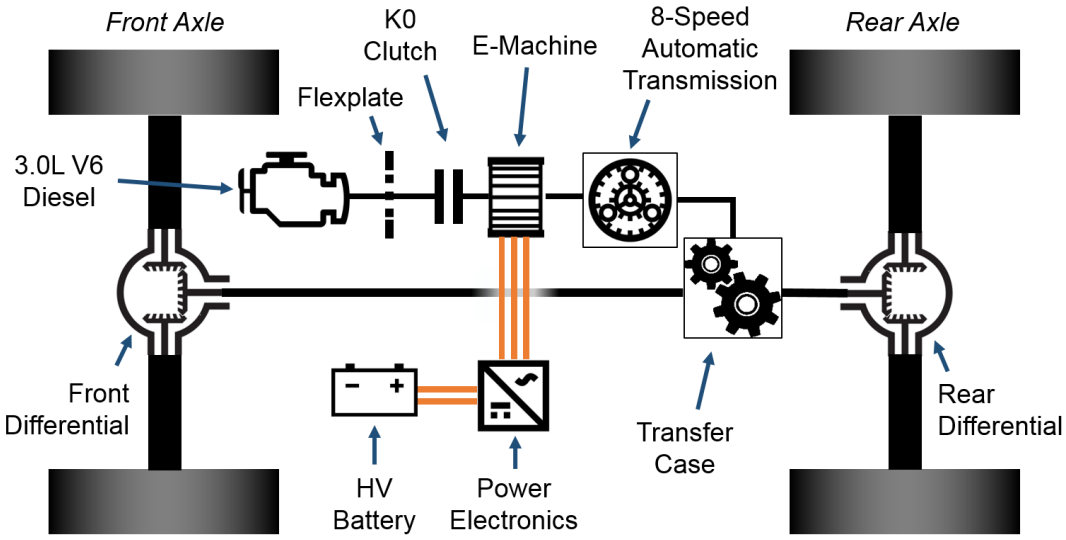

Figure B-2: Schematic of Range Rover Hybrid HEV powertrain. Black connections indicate mechanical coupling, and orange indicates HV electrical connections.

B-2.1 – Simscape Model

Using the available known parameters of the vehicle, a Simscape model of the Range Rover Hybrid was created in Simulink, with the top-level model layout is shown in Figure B-3.

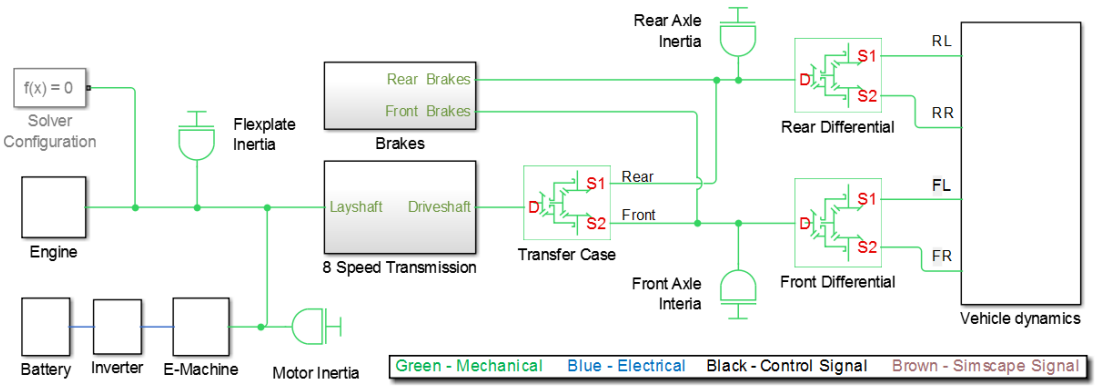

Figure B-3: Range Rover Hybrid Simscape Vehicle Model

The model contains rotating mechanical connections (green), electrical connections (blue), Simulink control signals and calculations (black), and Simscape physical signals (brown). A high-fidelity model is not necessary for the purpose of testing safety concepts, especially at

the expense of runtime. A fast runtime was an important design driver, as it would allow a fast throughput of automated design of experiments test schedules, ideal for any verification, data collection, and comprehensive performance analysis of safety concepts. Additionally, since the safety concepts will be novel, a high-fidelity model would contain much complexity, slowing research time.

Many of the top-level components are modified versions of example models that are included in the Simscape package. The initial model used a distributed control strategy in order to track a target vehicle speed, whereby an individual PID controller and surrounding gains and saturation limits would produce a control signal for each torque actuator (ICE engine, e-machine, brakes), based on the error created between target vehicle speed and current vehicle speed. While this was not ideal for testing safety concepts, seeing as the effective functional software was distributed and not controlled by a central supervisory controller, this allowed rapid development of the individual components. These controllers were then calibrated together once each component performed correctly and the powertrain is assembled. Problems with a fully cohesive clutch control program and stable runtime led to the omission of a K0 Clutch for simplicity, which only affects the model during gearshift events. An 8-speed epicyclical automatic transmission model was attempted, but a working version was unable to be completed. Therefore, a simulated 8-speed automated manual gearbox with synchronisers was created, which is acceptable within the scope of the project. However, this component, required very small simulation time steps during gear changes to avoid diverging continuous states, leading to longer runtimes. The brakes were simply modelled with negative ideal torque actuators, as was engine friction using a tabulated friction model supplied by Jaguar Land Rover. The vehicle dynamics uses a half-car model, since this project is focused on forward longitudinal motion (as per the safety goal).

The inverter is, in fact, a DC:DC inverter, and the e-machine technically a DC motor/generator as opposed to AC. The e-machine is an ideal torque source controlled directly by the e-machine controller, and the required current draw to deliver the torque demand and account for mechanical and electrical losses would be calculated. Such a ‘backwards’ strategy differs from the actual powertrain, in which the torque demand would instead be received by the inverter, which then decides the correct amount of current to drive the e-machine. However, within the scope of this project, the torque demand simply needs to be actualised by the e-machine, and how this is calculated does not matter. The equations governing the calculation of current draw are as follows:

$$I_{em} = \begin{cases} \dfrac{P_{em} + (\tau_{em}^2 \eta_{el})}{V_{inv}} & if\ P_{em} > 0 \\ \dfrac{P_{em} - (\tau_{em}^2 \eta_{el})}{V_{inv}} & if\ P_{em} < 0 \\ 0 & if\ P_{em} = 0 \end{cases} \quad \text{(B-1)}$$

where I_{em} the current draw by the e-machine, V_{inv} the voltage from the inverter, P_{em} and τ_{em} the mechanical power and torque generated by the e-machine, and η_{el} a constant electrical efficiency, which includes the resistance of the system and also the proportional torque-to-current coefficient. The numerator being subtracted from P_{em} is a simple model of electrical losses, proportional to the square of the current. When $P_{em} > 0$, the e-machine is in motoring mode so these losses are added to the total motor current draw. When $P_{em} < 0$, the e-machine is in generating mode, so the current draw is negative, and the current losses must be subtracted. Figure B-4 shows the e-machine model, based on the model provided by the Simulink Simscape Driveline package [139]:

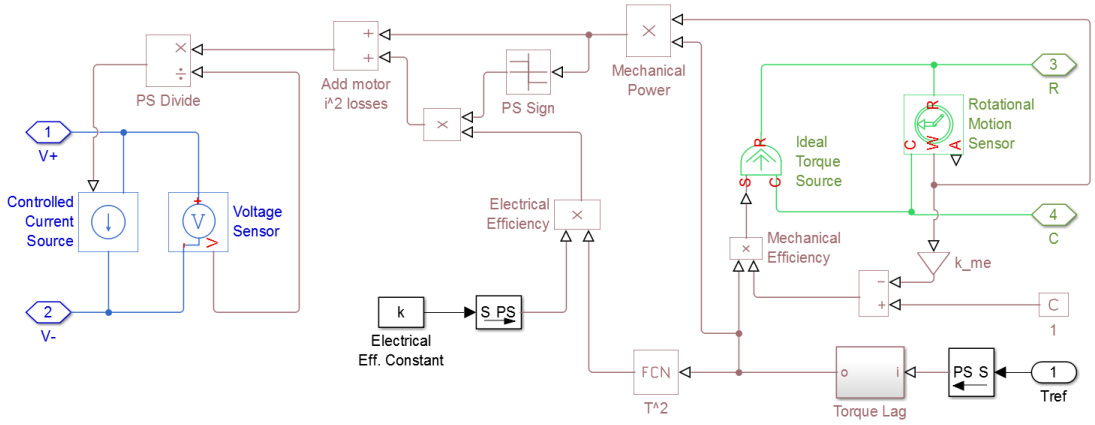

Figure B-4: E-machine Simscape Model, with controller torque demand input (black, bottom right), mechanical connection (green), and electrical connection (blue).

B-2.2– Model Validation

The simplified Simscape model of the Range Rover Hybrid powertrain was validated using vehicle test data collected at the Gaydon Proving Ground [138]. The control target of each of the individual actuator controllers is a desired vehicle speed. A recorded speed trace was used as the control target for a simulation, with the notion that if the model vehicle speed closely matches the recorded vehicle speed, a valid comparison of the model and recorded powertrain torque outputs could be made. The transmission was controlled by the recorded gear position signal from data. It was expected that there would be some error in the torque comparison

owing to the simplified nature of the vehicle model versus a production vehicle. However, the aim of this exercise was to identify whether the torque outputs are generally within a reasonable range at the vehicle level, not that they necessarily match up entirely at all points.

The dataset chosen to for validation was dataset number 2 (see Table 4-4), due to the fact that the least number of noise factors are present, such as wheel spin from varying road surface conditions and wading, and since it has special programs turned off that could interfere with torque demand and delivery. These are not accounted for in the model, with the exception of road surface through a set constant friction coefficient similar to the tarmac of the Emissions Circuit. Figure B-5 shows the vehicle speed for both recorded and model vehicle speed over a period in dataset 2; only a portion of the dataset is shown for detail in subsequent figures.

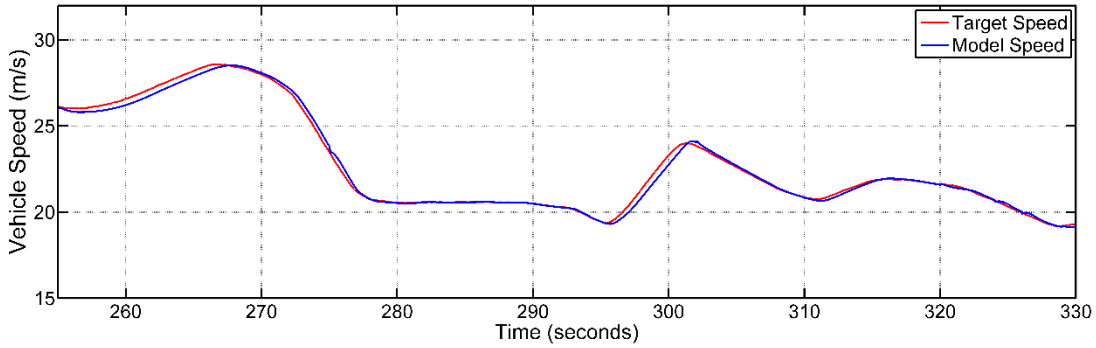

Figure B-5: Simscape model tracking recorded vehicle speed in Dataset 2.

Figure B-5 shows the model is able to track the vehicle data very well, with the largest error reading below 0.5 m/s. There is a small amount of lag, visible in the acceleration events around 260 s and 300 s, which can be attributed to transient dynamics and lag in the controllers; while this could be fixed with some feed-forward control or recalibration, for the purposes of this project this is sufficient. Figure B-6 compares the total torque output of the model actuators (no brake) with the combined torque output from the recorded vehicle data, measured at the transmission layshaft.

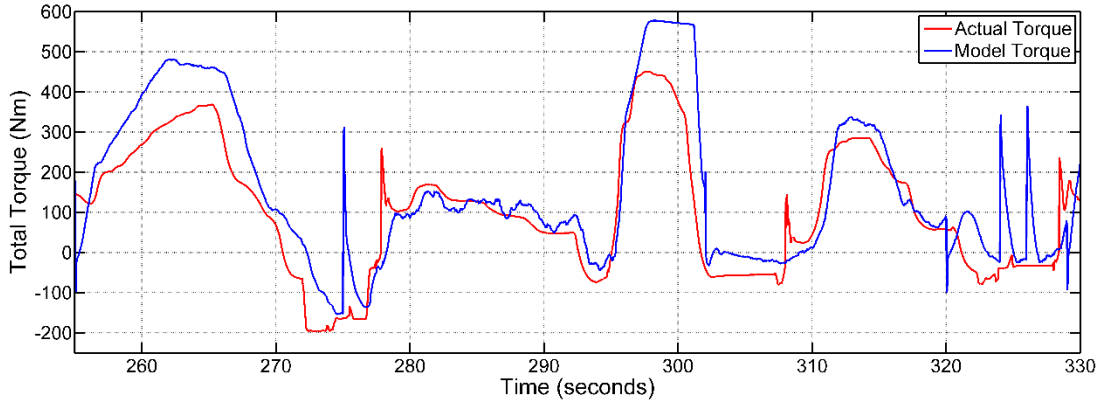

Figure B-6: Model powertrain torque output compared to total torque output from recorded vehicle data.

In general, the torque outputs from the model are within reasonable range of the recorded data. Particularly at steady state, the model torque stays very close to the recorded torque (averaging 4.6% of the maximum 600 Nm, or 27.6 Nm error-per-second between 279 s and 295 s, for example) with some noise that is likely due to the various controllers working individually and not cohesively. There are, however, two areas where the model differs from real world data. First is during gearshift events, such as at 275 s, and a number of times between 320 s and 330 s. The result is a large, sudden increase in torque as vehicle speed drops or increases relative to the recorded vehicle speed in between gear changes, leading to an undesired reaction by the controllers. The model does, however, quickly settle back to a reasonable torque output following the gearshift. The second discrepancy occurs during the acceleration events seen in Figure B-5, where torque tends to be slightly higher in the model than the recorded data. Between 250 s and 275 s an average error of 16.4% is seen, and between 295 s and 303 s an average error of 23.2% is experienced. This is primarily due to a combination of modelling limitations in the ICE engine, and a reactive control strategy attempting to 'catch up' with the velocity target. The test vehicle was a development vehicle, whereas some vehicle parameter values (such as the vehicle mass and component inertias) are from the production vehicle. However, in general, the vehicle model is a close match of overall performance of the test vehicle recorded data, close enough to be considered a valid test bed for qualitative concept testing.

B-3 – Twin-EV Powertrain Model and Functional Software

The rapid development of the Range Rover Hybrid powertrain in the first year of research led to a distributed control system developed to match the general performance of the test vehicle. However, no design information was available regarding the central functional software of the

vehicle, so much of the calibration was a system identification exercise. As is apparent in Figure B-6, the calibration lacks refinement, and sometimes the systems could work against each other. It was planned, therefore, to change the control system such that a total torque demand was calculated by a controller, and the responsibility of meeting that demand split between the e-machine and ICE; this matches the Continuous Torque Demand Monitoring concept adapted from E-Gas, and is the basis for the current functional safety concept used by the OEM. Before this modification could be implemented for the Range Rover Hybrid, at the start of the second year of research, the OEM provided control software for a pre-production twin-EV powertrain, which led to the decision to work on this platform instead of the hybrid one.

It is at this point the investigation in Section 3-3 is detailed.

Appendix C

C-1 – Future Research

The investigations in this project were completed insofar as meeting the objectives set out in the introduction. Throughout the thesis, it has been alluded to that plans have been made for future work. The overall aim of the future research activities is different for the two concepts: for the adaptive safety monitor, the aim is to deploy it on a production vehicle, which means addressing the verification and validation concerns; for the PCA safety monitor, it is continuing the investigation to overcome the limitations of the concept. Plans for these future investigations will be discussed in this section. Verification and validation concerns will not be looked at for the PCA safety monitor until the functional suitability limitations are overcome.

C-1.1 – Adaptive Safety Monitor

The adaptive safety monitor showed the most promise of the concept candidates during the main investigation. In order to deploy the concept on a road vehicle, the validation and verification concerns in Chapter 6 need to be addressed.

C-1.1.1 – Driver Task Analysis

A concern was identified with ensuring the adaptive module adapts in a way that is commensurate with driver expectation. If it adapts too slowly robustness of the system suffers, and if too fast it could adapt to a hazardous malfunction faster than a human can, leading to a safety goal being violated without detection. Therefore, a driver task analysis has been suggested as a way to validate correct driver behaviour.

A driver performance testing framework has been proposed by Fastenmeister and Gstlater [140] through a driver task analysis and driver requirement assessment called SAFE (‘Situative Anforderungsanalyse von Fahraufgaben’, or ‘Situational Requirement Analysis of Driving Tasks’), for the purpose of aiding traffic safety research. SAFE seeks to approach driver task analysis by combining objective task properties (including the physical nature of the task, the subject matter etc.) with the behavioural requirements (mental and psychomotor performance, which establish the target level of task achievement). Through defining the road traffic

situations being investigated precisely, driving tasks can be derived as analytical units and analysed through a human information processing model, which the SAFE framework proposes using Rasmussen's model of information-processing [141]. These activities lead to a compiled list of all the necessary behavioural requirements – such as driver reaction time – that can be used to drive tuning of the adaptive safety monitor. The outline of the SAFE framework is shown in Figure C-1.

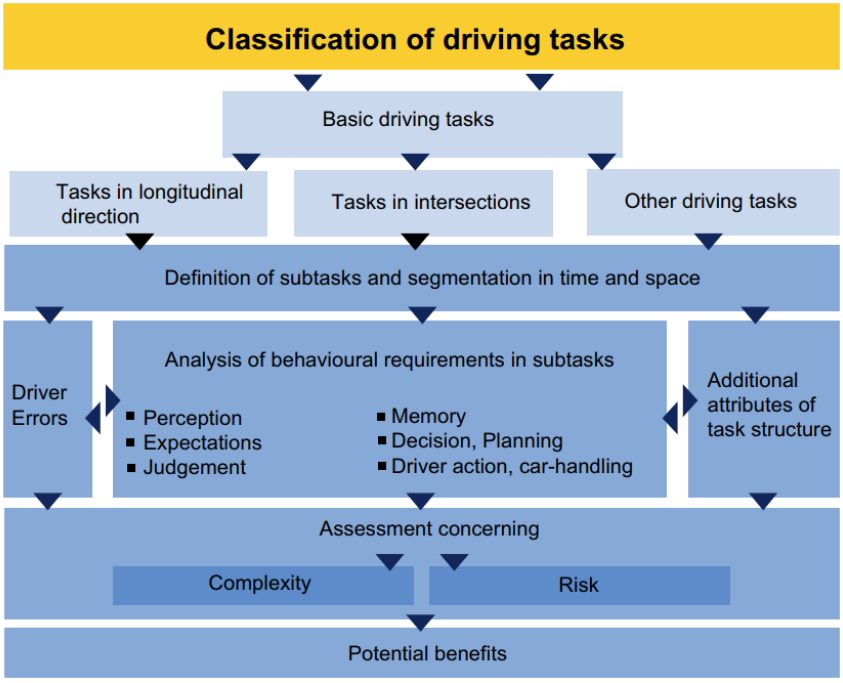

Figure C-1: Structure of SAFE driver task analysis and behavioural requirements derivation [140]

Surprisingly, but perhaps predictably, Rasmussen's model of informational processing resembles the two-stage adaptive algorithm in many ways, with the inclusion of long and short-term memory functions, generation of expectation, decision making, and error checking aspects; this is encouraging as it gives further credibility to the foundational reasoning of the adaptive safety monitor. A schematic of a model for deriving behavioural requirements as part of the SAFE driver task analysis – modified from Rasmussen's model – is shown in Figure C-2.

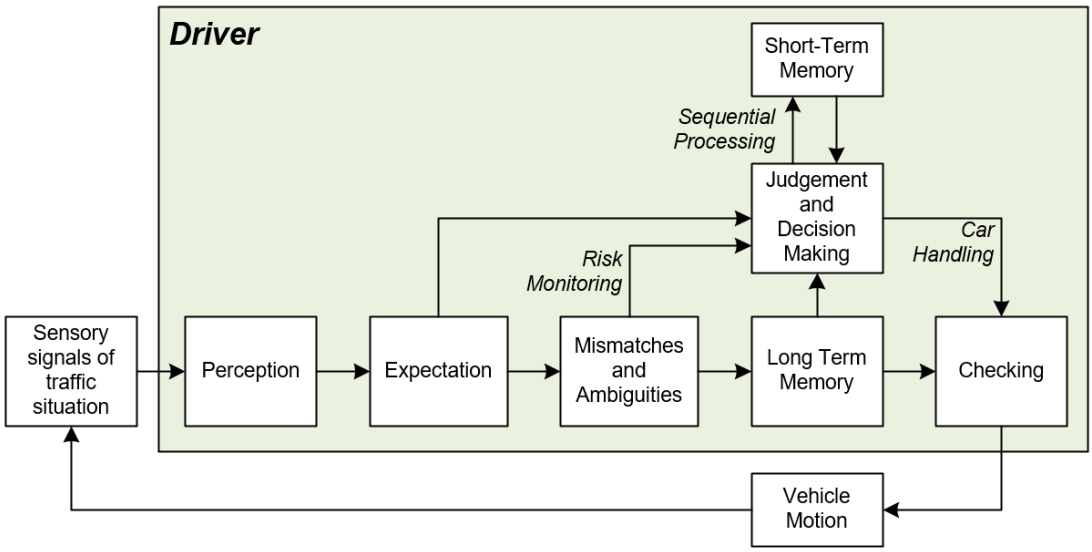

Figure C-2: SAFE behavioural requirement derivation model for driving tasks [140].

The driver task analysis will be an interesting avenue to explore in future research, as it leads further into human-machine interaction, and how this can be leveraged to both improve safety, and maximise the results of the efforts spent by automotive OEMs developing safe and reliable systems. It will also form a justifiable basis from which to validate adaptive safety monitor parameters.

C-1.1.2 – Systematic Test Environment

Another concern was that of the verification of the simplified safety model. If the safety software is simplified in such a way that a noise factor causes a sudden switch between modes that changes the nominal torque error very suddenly, it would cause a sudden change in the offset torque error too, which either results in a false positive, or a torque error that would be adapted to over time. If a true fault were to occur during this period, it could be possible that an inaccurate offset torque error is produced, depending on the change in nominal torque error. Therefore, the safety model needs to be verified in such a way that this does not occur, as well as the other verification procedures.

The suggested method would be to build a systematic testing environment to identify these potential issues in the software, their source, and let the engineer decide whether to adapt to them or add them to the simplified safety model, at the expense of simplicity. The investigation will seek to determine if a full vehicle model is needed for the systematic testing environment, or if simply a software perturbation or Monte Carlo simulation [132] would be enough to identify these types of noise factors. Combined full-factorial and Monte Carlo approaches have

also seen success previously [135]. The experiment would be completed in Matlab/Simulink to make use of the existing code of the adaptive safety monitor.

C-1.1.3 – Parameter Scheduling

The investigation will also look at improving the performance of the adaptive safety monitor, as discussed in Chapter 6. A set of adaptive parameters will often be the "best compromise" in order to meet functional suitability requirements, but will not necessarily be the most robust, or least complex, or won't take full advantage of the adaptive safety monitor capabilities. As mentioned in Chapter 6, subdividing the operating range into cells, each with a set of adaptive parameters calibrated to it for optimised localised performance. In principle, this is similar to the PCA subdivision concept which significantly improved performance of the PCA safety concept. Such a method is expected to enable even further simplification of the safety software through improved localised – and thus global – robustness.

The process will effectively involve performing optimisation processes (as detailed in Chapter 4) on tests covering limited operational boundaries to produce a set of localised adaptive parameter values. Within a single function, this could be localised calibration regions, and on discrete functions this could be separate adaptive safety monitor modules altogether. When the monitor is deployed, these values can then be referenced in a lookup-table based on those operating conditions for optimum performance. A necessary caveat, however, would be that only readily available ASIL inputs (e.g. pedal position and vehicle velocity) can be used for parameter scheduling, as performance gains would likely be minimal for the effort required to conduct additional ASIL verification on another input during development

Furthermore, investigation into the potential of blended or interpolated parameters across the operating space could yield even further improved performance, balancing outright functional suitability with hardware storage and computational requirements, but could also bring with it further verification concerns. Similar blending studies were initialised as part of the PCA safety monitor investigation, but tabled due to more promising avenues of research; those initial findings could potentially be better suited for this avenue of future research.

C-1.2 – PCA Safety Monitor

C-1.2.1 – PCA Module Architectures

The most significant limitation of PCA is that of handling temporal-dynamics found within the model. Initial attempts were made at using one or many PCA modules in an arrangement such that the time-based dynamics can be separated or accounted for explicitly, and more investigation could yield alternative arrangements that overcome some of the PCA concept limitations; identification of possible research avenues were shown in Chapter 5. One example, is to extract and explicitly model temporally-dynamic functions in the safety software outside of the PCA safety monitor, on either the output or the inputs of the PCA module.

The investigation could also look at whether separate PCA modules are useful for some discrete noise factors, such as switching between driving modes. Some initial work was conducted on this topic during the Chapter 5 investigation, whereby three pedal maps could be switched between. Using a discrete input such as drive mode led to data resembling a layered surface, with some sections of the data surfaces intersecting and passing through one another. This leads to a very difficult dataset to train PCA models on, with poor performance. Instead, an arrangement with separate PCA modules for each discrete operational mode could extract the nonlinearity and better manage the transition between these modes or whichever discrete signals are noise factors.

In summary, the investigation will seek to find what software systems or functional software components could benefit from a PCA-based safety monitor in such a way, which dynamics can be extracted and modelled explicitly.

C-1.2.2 – Using last eigenvalue for determining T^2 scalar c_k

One of the novel developments during the PCA investigation was that of method of normalising all cell outputs through the derivation of the T^2 scalar, c_k, for each subdivision cell k. Without scaling, the T^2 values after offset would vary significantly, seen before scaling was introduced in Chapter 5. Prior to this development, a number of possible avenues were explored for deriving this scalar, the most promising of which is related to the smallest eigenvalue (the m^{th} eigenvector or principal component in an m-dimensional PCA model). To illustrate this, PCA models were derived for a four-dimensional plant (pedal position, velocity, actuator temperature, and torque demand) but T^2 normalisation through scaling was not applied. Then, the PCA module was tested on a set of data that conducted input value sweeps,

and the cells ordered by increasing value of the last (4th) eigenvalue; Figure C-3 compares these to the maximum T^2 value for a constant 100 Nm fault each cell outputs.

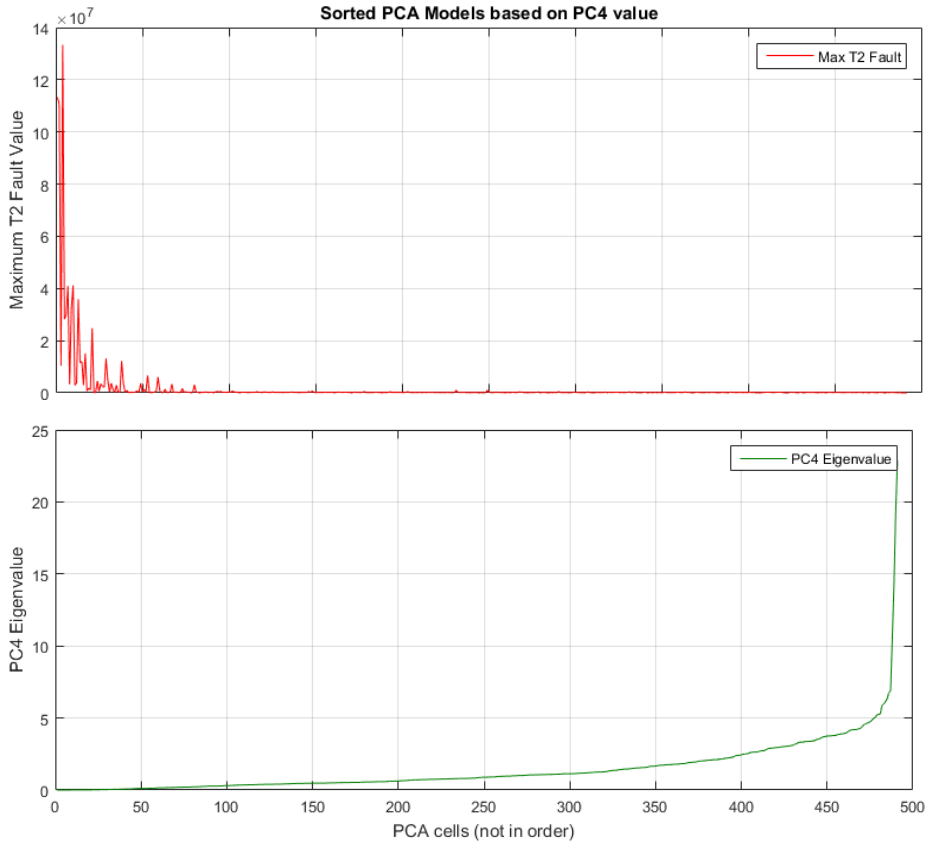

Figure C-3: Maximum T^2 value in each cell for 100 Nm fault, sorted in ascending order by PC4 eigenvalue, scaled linearly on y-axes.

Figure C-3 does seem to show some relationship between PC4 and maximum T^2 fault, where the cells with the largest maximum T^2 values roughly correlate to where the smallest PC4 eigenvalues are. Since the eigenvalues appear to change logarithmically, the y-axes of Figure C-3 are scaled logarithmically in Figure C-4.

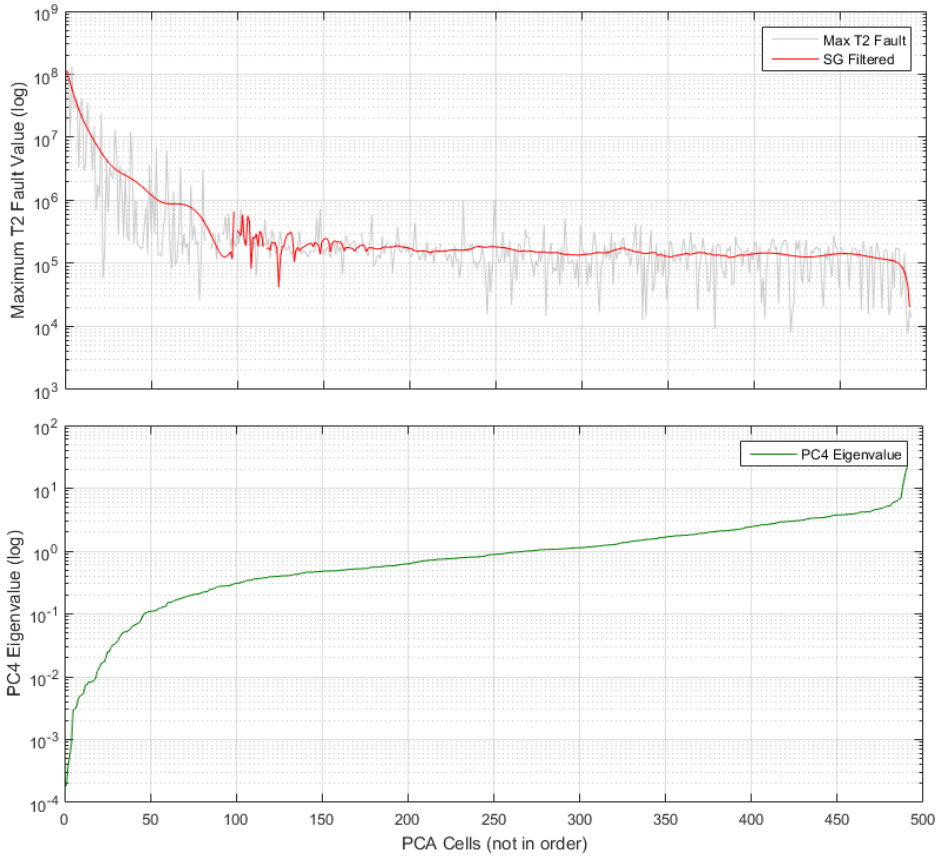

Figure C-4: Maximum T^2 value and (unfiltered and Savitzky-Golay Filtered) in each cell, sorted in ascending order by PC4 eigenvalue, scaled logarithmically on y-axes.

Savitzky-Golay filtering [119] was applied to better identify the general trend the maximum T^2 values follow by ascending PC4 value. While the unfiltered maximum T^2 values are quite noisy, through filtering it is clear that there is some inverse relationship between maximum T^2 value and eigenvalue. It is theorised that there is some way that the last eigenvalues can be used to better predict c_k for each, possibly reducing development time and improve model accuracy.