Uncertainty Management Framework for Automotive Crash Applications

Jonas Siegfried Jehle

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Gutachter:

1. Prof. Dr. rer. nat. Matthias Gerdts
2. Prof. Dr. rer. nat. Stefan Volkwein

Die Dissertation wurde am 30.11.2021 bei der Universität der Bundeswehr München eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 25.04.2022 angenommen. Die mündliche Prüfung fand am 10.05.2022 statt.

Istud quod tu summum putas gradus est.
— Lucius Annaeus Seneca. Epistulae Morales ad Lucilium. 62-65.

Danksagung

Die vorliegende Arbeit entstand während meiner Zeit als Doktorand an der Universität der Bundeswehr München und im ProMotion Programm der Bayerische Motoren Werke Aktiengesellschaft (BMW AG) in der Hauptabteilung Fahrzeugsicherheit. An dieser Stelle bedanke ich mich bei allen Personen, die mich während dieser Zeit unterstützt und motiviert haben.

Mein ganz besonderer Dank gilt meinem Doktorvater Prof. Dr. Matthias Gerdts von der Universität der Bundeswehr München für die Betreuung dieser Arbeit, die permanent ermutigende Unterstützung sowie für die eingeräumten Freiräume. Ich bin stets motiviert zur Universität gefahren, um meine Forschungsfortschritte zu präsentieren. Gerne habe ich seine hochwertigen und fördernden Ratschläge angenommen. Unvergessen bleiben für mich die intensiven Gespräche auf wissenschaftlicher und persönlicher Ebene als bereichernder und konstruktiver Austausch.

Ich danke zudem meinem Betreuer Dr. Volker Andreas Lange von der BMW AG. Ohne seine beharrliche und fachliche sowie menschliche Betreuung wäre diese Arbeit nicht zustande gekommen. Seine enorme Expertise gepaart mit der positiven Leichtigkeit, dieses Fachwissen mit großer Leidenschaft auf mich zu übertragen, werde ich mir weiterhin zum Vorbild nehmen. So habe ich diese Dissertation niemals als Last und Arbeit empfunden, sondern vielmehr als Herausforderung, diverse Aufgabenstellungen ehrgeizig und lösungsorientiert anzugehen.

Ebenfalls danken möchte ich meinem Vorgesetzten Franz Hoiss von der BMW AG. Vertrauen und Offenheit spiegelte die Zusammenarbeit. Jederzeit stellte er mir die notwendigen Ressourcen bereit, die zum Gelingen dieser Arbeit erforderlich waren. Wichtige Partner waren zudem die Mitarbeiter der Hauptabteilung Fahrzeugsicherheit mit ihrer Erfahrung und Hilfestellung. Dazu gehören vier Masteranden, die ich während dieser Zeit betreute und deren Ergebnisse teilweise in dieser Dissertation berücksichtigt wurden.

Als leidenschaftlicher Sportler und Fußballer attestiere ich dem Team meiner Doktorarbeit, angefangen vom Doktorvater als Trainer und den beteiligten Vorgesetzten sowie Kollegen als Spieler, ein weltmeisterliches Prädikat. Für diese Konstellation bin ich der BMW AG dankbar.

Zu guter Letzt fand ich bei meiner Familie und meinen Freunden Unterstützung, Aufmunterung und Halt. Soweit ich denken kann, waren sie immer für mich da, hatten offene Ohren und schufen den richtigen Ausgleich, den ich benötigte, um auf der richtigen Spur zu bleiben.

Hervorheben möchte ich meine Eltern Dagmar und Dieter Jehle und meine Schwester Hannah. In meiner Heimat in Neuburg an der Kammel im Landkreis Günzburg und im Kreise meiner Familie finde ich Rückhalt, Energie, Zeit zum Entspannen und Abschalten. Dort habe ich gelernt, logisch, ambitioniert und selbstbewusst an Herausforderungen heranzugehen, nicht aufzugeben und geeignete Lösungen zu finden. Diese Eigenschaften waren der Schlüssel zu meinem bisherigen Werdegang, zu dem auch diese Arbeit zählt.

Deutsche Zusammenfassung

Diese Arbeit stellt ein Rahmenkonzept für den Umgang mit Unsicherheiten von Fahrzeugcrashanwendungen vor. Der Bedarf an diesem Rahmenkonzept ist auf die vielfältigen Unsicherheiten von unterschiedlichen Quellen in Crashszenarien zurückzuführen, die sich entscheidend auf die Ergebnisse und damit auf die Sicherheit von Fahrzeugen auswirken. Ingenieure müssen sich daher mit den Unsicherheiten von Crashmodellen, die rechenintensiv sein können, auseinandersetzen.

Das Rahmenkonzept zeigt verschiedene Möglichkeiten auf, um Unsicherheitsanalysen zu befähigen. Der passende sowie schnellste Weg hängt vom Typ des zu untersuchenden Objekts ab. Modelle mit kleinen Rechenzeiten können direkt weiteren Analysen unterzogen werden. Modelle mit hohem Rechenaufwand können zunächst optional durch technisches Wissen vereinfacht werden. Das verbleibende Modell wird dann durch Approximationsmodelle ersetzt, welche sich durch geringe Rechenzeiten auszeichnen. Darüber werden anschließende Analysen zeitlich ermöglicht. Die Qualität der Approximationsmodelle kann unter Berücksichtigung vieler Parameter leiden. Screeningverfahren können eingesetzt werden, um den Eingaberaum vor Erstellung der Approximationsmodelle auf die wichtigen Dimensionen zu reduzieren. Hinterher lässt sich mithilfe der erstellten Approximationsmodelle eine Sensitivitätsanalyse rapide durchführen. Diese bewertet die Relevanz einzelner Parameter und deren Interaktionen. Als letzten Schritt beziffert die Unsicherheitsquantifizierung Auswirkungen der unsicheren Parameter auf definierte Ergebnisgrößen und bestimmt deren Beschaffenheit.

Finite-Elemente-Simulationen, die am häufigsten untersuchten Modelle im Crashbereich, wird besondere Aufmerksamkeit gewidmet. Das Rahmenkonzept wird demnach an diese angepasst. Als erstes werden die Simulationen als mathematische Abbildungen unterschiedlicher Informationsebenen definiert. Der zweite Schritt besteht aus Metamodellen, Vertretern von Approximationsmodellen, die sich den genannten Abbildungen angleichen. Als nächstes werden varianzbasierte Sensitivitätsanalysen eingesetzt. Zu guter Letzt wird die Evidenztheorie für die Unsicherheitsanalyse benutzt. Die vom Rahmenkonzept vorgegebene Kopplung dieser Methoden macht eine Unsicherheitsanalyse erst möglich. Diese ist dadurch zeitlich machbar, kann flexibel gestaltet werden und liefert eine für Ingenieure auswertbare Visualisierung.

Die Realisierbarkeit des Rahmenkonzept wird anhand eines realen Projekts geprüft. Das Untersuchungsobjekt ist eine Finite-Elemente-Simulation eines Seitenpfahltests. Die einzelnen Komponenten des Rahmenkonzepts werden Schritt für Schritt angewandt. Metamodelle, die Ergebnisgrößen verschiedener Informationsebenen approximieren, werden verglichen. Das beste Metamodell wird dann für die weitere Analyse verwendet, d.h. für die Sensitivitäts- und Unsicherheitsanalyse. Ingenieure können mithilfe der Ergebnisse dieser Analysen beurteilen, wie sicher und robust ihre Systeme sind. Das Rahmenkonzept trägt somit erfolgreich zur Bewertung und Verbesserung von Strategien für die passive Sicherheit bei.

Abstract

This thesis presents a general framework for managing uncertainties of vehicle safety applications. The need for this framework stems from the multiple uncertainties from different sources present in crash scenarios that have a decisive effect on the results and thus on the safety of vehicles. Engineers must therefore confront these uncertainties inherent in crash models that can be computationally expensive.

The framework shows different ways to enable uncertainty analysis. The appropriate route depends on the type of the object under investigation. Resource efficient models can be directly subjected to further analysis. Resource inefficient models can first be optionally simplified by technical expertise. The model under consideration is then replaced by approximated response surfaces characterized by low computational times. In terms of time, this enables subsequent analyses. When the model includes many parameters, the quality of the approximated response surfaces may suffer. Screening can therefore be used to reduce the input space to the important dimensions before creating the approximated response surfaces. By using them afterwards, sensitivity analysis can be performed rapidly. This evaluates the relevance of the individual parameters and their interactions. As a final step, uncertainty propagation measures the effects of the uncertain parameters for specified quantities of interest.

Finite element simulations, the most commonly studied models in crash, are devoted special focus. Accordingly, the framework is adapted to them. At first, the simulations are defined as mathematical mappings of different levels of information. The second step consists of metamodels, representatives of approximated response surfaces, which approximate the mappings. Next, variance-based sensitivity analysis is performed quickly by using the metamodels. In the end, the Dempster-Shafer evidence theory is used for uncertainty propagation. The framework that structures these methods makes this whole procedure efficient, temporally feasible, and flexible. It also yields a reasonable visualization for engineers.

The practicability of the framework is examined on the basis of a real-world project. The object of investigation is a finite element simulation of a side pole test. The individual components of the framework are applied step by step. Metamodels approximating quantities of interest at different levels of information are compared. The best metamodel is then used for further investigations, i.e. sensitivity and uncertainty analysis. Engineers can use the results of these analyses to assess the level of maturity of their systems. The framework thus successfully contributes to the evaluation and improvement of passive safety concepts.

Contents

Tables

Figures

Essential abbreviations

ATC	Anthropomorphic test device, dummy
AvP	Actual versus predicted plot
BMW	Bayerische Motoren Werke
CAD	Computer-aided design
CDF	Cumulative distribution function
CN	Nodes of a component of a FE model
CPU	Central processing unit
EE	Elementary effects
EN	Entire nodes of a FE model
ET	Evidence theory
FE	Finite element
GPR	Gaussian process regression
kPCA	Kernel principal component analysis
KR	Key result, i.e. scalar quantity of interest, of a FE model
MLR	Multiple linear regression
MNN	Multi-target NN
MOR	Model order reduction
MSE	Mean squared error
MTR	Multi-target regression
NCAP	New Car Assessment Program
NHTSA	National Highway Traffic Safety Administration
NN	Neural network
PCA	Principal component analysis
PCE	Polynomial chaos expansion
R^2	Coefficient of determination
RC	Regressor chains-based
SH	Time history of a single node or a processed quantity of a FE model
SNN	Scalar NN
ST	Single-target

Introduction

1.1 || Motivation and problem definition

Nowadays, over a billion vehicles are driven on roads all over the world, see [SG10]. In fact, this number is expected to further increase, see [Gro16]. Many vehicles mean many kilometers traveled. During these numerous trips, dangerous traffic situations can occur that may lead to accidents — or fatalities in the worst case. Vehicle safety aims to prevent accidents, mitigate occupant injuries, or even avert deaths. Among other factors, it is still instrumental in making the roads safer. For example, the fatality rate per vehicle miles traveled in the United States significantly decreased from 1975 to 2018 thanks to safety programs and improvements, see [Nat19].

Vehicle safety can be divided into active, passive, and integral categories, see [KGD10]. Active safety serves drivers in pre-emptive accident avoidance. This refers primarily to driver assistance systems and technologies. These include, for instance, electronic stability program, anti-lock braking system, rear view camera, and driver drowsiness detection. The passive counterpart comprises components designed to reduce or circumvent the risk of injury in events of accidents. These mainly encompass physical structures and restraint systems, most notably crumple zone, seat belt and its tensioner, airbags, head restraints, and child seats, see [Bau98]. Integral safety combines passive with active features, e.g. restraint systems profit from early detection of accident situations, see [KH09].

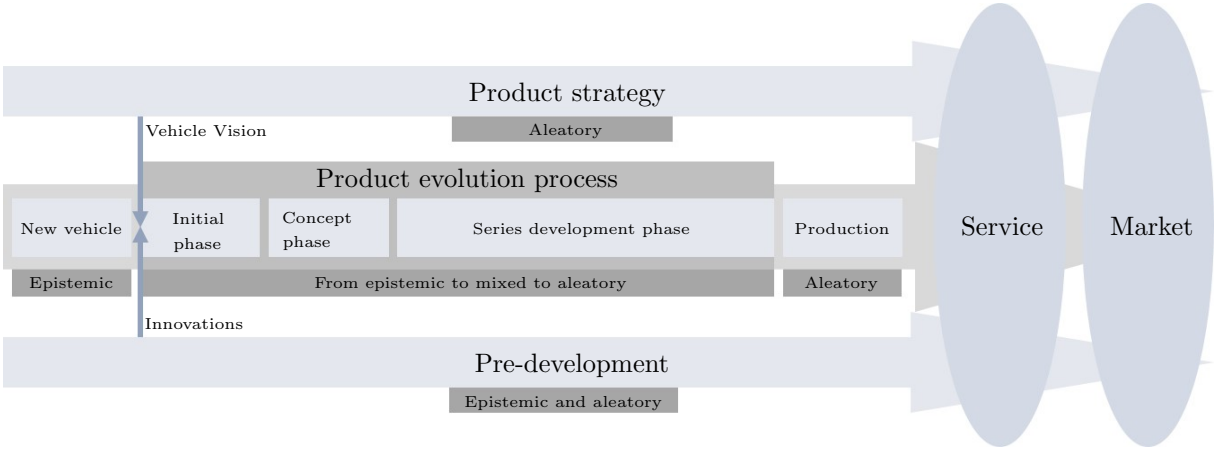


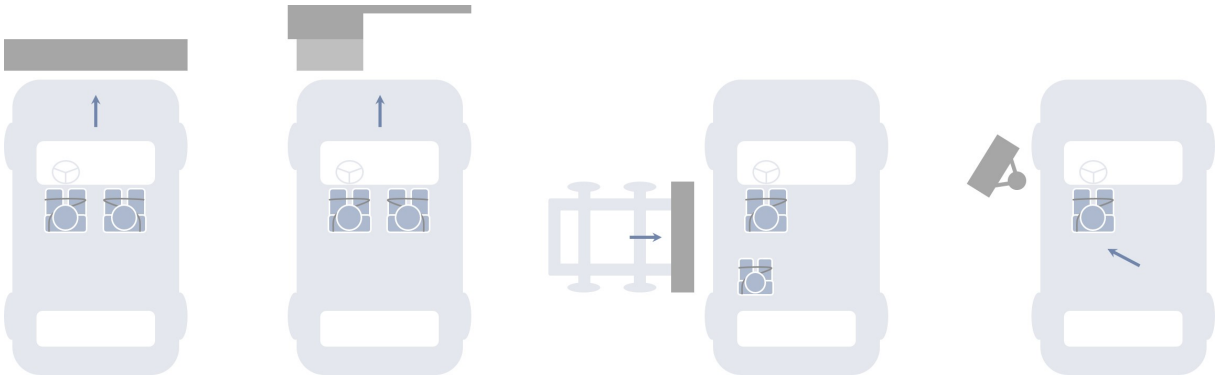
Figure 1: The product evolution process. Fig. 1.4 from [Web09] is extended by the allocation of uncertainties.

The focus of this thesis is on passive safety for which the terms crash safety, accident safety, and occupant protection are also consulted. In concrete terms, events are examined in which collisions take place. To assess and improve the crashworthiness of an arbitrary

but fixed car project, vehicle safety engineers analyze diverse impact scenarios of objects in the laboratory under realistic controlled conditions. These experiments are called crash tests. Engineers consider crash tests of components and total vehicles from various development stages. Before the production of a vehicle is started, it passes the product evolution process. It consists of three phases and is accompanied by product strategy and pre-development, see [Web09] and Figure 1.

A product strategy is devised and constantly updated, in which the wishes and needs of the customer are captured. In parallel, pre-development continuously designs and refines components and technologies. The technical and business aspects of the project are settled in the initial phase of the product evolution process. The concept phase then condenses the project into a systematic target catalog. These targets are realized in the series development phase. Prototypes, pre-series, and series cars are therefore developed and validated. Regarding vehicle safety, car companies carry out hardware crash tests especially in the pre-development and series development phase.

Besides internal studies, these tests are defined by two instances. On the one hand, the automotive homologation needs to be approved. It represents the procedure for certifying vehicles or particular components in vehicles for fulfilling the requirements of regulatory authorities. On the other hand, crash test programs exist worldwide to serve as a source of comparative information for consumers on the safety performance of new and used vehicles. These information are provided through ratings typically illustrated by stars. Both — the legal requirements as well as the consumer safety programs — specify safety standards. They consist of multiple load cases. There are frontal-, side-, pole-impact, overlap, roll-over tests, etc. An overview is given in Fig. 2 and [car21].



(a) Full frontal test (b) Frontal overlap test (c) Side barrier test (d) Side pole test

Figure 2: Different crash test disciplines. The images from the Safety Companion 2021 brochure, see [car21], were used as a template.

Engineers design and construct vehicles to satisfy the homologation conditions while achieving desired scores in consumer tests at the same time. To this end, during the development process, the vehicle is already subjected to numerous crash tests to ascertain the maturity level of current designs. This must be done for each car project and its derivatives, i.e. not only for the base car but also for convertibles, coupes, etc. Many automotive companies offer more than one car series, see e.g. [BMW21] for the online

catalog of the Bayerische Motoren Werke (BMW) Group. This requires a large number of crash tests. Each test incurs high financial costs as the test equipment is expensive and the examined car usually has to be scrapped afterwards.

Alongside these hardware tests, many mathematical models are created, cf. [DW15]. These models comprise numerical simulations and analytical functions. Investigating these models and their results can yield useful insights about the behavior of the experimental objects in the considered hardware load case. Unlike hardware testing, these simulations or functions cause comparatively low financial costs, e.g. due to acquisition of and electricity for high-performance computers. From a financial point of view, they can therefore be evaluated repeatedly enabling all kinds of mathematical analysis in theory.

Part of these investigations is the quantification and propagation of uncertainties. When performing crash tests, a variety of uncertainties is existent that can be crucial for the test result and also transfer to the accident behavior. It is thus important to identify and assess these uncertain factors. Depending on the phase of the product evolution process, which load case is considered, which object is analyzed, and how the test conditions are arranged, etc., uncertainties appear in different compositions and characteristics. All of these uncertainties can hardly be assessed by just the handful of possible hardware tests. For this purpose, mathematical surrogate models, e.g. virtual simulations, are utilized or serve as support. Their inputs and parameters shall imitate the nature of the uncertainties. The literature typically divides uncertainties into two classes.

The distinction is made between the epistemic and aleatory category, see [DKD09, BS07], which specify the mathematical perspective on the parameters. Epistemic uncertainties are provoked by lack of knowledge if no appropriate values can be assigned to specific quantities. They can be diminished by a greater understanding through further or enhanced data, see [SG07]. Aleatory uncertainties cannot be removed — they are irreducible, i.e. the investigated system reaction is inherently random [OHJ$^+$85]. The product evolution process is confronted with both forms of uncertainty.

The initial and concept phase is dominated by lack of knowledge, i.e. epistemic uncertainties, see Fig. 1. The form of the final product cannot yet be recognized. The architecture and systems, e.g. the structure of the crumple zone or the size and settings of the airbag or belt, are revised several times in the series development phase. That is, such quantities vary between prototypes, pre-series, and series cars. By renewing concepts, more experience, and data derived from tests, the epistemic uncertainties of those design parameters vanish.

Operating with surrogate models instead of hardware tests means presence of approximation inadequacy and numerical uncertainties, see [KO01] for an overview of uncertainty sources. Model inadequacy occurs as underlying physics of the hardware test cannot be reproduced exactly. Examples for numerical uncertainties are algorithmic, rounding, or interpolation errors. Either source is considered aleatory along with parametric variability that stems from production as well as experimental scatter. A crash test may be performed unintentionally with conditions divergent from the exact test program. The velocity may deviate from the prescribed test schedule. Depending on the positioning of the measuring instruments, the recorded signals could exhibit noise.

In the product evolution process, parameters can be associated with both uncertainty classes. Uncertainties of epistemic class can transition into aleatory uncertainties from

time to time, see Fig. 1. Although concrete values are established for particular quantities, thereby eliminating epistemic uncertainties, they may inhere aleatory irregularities. Shape and state of specific prototype components may not be finalized. Later, for series cars, decisions are made and manufacturing plans are prepared. Not all components, however, can be built in the factory exactly as plans dictate. For instance, the wall thickness of the crashbox has a fixed value but materials used have impurities in some places.

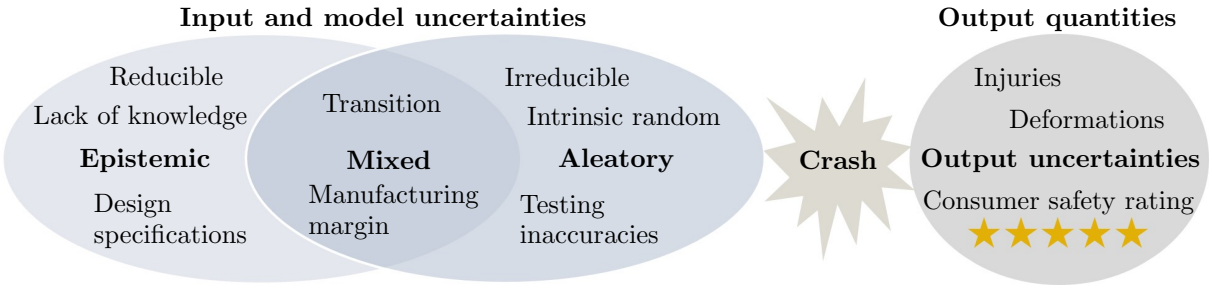


Figure 3: Uncertainties in the crash world.

Summarizing, vehicle safety engineers may face several sources of uncertainties: uncertain design variables and parameters, manufacturing, testing, and numerical uncertainties as well as model inadequacy. These sources possess elements of both classes — epistemic and aleatory. According to the phase of the product evolution process, the proportion of epistemic or aleatory uncertainties predominates. Engineers therefore have to tackle various uncertainties encountered in diverse objects and models at different times in order to plan and ultimately develop safe and robust cars. These multifaceted tasks must be completed in the shortest possible time and in the most cost-efficient way, thus posing a great challenge.

1.2 || State-of-the-art and research question

This section highlights the confrontation of well-established and current research with uncertainties. At first, examples from different fields of study that take uncertainties into account are presented. Then, the uncertainty analysis is divided into two types — forward and backward quantification. It is discussed that its high computational effort can be resolved using approximated response surfaces. Methods for forward uncertainty quantification, which is the focus of this thesis, are described. Sensitivity analysis is introduced next that provides information about the considered uncertain parameters. Afterwards, an overview of the models used in vehicle safety is given. These models are to be subjected to uncertainty analysis. Last but not least, the research question of this thesis is formulated.

Uncertainties appearing in different fields

Dealing with uncertainties is an important part of science. The influence of uncertainties can crucially change results of experiments. For this reason, the investigation of uncertainties in more and more fields is gaining in priority. Uncertainties are detected, quantified, and propagated on the one hand. On the other hand, they are in-

corporated into analyses like optimization, optimal control, robustness studies, etc, see [FCJR17, Ger11, BMNHG18]. These studies can make systems safer by identifying how likely certain outcomes or scenarios are. Negative surprises can be predicted beforehand. They can be prevented by revising concepts or revealed risks can be accepted.

In aerospace engineering, components of the aircraft, e.g. airframe, wings, or airfoil, are optimized with regards to crashworthiness, drag, or lift under uncertain conditions, see [Zan02, PGL06]. Architects and civil engineers want to build durable and stable houses, bridges, platforms, etc. For this, they want to remove vulnerabilities of their objects that often come from uncertainties. Thus, they consider strength properties of materials, e.g. cement, structural load characteristics, e.g. water wave loads on offshore oil platforms, etc., as aleatory, see [AK06]. Environmental researchers use uncertainty analysis to evaluate risks, potential adverse effects, and effectiveness of proposed remediation measures when hazardous substances are released into the atmosphere, see [HHB94].

Meteorologists make forecasts taking into account uncertainty. For example, parametric and structural uncertainties in the climatological planetary boundary layer are examined in [SAL10b] that highly influence climate, weather, and air quality. The recent pandemic, COVID-19, had physicians analyzing the impact of diagnostic uncertainties in virus detection testing, see [GCW$^+$20]. In the automotive sector, there are likewise many places where uncertainty analyses are or should be made: design optimization [ZZC11, ABH$^+$17], autonomous driving [CCKL19, HFRD13, CGEO15], but also for crashworthiness [DW15], and many more.

Types of uncertainty analysis and their time burden

One goal unites different sciences in analysis of uncertainties: all aim for fast, accurate, and reliable results. Concerning detection and quantification of uncertainties, literature distinguishes between two types, forward and backward uncertainty quantification, see [AKVS03, CI12]. Forward quantification or propagation of uncertainty is the determination of output uncertainty caused by uncertain system parameters. In more detail, this type is used to provide information about mean, variance, and reliability of outputs or to estimate their probability distribution. By contrast, backward or inverse uncertainty quantification deals with the opposite task. Given the output distribution function, it searches the values of unknown parameters, e.g. under a Bayesian framework, see [KO01]. The distinction between forward and backward uncertainty quantification is elaborated in more detail in Section 3.4.1.

The focus here is on forward propagation. In the following, this type is meant when uncertainty quantification or analysis is mentioned. Methodologies for forward uncertainty propagation often overlap, especially for engineering applications. Particularly there, several models, e.g. analytical functions derived from physics or computer simulations, are used to represent the actual situation. Often, they incur high computational efforts, e.g. finite element models. To perform uncertainty analyses in a reliable and profound manner, thousands or millions of evaluations may be required. This can make the investigation infeasible in terms of time and resources. Pure sampling-based methods like Monte Carlo simulations [KBTB14], importance sampling [Sri02], etc. are often too expensive to be applied. So, auxiliary methods must be implemented.

Approximated response surfaces as a remedy
As a workaround, approximated response surfaces can be employed to emulate costly models. They require some sample points to be trained with. There exist two main approaches for response surfaces, an intrusive and a non-intrusive option. For intrusive approaches, the governing equations of the model must be known. It can then be tried to simplify these equations and thereby accelerate the computational time of the model. For instance, complexity of equations, e.g. the size of matrices, can be reduced via dimensionality reduction. This technique is also called model order reduction, see [KMV18, BSV14, LV13, GV17].

For finite element crash simulations, a dissertation was published recently that describes an intrusive model order reduction method, see [Bac19]. This method is able to reproduce the already computed results of the full model. However, the method was not yet sophisticated enough to make good predictions for parameters outside the training set. In addition, the author had a special license with insights into the solver code. Generally, finite element crash models have to be considered as black boxes, i.e. non-intrusive mappings, as their underlying code is hard to access or is denied to be touched.

For these black boxes, metamodels as non-intrusive approaches are better suited. They are mathematical functions whose hyperparameters have to be optimized for a sufficient replicate of the actual model. Several methods are present that assume different mathematical source terms and hyperparameters. For instance, there are linear regression routines [MPV21], Gaussian process regression [Wil98], polynomial chaos expansion [TMES19], support vector machines [BL10], decision trees [XWVA05], random features [YLL$^+$14], etc. Methods from the field of machine learning, see [RN10], like artificial neural networks [Abr05] can also be used. They are here also referred to as metamodels. All have advantages and disadvantages regarding fitting quality subject to the data to be observed. Nevertheless, each of them can be evaluated quickly. Detailed information about metamodel concepts can be found in Section 3.2.

Integration of these metamodels into multifidelity methods may improve performances. Multifidelity methods combine models of high-fidelity, e.g. finite element simulations, with models of low-fidelity, e.g. metamodels, see [PWG18, KGH20]. Low-fidelity models speed up evaluations, where possible. Where impossible, the high-fidelity model is invoked to retain accuracy. In theory, this sounds promising but e.g. the prohibitively high costs of finite element crash simulations — even when used solely for corrections within multifidelity methods — complicates the realization in terms of computing time. In this case, it is proposed to work only with metamodels. If they are of high quality, multifidelity methods become obsolete anyway.

Methods for forward uncertainty quantification
After constructing an approximated response surface, uncertainty analysis can be executed, e.g. using sampling-based methods, on the fast response surface. Moments of outputs, e.g. mean and variance, can be calculated. Probability of system failures can be identified by analyzing the reliability of outputs, see [OK12]. On that account, a limit state function is defined. It indicates where a system satisfies certain conditions and where it fails. Response surfaces accelerate calculating the probability of failure. Moreover, other popular methods exist to measure the probability of failure without using response sur-

faces, e.g. first order or second order reliability method, see [ZO99]. However, they are prone to become inaccurate for nonlinear limit state functions, see [MS18].

Depending on the form of the input uncertainty, different methods can be used to propagate them through the system via response surfaces. Results of these methods are histograms, probability density, and cumulative distribution functions, or corridors thereof. All of them can help engineers to better understand their systems and inform whether concepts should be revised. Popular methods are now briefly introduced.

Interval analysis only assumes that the uncertain parameters lie in intervals, see [JKDW01]. Interval fields inferred by interval analysis are an interesting approach for interval finite element models, see [MDMDV11]. They model spatial varying input uncertainty. Via machine learning, they can be extended to supervised interval fields, see [BMM18]. Finite element crash models, however, are usually deterministic, i.e. there is no clear interval model expression yet. So, this calls for further research.

Possibility theory considers uncertainties as variables that endow quantities with attributes ranging from impossible to possible and from unnecessary to necessary, see [DP01]. One special case of possibility theory is fuzzy theory, see [Kli99]. Fuzzy logic deals with the concept of partial truth where the observation can be between completely true and completely false, see [NPM12]. However, the uncertainty descriptions of these methods are neither optimal for the parameters of vehicle crashes nor straightforward for engineers to interpret. They need unambiguous terms and no fuzzy terminology.

Second-order probability uses an inner and an outer loop to propagate uncertainties. It deals with the case that the distribution family of a parameter is known but its defining hyperparameters, e.g. the mean and the standard deviation for a Gaussian distribution, are unknown, see [SPM09]. In the outer loop, the hyperparameters are specified and sent to the inner loop where sampling is performed from the realized distributions. In vehicle safety applications, parameters are usually designed to have a clear uncertainty allocation. It is not necessary to introduce double uncertainties, i.e. an inner and an outer loop, for the parameter definitions themselves. If another allocations shall be tested, this can also be done by running other methods multiple times.

Probability boxes imprecisely surround probability distributions, see [DMS21]. Probability boxes for parameters can be evaluated to obtain another probability box for the output. Moreover, a version of the Dempster-Shafer evidence theory propagates uncertainties through the system, see [EST11]. The method — described more precisely in Section 3.4.2 — works with intervals for the uncertain parameters. These intervals are then assessed by experts: they are divided into subintervals and afterwards, probabilities are allocated to these subintervals. All combinations of subintervals are generated and propagated through the model. This leads to a corridor for the cumulative distribution function including worst and best case boundaries.

Probability boxes and the Dempster-Shafer evidence theory are related, see [FKG+15]. Dempster-Shafer structures can be interpreted as probability boxes. Conversely, probability boxes can be approximated with Dempster-Shafer curves. In this thesis, Dempster-Shafer structures are preferred as the uncertainties of the parameters are based on the mentioned expert opinions. Vehicle safety engineers can handle the uncertainty assignments for the parameters with ease and flexibility. Their opinion can additionally be fused with assignments coming from, for example, old data or associated information before the

propagation is absolved. Note that it can be attempted to perform the discussed methods straightly with the models and without the use of approximated response surfaces. As already mentioned, the computational effort are too expensive in most cases. Therefore, approximated response surfaces will be a key feature to enable uncertainty propagation in this thesis, especially for finite element models.

Input exploration via sensitivity analysis

Besides the quantification of uncertainties, engineers are interested in understanding the parameters of their system. Global sensitivity analysis explores how output uncertainty can be apportioned and attributed to the uncertainty in the parameters, see [SRA+08]. This must not be confused with local sensitivity analysis — often based on derivatives — which examines the effects of slight changes in a chosen set of factor values, see [ZL08b]. In global sensitivity analysis, the uncertainty in the outputs is assigned to the parameters over their full range of interest, see [ZL08a]. The difference between local and global sensitivity analysis is described in more detail in Section 3.3.1. In this thesis, the global approach is meant when speaking about sensitivity analysis as comprehensive findings need to be obtained.

Methods for global sensitivity analysis usually require a number of model evaluations. There are qualitative approaches that superficially identify significant parameters and operate with relatively low computational effort, i.e. number of sample points. In contrast, approaches exist that make quantitative statements about the influence of the parameters. They rely on larger computation expenses, see [IS17].

Incurred computational costs can again be addressed through use of approximated response surfaces, e.g. metamodels. However, the so-called curse of dimensionality can occur, see [Don00, Tru79]. On the one side, it describes the problem that costs for an entire study of the input space exponentially increase with the dimension of this space. On the other side, some metamodels may stop functioning well when dimensions become too high. In short, one needs to find the right number of inputs to balance the costs to create the model and the quality of its predictions. Therefore, cheap qualitative approaches may be employed before constructing metamodels and applying quantitative measures.

Depending on the model properties and the assumptions on the model parameters, qualitative approaches lead to the correct relevant parameters. They are also referred to as screening methods. The Cotter method is a very simple and cheap screening method to determine the importance of parameters, see [Cot79]. Its drawback is that it may not be in-depth enough to yield solid results.

The Morris method, see [Mor91] and Section 3.3.2, tries to cover the input space more carefully. It determines so-called elementary effects — basically difference quotients — of each parameter at different random starting points. Then, mean and standard deviation of these effects indicate the importance of parameters. The number of random starting points can be chosen arbitrarily. The more points are used, the more trustworthy the method becomes. Campolongo et al. use a radial design-based sampling in [CSC11] that enhances the method to make it quantitative.

The active subspace method, see [CDW14], reduces model parameters by rotating their coordinates to directions of strongest variation. It therefore requires the gradient which is not accessible when facing black boxes. Gradients can be sketched through

finite differences, see [CEW15]. This, however, asks for additional model evaluations. Furthermore, for non-smooth problems, one needs approximations of subgradients.

Examples for quantitative measures are Borgonovo, Sobol', or Kucherenko indices. Borgonovo indices of a parameter declare the expected shift in the output probability distribution when the observed parameter is fixed, see [Bor07]. The idea of Sobol' indices is to decompose the model into an expansion of summands growing in dimension, see [Sob01] and Section 3.3.3. The variance of the expansion can then be expressed in terms of the sum of all single variances of the summands. For this reason, Sobol' indices are also called variance-based sensitivity measures.

The decomposition applies only to independent parameters. Therefore, Sobol' indices are not suited for dependent parameters. Kucherenko et al. extended Sobol' indices in [KTA12] for dependent parameters. Kucherenko indices are based on a direct decomposition of the output variance. A great review of global sensitivity analysis methods can be found in [IL15].

Models considered in passive vehicle safety

Vehicle safety engineers encounter different models during car development. To arrive at a broad understanding of emerging phenomena, uncertainties should be considered for all of these models. The most popular and frequently employed models to imitate hardware crash events in high fidelity are explicit finite element simulations, see [DW15]. These simulations consists of many finite elements, e.g. several millions in full vehicle models, that form a mesh for the underlying geometry. To calculate the crash characteristics, differential equations are formulated and solved on the mesh. Due to the complex geometry and the nonlinear behavior, solving finite element models demands multiple hours on a great number of parallel computing central processing units (CPU).

Finite element models can be simplified — also known as physical surrogates, e.g. by applying sub-structure modeling, hybrid nonlinear finite element rigid body or elastic approaches, or hybrid fine-rough finite element meshes, see [Ray14, GJP13, DW15]. They maintain physical processes in the model. Some concentrate on components or consider parts to be elastic. Others replace the fine mesh by rigid bodies or alternative structures. All aim for faster calculations but often simplified models remain computational expensive.

Another approach similar to finite element simulations is called isogeometric analysis, see [CHB09]. It integrates finite element methodologies into computer-aided design (CAD). By doing so, the geometry does not have to be meshed. The solution is carried out directly on the CAD geometry where non-uniform rational basis splines are used. This saves time and effort due to omitting meshing procedure. A recent doctoral thesis, see [Lei20], discusses this technique for crash applications, i.e. contact problems. The method, while promising, needs further research for use in large impact scenarios. Furthermore, first results demonstrate that solving isogeometric analysis models is comparably expensive to running meshed finite element simulations.

Other examples for simplified models are multi-body system [CAE11, Fen13], deformation space models [Lan21, LFSD18], analytical and semi-analytical crash modeling, and equivalent static load methods [HHMR08], see [DW15, LFSD18] for an overview. In many cases, these models are suited merely for early phase development as they are derived by drastic simplifications, e.g. linearizations, heavy coarsening, etc. They give intuitions and

rough guidelines for design possibilities but they are not suitable for in-depth analysis. Moreover, engineers work with crash performance functions like safety ratings, see [car21], that are concerned with processed measurement quantities. These are not provided by the early phase models.

Posing the research question
A clear, uniform procedure to enable uncertainty investigations for these diverse crash models is to be found. It has to tackle the three main challenges: numerous, multifaceted parameters and their uncertainty definitions, large computational costs, and appropriate visualizations for interpretation. Accordingly, the following research question arises:

> *How can uncertainty management be structured for fast, flexible, and visualizable usage towards vehicle crash models?*

This thesis will provide an answer by means of a framework that brings required global methods in the correct order. The framework will be specifically trimmed for finite element crash simulations — the most common, precise, and expensive models in the field of vehicle safety. A selection of state-of-the-art methods from different science applications and their adaptions will be proposed as proxies of the global methods. The structure of the work is presented in the following section.

1.3 | Thesis structure

Chapter 2 proposes a global uncertainty management framework that answers the posed research questions by presenting a way to analyze uncertainties in arbitrary vehicle crash models. For this purpose, it provides a unified, generic process with general methodologies in a compact manner. Chapter 3 customizes the global framework to finite element crash simulations. It is divided into four sections. First, preliminaries are provided and finite element crash simulations are formulated as mathematical mappings. Second, different forms of metamodels are introduced for the mathematical mappings. For each type, a choice of state-of-the-art metamodels is presented. Third, sensitivity analysis including the elementary effects method and variance-based sensitivity indices are discussed. At last, uncertainty propagation via the Dempster-Shafer evidence theory is shown and its benefits are explained. Metamodels as well as sensitivity and uncertainty analysis are demonstrated on several examples.

In Chapter 4, the adjusted framework is applied to a large real-world project, a full vehicle side pole impact with nine parameters that exhibit uncertainties of the epistemic or aleatory type. The model is replaced by different metamodels to quickly approximate the quantities of interest, i.e. the key results. The best performing metamodel is then used to calculate Sobol' indices. Afterwards, Dempster-Shafer structures are assigned to the most relevant parameters. This package is subsequently evaluated with the evidence theory to obtain informative results about the output uncertainty, i.e. bounds for the cumulative distribution function of the key results. Chapter 5 critically reviews the chosen methods

for the framework components and provides an outlook on future efforts to refine and further extend the findings of this thesis. Chapter 6 delivers the final conclusion.

1.4 || New scientific contributions

The main contribution of this thesis is the framework that enables global sensitivity analysis and uncertainty propagation for crash applications that can be computationally expensive, are considered as black boxes, and may have multiple uncertain parameters. Each component and step of the framework is placed to best accommodate the application of uncertainty propagation. The generic framework is adapted to the most common computer models in vehicle crash, i.e. finite element simulations. The final aim of the tailored framework is to enable the Dempster-Shafer evidence theory for uncertainty propagation. This method appears to be most suitable for this field of research. Therefore, its concept is rewritten mathematically to make it understandable and manageable for engineers. For describing the single components of the framework, novel model descriptions of industrial finite element simulations are introduced. This allows to easily determine which types of metamodels can be used to approximate the time-consuming finite element models. These types, i.e. scalar metamodels, multi-target regression metamodels, and model order reduction metamodels are trimmed to the data, tested, and compared. Moreover, an adaptive algorithm for the elementary effects method is proposed to reduce the problem dimensionality at the right stage of the analysis.

Additional scientific contributions by the author of this thesis related to the topic here are [JLG21a] and [JLG21b]. The first work applies a non-intrusive model order reduction metamodel to a crashbox simulation to enable the Dempster-Shafer evidence theory. The second contribution suggests a generic framework similar to the one presented in this thesis and demonstrates it using a simplified occupant simulation. Furthermore, the author of this work advised four Master theses [Dek20, LN21, Go21, Her21] that were incorporated here. All of them investigated metamodels of different types, i.e. scalar metamodels, multi-target regression metamodels, and model order reduction metamodels.

General uncertainty management framework for vehicle crash applications

Vehicle safety engineers work with diverse models. Here, a model is defined as a construct that assigns n-dimensional inputs to outputs, i.e.

$$\mathcal{F} : \mathcal{X} \subset \mathbb{R}^n \to \overline{\mathcal{Y}} \tag{2.1}$$

where $\mathcal{X} \subset \mathbb{R}^n, n \in \mathbb{N}$, and $\overline{\mathcal{Y}}$ designate the input and the output space, respectively. Thereby, it is not mandatory to know explicitly how the mapping works, i.e. the model can be non-intrusive or simply a black box function. An input reflects the uncertain parameters of the model. Different input settings effect output uncertainties. This process, its reasons, and its outcomes are to be uncovered via sensitivity and uncertainty analyses. The right strategy for these studies must be found for each model individually. In the following, a global framework illustrated in Fig. 4 is proposed and subsequently discussed. It shows the paths to follow in order to enable global sensitivity and forward uncertainty analysis for vehicle safety models. To this end, various properties of the underlying model have to be examined. According to these, it is decided which steps must be taken to perform analyses in a fast, efficient, and accurate way.

First and foremost, the type of the model decides whether analyses can only be run with prior application of other techniques or directly without them. By type, it is meant if the model is resource efficient or inefficient. This separation is depicted as the first branch in Fig. 4. A resource efficient model can be evaluated quickly. On the contrary, resource inefficient models have large computational costs making profound investigations infeasible with respect to time. This is due to the many evaluations that are needed for reliable and precise analyses.

Known analytical functions such as safety ratings from crash test programs, see [car21], or mathematical formulas based on physics describing the crash event are computationally cheap, i.e. resource efficient. By contrast, high-fidelity virtual simulations, e.g. finite element models, are typically considered resource inefficient. Analyses of resource inefficient models must be enabled by additional methods whereas sensitivities and uncertainties of resource efficient models can be directly analyzed. This is illustrated in Fig. 4 where resource efficient models can skip the intermediary methods. For resource inefficient models, further branches must be traversed.

The next step for resource inefficient models is to check if the model can be simplified. Full vehicle models with occupants for instance can be partitioned into smaller models. In engineering practice, it is common — though not necessary — to observe the structure of the vehicle first. Then, the obtained information on the movement activity of the vehicle is forwarded to a compact occupant model, cf. the sub-structure modeling approach in [Ray14]. Usually, this still results in resource inefficient models as occupant models have long computation times. There are, however, other possibilities to reduce the model to

make it resource efficient. Examples are to consider only small components that might incur lower calculation efforts, to use drastic simplifications of the physics, see [LFSD18, Bra17], or to break it down to straightforward mathematical formulas. This would in turn enable the desired analyses immediately — portrayed as the arrow at the simplified modeling level in Fig. 4 pointing from left to right. Note that simplifying also demands time and effort for developing sophisticated concepts that imitate the complex behavior.

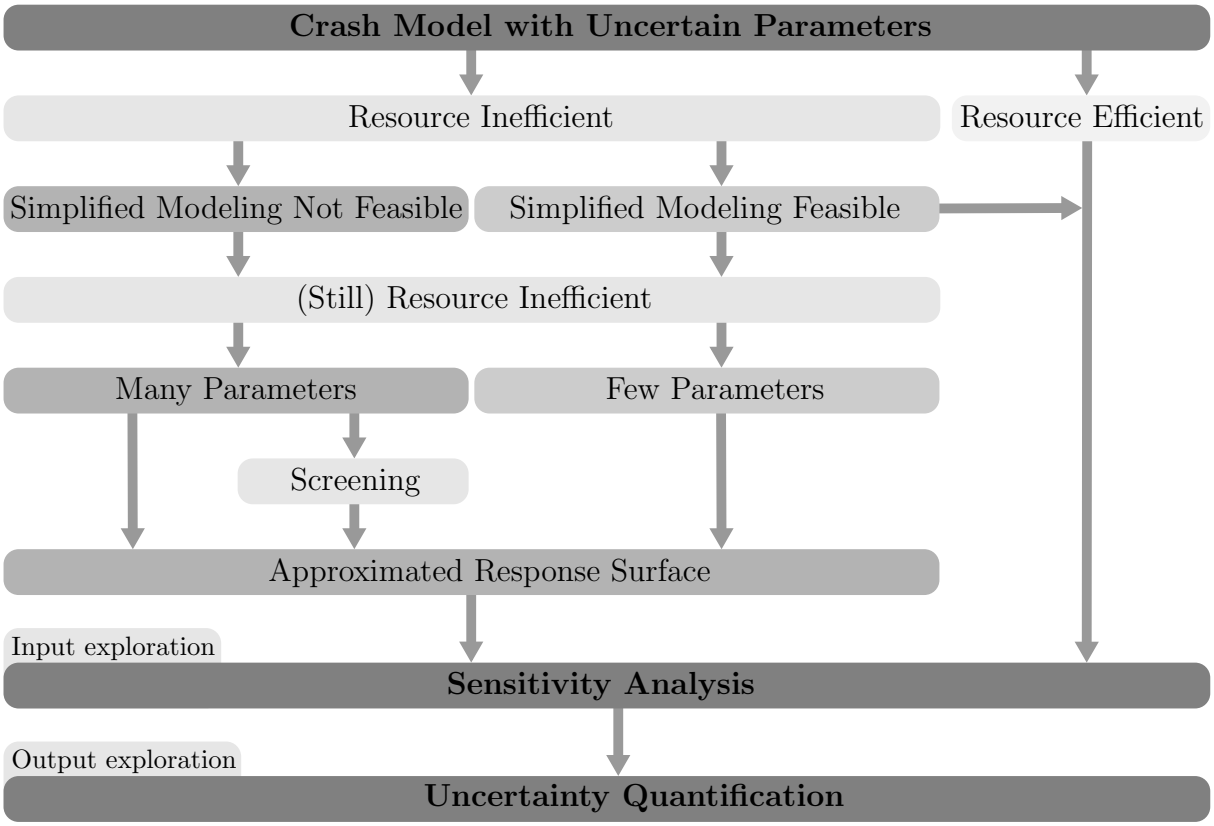


Figure 4: Uncertainty management framework for vehicle crash models. The framework is similar to the one presented in [JLG21b].

The framework recommends to replace resource inefficient models by approximated response surfaces,

$$\mathcal{M} : \mathcal{X} \subset \mathbb{R}^n \to \overline{\mathcal{Y}}, \tag{2.2}$$

to accelerate the output generation. To keep it general, i.e. concerning black boxes, metamodels are used as approximation response surfaces. Metamodels are mathematical surrogates that emulate the behavior of the considered model. They aim for precise approximations, i.e.

$$\mathcal{M}(x) = \mathcal{F}(x) \tag{2.3}$$

for all $x \in \mathcal{X}$. Depending on the investigated model and the data of interest, several approaches exist and different variants are available to utilize them. For instance output

dimensions of diverse complexity can be considered when dealing with finite element simulations, see Chapter 3 for details. All have in common that they have to be trained by a sample of actual model input-output data,

$$\left(x^1, \mathcal{F}\left(x^1\right)\right), \ldots, \left(x^{n_{\text{train}}}, \mathcal{F}\left(x^{n_{\text{train}}}\right)\right) \in \mathcal{X} \times \overline{\mathcal{Y}}, \quad n_{\text{train}} \in \mathbb{N}. \tag{2.4}$$

After this offline training stage, a well-fitted metamodel can quickly make predictions close to the actual outputs in the online phase. For the creation of appropriate response surfaces, it can be beneficial to take the number of parameters, i.e. n, into account. Broadly speaking, the higher the dimension of the input space is, the more mathematical terms must be used to construct the metamodel. The more terms there are, the more equations have to be solved to do the regression. Consequently, the harder it gets to optimize the metamodel and possibly, the worse the quality may become.

Therefore, in case of many parameters, it may help to install a screening method before training metamodels. Screening belongs to sensitivity analysis methods that qualitatively identify the contributions of parameters to the output, see [IL15]. Only the most relevant parameters, e.g. in total $k \in \mathbb{N}, k < n$, are then considered in the following steps, e.g. metamodel creation. Some parameters may have little or no influence on the output. These irrelevant parameters do not need to be investigated further. They can be eliminated from the model, i.e. the dimension of the input space, $\mathcal{X}$, shrinks from n to k. This can be advantageous because examining a model with more parameters normally demands more evaluations. So, it is advisable to limit the model to its most important parameters by exploiting the information from screening. In addition, screening methods require a relatively small number of sample points which makes them attractive.

Besides, there exist metamodels that can handle high dimensional input spaces. For example, there are techniques that are coupled with the high dimensional model representation, see [LWR00, SW10, CWYH19, HQZ+15]. Others try to diminish the expression of the metamodel. They are supposed to detect its significant terms and discard the insignificant ones, e.g. different possibilities are available to make the expansion of polynomial chaos sparse, see [LMS21]. Generalized, this can also be done by connecting metamodels with dimensionality reduction methods, see [LMS20]. In a nested optimization approach, the dimension of the problem is reduced in the outer loop and afterwards, the metamodel is fit within the reduced space in the inner loop.

All options can help to deal with a high input dimension and facilitate training of metamodels. There can be indeed many parameters. Engineers face test uncertainties, uncertain design specifications, production uncertainties, etc., see Fig. 3. The initial speed, the impact angle resulting from the rocking of the vehicle, the position of the dummy or other measuring devices, etc. can lead to different results. These parameters are treated as test uncertainties that are aleatory as they cannot be reduced and may appear in each test. Uncertainties can also stem from production instructions to be defined by the pre-development or they come from manufacturing inaccuracies.

The definitions of the input uncertainty for e.g. the choice of material properties or wall thicknesses thus depends on the point in time of the product evolution process. In the initial and concept phase, see Fig. 1, the lack of knowledge, e.g. of the wall thicknesses of the crashboxes, might dominate. Hence, the parameter is to be regarded as epistemic. In later series development phases, this could transition to an aleatory view. The size

might be defined on the plan but it may be realized slightly imprecisely in the production possibly which could cause uncertain behaviors.

Design specifications of vehicle restraint systems can be epistemic or aleatory as well. In the concept phase, it is not clarified what the optimal time is for the airbag to deploy since the structure of the passenger compartment — a production uncertainty — is not yet settled. Later, it is determined but the airbag itself could respond with delay that can influence results drastically. For the protection against rib injuries in a side crash, the deployment of the side airbag is crucial. Moreover, numerical effects and model inadequacy could also be modeled by specific parameters. Here, however, the focus lies on the presented test, design, and production parameters.

Through the metamodels, a fast output generation is guaranteed which enables profound studies of the model, i.e. the approximated response surface level in Fig. 4 is pierced through. Sensitivities can be extensively analyzed, i.e. quantitative measures can be calculated as opposed to qualitative information from screening. Global sensitivity indices expose important properties of the model, especially its parameters. They indicate which parameters control the global output variation and to what extent. From a mathematical perspective, the input, $x \in \mathcal{X}$, is considered to be the realization of a multivariate random variable, $\mathscr{X}$, see Section 3.1 for notations and definitions. The global sensitivity index of first order then represents the proportion,

$$\frac{\mathscr{V}_i}{\mathrm{Var}\,(\mathscr{Y})}, \quad i \in \{1, \ldots, n\}, \tag{2.5}$$

see (3.128), that the influence, $\mathscr{V}_i$, of the i-th dimension of the random vector, $\mathscr{X}$, has on the total output variance, $\mathrm{Var}\,(\mathscr{Y})$. There, $\mathscr{Y} = \mathcal{F}(\mathscr{X})$ is the random vector that belongs to the output space, $\overline{\mathcal{Y}}$. Other global sensitivity indices highlight which parameters interact and can also indicate whether the model is additive. Extensive sensitivity analysis can concretely determine which parameters need further investigation, whereas screening can only give an idea of relevant and irrelevant parameters. It is therefore possible that further parameters will be deleted for the upcoming uncertainty analysis. Note that global instead of local approaches are taken in this thesis since influences of entire parameter ranges are of interest, see Section 3.3.1 for more details.

After the knowledge about the parameters, uncertainty quantification is performed as last level of the total framework, see Fig. 4. In fact, forward uncertainty propagation is meant here while backward uncertainty quantification is not part of this thesis. The distinction between both is elucidated in Section 3.4.1. The forward approach is applied to quantify uncertainties in the outputs caused by epistemic and aleatory input parameters following a known distribution, $p_{\mathscr{X}}$. For example, the output range, its distribution, $p_{\mathscr{Y}}$, and, by association, the probability, $\mathscr{P}(\mathscr{Y} = \mathcal{F}(x))$, at which a particular output value occurs, can be ascertained. This allows conclusions to be drawn about safety, reliability, risks, and chances of the underlying event and components thereof. Depending on the assessment, engineers can decide whether systems need to be revised. Aleatory uncertainties cannot be erased but they can be made benign through enhanced concepts. All in all, the framework can lead to an improvement of vehicle safety by detecting possible uncertainties as well as being aware of where they come from and which parameters control them. Note that sensitivity analysis is intentionally used prior to uncertainty analysis

because it facilitates uncertainty analysis by reducing the number of relevant parameters and thus avoiding the curse of dimensionality.

The framework was explained in a forward fashion. Actually, it was created backwards. The main intention is to run uncertainty analysis. For that, it is useful to know which parameters actually contribute to the output variation to assist uncertainty quantification. Sensitivity analysis identifies these information. Irrelevant parameters can be rejected and set to their nominal values. Special attention can be paid to the most important parameters. Then, complex models, e.g. finite element simulations, posed the challenge of being resource inefficient. So, they had to be exchanged by fast approximations, i.e. metamodels. Metamodels in turn may suffer from the curse of dimensionality. This can be tackled by screening the most relevant parameters in advance, by using sparse methodologies or dimensionality reduction techniques. Before these methods are applied, the model may be simplified to a simpler surrogate model which potentially avoids the whole metamodel procedure. In the end, the framework in Fig. 4 was established.

A model can also simply consist of collected data from e.g. hardware crash tests. Parameters could be test or vehicle conditions. The measured outputs could be motion related quantities of the experiment. This data certainly requires high effort to be obtained. A real crash test is an expensive time-consuming event. Observed data is therefore resource inefficient. Furthermore, hardware crash tests of full vehicles are rarely repeated several times. Therefore, it is difficult to perform reliable analyses only on the small sample of these hardware tests. In some cases, however, enough data is available, i.e. the methodologies can be applied with better results. Nevertheless, the framework focuses on surrogate models for these crash events. Note that this procedure can also be applied to applications from technical fields other than vehicle safety.

The next chapter provides precise technical details how the framework can be tailored to finite element simulations. These simulations are the standard for computer models approximating crash scenarios, see [DW15]. They are solved explicitly — due to contact issues — by a commercial solver that must be seen as a black box function. Furthermore, they are designed using a large complex geometry including complex material properties. Solving them can entail tackling highly nonlinear phenomena. This all implies long computational times. Different options for metamodels replacing the simulations will be shown. For the sensitivity analysis, the variance-based Sobol' indices will be presented as parameter independence is assumed. The propagation of the uncertainties will be carried out with the Dempster-Shafer evidence theory. In principle, the framework customized for finite element simulations and its methods can also be transferred to other virtual or non-virtual models.

Methods specific to finite element models

In this chapter, finite element (FE) simulations as the most commonly used computer models in the crash domain are subjected to the uncertainty management framework shown in Fig. 4. For this purpose, the framework can be formulated more concisely. In most cases, finite element models are resource inefficient whether simplified or not. The optional screening component is integrated into the approximated response surface strategy, i.e. the metamodel. Fig. 5 features the resulting more compact framework.

FE Model (Section 3.1.2)	**Metamodeling** (Sections 3.2, 3.3.2)	**Sensitivities** (Section 3.3.3)	**Uncertainties** (Section 3.4)
Resource inefficient, black box solver, various parameters	Different options, screening, dimensionality reduction	Parameter assessment, ranking, deletion	Output analysis, visualization, decision making

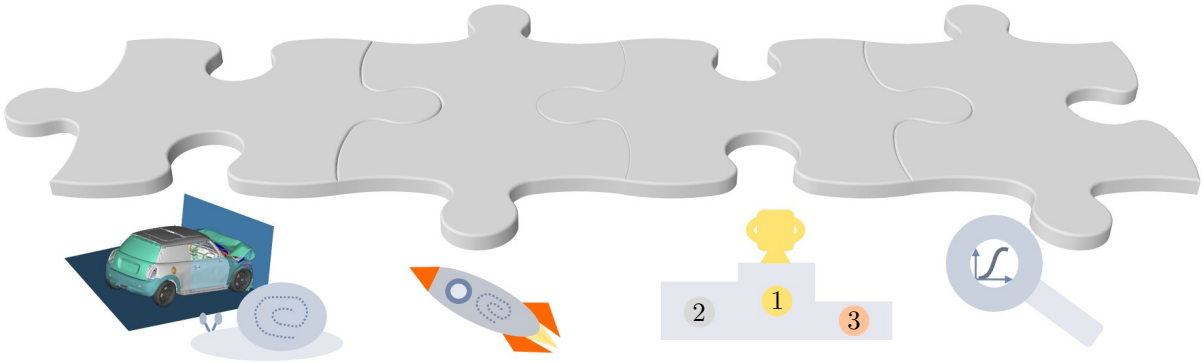


Figure 5: Compact uncertainty management framework for finite element crash simulations.

3.1 Preliminaries and model description

Before concrete methods are introduced for the framework components, preliminaries are provided and representations of finite element models for this work are discussed. These representations, i.e. treating finite element methods as mappings containing different levels of information, enable various options of metamodels mentioned in Fig. 5.

3.1.1 Preliminaries and notation

To begin with, regularly occurring abbreviations along with their descriptions can be found in the list of essential abbreviations after the table of contents. Additionally, Table 1 records repeatedly used mathematical symbols.

Table 1: Mathematical symbols often appearing in this thesis.

Symbol	Description
$\cdot$	Standard multiplication or placeholder
$\circ$	Function composition
$\mathbb{R}$	Real numbers
$\mathbb{N}$	Natural numbers
Lower case letters, e.g. x	Scalar values or vectors
Upper case letters, e.g. X	Matrices
Script letters, e.g. $\mathscr{X}$	Random variables, vectors, or resulting quantities
$\mathrm{E}(\cdot)$	Expected value of a random variable
$\mathrm{Var}(\cdot)$	Variance of a random variable
$\mathcal{F}$ and $\mathcal{M}$	Mapping, e.g. FE black box function, and its metamodel
$\mathcal{X}$ and $\mathcal{Y}$	Input and output spaces of mappings, e.g. $\mathcal{F}$
n	Initial number of (model) parameters
n_{train}	Size of the training sample
n_{val}	Size of the validation sample
n_{red}	Number of parameters after dimensionality reduction
k	Number of parameters after screening
d	Number of parameters after sensitivity analysis
N_{T}	Number of discrete time steps of a FE model
N_{EN}, N_{CN}, and $\overline{N}$	Number of entire or component nodes of a FE model
N_{red}	Number of reduced nodes, e.g. principal components

Now, further required notions and concepts are discussed. A model maps from an input space to an output space. In this thesis, one input consists of several parameters. In other words, a parameter configuration defines one input that can be evaluated via the model to obtain the corresponding output. Furthermore, the term hyperparameters, e.g. coefficients of basis functions, describes settings of models or techniques, e.g. metamodels or dimensionality reduction. They need to be optimized to yield optimal results.

The last terminologies being introduced are related to probability theory and statistics. The definitions orientate on the works [Kle13, Dur19, Ash08]. Let $(\Omega, \mathscr{F}, \mathscr{P})$ be a probability space, i.e. a triple consisting of the event space Ω, the σ-algebra $\mathscr{F}$, and the probability measure $\mathscr{P}$. A random variable is a measurable function, $\mathscr{X} : \Omega \to B, \omega \mapsto \mathscr{X}(\omega)$, from the event space, Ω, i.e. a set of possible outcomes, to a measurable space, B. The expected value of a random variable, $\mathscr{X}$, is defined as

$$\mathrm{E}(\mathscr{X}) = \int_{\Omega} \mathscr{X}(\omega)\, d\mathscr{P}(\omega). \tag{3.1}$$

The variance of $\mathscr{X}$ is denoted as

$$\mathrm{Var}(\mathscr{X}) = \mathrm{E}\left((\mathscr{X} - \mathrm{E}(\mathscr{X}))^2\right) = \mathrm{E}\left(\mathscr{X}^2\right) - \mathrm{E}(\mathscr{X})^2. \tag{3.2}$$

The expected value of a multivariate random variable, i.e. a random vector, $(\mathscr{X}_1, \ldots, \mathscr{X}_n)$, $n \in \mathbb{N}$, is defined as $\mathrm{E}(\mathscr{X}_1, \ldots, \mathscr{X}_n) = (\mathrm{E}(\mathscr{X}_1), \ldots, \mathrm{E}(\mathscr{X}_n))^T$.

In this work, event spaces, Ω, of probability spaces are equipped with the Borel-σ-algebra as σ-algebra and the Lebesgue measure as probability measure. The expected value of a random variable, $\mathcal{X} : \Omega \to \mathbb{R}, \omega \mapsto x = \mathcal{X}(\omega)$ with the probability density function, $p_{\mathscr{X}}$, and the cumulative distribution function, $P_{\mathscr{X}}$, can then be formulated as

$$\mathrm{E}\left(\mathscr{X}\right) = \int_{\mathbb{R}} x p_{\mathscr{X}}(x)\, dx. \tag{3.3}$$

Two random variables, $\mathscr{X}_1, \mathscr{X}_2 : \Omega \to \mathbb{R}$, are called independent if and only if the combined random variable $(\mathscr{X}_1, \mathscr{X}_2)$ possesses a joint cumulative distribution function that fulfills

$$P_{\mathscr{X}_1,\mathscr{X}_2}(x_1, x_2) = P_{\mathscr{X}_1}(x_1) P_{\mathscr{X}_2}(x_2), \tag{3.4}$$

or equivalently, it has a joint probability density function that satisfies

$$p_{\mathscr{X}_1,\mathscr{X}_2}(x_1, x_2) = p_{\mathscr{X}_1}(x_1) p_{\mathscr{X}_2}(x_2). \tag{3.5}$$

Later, uniform distribution is assumed when, for example, training points are sampled for metamodels. The uniform distribution is defined for an interval, $[a, b] \subset \mathbb{R}, a < b$. Its probability density function for a random variable, $\mathscr{X} : \Omega \to \mathbb{R}$, reads

$$p_{\mathscr{X}}^{\text{uniform}}(x) = \begin{cases} \frac{1}{b-a}, & \text{for } a \leq x \leq b, \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

Its cumulative distribution function is declared as

$$P_{\mathscr{X}}^{\text{uniform}}(x) = \begin{cases} 0, & \text{for } x < a, \\ \frac{x-a}{b-a}, & \text{for } a \leq x \leq b, \\ 1, & \text{for } x > b. \end{cases} \tag{3.7}$$

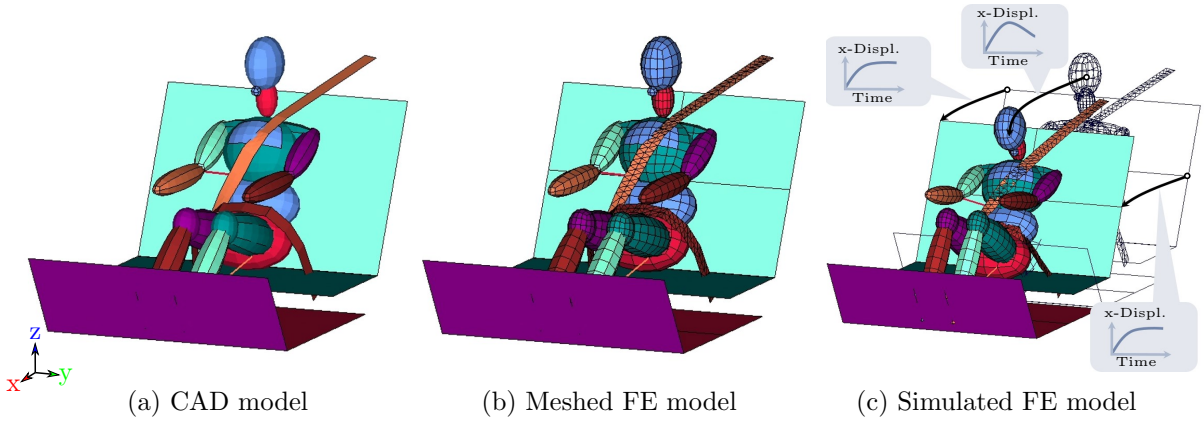
As a final note, other required definitions are provided directly in the respective sections.

3.1.2 Model description

Finite element (FE) models are now described in a nutshell — for more information on this topic see e.g. [ZTZ05, Red19, Log16]. First, a geometry of the object under investigation is usually drawn in a computer-aided design (CAD) program. This geometry is meshed, i.e. it is discretized into small parts — the so-called finite elements, see Fig. 6. After defining the trial functions and setting the boundary conditions, material properties, etc., the governing equations are solved on the nodes of the finite elements to model the physical behavior. This means the solution of a complex system of differential equations must be found numerically.

For vehicle crash models, explicit solvers are used due to contact issues, multiple material definitions, etc. The FE models in this thesis are solved using commercial software, LS-DYNA or Abaqus FEA. Although manuals, see [LS-06, FEA15], exist that explain theoretical foundations of the solvers, exact mathematics behind them — including many heuristics — are hard to access in most cases. In the end, only the results of the simulations and their prescribed processed quantities can be analyzed, i.e. models must be seen

as black box functions. Outputs that can be demanded from the model are time developments of quantities like displacements, velocities, accelerations, forces, moments, etc., of all finite element nodes. These discrete time developments are also called histories or profiles. Due to their typical high sampling rate, e.g. every 0.1 ms, they are referred to as curves as well — although they are still non-continuous objects from a strictly mathematical point of view. The output quantities are reciprocally linked, e.g. displacements can be differentiated to obtain velocities. Differentiation of velocities yields accelerations. On the opposite, integration provides the transition to the other direction. Other quantities can be obtained by using basic physics, e.g. Newton's law of motion to calculate forces from masses and accelerations.



(a) CAD model (b) Meshed FE model (c) Simulated FE model

Figure 6: From a CAD model to a simulated FE model. The simulation code is taken from the LS-DYNA example homepage, see [LS-20].

The solver internally retrieves further single quantities from the node histories that are required for later analysis or evaluation of the crashworthiness. This postprocessing is actually deterministic and defined a-priori but its formulas are not invariably accessible. Mandatory information such as modeling formulations and material plans may be contained in encrypted software code resulting in a second and unresolvable black box. Therefore, it may not be sufficient to solely know the node profiles. As a remedy, engineers can tell the solver to additionally output quantities of several measuring nodes, elements, or objects to figure out desired crash performance values.

For example, the chest deflection of specific FE dummies are measured by the change in length of connector beams. It is complicated to flawlessly reconstruct this change, i.e. compression, by corresponding node histories due to the inaccessible modeling topology of the beam within the encoded dummy. However, the single chest deflection curve can be requested from the model. In contrast, it is straightforward to find the acceleration of the node describing the center of gravity of the dummy head among the entire node acceleration histories.

To investigate the crash scenario under uncertainty, model parameters are defined and varied for the FE simulation. One parameter configuration is here called an input. Each input, $x = \mathscr{X}(\omega) \in \mathbb{R}^n$, is a realization of a multivariate random variable, i.e. a random vector, $\mathscr{X} : \Omega \to \mathbb{R}^n$, from the probability space $(\Omega, \mathscr{F}, \mathscr{P})$, cf. Section 3.1.1. Its joint probability density function is denoted as $p_{\mathscr{X}}$. Its components, $\mathscr{X}_i$, $i \in \{1, \ldots, n\}$, are

assumed to be independent. The joint density function can thus be described as the product of the marginal probability density functions, i.e.

$$p_{\mathscr{X}} = \prod_{i=1}^{n} p_{\mathscr{X}_i}. \tag{3.8}$$

Each realized input component, x_i, is supposed to lie within a compact interval,

$$I_i = [a_i, b_i] \subset \mathbb{R}, a_i, b_i \in \mathbb{R}, a_i < b_i, i \in \{1, \ldots, n\}, \tag{3.9}$$

that largely covers the support of the corresponding marginal probability density function, $p_{\mathscr{X}_i}$. Consequently, the input space,

$$\mathcal{X} = \prod_{i=1}^{n} I_i \subset \mathbb{R}^n, n \in \mathbb{N}, \tag{3.10}$$

is regarded as a Cartesian product of compact intervals.

Together, from a mathematical point of view, there arise several options to consider FE models as black box functions. Four global and interrelated variants covering different levels of information are presented below. Depending on the key result, i.e. scalar quantity of interest, to be examined, a variant can or cannot be selected. Histories of all $N_{\mathrm{EN}} \in \mathbb{N}$ model nodes in one degree of freedom or the resultant direction can be regarded as a black box called the entirety function,

$$\mathcal{F}_{\mathrm{EN}}^{i,q} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y}_{\mathrm{EN}}^{i,q} \subset \mathbb{R}^{N_{\mathrm{EN}} \times N_{\mathrm{T}}}, \tag{3.11}$$

where $\mathcal{Y}_{\mathrm{EN}}^{i,q}$ is the output space containing real-valued matrices. The variable $N_{\mathrm{T}} \in \mathbb{N}$ is the number of model time steps and i is the direction of the behavior, e.g. x, y, z, or resultant (res), i.e. $\sqrt{\mathrm{x}^2 + \mathrm{y}^2 + \mathrm{z}^2}$. Displacements, velocities, accelerations, forces, etc., are exemplary quantity types, q. Where possible, the superscript indices, i and q, are omitted from now on for better readability, e.g. $\mathcal{F}_{\mathrm{EN}} \coloneqq \mathcal{F}_{\mathrm{EN}}^{i,q}, \mathcal{Y}_{\mathrm{EN}} \coloneqq \mathcal{Y}_{\mathrm{EN}}^{i,q}$. The rows of one output realization $Y \in \mathcal{Y}_{\mathrm{EN}}$ represent the nodes. The columns stand for the prescribed discrete N_T time steps.

A concentration function,

$$c : \mathcal{Y}_{\mathrm{EN}} \subset \mathbb{R}^{N_{\mathrm{EN}} \times N_{\mathrm{T}}} \to \mathcal{Y}_{\mathrm{CN}} \subset \mathbb{R}^{N_{\mathrm{CN}} \times N_{\mathrm{T}}}, \tag{3.12}$$

can be applied to focus on specific components of the model, e.g. the dummy. It selects the $N_{\mathrm{CN}} \in \mathbb{N}$ nodes that are located on the desired component, i.e. it withdraws the N_{CN} relevant rows of the matrix $Y \in \mathcal{Y}_{\mathrm{EN}}$. For instance, the deformation behavior of the vehicle's front end in a frontal impact is clearly different from that of the rear end. The concentration function can help to better analyze both behavior by isolating them.

With it, the component function

$$\mathcal{F}_{\mathrm{CN}} \coloneqq c \circ \mathcal{F}_{\mathrm{EN}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y}_{\mathrm{CN}} \subset \mathbb{R}^{N_{\mathrm{CN}} \times N_{\mathrm{T}}} \tag{3.13}$$

is declared as the mapping from input space, $\mathcal{X}$, to the set of histories of the component nodes, $\mathcal{Y}_{\text{CN}}$. It composes the function c with the black box $\mathcal{F}_{\text{EN}}$ implying that $\mathcal{F}_{\text{CN}}$ is a black box as well. In the case that $N_{\text{CN}} = 1$, the concentration function reduces the problem to one node history lying in $\mathbb{R}^{1\times N_{\text{T}}}$. This is interesting when engineers want to observe a specific measurement node, e.g. the center of gravity acceleration of the head in driving direction.

Existent quantities can be processed to a single desired object history. Postprocessing functions of the form

$$g : \mathcal{Y}_{\text{CN}}^{i_1,q_1} \times \mathcal{Y}_{\text{CN}}^{i_1,q_2} \times \cdots \times \mathcal{Y}_{\text{CN}}^{i_d,q_m} \to \mathcal{Y}_{\text{SH}} \subset \mathbb{R}^{1\times N_{\text{T}}}, \tag{3.14}$$

represent this operation for $m \in \mathbb{N}$ quantity types, q_l where $l \in \{1, \ldots, m\}$, and their $d \in \mathbb{N}$ behavior directions, i_k where $k \in \{1, \ldots, d\}$. Some postprocessing functions are encrypted within the solver code, e.g. for determining the chest deflection of the dummy. Others are available or can be easily designed. For example, the resultant head acceleration is the 2-norm of the x-, y-, z-directions. This means the component of interest could be the single head measurement node — $N_{\text{CN}} = 1$ — and the required quantity, q, could be the acceleration — $m = 1$. The x-, the y-, and the z-accelerations, i.e.

$$\mathcal{Y}_{\text{Head Node}}^{\text{x, Acceleration}}, \mathcal{Y}_{\text{Head Node}}^{\text{y, Acceleration}}, \mathcal{Y}_{\text{Head Node}}^{\text{z, Acceleration}}, \tag{3.15}$$

can then be used to calculate the 2-norm — $d = 3$. There are truly several options. The resultant acceleration could be directly considered, i.e. d shrinks to 1, or the velocity could be taken and differentiated. Note that postprocessing functions, g, can also be applied to output spaces of the entirety function, $\mathcal{F}_{\text{EN}}^{q_l,i_k}$.

Composing postprocessing, g, with functions $\mathcal{F}_{\text{CN}}^{q_l,i_k}$ yields the next black box, the history function,

$$\mathcal{F}_{\text{SH}} \coloneqq g \circ \left(\mathcal{F}_{\text{CN}}^{q_1,i_1}, \mathcal{F}_{\text{CN}}^{q_1,i_2}, \ldots, \mathcal{F}_{\text{CN}}^{q_m,i_d}\right) : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y}_{\text{SH}} \subset \mathbb{R}^{1\times N_{\text{T}}}, \tag{3.16}$$

that sends inputs to individual output behavior, e.g. single, averaged, or aggregated node profiles, element histories, component part curves, etc. Actually, the most obvious manner to evaluate performances is to look at scalar values. Therefore, extraction functions,

$$e : \mathcal{Y}_{\text{SH}} \subset \mathbb{R}^{1\times N_{\text{T}}} \to \mathcal{Y} \subset \mathbb{R}, \tag{3.17}$$

are applied to $\mathcal{Y}_{\text{SH}}$. Often these deterministic functions seek the maximum or minimum of the curve that $\mathcal{F}_{\text{SH}}$ outputs. However, they can have complex expressions as well — they may be nonlinear or even discontinuous. Their function equations are well-known from the respective test protocol.

In hardware crash tests, for example, the maximum Head Injury Criterion of 15 milliseconds (HIC_{15}), see [U.S21], is an indication of how well the restraint system can protect the occupant's head. It is extracted from the resultant and filtered head acceleration curve, a_{res}, expressed as a multiple of $9.81\,\text{m/s}^2$ — the gravitational acceleration. More

precisely, this scalar value is calculated through

$$\mathrm{HIC}_{15}(a_{\mathrm{res}}) = \max_{\substack{t_1,t_2 \\ t_1<t_2 \\ t_1-t_2\leq 15}} \left\{ \left(\frac{1}{t_2-t_1} \int_{t_1}^{t_2} a_{\mathrm{res}}(t)\, dt \right)^{2.5} (t_2-t_1) \right\}, \tag{3.18}$$

where $t_1 \geq 0, t_2 > 0$ are two points in time during the event, specified in milliseconds, that are not more than 15 ms apart from each other. For FE models, this can be translated to a discrete HIC_{15} extraction function, e, applied on head acceleration matrices, $Y \in \mathcal{Y}^{\text{res, Acceleration}}_{\text{Head Node}}$, in the correct unit.

The extraction finally leads to the key result function,

$$\mathcal{F}_{\mathrm{KR}} = e \circ \mathcal{F}_{\mathrm{SH}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y} \subset \mathbb{R}. \tag{3.19}$$

Key result functions connect the input space, $\mathcal{X}$, with scalar key results, i.e. quantities of interests. Altogether, a key result function can be written as a composition of several functions, i.e.

$$\begin{aligned} \mathcal{F}_{\mathrm{KR}} &= e \circ \mathcal{F}_{\mathrm{SH}} = e \circ g \circ \left(\mathcal{F}_{\mathrm{CN}}^{q_1,i_1}, \mathcal{F}_{\mathrm{CN}}^{q_1,i_2}, \ldots, \mathcal{F}_{\mathrm{CN}}^{q_m,i_d} \right) \\ &= e \circ g \circ \left(c \circ \mathcal{F}_{\mathrm{EN}}^{q_1,i_1}, c \circ \mathcal{F}_{\mathrm{EN}}^{q_1,i_2}, \ldots, c \circ \mathcal{F}_{\mathrm{EN}}^{q_m,i_d} \right). \end{aligned} \tag{3.20}$$

To recall, the functions $\mathcal{F}_{\mathrm{SH}}, \mathcal{F}_{\mathrm{CN}}, \mathcal{F}_{\mathrm{EN}}$ are black boxes. The formulas of the postprocessing, g, are also unknown in some cases. Accordingly, the key result mapping, $\mathcal{F}_{\mathrm{KR}}$, is regarded as a black box function.

The equality in (3.20) demonstrates that the key results can theoretically be obtained through different routes, cf. Fig. 7. If the formulas of g are accessible, one can use the entirety function (3.11), concentrate on the desired component, use postprocessing to obtain an individual history, and then extract the key result. Option two is to start from the component function (3.13) and then break it down to the key result. The third possibility is to leave out the component and analyze the history function (3.16). The fourth and last presented alternative deploys the key result function (3.19) itself. If the formulas of g are not available, one cannot start later than from the history function.

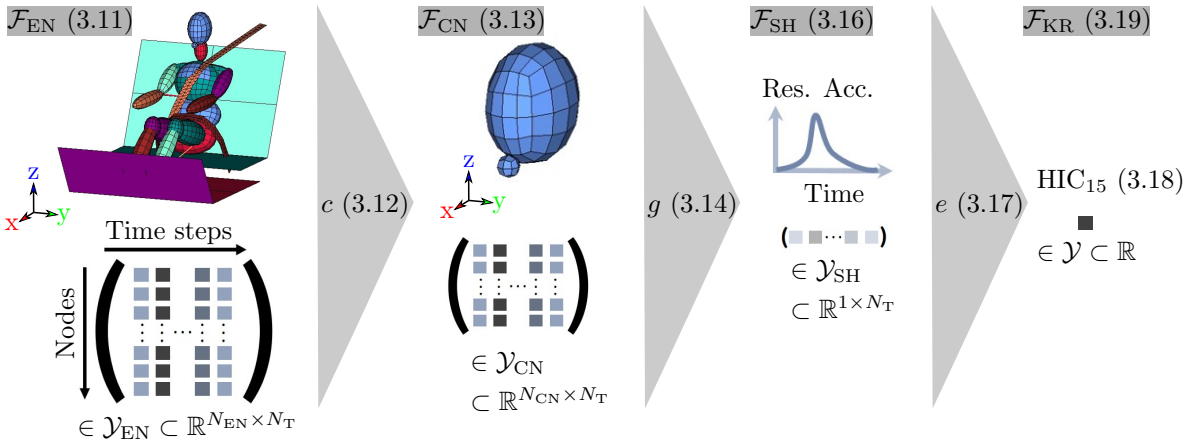


Figure 7: From a FE model to a key result, e.g. HIC_{15}.

Summarizing, four ways can lead to desired key results. In the following, $\mathcal{F}_{\mathrm{KR}}$ is mainly taken to explain the proposed methods of the uncertainty management framework as essentially its scalar outputs, i.e. the key results, are analyzed by engineers. It should be kept in mind that the functions, $\mathcal{F}_{\mathrm{EN}}, \mathcal{F}_{\mathrm{CN}}, \mathcal{F}_{\mathrm{SH}}$, may also be used for the upcoming methodologies due to their relation with $\mathcal{F}_{\mathrm{KR}}$. In the end, note that FE models can be represented via other mappings as well. The time dependence, for example, could be more clearly emphasized by grid functions. However, this is not necessary here. Consequently, the novel definitions proposed are preferred as they are more user friendly — in the author's opinion — for the purpose, i.e. the following methodology, of this thesis.

3.2 || Metamodeling

Everyday tasks of engineers include design space exploration, optimization, sensitivity analysis, see Section 3.3, or studies on other aspects of underlying models. Most analyses require thousands or even millions of model evaluations. In real-world problems, a single model evaluation can last minutes, hours, or even days, e.g. full vehicle crash simulations. Together, those analyses are not feasible due to huge time burdens. One possibility to tackle this bottleneck is to use metamodels as approximations. They intend to replace the actual model which yields high speed-ups as they are computationally cheap.

Metamodels — also called surrogate models, response surface models, or emulators — are mathematical constructs. They map input values of the actual model to an approximation of the actual model output. This output is originally conceived as a scalar quantity. In general, a classical metamodel consists of chosen mathematical terms, e.g. basis functions, and their hyperparameters, e.g. coefficients. These terms are combined via prescribed mathematical operations. To create the metamodel after defining its terms, the optimal values for their hyperparameters have to be found. These values can be determined by minimizing a selected loss function for a sample of actual model evaluations denoted as training sample. This leads to a trained metamodel that is meant to appropriately approximate outputs of the actual model for new input values.

In case there are outputs of higher dimension, several ways exist to enable metamodeling. A dimensionality reduction technique can be coupled to the metamodel when, for example, the output consists of spatio-temporal quantities, i.e. time histories of several different objects. Usually, in this context, output instances are represented in a matrix where the rows stand for the objects and the columns describe the time steps. The metamodel is then referred to as model order reduction metamodel. When the time history of a single object — reported in a vector — is considered, multi-target regression metamodels can be applied.

To distinguish the three possibilities for outputs of different magnitude, metamodels for scalar quantities are named scalar metamodels from now on. Summarizing, these three options — scalar, multi-target regression, model order reduction metamodels — are interesting for our applications: they can be applied to the four black box functions, $\mathcal{F}_{\mathrm{EN}}$ (3.11), $\mathcal{F}_{\mathrm{CN}}$ (3.13), $\mathcal{F}_{\mathrm{SH}}$ (3.16), $\mathcal{F}_{\mathrm{KR}}$ (3.19), that were described in Section 3.1.2. Diverse representatives of these methods will be discussed in the following. It is recommended to test several methods and compare their accuracy as the no free lunch theorems imply that there exists no universally best one, see [Wol02, HP02, Mur12, KJ13].

3.2.1 Creating the training sample

To find the optimal hyperparameters of metamodels, they are trained with $n_{\text{train}} \in \mathbb{N}$ actual model observations,

$$\left(x^1, y^1\right), \ldots, \left(x^{n_{\text{train}}}, y^{n_{\text{train}}}\right) \in \mathcal{X} \times \overline{\mathcal{Y}} \subset \mathbb{R}^n \times \overline{\mathcal{Y}}, \tag{3.21}$$

where it holds $y^i = \mathcal{F}\left(x^i\right)$ for $i \in \{1, \ldots, n_{\text{train}}\}$. Depending on the choice

$$\mathcal{F} \in \{\mathcal{F}_{\text{KR}}, \mathcal{F}_{\text{SH}}, \mathcal{F}_{\text{CN}}, \mathcal{F}_{\text{EN}}\}, \tag{3.22}$$

the output space can be of arbitrary but fixed dimension, i.e.

$$\overline{\mathcal{Y}} \in \left\{\mathcal{Y} \subset \mathbb{R}, \mathcal{Y}_{\text{SH}} \subset \mathbb{R}^{1 \times N_{\text{T}}}, \mathcal{Y}_{\text{CN}} \subset \mathbb{R}^{N_{\text{CN}} \times N_{\text{T}}}, \mathcal{Y}_{\text{EN}} \subset \mathbb{R}^{N_{\text{EN}} \times N_{\text{T}}}\right\}, \tag{3.23}$$

see Section 3.1.2. To create a design of experiments, i.e. the training sample, values for the training inputs, $x^1, \ldots, x^{n_{\text{train}}}$, are generated by sampling strategies. These inputs are then evaluated via the model, i.e. one simulation is started for each input. After the simulations are finished, outputs can be collected. It depends on the analyst which dimension for the output is chosen. The metamodel then tries to find a mathematical connection between the training inputs, $x^1, \ldots, x^{n_{\text{train}}}$, and their corresponding outputs, $y^1, \ldots, y^{n_{\text{train}}}$, by tuning the hyperparameters of chosen mathematical terms.

To achieve an appropriate fit, the training inputs should cover the input space as comprehensively as possible. That is, all regions of the input space should be included in the training set. This wishful thinking is countered by the fact that, for reasons of time, as few simulations as necessary should be calculated. Two sampling strategies are presented that, in theory, distribute the training inputs fairly throughout the space. For overall good approximations, the strategies are used to produce uniformly distributed numbers that best cover the whole input range,

$$\mathcal{X} = \prod_{i=1}^{n} I_i \subset \mathbb{R}^n, \tag{3.24}$$

cf. (3.10). For explaining the methods and their numerical realization, the single intervals, $I_i = [a_i, b_i] \subset \mathbb{R}$, see (3.9), are w.l.o.g. assumed to be equal to the interval $[0, 1]$, i.e. $\mathcal{X} = [0, 1]^n$. In fact, a number, ζ, of the interval $[0, 1]$ can be converted into a number of the compact interval I_i through the linear operation

$$(b_i - a_i)\zeta + a_i. \tag{3.25}$$

Certainly there are many sampling methods existent in literature, e.g. random sampling, orthogonal array sampling, etc., see [Tah16, LXW16]. Some are combined with the metamodel to sample more in the regions where the approximation is not satisfactory, e.g. adaptive sampling strategies, see [LOC18, LCO17, VMQ+13]. Nevertheless, the choice fell on the well-established methods Latin hypercube and Sobol' sequence sampling whose theories and advantaged are briefly described in the following.

Latin hypercube sampling
Latin hypercube sampling is a method for producing near-random points from a multi-dimensional distribution, see [MBC00]. It is based on the Latin square design: a square grid that contains sample positions where there is exactly one point in each row and column. For Latin hypercube sampling, the Latin square is extended to draw samples from multiple dimensions and hyperplanes. Each sample point is unique in any axis-aligned hyperplane that encloses it, see [HD03].

Based on how many sample points are desired, say $n_{\text{train}} \in \mathbb{N}$, the strategy divides each dimension, $I_i = [0, 1]$, of the input space into n_{train} equally probable disjoint intervals. As the elements of the input space, $\mathcal{X} = [0, 1]^n$, are considered to be uniformly distributed, each I_i can be divided into

$$\left[0, \frac{1}{n_{\text{train}}}\right), \left[\frac{1}{n_{\text{train}}}, \frac{2}{n_{\text{train}}}\right), \ldots, \left[\frac{n_{\text{train}} - 1}{n_{\text{train}}}, 1\right]. \tag{3.26}$$

For each input dimension, $i \in \{1, \ldots, n\}$, exactly one value is randomly drawn from each of the respective n_{train} intervals, i.e. for each of the n input parameters, n_{train} values are created. The established values for the dimensions are then combined — again at random — resulting in a $n_{\text{train}} \times n$ Latin hypercube sample matrix.

In [OSD03], the efficiency of the Latin hypercube sampling is shown and compared with standard sampling methods. One drawback of the strategy is its dependence on the desired sample size. The chosen sample size may not be large enough, e.g. to build well-approximating metamodels. Increasing this size by including another points may not be optimal in terms of covering the space as the initial sample point distribution is designed specifically for the original number of points chosen. For example, additional sampling points from a second Latin hypercube sample might not fill the empty space left by the first sample ideally.

Sobol' sequence sampling
In 1967, Sobol' introduced a quasi-random sequence whose numbers cover the sampling domain rapidly and evenly, see [Sob67]. This later called Sobol' sequence generates numbers taking into account the previously sampled points. It thereby avoids gaps and clusters, see [BJH11]. Numbers are generated using a base of two to create gradually finer uniform partitions of the unit interval. Afterwards, the coordinates are rearranged in each dimension. This minimizes the discrepancy between the points, see [Sob76].

Sobol' sequence sampling is of advantage when additional points are required to guarantee a good approximation. When using quasi-random sequences, accuracy typically increases continuously as more data points are added whereby the existing points are fully reused. Moreover, in [BJH11], the strategy is compared to other standard sampling procedures, e.g. Latin hypercube sampling, where it outperforms them in most of the analyzed aspects. Therefore, Sobol' sequence sampling is slightly preferred to Latin hypercube sampling in this thesis.

Data preprocessing
The approximation quality of metamodels highly depends on the quality and structure of the training data. Proper preparation of this data can result in improved accuracy.

Data preprocessing encompasses data cleaning, selection, scaling, etc., see [KKP06]. As the data comes from computer simulations, it is expected to be clean. There ought to be no unintended outliers as can occur when analyzing hardware instances, e.g. due to measurement errors or inaccuracies. On top of that, the data points should be well elected via the sampling method. Data selection can hence be skipped. Conversely, data scaling is worth to be taken into consideration.

Scaling is usually a linear transformation of the data. It is recommended to convert the n parameters of the model to have the same level of magnitude, i.e. scaling them to practically consistent ranges. Actually, scaling does not affect the metamodel theoretically as linear transformations can mostly be captured by adapting its hyperparameters. Indeed, after scaling, the calculation of these hyperparameters, e.g. the coefficient of the monomials in polynomial regressions, may become numerically more stable which can accelerate the training procedure, see [Wan19, AZI16, YHHZ20, IS15]. Scaling the output values can again improve the methods. Especially for neural networks, training outputs should have a similar range as the image of the chosen activation function, see Section 3.2.4.

There exist two common scaling techniques. On the one hand, data can be normalized, i.e. rescaled from its original range to the interval $[0, 1]$. For example, this can be done by the so-called MinMaxScaler,

$$\text{MinMaxScaler}\left(\xi^i\right) = \frac{\xi^i - \min\limits_{i\in\{1,\dots,n_{\text{train}}\}} \xi^i}{\max\limits_{i\in\{1,\dots,n_{\text{train}}\}} \xi^i - \min\limits_{i\in\{1,\dots,n_{\text{train}}\}} \xi^i}, \tag{3.27}$$

where $\xi^i, i \in \{1, \dots, n_{\text{train}}\}$, can be equal to one of the sampled parameters, $x_1^i, \dots, x_n^i$, or the sampled output, y^i, in case it holds $\overline{\mathcal{Y}} = \mathcal{Y}$, see [JNR05, LL11]. For higher dimensional spaces $\overline{\mathcal{Y}} \in \{\mathcal{Y}_{\text{SH}}, \mathcal{Y}_{\text{CN}}, \mathcal{Y}_{\text{EN}}\}$, this can be done for each output entry separately or for all together. On the other hand, data can be standardized that sets the mean of the values to zero and their standard deviation to one. This is accomplished by

$$\text{StandardScaler}\left(\xi^i\right) = \frac{\xi^i - \mu_\xi}{\sigma_\xi} = \frac{\xi^i - \frac{1}{n_{\text{train}}} \sum\limits_{i=1}^{n_{\text{train}}} \xi^i}{\sqrt{\frac{1}{n_{\text{train}}} \sum\limits_{i=1}^{n_{\text{train}}} \left(\xi^i - \frac{1}{n_{\text{train}}} \sum\limits_{i=1}^{n_{\text{train}}} \xi^i\right)^2}}, \tag{3.28}$$

with the same notations as for (3.27), see [Joe15]. After the metamodel is trained, new parameter values must obviously be scaled before evaluation. Gained output values need to be returned to their original range by corresponding inverse operations.

3.2.2 Quality measures

After training metamodels, their quality should be assessed. There are different measures that can be used for comparison. Basically, it can be distinguished between metrics that are calculated using the n_{train} training sample points and metrics that require new $n_{\text{val}} \in \mathbb{N}$ points forming the validation sample. Popular methods that only use the training sample are cross-validation techniques, e.g. π-fold or leave-one-out cross-validation, see

[Ber19]. They can also be used to train metamodels, i.e. to tune their hyperparameters, by employing them as regression cost functions.

Usually, several metamodels have to be built to compute cross-validation techniques. The training sample is therefore divided into two sets — often multiple times. One set is used to optimize the metamodel hyperparameters. The other set is taken to evaluate a loss function to assess how good the metamodel was trained to forecast the corresponding outputs. The π-fold cross validation repeats this process $\pi \in \mathbb{N}$ times. It considers π different pairs of actual training and evaluation sets, trains π metamodels, calculates their losses, and averages them.

The leave-one-out cross validation is a special case of the π-fold cross validation where it holds $\pi = n_{\text{train}}$. This means n_{train} metamodels are trained with $n_{\text{train}} - 1$ points and evaluated with the point missing that completes the training sample of size n_{train}. For some metamodels, computation of cross-validation techniques, e.g. the leave-one-out error, can be done by evaluating an analytical function, e.g. for Gaussian process regression and polynomial chaos expansion, see [SSW15]. However, they do not exist in general demanding to construct numerous independent metamodels.

Several single metamodels already have to be created or updated for finalizing specific metamodel variants that will be discussed in the next sections. Examples are scalar metamodels coupled to dimensionality reduction, certain multi-target regression and model order reduction metamodels. The number of metamodels to be trained would become tremendously large if cross-validation techniques were to be utilized. For this reason, these techniques are not chosen as quality measures. Instead, a second sample — the validation sample of size n_{val} — that is not shown to the metamodel during training is generated to evaluate selected error functions in order to compare performances.

The mean squared error (MSE) is a well-established option for these loss functions, see [WB09]. It calculates the average of the squares of the errors to measure the discrepancy between the actual and predicted data. The lower its values are, the better the approximation is. For the validation sample points,

$$\left(x^1, y^1\right), \ldots, \left(x^{n_{\text{val}}}, y^{n_{\text{val}}}\right) \in \mathcal{X} \times \overline{\mathcal{Y}} \subset \mathbb{R}^n \times \overline{\mathcal{Y}}, \tag{3.29}$$

in case $\overline{\mathcal{Y}} = \mathcal{Y} \subset \mathbb{R}$, the MSE is determined by

$$\text{MSE}(x^1, \ldots, x^{n_{\text{val}}}) = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \left(\mathcal{F}(x^i) - \mathcal{M}(x^i)\right)^2 = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \left(y^i - \hat{y}^i\right)^2 \tag{3.30}$$

where $\hat{y}^i \coloneqq \mathcal{M}(x^i), i \in \{1, \ldots, n_{\text{val}}\}$, are the outputs of the metamodel,

$$\mathcal{M} : \mathcal{X} \subset \mathbb{R}^n \to \overline{\mathcal{Y}}. \tag{3.31}$$

For $\overline{\mathcal{Y}} = \mathcal{Y}_{\text{SH}} \in \mathbb{R}^{1 \times N_{\text{T}}}$, MSE can be independently determined for each time step and, if desired, averaged to obtain a scalar value for all time steps. If $\overline{\mathcal{Y}} = \mathcal{Y}_{\text{CN}} \in \mathbb{R}^{N_{\text{CN}} \times N_{\text{T}}}$ or $\overline{\mathcal{Y}} = \mathcal{Y}_{\text{EN}} \in \mathbb{R}^{N_{\text{EN}} \times N_{\text{T}}}$, it can be done as for $\overline{\mathcal{Y}} = \mathcal{Y}_{\text{SH}}$, i.e. MSE can be computed separately for each time step and for each node. Then, individual errors can be averaged to errors for specific time steps, errors for chosen node histories, or the overall error.

A second popular metric is called the coefficient of determination (R^2), see [FHK+16]. It is measured by means of the proportion of the total variation in outputs represented by the metamodel. For $\overline{\mathcal{Y}} = \mathcal{Y}$, it can be computed through

$$R^2(x^1, \ldots, x^{n_{\text{val}}}) = 1 - \frac{\text{SS}_{\text{res}}(x^1, \ldots, x^{n_{\text{val}}})}{\text{SS}_{\text{tot}}(x^1, \ldots, x^{n_{\text{val}}})} = 1 - \frac{\sum_{i=1}^{n_{\text{val}}} (y^i - \hat{y}^i)^2}{\sum_{i=1}^{n_{\text{val}}} (y^i - \overline{y})^2} \tag{3.32}$$

where

$$\overline{y} = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} y^i \tag{3.33}$$

is the mean of the actual output values $y^1, \ldots, y^{n_{\text{val}}}$.

The nominator of the fraction in (3.32) is called the residual sum of squares (SS_{res}). Besides the factor $1/n_{\text{val}}$, it is equal to the MSE from (3.30). The denominator is called the total sum of squares (SS_{tot}). It indicates the total variation of the observed outputs. The coefficient of determination can take values up to one. If it is less than zero, the approximation quality of the metamodel is worse than just taking the mean value as the estimator for each output. This is due to the fact that it holds

$$\sum_{i=1}^{n_{\text{val}}} (y^i - \hat{y}^i)^2 = \sum_{i=1}^{n_{\text{val}}} (y^i - \overline{y})^2 \tag{3.34}$$

for $\hat{y}^1 = \hat{y}^2 = \ldots = \hat{y}^{n_{\text{val}}} = \overline{y}$ implying $R^2(x^1, \ldots, x^{n_{\text{val}}}) = 0$. Values between zero and one for R^2 mean obtaining better approximations than just using the mean values. The nearer they are to one, the tighter the fits are, i.e. $R^2 = 1$ describes a perfect emulation. Analogous to the MSE, the coefficient of determination can be broadened to $\overline{\mathcal{Y}} \in \{\mathcal{Y}_{\text{SH}}, \mathcal{Y}_{\text{CN}}, \mathcal{Y}_{\text{EN}}\}$. Alternatively, to expand R^2 for $\mathcal{Y}_{\text{EN}}$ for instance, one can let the sum in (3.32) run until $n_{\text{val}} N_{\text{EN}} N_{\text{T}}$ and consider the entries of the matrices, Y^i and $\hat{Y}^i \in \mathcal{Y}_{\text{EN}}, i \in \{1, \ldots, n_{\text{val}}\}$, as the scalar output values, y^j and $\hat{y}^j, j \in \{1, \ldots, n_{\text{val}} N_{\text{EN}} N_{\text{T}}\}$.

For both measures, there are no general thresholds at which one speaks of a good fit. This is totally problem dependent and will be declared for each application separately. Furthermore, there exist many more error measures. The mean absolute error, see [CD14], is similar to the mean squared error with the difference that it does not square the residuals but uses its absolute values. In this way, outliers are not punished as severely as in the MSE. Besides, the R^2 can be adapted to the R^2 adjusted, see [Mil05], that sentences the number of hyperparameters used. This is to prevent using unnecessary extra terms for the metamodel. However, it is not necessary here to keep the size of the metamodel small. In fact, the best-performing model — but possibly entailing many terms — should be chosen as, in particular for crash applications, each digit can count. Besides that, there exist several other measures, e.g. the mean squared weighted deviation [WC91], the root mean squared error [HK06], the mean absolute percentage error [DMGLGR16], etc.

Additionally, actual versus predicted plots (AvP) illustrate how well the data is approximated. Actual model outputs from the validation sample are plotted against corresponding outputs of the metamodel. The closer the points lie at the bisectrix, the better the fit is. For time histories of objects, e.g. nodes, all output entries can be shown in one AvP. Alternatively, separate AvPs can be plotted for each time step or each object. The actual and the predicted time history of one parameter configuration can be compared in quantity-time diagrams. The last two options, however, can require plenty of plots dependent on the number of time steps, N_{T}, and the number of validation points, n_{val}.

In the following, to the best of the author's knowledge, a selection of the currently most sophisticated methods for different metamodel options are briefly introduced. However, there exist several more techniques that can be employed within the framework. For comparison purposes, the well-known multiple linear regression and versions of it are additionally given.

3.2.3 Scalar metamodels

The most common and probably easiest way is to use metamodels for scalar outputs. Scalar metamodels emulate black box functions that have the form of the key result function,

$$\mathcal{F}_{\mathrm{KR}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y} \subset \mathbb{R}, \tag{3.35}$$

see (3.19), i.e. the output space, $\mathcal{Y}$, is embedded in the real numbers. Therefore, a metamodel,

$$\mathcal{M}_{\mathrm{KR}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y} \subset \mathbb{R}, \tag{3.36}$$

is a mapping between the spaces of the underlying model. It aims to approximate this model, i.e.

$$\mathcal{M}_{\mathrm{KR}}(x) \approx \mathcal{F}_{\mathrm{KR}}(x), \tag{3.37}$$

for $x \in \mathcal{X}$. It is trained with the sample points from (3.21) using $\overline{\mathcal{Y}} = \mathcal{Y} \subset \mathbb{R}$.

Multiple linear regression

Perhaps the simplest method for a metamodel with a single parameter, i.e. $n = 1$, is called simple linear regression, see [AK15]. It assumes that the relationship between this parameter and the output is linear. Graphically, this relationship can be expressed by a straight line. Multiple linear regression (MLR) extends this idea to a model with more than one parameter. It also supposes that a linear connection between the parameters, $x \in \mathcal{X} \subset \mathbb{R}^n$, and the outputs, $y \in \mathcal{Y} \subset \mathbb{R}$, can be established.

Mathematically speaking, it takes the function

$$\mathcal{M}_{\mathrm{KR}}^{\mathrm{MLR}}(x) = \beta^T x + \beta_0 \tag{3.38}$$

as a starting point. Its coefficients, $\beta = (\beta_1, \ldots, \beta_k) \in \mathbb{R}^k$ and $\beta_0 \in \mathbb{R}$, need to be adjusted to best fit the n_{train} observations, see [MPV21]. Usually, this is accomplished by solving a least-squares optimization problem, e.g. finding β and β_0 that minimize

$$\sum_{i=1}^{n_{\text{train}}} \left(y^i - \mathcal{M}_{\text{KR}}^{\text{MLR}}(x^i)\right)^2 = \sum_{i=1}^{n_{\text{train}}} \left(y^i - \hat{y}^i\right)^2. \tag{3.39}$$

Thanks to its simple nature, tuning the MLR and producing predictions is quick — notwithstanding whether high dimensional parameters appear, i.e. for large n. The shortage in terms of approximation quality is reached as soon as there exist nonlinear behavior between the spaces, $\mathcal{X}$ and $\mathcal{Y}$.

Gaussian process regression

Also known as kriging, Gaussian process regression (GPR) is a widely practiced regression method for interpolation, see [SWMW89, SWN03]. Danie Krige originally developed the method to find gold spots at the Witwatersrand reefs in South Africa based on information from some boreholes, see [Kri51]. His idea was then cast into a metamodel.

GPR provides a function $\mathcal{M}_{\text{KR}}^{\text{GPR}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y} \subset \mathbb{R}$ that interpolates between the n_{train} observations, i.e. the kriging function is exact at the training data,

$$\mathcal{M}_{\text{KR}}^{\text{GPR}}\left(x^i\right) = y^i, i \in \{1, \ldots, n_{\text{train}}\}. \tag{3.40}$$

In [MBGS18] for instance, the function is formulated as

$$\mathcal{M}_{\text{KR}}^{\text{GPR}}(x) = \beta^T \varphi(x) + Z(x) = \sum_{p=1}^{P} \beta_p \varphi_p(x) + Z(x), \tag{3.41}$$

where the first term, $\beta^T \varphi(x)$ called trend, is composed of the weight vector $\beta \in \mathbb{R}^P, P \in \mathbb{N}$, and the preselected regression function vector $\varphi : \mathbb{R}^k \to \mathbb{R}^P$ of length P, e.g. consisting of polynomials. The second term, $Z(x)$, is a realization of a Gaussian process with zero mean and variance σ^2.

Numerically, auto-covariance functions, e.g. Matérn kernel functions,

$$\mathcal{K}_{\theta,\nu}(x, x') = \prod_{i=1}^{n} \frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\sqrt{2\nu}\frac{|x_i - x_i'|}{\theta_i}\right)^{\nu} \kappa_\nu \left(\sqrt{2\nu}\frac{|x_i - x_i'|}{\theta_i}\right), \tag{3.42}$$

see [SSW15], where it holds $x, x' \in \mathcal{X}$, $\nu \geq 1/2$, Γ is the Euler gamma function, and κ_ν is the modified Bessel function of second kind, are chosen to model $Z(x)$, see [RW06] for other possible choices. The kriging model is calibrated by finding the optimal hyperparameter values, θ, of the auto-covariance functions. They may be obtained by maximum-likelihood-estimation or leave-one-out cross-validation, see [SSW15]. Least-squares optimization can then yield the kriging parameters, $\beta = \beta(\theta)$ and $\sigma^2 = \sigma^2(\theta)$, see [SSM16] for analytical expressions. Inserting tuned hyperparameters into the kriging model, $\mathcal{M}_{\text{KR}}^{\text{GPR}}$, allows for quick output prediction of a new input configuration. The calculated estimate, $\hat{y} = \mathcal{M}_{\text{KR}}^{\text{GPR}}(x)$, for $x \in \mathcal{X}$, is a Gaussian random variable. Its mean is taken as proxy for the actual outcome. Additionally, its variance measures local error.

Three types of kriging are named in the literature: simple, ordinary, and universal kriging, see [Jos06]. The first term, $\beta^T \varphi(x)$, in (3.41) sets the type:

1. For simple kriging, it holds $\beta^T \varphi(x) = \beta_0$, where $\beta_0 \in \mathbb{R}$ is a prescribed fixed constant.
2. One parameter, $\beta_1 \in \mathbb{R}$ for $P = 1$ where $\varphi_1(x) = 1$, must be optimized for ordinary kriging.
3. The general formulation $\beta^T \varphi(x)$ is called universal kriging.

Many works exist that suggest ways to improve the presented concept of kriging for specific problems, see [UCD+16, BBOM16], and show practical applications, see [MSBG14, MSBG15] — even for vehicle safety examples.

Polynomial chaos expansion

Another state-of-the-art method is called polynomial chaos expansion (PCE). The core idea of PCE is to emulate the actual model by an infinite series of polynomials. It works with orthogonal polynomial basis functions that build metamodels of the form

$$\mathcal{M}_{\mathrm{KR}}^{\mathrm{PCE}}(x) = \sum_{\alpha \in \mathcal{A}} a_\alpha \psi_\alpha(x), \tag{3.43}$$

see [BS11, BS08]. The quantity $\mathcal{A} \subset \mathbb{N}^n$ is called the truncation set. It makes the expansion finite. This is required as infinite series cannot be implemented in practice.

So, the truncation set contains a finite number of multi-indices for the polynomial basis functions, ψ_α, and their coefficients, $a_\alpha, \alpha \in \mathcal{A}$. There exist different techniques to choose this set, e.g. using the hyperbolic truncation set, see [SSW15], running an adaptive algorithm, see [BS10], etc. The bases, ψ_α, are selected dependent on the distribution of the parameters. Considering uniformly distributed variables, it is common practice to choose Legendre polynomials, see [XK02] for other distributions. The degrees of ψ_α are a-priori restricted to range up to a preset number $P \in \mathbb{N}$.

Once the set of candidate polynomials is defined, optimal coefficients, $a_\alpha, \alpha \in \mathcal{A}$, of the basis functions, ψ_α, must be calculated. A least-squares minimization on the training data can be utilized for this task, cf. (3.39). Over the last years, many other variants were developed that are compared and explained in [LMS21]. Some of them use sparse regression to make the expansion even shorter. By doing so, the significant coefficients of the polynomial chaos expansion are detected leading to a smaller size of $\mathcal{A}$. This can avoid the curse of dimensionality which may appear when n is large. The expansion is hereby shortened again and therefore computationally tractable. This, for example, can be realized by adding regularization terms to the least-squares optimization. Furthermore, least angle regression, the least absolute shrinkage operator, or a combined version can be used, see [BS11, LMS21]. Another strategy comes from the context of compressive sensing that allows to solve an underdetermined system of equations, see [LMS21].

In recent developments, the combination of polynomial chaos expansion and kriging was investigated, see [SSW15, SSM16]. So-called polynomial chaos kriging is categorized

as universal kriging. There, the polynomials are chosen as the regression function vector, φ, and their coefficients as the weight vector, β, see (3.41). This leads to the formula

$$\mathcal{M}_{\mathrm{KR}}^{\mathrm{PCK}}(x) = \sum_{\alpha \in \mathcal{A}} a_\alpha \psi_\alpha(x) + Z(x), \tag{3.44}$$

where $Z(x)$ is again a realization of a Gaussian process with zero mean and variance σ^2. The optimal set of polynomials is determined first before the kriging trend, i.e. the PCE coefficients, is tuned and its auto-covariance function is tweaked.

3.2.4 Multi-target regression metamodels

The second option for metamodels is their application to vectors, e.g. time histories, from which scalar quantities of interest may be extracted. Multi-target regression (MTR) can be employed for this job. A survey on this strategy can be found in [BVBL15] that lists multi-output, multi-variate, or multi-response regression as its synonyms. MTR metamodels approximate models like the history function,

$$\mathcal{F}_{\mathrm{SH}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y}_{\mathrm{SH}} \subset \mathbb{R}^{1 \times N_{\mathrm{T}}}, \tag{3.45}$$

see (3.16). Consequently, the metamodel,

$$\mathcal{M}_{\mathrm{SH}} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y}_{\mathrm{SH}} \subset \mathbb{R}^{1 \times N_{\mathrm{T}}}, \tag{3.46}$$

maps from the input space to the space of vectors, $\mathcal{Y}_{\mathrm{SH}}$. Hence, the training points must all be vectors, i.e.

$$y^i = \left(y_1^i, \ldots, y_{N_{\mathrm{T}}}^i\right) \in \mathcal{Y}_{\mathrm{SH}}, i \in \{1, \ldots, n_{\mathrm{train}}\}. \tag{3.47}$$

The scalar metamodels from Section 3.2.3 can be extended to MTR metamodels either via problem transformation methods or algorithm adaption methods, see [BVBL15]. Two problem transformation methods are shown in the following. Moreover, artificial neural networks that may be count to algorithm adaption methods are introduced. Actually, the networks can solve a multi-task learning problem, see [ZY21]. This problem may deviate from multi-output regression: its tasks are allowed to have different training sets or disparate descriptive features, see [BVBL15]. In this thesis, however, these circumstances are identical. The networks thus are considered as MTR methods. Other algorithm adaption methods are not treated as their creation depends on the scalar metamodel chosen. They usually need to be formulated specifically for each emulator, see the adaptions of Gaussian process regression for multiple outputs in [AL08, GB16].

Single-target method

The single-target (ST) method is the most basic MTR metamodel that gathers N_{T} independent scalar metamodels — one for each output, see [BVBL15]. Therefore, the output vector, $y = (y_1, \ldots, y_{N_{\mathrm{T}}}) \in \mathcal{Y}_{\mathrm{SH}} \subset \mathbb{R}^{1 \times N_{\mathrm{T}}}$, is divided into N_{T} scalar values, i.e. its components $y_1, \ldots, y_{N_{\mathrm{T}}}$. The j-th scalar metamodel, $\mathcal{M}_{\mathrm{KR}}^j$, inside the single-target method

is trained with all j-th entries of the training vectors, i.e. $y_j^1, \ldots, y_j^{n_{\text{train}}}, j \in \{1, \ldots, N_{\text{T}}\}$. After the independent training sessions, ST method can be written as

$$\mathcal{M}_{\text{SH}}^{\text{ST}} = \left(\mathcal{M}_{\text{KR}}^1, \ldots, \mathcal{M}_{\text{KR}}^{N_{\text{T}}}\right). \tag{3.48}$$

Regressor chains-based approach

The concept of regressor chains is to link scalar metamodels. Originally, a random permutation, i.e. chain, of output instances is drawn. Then, individual scalar metamodels are created for the output instances step by step following the order of the chain, see [BVBL15]. Thereby, in the current stage, the metamodel shall profit from the information gained in the previous stage.

Let the output $y = (y_1, \ldots, y_{N_{\text{T}}}) \in \mathcal{Y}_{\text{SH}}$ without loss of generality be already permuted. To begin with, the first ordered output target, y_1 of $y \in \mathcal{Y}_{\text{SH}}$, is approximated by the first scalar metamodel that takes merely the original n parameters, $x \in \mathcal{X}$, into account. The second scalar metamodel uses $n+1$ parameters, namely (x, y_1), to forecast the second ordered output, y_2. Next, the third ordered target, y_3, is the outcome of the third scalar metamodel that takes $n+2$ parameters, (x, y_1, y_2), into account. In general, the i-th scalar metamodel maps $(x, y_1, \ldots, y_{i-1})$ to the i-th ordered output, $y_i, i \in \{2, \ldots, N_{\text{T}}\}$. In this way, the single scalar metamodels are connected.

The regressor chains method entails that the evaluation must be done step by step as the previous output or outputs are taken as additional parameters for the current output. FE crash simulations face long temporal sequences as output vectors. For them, especially the last scalar metamodel and its predecessors might suffer the curse of dimensionality, see [Don00, Tru79], leading to poor performances. This might occur as they operate on large input spaces, e.g. the space of the last metamodel would be $(n + N_{\text{T}} - 1)$-dimensional.

Accordingly, the regressor chains method is adjusted for long time histories to avoid the curse of dimensionality. For non-sequential targets, testing different permutations might lead to improved results. The adapted regressor chains-based (RC) approach, however, does not permute the time sequence. This sequence should not be touched as the standard order is correct: the i-th time step follows the $(i-1)$-th in a dependent manner due to the FE model solver. In addition, to keep the input dimension low, only the previous scalar output is considered as extra parameter, respectively.

The obtained MTR metamodel hence reads

$$\mathcal{M}_{\text{SH}}^{\text{RC}}(x) = \left(\mathcal{M}_{\text{KR}}^1(x), \mathcal{M}_{\text{KR}}^2\left(x, \mathcal{M}_{\text{KR}}^1(x)\right), \ldots, \mathcal{M}_{\text{KR}}^{N_{\text{T}}}\left(x, \mathcal{M}_{\text{KR}}^{N_{\text{T}}-1}(x)\right)\right). \tag{3.49}$$

Note that one could also use the previous two, three, or more outputs in addition to the parameters, x. In [LN21] — a Master thesis tutored by the author of this work — a similar approach is applied to time histories of dummy signals from FE simulations of a full frontal crash. The results indicate that, if at all, taking into account the previous output as proposed in (3.49) leads to improved results. Prior time steps did not enhance the metamodel. Due to its knowledge of past information, the RC approach slightly outperformed the ST method. Both approaches were beaten by feed-forward neural networks

that are introduced next. The ST method and the RC approach, however, were coupled with MLR. Other scalar metamodels, e.g. GPR or PCE, should be employed within multi-target regression as well and then again compared with neural networks. This is done in Section 3.2.7.

Neural Networks

Artificial neural networks belong to machine learning of artificial intelligence, see [NJ18]. If they are to learn a function based on observed input-output pairs, they are specifically assigned to the subcategory called supervised learning. Metamodels from uncertainty quantification and supervised learning methods are related. In fact, they differ slightly in where they originate but not in what can be done with them and how they work. Both need observations to be trained with. They attempt to find the mapping between input and output spaces using mathematical expression, see [TMES19]. Metamodels replace computationally expensive models whose parameters are uncertain variables. Training data is generated by evaluating the actual model at prescribed parameter configurations, e.g. produced by a sampling strategy, see Section 3.2.1.

Supervised learning tries to set up a function connecting the presented input-output pairs. The training data must not come from sampling parameter values and evaluation of the actual model at these points. On the contrary, there does not necessarily have to be a model that shall be approximated. Data can come from other sources, e.g. real-world measurements. This data is then linked to a mapping by neural networks. Nevertheless, supervised learning methods can handle sample points from actual models. Neural networks can thus be used for metamodeling. The other way round, most metamodels are naturally designed, e.g. MLR [KVR16] and GPR [Ras03], or can be adapted, e.g. PCE [TMES19], to tackle supervised learning problems.

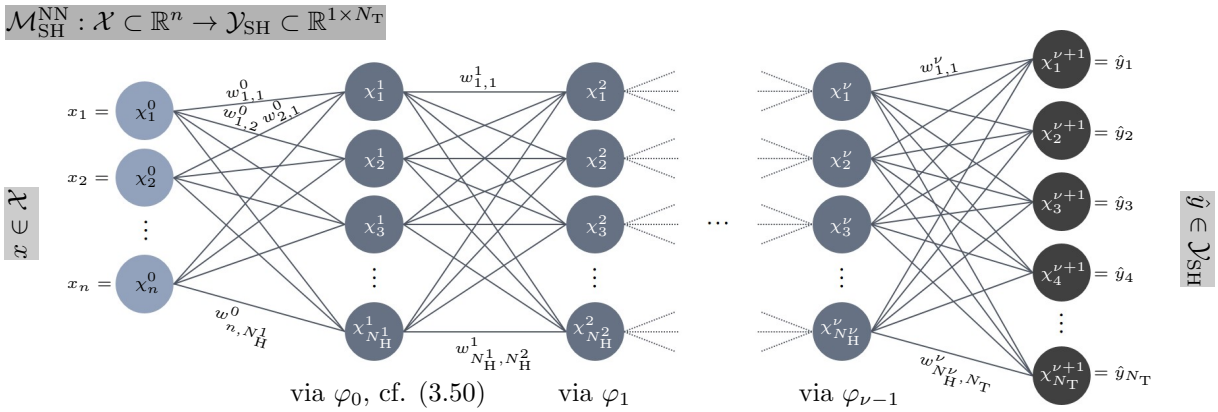


Figure 8: Sketch of a feed-forward neural network architecture. Exemplary components — neurons, weights, and activation functions — of the network are labeled.

As a metamodel, neural networks reproduce complex relationships between sampled input-output training points by forming hierarchically connected layers, see [Spe91]. Conformably, the universal approximation theorem, see [HSW89, KB20], states that standard feed-forward neural networks are capable of emulating any measurable function to any level of precision. Feed-forward neural networks let information flow in forward direction

without using cycles, see [Zel94]. They are also called multilayer perceptron and consist of at least three layers: the input layer, one or more hidden layers, and the output layer, see Fig. 8. The input layer receives the input parameters and passes them to the first hidden layer. Inside the hidden layers, computations are performed which are — from the last hidden layer — sent to the output layer. There, they are converted to results of the actual output space, $\mathcal{Y}_{\mathrm{SH}}$. Note that neural networks (NN) with more than one hidden layer are often called deep neural networks, see [HDY+12].

Each layer is composed of units also called neurons. The input and the output layer have the same dimension as the input and the output space, i.e. the input layer comprises n units and the output layer is made of N_{T} neurons. The $\nu \in \mathbb{N}$ hidden layers consist of arbitrary but fixed numbers of units, $N_{\mathrm{H}}^i, i \in \{1, \ldots, \nu\}$. By raising the complexity of the neural network, i.e. increasing the number of layers or units, the information is gradually decomposed into finer individual features. Thereby, each unit of a given layer is fully connected to the units of the previous layer. Mathematically, fully connected means summing up all weighted neurons of the previous layer and subsequently applying a chosen activation function, $\varphi_i, i \in \{0, 1, \ldots, \nu\}$, see [Hay10]. The activation functions for the neurons between two layers are the same. They can be chosen differently when the connection between other two layers must be prepared. For example, the activation functions between hidden layer 1 and hidden layer 2 can be of type A whereas the functions between hidden layer 2 and hidden layer 3 can be of type B.

In more detail, for the j-th neuron, χ_j^{i+1}, of the $(i+1)$-th hidden layer, the neural network connections can be expressed as

$$\chi_j^{i+1} = \varphi_i \left(\sum_{k=1}^{N_{\mathrm{H}}^i} w_{k,j}^i \chi_k^i + \beta_j^{i+1} \right) \tag{3.50}$$

for $j \in \left\{1, \ldots, N_{\mathrm{H}}^{i+1}\right\}, i \in \{0, 1, \ldots, \nu\}$. The weights, $w_{k,j}^i \in \mathbb{R}$, as well as the biases, $\beta_j^{i+1} \in \mathbb{R}$, are the hyperparameters that have to be adjusted to the training data. The neurons of the input and the output layer are denoted as $\chi_1^0, \ldots, \chi_n^0$ and $\chi_1^{\nu+1}, \ldots, \chi_{N_{\mathrm{T}}}^{\nu+1}$ with $N_{\mathrm{H}}^0 = n$ and $N_{\mathrm{H}}^{\nu+1} = N_{\mathrm{T}}$, respectively.

There are many possibilities for activation functions, φ_i, see [SS17]. Three of them are shown here. The identity (id) activation function,

$$\varphi_{\mathrm{id}}(\chi) = \chi, \tag{3.51}$$

does not influence the weighted sum. A range from -1 and 1 is provided by the hyperbolic tangent (tanh) activation function,

$$\varphi_{\tanh}(\chi) = \frac{\exp(\chi) - \exp(-\chi)}{\exp(\chi) + \exp(-\chi)}. \tag{3.52}$$

The rectified linear unit (relu) activation function

$$\varphi_{\mathrm{relu}}(\chi) = \max\{0, \chi\} \tag{3.53}$$

removes negative values. To work in the correct ranges, the data is normalized during preprocessing using scaling functions, e.g. from Section 3.2.1. In this thesis, MinMaxScaler (3.27) is employed.

Formula (3.50) shows the recursive relationship between the neurons of the current layer and the units of the previous layer, which in turn are related to the neurons of the preceding layer, and so on. These relationships can be found in Fig. 8 which represents the overall appearance of the network. The beauty of neural networks as a multi-target regression metamodel is that — unlike the ST method and the RC approach — only one closed model is set up to predict the complete output vector. This means it holds

$$\hat{y} = (\hat{y}_1, \dots, \hat{y}_{N_\mathrm{T}}) = \left(\chi_1^{\nu+1}, \dots, \chi_{N_\mathrm{T}}^{\nu+1}\right) = \mathcal{M}_{\mathrm{SH}}^{\mathrm{NN}}(x) \approx y \in \mathcal{Y}_{\mathrm{SH}} \tag{3.54}$$

for the corresponding input, $x = (x_1, \dots, x_n) = (\chi_1^0, \dots, \chi_n^0) \in \mathcal{X}$, cf. Fig. 8. Neural networks are also powerful instruments to approximate scalar values. In this case, the output layer consists of one neuron, $N_\mathrm{T} = 1$. The space $\mathcal{Y}_{\mathrm{SH}} \subset \mathbb{R}^{1 \times N_\mathrm{T}}$ can be replaced by the space $\mathcal{Y} \subset \mathbb{R}$. Therefore, neural networks can also be allocated to scalar metamodels, see (3.36), i.e. they approximate the key result function, see (3.19).

To train feed-forward neural networks, i.e. to update their weights and biases, a cost function is minimized. The cost function, e.g. a least-squares formula, is evaluated through forward propagation of the information through the network. The hyperparameters, i.e. the weights and the biases, are then updated through sending the cost backwards through the network. More precisely, gradients are calculated using an inexpensive procedure called back-propagation, see [GBC16]. A first-order optimization can then be used to revise the current hyperparameters. Normally, versions of the gradient descent algorithm, e.g. stochastic gradient descent [Bot91] or adaptive moment estimation [BW19], are applied. For an appropriate fit, these methods include adjusting the learning rate, e.g. the step size of the gradient descent, and the number of epochs, i.e. the iterations to be performed. Moreover, it often helps to use preprocessing for the training data, see Section 3.2.1. The convergence speed may be improved as the activation functions are more effective for specific ranges.

Recently, feed-forward neural networks are further developed to additionally yield uncertainty assessments of predictions. Stochastic neural networks like Bayesian networks quantify inference uncertainty, see [JBB+20]. This is an interesting extra information that can be used for error analysis but it is not necessary for the proposed framework. Besides feed-forward neural networks, there exist other architectures that may be applied to the considered problem. Recurrent neural networks, especially with long short-term memory, see [HS97], seem to be ideal to be dedicated to long time sequences, e.g. history function, see (3.16). Internally, they integrate loops, i.e. cycles, to establish sequential dependencies, see [GBC16]. At first glance, they should improve predictions.

In the mentioned Master thesis [LN21], it is shown that there is no benefit in using recurrent neural networks for the considered task, i.e. mapping independent parameters to time sequential output vectors. To start with, there is no temporal sequence in the input space as, for example, in time series forecasting where recurrent neural networks are attractive to continue the time development of historical data, see [SK19]. Here, temporal dependencies only exist in the output space. In this setting, recurrent neural networks may outperform feed-forward networks when there is a probabilistic context for

the output, see [LN21]. That is, when it is advantageous or even necessary to know the previous members of the sequence. For instance, in text-related problems, e.g. image captioning, the current word is highly dependent on the previous word. Recurrent neural networks can — by means of hidden states — provide probabilities for words to be chosen at specific locations, remember them, and thus improve the quality of the output sentence.

The examined problem, however, does not possess this kind of randomness. Time steps are to keep in a fixed order. For each time step, the single error must be minimized. The hidden state of the recurrent neural network does not contribute new information as it merely stores deterministic transformations of the input parameters. Moreover, training recurrent neural networks may be difficult and time-consuming, see [LN21]. Therefore, feed-forward neural networks are used for the applications of this thesis.

3.2.5 Model order reduction metamodels

Model order reduction (MOR) metamodels — also known as reduced order modeling — are the third and last variant presented here to approximate black boxes. They are used when it is intended to emulate output matrices. These matrices can have large sizes, e.g. when a whole crash model must be reconstructed. They would then have dimension $N_{\mathrm{EN}} \times N_{\mathrm{T}}$, see the entirety function, $\mathcal{F}_{\mathrm{EN}}$ (3.11). The same can be done for the component function, $\mathcal{F}_{\mathrm{CN}}$ (3.13), where the output dimension is $N_{\mathrm{CN}} \times N_{\mathrm{T}}$. Thus, the larger the values of N_{EN} or N_{CN} become, the more rows and hence entries appear in the matrices and the more values must be predicted.

MTR metamodels like ST method or RC approach must create $\overline{N} \cdot N_{\mathrm{T}}$, $\overline{N} \in \{N_{\mathrm{EN}}, N_{\mathrm{CN}}\}$, single scalar metamodels — one for each matrix entry. This can become practically infeasible, e.g. when the product $\overline{N} \cdot N_{\mathrm{T}}$ tends to a million or more. In practice, the number of nodes, $\overline{N}$, in full vehicle simulations can alone be equal to a few millions.

Therefore, when a metamodel of the form

$$\mathcal{M}_\gamma : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N} \times N_{\mathrm{T}}}, \tag{3.55}$$

for $\gamma \in \{\mathrm{EN}, \mathrm{CN}\}$, where $\overline{N} \cdot N_{\mathrm{T}}$ is large, shall be implemented, the dimension of the output space should be reduced with dimensionality reduction techniques, cf. [GH18, HU18, MP21, SMA19]. This combination of dimensionality reduction and metamodel is here referred to as MOR metamodel. Here, only the number of matrix rows, e.g. nodes, $\overline{N}$, are reduced although it can also be attempted to reduce the number of columns, e.g. time steps, N_{T}. The reason for not shrinking both is that the less fidelity is removed, the more likely it is to keep the reduction error low. As $\overline{N}$ is usually larger than N_{T}, $\overline{N}$ is preferred to be decreased — to reach a small number of required MTR metamodels.

To achieve that, the dimensionality reduction seeks to find the important characteristics of the rows, e.g. the most significant time histories of finite element nodes. It then transforms the original, long $\overline{N} \times N_{\mathrm{T}}$ matrix to a short matrix of size

$$N_{\mathrm{red}} \times N_{\mathrm{T}}, N_{\mathrm{red}} \in \mathbb{N}, N_{\mathrm{red}} < \overline{N}, \tag{3.56}$$

by compressing the principal information of all $\overline{N}$ rows. This process can be denoted as the projection mapping

$$\text{Proj} : \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N} \times N_\text{T}} \to \mathcal{Y}_\text{red} \subset \mathbb{R}^{N_\text{red} \times N_\text{T}}, Y \mapsto Y_\text{red}. \tag{3.57}$$

In this way, only N_red MTR metamodels must be created making the procedure feasible if N_red is sufficiently small.

These MTR metamodels together operate in the reduced space, $\mathcal{Y}_\text{red} \subset \mathbb{R}^{N_\text{red} \times N_\text{T}}$. To have the predictions in the correct dimensions, the approximated matrices from the reduced space, $\mathcal{Y}_\text{red}$, need to be projected back into the full space, $\mathcal{Y}_\gamma$, see Fig. 9 for the whole process. This can be done by the exact or an approximated inverse operation of the forward projection, Proj, also called back projection,

$$\text{Back} : \mathcal{Y}_\text{red} \subset \mathbb{R}^{N_\text{red} \times N_\text{T}} \to \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N} \times N_\text{T}}, Y_\text{red} \mapsto Y. \tag{3.58}$$

The error made by this procedure can be computed, for example, by using the Frobenius norm. This norm must be applied to the differences between actual matrices, $Y^k \in \mathcal{Y}_\gamma$, and their projected and subsequently back-projected versions,

$$\hat{Y}^k = \text{Back}\left(Y^k_\text{red}\right) = \text{Back}\left(\text{Proj}\left(Y^k\right)\right) \in \mathcal{Y}_\gamma, \quad k \in \{1, \dots, n_\text{train}\}, \tag{3.59}$$

respectively. This can then be averaged, i.e. the error is written as

$$\frac{1}{n_\text{train}} \sum_{k=1}^{n_\text{train}} \left\| Y^k - \hat{Y}^k \right\|_\text{F} = \frac{1}{n_\text{train}} \sum_{k=1}^{n_\text{train}} \sqrt{\sum_{i=1}^{\overline{N}} \sum_{j=1}^{N_\text{T}} \left| y^k_{ij} - \hat{y}^k_{ij} \right|^2}. \tag{3.60}$$

Note that the mapping Back may be approximated by regression models as its form may not be derived analytically depending on the projection mapping, Proj, chosen.

The dimensionality reduction is coupled with MTR metamodels, e.g. from Section 3.2.4, as follows. The projection mapping, Proj, converts the training matrices,

$$Y^1, \dots, Y^{n_\text{train}} \in \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N} \times N_\text{T}}, \tag{3.61}$$

to matrices,

$$Y^1_\text{red} = \text{Proj}\left(Y^1\right), \dots, Y^{n_\text{train}}_\text{red} = \text{Proj}\left(Y^{n_\text{train}}\right) \in \mathcal{Y}_\text{red} \subset \mathbb{R}^{N_\text{red} \times N_\text{T}}, \tag{3.62}$$

of reduced dimension. Then, N_red MTR metamodels are trained — one for each row of the reduced matrices. Concretely, the i-th rows of $Y^1_\text{red}, \dots, Y^{n_\text{train}}_\text{red}$ are used for training the i-th MTR metamodel, $i \in \{1, \dots, N_\text{red}\}$. These MTR metamodels are merged to output a matrix of dimension $N_\text{red} \times N_\text{T}$. Via the mapping Back, this matrix is then projected up into full space, $\mathcal{Y}_\gamma$, leading to the approximation of the actual output matrix.

Summarizing, the mapping (3.55) can be read as the function composition

$$\mathcal{M}_\gamma = \text{Back} \circ \begin{pmatrix} \mathcal{M}^1_\text{SH} \\ \vdots \\ \mathcal{M}^{N_\text{red}}_\text{SH} \end{pmatrix}, \tag{3.63}$$

where models $\mathcal{M}_{\mathrm{SH}}^{i}$ are trained with the i-th rows of $Y_{\mathrm{red}}^{j} = \mathrm{Proj}\left(Y^{j}\right), i \in \{1, \ldots, N_{\mathrm{red}}\}$, $j \in \{1, \ldots, n_{\mathrm{train}}\}$, cf. Fig. 9. For this combined approach, hyperparameters for the dimensionality reduction as well as for the MTR metamodels must be tuned. This can be done in two ways.

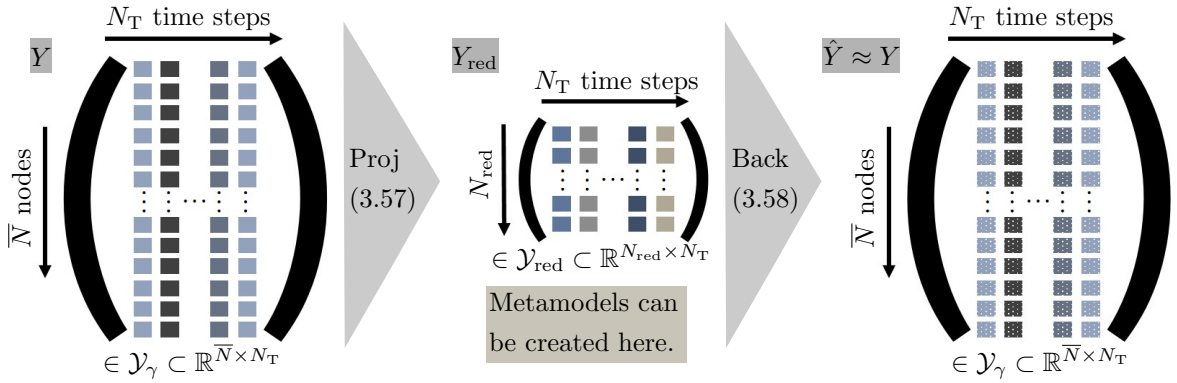


Figure 9: Process of dimensionality reduction including the place for metamodels. The bright pixels in the rectangles, i.e. the entries of matrix $\hat{Y}$, represent the projection error.

On the one hand, both methods can be optimized independently. By doing so, the hyperparameters of the dimensionality reduction and its inverse are found first, e.g. by minimizing the error of the averaged Frobenius error formula, see (3.60). Then, N_{red} MTR metamodels are trained — one for each row of the reduced matrices. Executing an output compression without direct consideration of the metamodel performances may lead to highly complex input-output mappings that are afterwards difficult to approximate. So, on the other hand, a nested optimization problem can be formulated as a remedy.

This concept orientates on [LMS20] where, in fact, the input dimension is reduced via dimensionality reduction to improve scalar metamodels. This can be translated to be used for MOR metamodels as done in a Master thesis [Go21] tutored by the author of this work. The mentioned optimization problem consists of an inner and an outer loop. The outer loop searches for the optimal hyperparameters of the dimensionality reduction. The inner loop fits the chosen MTR metamodels to the then reduced data. The algorithm stops when the approximation error that is made by the combination, i.e. the present MOR metamodel, regarding the training data is minimized. By doing so, the dimensionality is lowered in a way that endows the resulting matrices with suitable structures that facilitate the creation of MTR metamodels. Further details can be found in Section 3.2.6 where the original method from [LMS20] is presented.

There exist several methodologies for the dimensionality reduction and its inverse operation. For this purpose, principal component analysis and its nonlinear variant, kernel principal component analysis, are discussed below. They are used in several works as components of MOR metamodels, see [JLG21a, Go21, GH19, RGGZ^{+}21, Roh15]. Other methods, e.g. isomap, local linear embedding, autoencoder, discrete empirical interpolation method, see [vdMPvdH09, Kli13, KE19], however, could also be employed but are not part of this thesis. These methods can be connected with any MTR metamodel from Section 3.2.4.

Principal component analysis
A popular dimensionality reduction technique is called principal component analysis (PCA). It is equivalent to other methods in various fields of study. For instance, in signal processing, it is known as the discrete Karhunen-Loève transform, see [HL98]. Linear algebra names it singular value decomposition, see [KL80]. In engineering applications, it is familiar as proper orthogonal decomposition, see [KGVB05]. Thus, this term is often used in academic works dealing with model order reduction, see [GHV20, BBV17, BSV14]. Therefore, it would be also suited here. However, the name principal component analysis and its workflow are chosen since its relative, kernel principal component analysis, is introduced as well.

PCA is an orthogonal linear transformation of the considered data that maps it into a new coordinate system, see [Jol02]. In this coordinate system, the variance portion of the data resides in the coordinates in descending order — from highest to lowest. By taking advantage of that, PCA can reduce the number of columns of a matrix,

$$A = (a_{ij})_{1\leq i\leq \eta, 1\leq j\leq \rho} \in \mathbb{R}^{\eta\times\rho}, \quad \eta, \rho \in \mathbb{N}. \tag{3.64}$$

For this task, practice exploits the sample covariance matrix with Bessel's correction,

$$C = \mathrm{Cov}\left(\tilde{A}\right) = \frac{1}{\eta - 1}\tilde{A}^T\tilde{A} \in \mathbb{R}^{\rho\times\rho}, \tag{3.65}$$

of the centered matrix, $\tilde{A}$, see [LMS20, Jol02]. Centering the matrix means that each row must have zero mean, i.e. it can be computed through

$$\tilde{A} = A - \mathbb{1}_\eta \left(\mu^A\right)^T. \tag{3.66}$$

In this formula, $\mathbb{1}_\eta \in \mathbb{R}^\eta$ is a vector of ones and $\mu^A \in \mathbb{R}^\rho$,

$$\mu_j^A = \frac{1}{\eta}\sum_{i=1}^{\eta} a_{ij}, \tag{3.67}$$

is the vector of the means along each column, $j \in \{1, \ldots, \rho\}$. Centering is beneficial for finding an appropriate solution, see [MLBB08].

Now, the eigenvectors, v^i, and eigenvalues, λ_i, of the covariance matrix, C, must be found, i.e. the problem

$$Cv^i = \lambda_i v^i, \quad i \in \{1, \ldots, \rho\}, \tag{3.68}$$

is considered. Many algorithms exist for numerical calculation of these quantities, see [Saa11]. After their computation, the so-called projection matrix, $V \in \mathbb{R}^{\rho\times N_{\mathrm{red}}}$ can be arranged. From left to the right, it consists of the N_{red} eigenvectors that correspond to the N_{red} highest eigenvalues — in descending order. These eigenvectors, i.e. columns of V, are also referred to as the principal components. The reduction of A can now be achieved by multiplication with the projection matrix, i.e. the reduced matrix is obtained by

$$A_{\mathrm{red}} = AV \in \mathbb{R}^{\eta\times N_{\mathrm{red}}}. \tag{3.69}$$

Vice versa, the reduced matrix can be projected back into the full space via multiplication with the transposed projection matrix, i.e.

$$A_{\text{red}}V^T = AVV^T \approx A \in \mathbb{R}^{\eta\times\rho}. \tag{3.70}$$

For the MOR metamodels, the so-called snapshot matrix, S, is set up, see [GH18]. It is the concatenation of all n_{train} training output matrices, $Y^i \in \mathcal{Y}_\gamma$, $i \in \{1,\dots,n_{\text{train}}\}$, i.e. it holds

$$S = \left(Y^1,\dots,Y^{n_{\text{train}}}\right) \in \mathbb{R}^{\overline{N}\times N_{\text{T}}\cdot n_{\text{train}}}. \tag{3.71}$$

Using the snapshot matrix, different characteristics of the output caused by different parameter manifestations come into play. To reduce the output dimension for $\mathcal{M}_\gamma$ (3.55), PCA is applied to the snapshot matrix. As the rows of $Y \in \mathcal{Y}_\gamma$ are supposed to be reduced, it is applied on the transposed snapshot matrix, $S^T \in \mathbb{R}^{N_{\text{T}}\cdot n_{\text{train}}\times\overline{N}}$, i.e. $\eta = N_{\text{T}}\cdot n_{\text{train}}$, $\rho = \overline{N}$. The sample covariance matrix of $\tilde{S}^T$ thus equals

$$C = \text{Cov}\left(\tilde{S}^T\right) = \frac{1}{N_{\text{T}}\cdot n_{\text{train}}-1}\left(\tilde{S}^T\right)^T\tilde{S}^T = \frac{1}{N_{\text{T}}\cdot n_{\text{train}}-1}\tilde{S}\tilde{S}^T \in \mathbb{R}^{\overline{N}\times\overline{N}}, \tag{3.72}$$

cf. (3.65). Its eigenvectors and eigenvalues must then be found to derive the projection matrix, $V \in \mathbb{R}^{\overline{N}\times N_{\text{red}}}$.

Recalling the transposition, the reduction is carried out via

$$Y_{\text{red}} = \left(Y^TV\right)^T = V^TY \in \mathcal{Y}_{\text{red}} \subset \mathbb{R}^{N_{\text{red}}\times N_{\text{T}}} \tag{3.73}$$

for arbitrary matrices $Y \in \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N}\times N_{\text{T}}}$. Consequently, using the projection matrix, the projection mapping (3.57) for the MOR metamodels can be designed as

$$\text{Proj} : \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N}\times N_{\text{T}}} \to \mathcal{Y}_{\text{red}} \subset \mathbb{R}^{N_{\text{red}}\times N_{\text{T}}}, Y \mapsto Y_{\text{red}} = V^TY. \tag{3.74}$$

The inverse operation, Back (3.58), can then be formulated as

$$\text{Back} : \mathcal{Y}_{\text{red}} \subset \mathbb{R}^{N_{\text{red}}\times N_{\text{T}}} \to \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N}\times N_{\text{T}}}, Y_{\text{red}} \mapsto VY_{\text{red}} = VV^TY. \tag{3.75}$$

The only hyperparameter that has to be tuned for principal component analysis is N_{red}, i.e. the number of principal components. This variable is not calculated via optimization. Instead the formula,

$$\sum_{i=1}^{N_{\text{red}}}\lambda_i^{\text{h}} \Bigg/ \sum_{j=1}^{\overline{N}}\lambda_j \geq \varepsilon_\lambda \tag{3.76}$$

is considered, see [JLG21a], where $\lambda_1^{\text{h}},\dots,\lambda_{N_{\text{red}}}^{\text{h}}$ are the N_{red} highest eigenvalues of C. The threshold $\varepsilon_\lambda \in (0,1]$ ensures that at least $\varepsilon_\lambda\cdot 100\%$ of the variance can be explained by the N_{red} associated eigenvectors. The value for N_{red} is chosen as the lowest natural number that fulfills (3.76) for a fixed ε_λ. The threshold, ε_λ, is selected dependent on the problem — generally, it is set to at least 0.9. PCA therefore enables using N_{red} MTR metamodels to emulate $\mathcal{F}_{\text{EN}}$ (3.11) or $\mathcal{F}_{\text{CN}}$ (3.13), see Fig. 9, while both methods are established independently, i.e. without applying the nested optimization problem.

Kernel principal component analysis
A nonlinear extension of PCA is kernel principal component analysis (kPCA), see [SSM97]. Via a kernel function, it reforms principal component analysis in a high-dimensional space, [LMS20]. A kernel function, $\mathcal{K}$, processes two vectors of same dimension $\rho \in \mathbb{N}$, e.g. the i_1-th column, $a^1_{i_1} \in \mathbb{R}^\rho$, of

$$A^1 = \left(a^1_{i_1}\right)_{1\leq i_1\leq \eta_1} = \left(a^1_{i_1,j}\right)_{1\leq i_1\leq \eta_1, 1\leq j\leq \rho} \in \mathbb{R}^{\eta_1\times\rho}, \quad \eta_1 \in \mathbb{N}, \tag{3.77}$$

and the i_2-th column, $a^2_{i_2} \in \mathbb{R}^\rho$, of

$$A^2 = \left(a^2_{i_2}\right)_{1\leq i_2\leq \eta_2} = \left(a^2_{i_2,j}\right)_{1\leq i_2\leq \eta_2, 1\leq j\leq \rho} \in \mathbb{R}^{\eta_2\times\rho}, \quad \eta_2 \in \mathbb{N}. \tag{3.78}$$

The operation can be defined by $\mathcal{K} : \mathbb{R}^\rho \times \mathbb{R}^\rho \to \mathbb{R}$ with

$$\mathcal{K}\left(a^1_{i_1}, a^2_{i_2}\right) = \langle \Phi\left(a^1_{i_1}\right), \Phi\left(a^2_{i_2}\right)\rangle_{\mathcal{H}}, \quad i_1 \in \{1,\dots\eta_1\}, i_2 \in \{1,\dots\eta_2\}, \tag{3.79}$$

where $\Phi : \mathbb{R}^\rho \to \mathcal{H}$ is a mapping into an intractable, high-dimensional space $\mathcal{H}$.

Practically, Φ is not evaluated explicitly. Instead, kernel functions are defined without the use of Φ that avoids working in $\mathcal{H}$. This relief is also referred to as the kernel trick, see [NDLT08]. Moreover, the so-called kernel matrix of A^1 and A^2, denoted as

$$K = \mathcal{K}\left(A^1, A^2\right) = \left(k_{i_1,i_2}\right)_{1\leq i_1\leq \eta_1, 1\leq i_2\leq \eta_2} \in \mathbb{R}^{\eta_1\times\eta_2}, \tag{3.80}$$

is composed of

$$k_{i_1,i_2} = \mathcal{K}\left(a^1_{i_1}, a^2_{i_2}\right) = \langle \Phi\left(a^1_{i_1}\right), \Phi\left(a^2_{i_2}\right)\rangle_{\mathcal{H}} \in \mathbb{R}, \tag{3.81}$$

see [HSS08].

By using the kernel trick and matrix, kPCA reduces the number of columns of

$$A = (a_i)_{1\leq i\leq \eta} = (a_{ij})_{1\leq i\leq \eta, 1\leq j\leq \rho} \in \mathbb{R}^{\eta\times\rho}, \quad \eta, \rho \in \mathbb{N}, \tag{3.82}$$

with $a_i \in \mathbb{R}^\rho, i \in \{1,\dots,\eta\}$. The kernel matrix, $K = \mathcal{K}(A, A)$, is thereto passed through the standard principal component analysis procedure. This means it is centered, its covariance matrix,

$$\mathrm{Cov}\left(\tilde{K}\right) \in \mathbb{R}^{\eta\times\eta}, \tag{3.83}$$

is established, and consequently eigenvalues and eigenvectors are determined. In this way, the projection matrix, $V \in \mathbb{R}^{\eta\times N_{\mathrm{red}}}$, can be derived. This matrix can be multiplied to kernel transformed data, e.g. to $K \in \mathbb{R}^{\eta\times\eta}$ yielding $K_{\mathrm{red}} = KV \in \mathbb{R}^{\eta\times N_{\mathrm{red}}}$. Vice versa, reduced kernel transformed matrices, e.g. K_{red}, are projected up by multiplying V^T, e.g. $K_{\mathrm{red}}V^T = KVV^T \approx K$. This construct must then be treated with the inverse, $\mathcal{K}^{-1}$, of the kernel function to obtain data in the correct space, i.e.

$$\mathcal{K}^{-1}\left(K_{\mathrm{red}}V^T\right) = \mathcal{K}^{-1}\left(KVV^T\right) \approx A \in \mathbb{R}^{\eta\times\rho}. \tag{3.84}$$

Numerically, the inverse function, $\mathcal{K}^{-1}$, may be intractable and is thus often approximated through regression techniques.

For the MOR metamodels, the snapshot matrix, $S \in \mathbb{R}^{\overline{N} \times N_{\mathrm{T}} \cdot n_{\mathrm{train}}}$ (3.71), is utilized again. Analogous to PCA, kPCA is applied to its transpose, S^T. The projection matrix, $V \in \mathbb{R}^{N_{\mathrm{T}} \cdot n_{\mathrm{train}} \times N_{\mathrm{red}}}$, is the concatenation of the eigenvectors corresponding to the N_{red} highest eigenvalues of the covariance matrix of the kernel matrix $K = \mathcal{K}\left(S^T, S^T\right)$. It is used to project between the full and reduced kernel transformed spaces. Summarizing, the projection mapping, Proj (3.57), can be written as

$$\mathrm{Proj} : \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N} \times N_{\mathrm{T}}} \to \mathcal{Y}_{\mathrm{red}} \subset \mathbb{R}^{N_{\mathrm{red}} \times N_{\mathrm{T}}}, Y \mapsto K_{\mathrm{red}} = V^T \mathcal{K}\left(S^T, Y^T\right), \tag{3.85}$$

cf. (3.74). Note that the transposition is taken into account. Furthermore, it holds $\mathcal{K}\left(S^T, Y^T\right) \in \mathbb{R}^{N_T \cdot n_{\mathrm{train}} \times N_T}$. The inverse operation, Back (3.58), can then be expressed as

$$\mathrm{Back} : \mathcal{Y}_{\mathrm{red}} \subset \mathbb{R}^{N_{\mathrm{red}} \times N_{\mathrm{T}}} \to \mathcal{Y}_\gamma \subset \mathbb{R}^{\overline{N} \times N_{\mathrm{T}}}, K_{\mathrm{red}} \mapsto \mathcal{K}^{-1}\left(V K_{\mathrm{red}}\right), \tag{3.86}$$

cf. (3.75). Therefore, kPCA allows $N_{\mathrm{red}} < \overline{N}$ MTR metamodels to approximate $\mathcal{F}_{\mathrm{EN}}$ (3.11) or $\mathcal{F}_{\mathrm{CN}}$ (3.13), see Fig. 9.

For kPCA, formula (3.76) cannot be used as the computed eigenvalues are transformed and do not belong to the original problem. Therefore, different numbers of principal components should be tested and compared. Moreover, the kernel function, $\mathcal{K} = \mathcal{K}_\theta$, consists of hyperparameters, θ, specific to the function chosen. They must be tuned through one of the two mentioned optimization procedures — the independent or the nested approach. Popular functions to use for $\mathcal{K}$ are the polynomial kernel,

$$\mathcal{K}_\theta\left(y^1, y^2\right) = \left(\theta_1 \left(y^1\right)^T y^2 + \theta_2\right)^{\theta_3}, \quad \theta_1 > 0, \theta_2 \geq 0, \theta_3 \in \mathbb{N}, \tag{3.87}$$

and the Gaussian kernel,

$$\mathcal{K}_\theta\left(y^1, y^2\right) = \exp\left(-\frac{1}{2} \sum_{i=1}^{\overline{N}} \frac{1}{\theta_i^2} \left(y_i^1 - y_i^2\right)^2\right), \quad \theta_i > 0, i \in \left\{1, \ldots, \overline{N}\right\}, \tag{3.88}$$

for $y^1, y^2 \in \mathbb{R}^{\overline{N}}$, see [LMS20]. Other choices for $\mathcal{K}$ can be found in [Sou10]. The polynomial kernel with the hyperparameter setting $\theta_1 = \theta_3 = 1$, $\theta_2 = 0$ is identical to standard PCA. The Gaussian kernel is called isotropic if all hyperparameters, $\theta_1, \ldots, \theta_{\overline{N}}$, are equal. Otherwise, it is called anisotropic. All kernels consist of different hyperparameters leading to diverse, complex optimization problems. Anisotropic Gaussian kernels might be difficult to tune as many, i.e. $\overline{N}$, optimal values have to be found making the problem high dimensional.

As seen above, kPCA as an generalization can be adjusted to be equal to PCA. In case standard PCA is the optimal choice, the kernel version may converge to it. Hence, kPCA may be preferred over the standard option. However, solving the optimization problem to gain appropriate kernel hyperparameters might be difficult. In addition, the conclusion of [vdMPvdH09] states that nonlinear dimensionality reduction techniques, e.g. the kernel version, often may not be capable of surpassing linear methods, e.g. standard PCA.

3.2.6 Dimensionality reduction of the input space

The curse of dimensionality, see [Don00, Tru79, VF05], regarding metamodels means that high-dimensional input spaces may cause problems. This happens for various methods. For instance, choosing the anisotropic Gaussian kernel function (3.88) for a scalar universal kriging metamodel, $\mathcal{M}_{\mathrm{KR}}^{\mathrm{GPR}}$ (3.41), requires tuning the weight vector, $\beta \in \mathbb{R}^P$, and the hyperparameter, $\theta \in \mathbb{R}^n$ — here, θ is of same dimension as the input, $x \in \mathbb{R}^n$. When n and P both are large and not sufficient training data is available, the approximation may get inappropriate due to facing an underdetermined system of equations. The same may occur for scalar polynomial chaos expansion metamodels, $\mathcal{M}_{\mathrm{KR}}^{\mathrm{PCE}}$ (3.43). The number of polynomial basis functions, $\psi_\alpha, \alpha \in \mathcal{A} \subset \mathbb{N}^n$, rapidly increase when n or the maximum degree, $P \in \mathbb{N}$, of the polynomials are raised. For large n, it is therefore hard to tweak their coefficients, a_α, possibly leading to bad approximated response surfaces.

A workaround is presented in [LMS20]. As mentioned in Section 3.2.5, this work reduces the dimension of the input space before a scalar metamodel is applied. This is done via dimensionality reduction techniques. The reductions and the scalar metamodels are combined via nested optimization problems also suggested for MOR metamodels, see Section 3.2.5. This approach is now presented in more detail. Another description can be found in a Master thesis, see [Her21], tutored by the author of this work.

The inclusion of dimensionality reduction leads to additional hyperparameters that need to be trimmed. To distinguish, the hyperparameters, e.g. the coefficients of polynomial chaos expansion, for metamodels are denoted as θ_{M}. The symbol, θ_{D}, signifies the hyperparameters for the dimensionality reduction. For optimizing the total metamodel, a cost function, C, e.g. the least-squares formula, the π-fold cross validation, or the leave-one-out error, must be chosen. It evaluates how good the predictions fit to the actual training data, $\tau_{\mathrm{train}} = \{(x^1, y^1), \ldots, (x^{n_{\mathrm{train}}}, y^{n_{\mathrm{train}}})\}$. Hence, it depends on the training data as well as on the dimensionality reduction and the metamodel along with their hyperparameters.

The procedure for combining the methods shown in Algorithm 1 consists of two loops. In the outer loop, the dimensionality reduction — dependent on the current settings, $\check{\theta}_{\mathrm{M}}$, of the metamodel and on its performance — is revised, i.e. θ_{D} is updated through

$$\check{\theta}_{\mathrm{D}} = \arg\min_{\theta_{\mathrm{D}}} C\left(\mathrm{Proj}\left(\cdot;\theta_{\mathrm{D}}\right), \mathcal{M}_{\mathrm{KR}}\left(\cdot;\check{\theta}_{\mathrm{M}}\right), \tau_{\mathrm{train}}\right). \tag{3.89}$$

The dimensionality reduction is applied to the input space. This means it compresses the number of parameters from n to $n_{\mathrm{red}} \in \mathbb{N}, n_{\mathrm{red}} < n$. The reduced parameters must not be projected back. As a consequence, only projection of the data to a lower dimensional subspace, cf. (3.57), without applying its inverse operation, cf. (3.58), is required. The projection mapping, Proj, is defined as

$$\mathrm{Proj} : \mathcal{X} \subset \mathbb{R}^n \to \mathcal{X}_{\mathrm{red}} \subset \mathbb{R}^{n_{\mathrm{red}}}, x \mapsto x_{\mathrm{red}}, \tag{3.90}$$

cf. (3.57). Proj converts the training input, $x^1, \ldots, x^{\mathrm{train}}$, to $x_{\mathrm{red}}^1, \ldots, x_{\mathrm{red}}^{\mathrm{train}}$ which facilitates the creation of the metamodel. The inner loop then optimizes the metamodel operating with the reduced number of parameters — found by the dimensionality reduc-

tion with argument $\check{\theta}_{\mathrm{D}}$. Its hyperparameters, θ_{M}, are adjusted by solving

$$\check{\theta}_{\mathrm{M}} = \arg\min_{\theta_{\mathrm{M}}} C\left(\mathrm{Proj}\left(\cdot;\check{\theta}_{\mathrm{D}}\right), \mathcal{M}_{\mathrm{KR}}\left(\cdot;\theta_{\mathrm{M}}\right), \tau_{\mathrm{train}}\right). \tag{3.91}$$

Algorithm 1: Nested optimization for combining dimensionality reduction with metamodels.

Input: Projection mapping, Proj; metamodel, $\mathcal{M}_{\mathrm{KR}}$; training data, τ_{train}; initial guess, $\check{\theta}_{\mathrm{D}}$; cost function, C; stopping value, $\varepsilon_{\mathrm{NO}}$

Output: Optimized hyperparameters, θ_{D} and θ_{M}

```
set flag = true
while flag do
    determine θ̌_M using (3.91) with θ̌_D
    update θ̌_D using (3.89) with θ̌_M
    if C(Proj(·; θ̌_D), M_KR(·; θ̌_M), τ_train) < ε_NO then
        set flag = false
    else
        continue
    set θ_D = θ̌_D, θ_M = θ̌_M
```

The work [LMS20] proposes to use standard or kernel PCA introduced above as the mapping Proj. For them, the training input matrix

$$X = \begin{pmatrix} x^1 \\ \vdots \\ x^{\mathrm{train}} \end{pmatrix} = \begin{pmatrix} x_1^1 & \cdots & x_n^1 \\ \vdots & \ddots & \vdots \\ x_1^{\mathrm{train}} & \cdots & x_n^{\mathrm{train}} \end{pmatrix} \in \mathbb{R}^{n_{\mathrm{train}} \times n} \tag{3.92}$$

is arranged and send through the strategies. The aim is to prune the columns, n, of X implying a reduced input dimension. Dimensionality reduction techniques may improve metamodels not merely via parameter reductions but also through resolving the given — possibly complex — input-output structures into simpler entities.

The outlined approach can be adapted for MTR metamodels from Section 3.2.4. The nested optimization can be done independently for each component of the ST method and the RC approach. Neural networks, however, do not suffer the curse of dimensionality due to its integral compressing architecture. Therefore, they do not rely on extra dimension reductions. In fact, they can be used for dimensionality reduction itself, e.g. autoencoders can be utilized, see [Sch15]. Similarly, MOR metamodels can be improved by this strategy. For independent optimization of the combined methods — dimensionality reduction and metamodels — parameter reduction proceeds in the same way as for the MTR metamodels. By contrast, a third loop is required for the nested optimization. The outer loop reduces the dimension of the output space, the middle loop the dimension of the input space, and the inner loop establishes a revised metamodel. It is also possible to swap tasks of the outer and the middle loop.

3.2.7 Exemplary applications of metamodels

This section shows that metamodels are suitable approximations for finite element model outputs. They accelerate the output generation which makes them central to the framework of this work. To show their workable quality, four applications are approximated by metamodels. For this purpose, methods from the Sections 3.2.3-3.2.6 are tested, i.e. scalar metamodels, multi-target regression metamodels, model order reduction metamodels, and scalar metamodels coupled with dimensionality reduction applied to the input space. To recall, the ingredients of these metamodels may be scalar metamodels — multiple linear regression (MLR), Gaussian process regression (GPR), Polynomial chaos expansion (PCE) — scalar or multi-target neural networks (SNN, MNN), and dimensionality reduction techniques — principal component analysis (PCA) and its nonlinear version using kernel functions (kPCA).

The numerical realization is done in Python [VRD09]. The routines for MLR, GPR, PCA, and kPCA are taken from the library Scikit-learn [PVG+11]. The library Chaospy [FL15] provides PCE. Neural networks, i.e. SNN and MNN, are implemented using the packages TensorFlow [AAB+15] and Keras [Cho15]. The GPR models throughout employ anisotropic Matérn 3/2 kernels as auto-covariance functions. As the applications treated with PCE assume uniformly distributed parameters, Legendre polynomial basis functions are utilized. Their maximal degrees, $P \in \mathbb{N}$, are varied. The suffix of the PCE title indicates the maximum degree, e.g. PCE-3 is the PCE model with monomials up to degree 3. The common truncation set,

$$\mathcal{A} = \left\{ \alpha \in \mathbb{N}^n \,\middle|\, \sum_{i=1}^{n} \alpha_i \leq P \right\}, \tag{3.93}$$

for the expansion is chosen, see [BS10, BS11]. This leads to an expansion of length

$$\#\mathcal{A} = \frac{(n+P)!}{n!P!}, \tag{3.94}$$

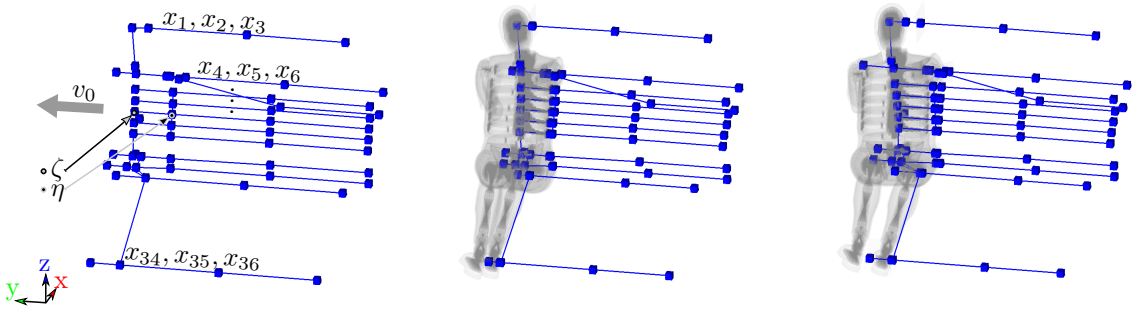
see [XK02, BS08].

Number of hidden layers, neurons, and activation functions of neural networks are specified in the text separately for each application. The number of iterations, i.e. epochs, to train the networks are included in their respective names, e.g. MNN-200 denotes a multi-target network trained for 200 epochs. Moreover, the reduced dimension, n_{red} or N_{red}, is written after the corresponding dimensionality reduction method. For a model order reduction metamodel that e.g. combines a single-target method using GPR (ST-GPR) with kPCA to reduce the dimensions of the problem to $N_{\text{red}} = 10$, the abbreviated title would read kPCA-10-ST-GPR. Analogously, this applies to metamodels coupled with dimensionality reduction for the input space.

Scalar metamodels for the simplified simulation of the WorldSID 50th percentile male dummy in a side crash scenario

In [JLG21b], the author of this thesis and his colleagues present an uncertainty management framework that is strongly related to the one in Fig. 4. There it is applied

to a simplified finite element side crash simulation of a WorldSID 50th percentile male dummy. The finite element simulation is solved with the commercial software Abaqus FEA [FEA16]. The applied simplification realized in [Bra17, SB19] lead to a condensed finite element construct of connected pushing masses, see Fig. 10. The simplified simulation has a computational time of 5 seconds. The key result considered in [JLG21b] is the maximum deflection at the middle rib of the dummy, thus a scalar. In total $n = 36$ parameters are defined that are all uniformly distributed — for the metamodeling task — and normalized to $[0, 1]$ with MinMaxScaler (3.27). The parameters describe the course of the plastic force-displacement curves for the mass point connectors, see [JLG21b] for more information.



(a) Initial configuration (b) Intermediate time state (c) Final time state

Figure 10: Simplified finite element side crash simulation of a WorldSID 50th percentile male dummy. The figure is taken from [JLG21b].

Using a combination of screening and a scalar metamodel, namely the elementary effects (EE) method, see Section 3.3.2, associated with GPR (EE-GPR), quantitative sensitivity analysis and uncertainty quantification were enabled. Screening required 148 simulations runs, i.e. $r = 4$ random starting points, to identify $k = 6$ relevant parameters. The kriging metamodel — using the default settings of MATLAB's [MAT18] GPR routine and operating on $k = 6$ parameters — was trained with $n_{\text{train}} = 102$ points produced by Sobol' sequence sampling.

Here, the performance of this screening-based metamodel is compared with other scalar metamodels, $\mathcal{M}_{\text{KR}}$ (3.36): multiple linear regression, kriging, and polynomial chaos expansion as standalone metamodels but also reinforced by dimensionality reduction. An anisotropic Gaussian kernel, see (3.88), is utilized for kPCA. This reduction technique is meant to eradicate the curse of dimensionality from the input space. The kPCA-based scalar metamodels are optimized via the nested approach from Section 3.2.6.

For a fair comparison, the new metamodels are trained with $n_{\text{train}} = 148 + 102 = 250$ Sobol' sequence sampling points as they do not use the information of screening, thus operate on $n = 36$ parameters. The quality of all metamodels are validated using an unseen Sobol' sequence sample of size $n_{\text{val}} = 350$, cf. [JLG21b]. The histogram of the $n_{\text{train}} + n_{\text{val}} = 600$ output points, i.e. the key results, is shown in Fig. 11.

The optimization of the dimensionality reduction-based metamodels is conducted using the particle swarm routine from the Python library Scikit-opt [Guo21]. As cost function, the empirical relative generalization error is used, see [LMS20]. The error is calcu-

lated using all 250 training points. Only 80% of this training set, i.e. the first 200 sample points, however, are fed into the metamodel to tune its hyperparameters. Using the other 20% as validation sample within the training process avoids overfitting. In [LMS20], an alternative is proposed to tackle overfitting via calculating the leave-one-out error. In general, n_{train} metamodels have to be determined for this measure. Note that for some metamodels, e.g. GPR and PCE, there exist analytical formulas for the leave-one-out error which enable its usage, see [Dub83, BS11].

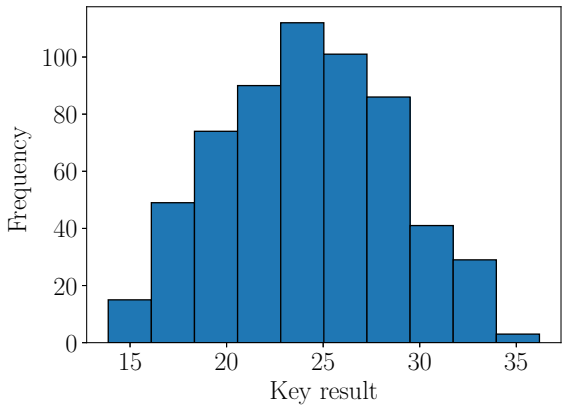


Figure 11: Histogram of the $n_{\text{train}} + n_{\text{val}} = 600$ key results from the simplified dummy simulation. It is constituted of ten equal-width bins which together cover the range from 13.87 to 36.20 mm.

To keep it generic, i.e. for metamodels for which this kind of formula is not provided, the already expensive nested optimization would become more costly using the leave-one-out error. So, the approach here takes the proposed 80% of the training set and thus prevents additional efforts. Moreover, the number of principal components is fixed a-priori. A loop determines the results for a range of 5 to 20 principal components. The best result is then taken as final configuration for the specific combination.

Table 2 shows a comparison of the results produced by different metamodels — MLR, GPR, PCE, and their combinations with kPCA, EE-GPR, and SNN. The neural network, SNN-500, has two hidden layers each equipped with 128 neurons and φ_{id} activation functions. All in all, the quality of EE-GPR from [JLG21b] is slightly the best. Besides PCE-2, the others are close to the performance of EE-GPR. Except for GPR, connecting the metamodel with kPCA improves the results. For GPR, it virtually stays the same.

For PCE-2, the curse of dimensionality noticeably decreases the approximation quality. Formula (3.94) shows that

$$\frac{(36+2)!}{36!2!} = 703 \tag{3.95}$$

hyperparameters, i.e. PCE coefficients, need to be optimized for PCE-2. As only $n_{\text{train}} = 250$ training points are available, the system that must be solved is underdetermined. The voluminous PCE-2 therefore lacks sufficient training data due to this curse of dimensionality. This leads to the inappropriate AvP plot shown in Fig. 12 (a). This changes when

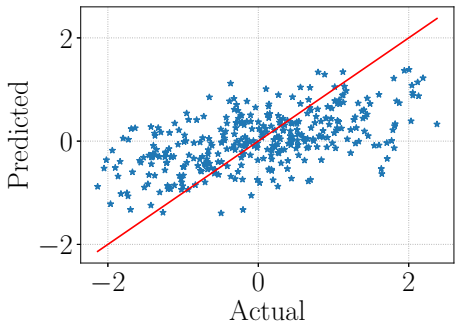
bringing kPCA into play. The best performing dimension found for PCE-2 with kPCA is $n_{\text{red}} = 17$, i.e. kPCA-17-PCE-2 is the final configuration after nested optimization of the combination comprising kPCA and PCE-2. The input space of size 17 can be treated without facing the curse of dimensionality as the number of hyperparameters shrinks to

$$\frac{(17+2)!}{17!2!} = 171. \tag{3.96}$$

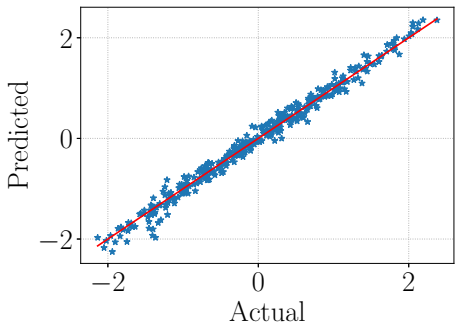
This produces a proper fit with high R^2 score, cf. Table 2, and a proper AvP plot depicted in Fig. 12 (b).

Table 2: Results of the different scalar metamodels for the simplified side crash simulation of the WorldSID 50th percentile male dummy determined with the validation sample. The best result is marked in bold. For comparison reasons, the standalone methods are listed on top of their combinations with dimensionality reduction or screening, respectively.

Approach	R^2
MLR	0.97559
kPCA-16-MLR	0.99041
GPR	0.98953
kPCA-13-GPR	0.98921
EE-GPR [JLG21b]	**0.99369**
PCE-1	0.97559
kPCA-15-PCE-1	0.98973
PCE-2	0.32406
kPCA-17-PCE-2	0.98318
SNN-500	0.97501



(a) PCE-2



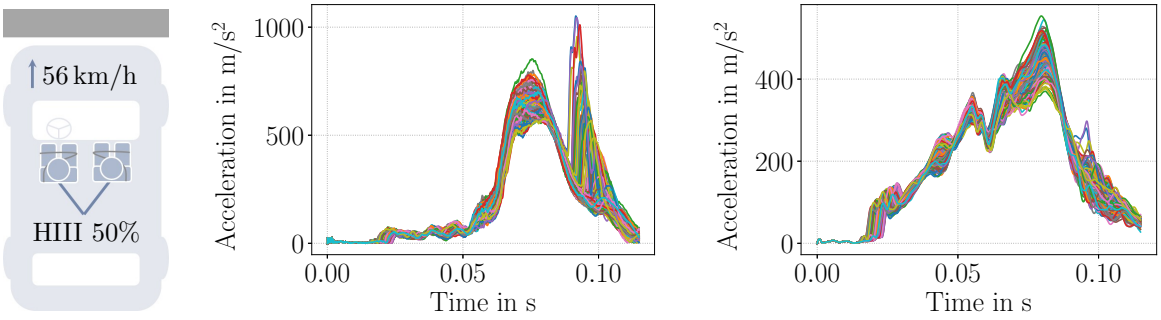
(b) kPCA-17-PCE-2

Figure 12: AvP plots of metamodels using PCE-2 for the simplified dummy model.

Moreover, it is worth mentioning that the results of PCE-1 and MLR — or kPCA-16-MLR and kPCA-15-PCE-1 — are similar. This is due to the fact that the considered PCE works with the same basis functions as MLR since it uses Legendre polynomials up to degree $P = 1$ that have the form $\psi(x) = x$ and $\psi(x) = 1$, cf. (3.43) and see [Jac12]. These monomials are also part of the MLR formula (3.38). Slightly deviating results may come from the different hyperparameter optimization techniques that are integrated in the corresponding Python libraries. Last but not least, these metamodels — except for PCE-2 — allow for efficient and accurate investigations afterwards, e.g. sensitivity and uncertainty analysis as done in [JLG21a].

Occupant simulation of a full frontal test

The Master thesis [LN21] uses multi-target regression metamodels, $\mathcal{M}_{\text{SH}}$ (3.46), to approximate specific processed time histories of a finite element simulation modeling a full frontal test. Target values for this test, measured e.g. with the aid of dummies, are stipulated by legal authorities and must be achieved in order to obtain automotive homologation. According to safety standards of the U.S. market, see [U.S08], one manifestation of the full frontal test places Hybrid III 50th percentile male (HIII 50%) dummies on the front seats. The car under consideration is driven head-on against a rigid barrier at a velocity equal to 56 km/h, see Fig. 13 (a).



(a) (b) Head acceleration (c) Chest acceleration

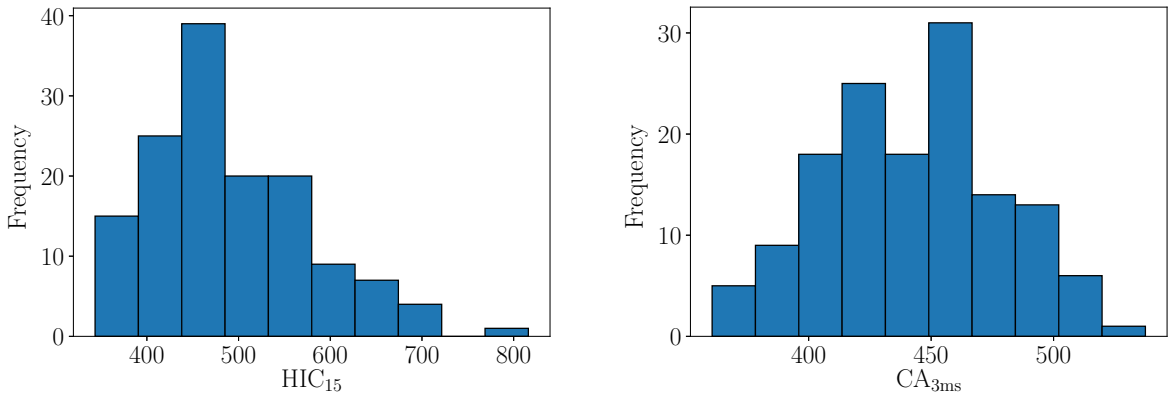
Figure 13: A sketch of the full frontal test is shown in (a). The simulation considers the passenger dummy in a separate occupant model. All 140 curves for the head (b) and chest acceleration (c) are displayed.

The simulation corresponding to this test is simplified: as mentioned in Chapter 2, it is oftentimes engineering practice to simulate the vehicle structure a-priori and then observe the dummy behaviors in extra occupant models subjected to the structure values obtained. The occupant model studied here investigates the passenger dummy seated in the interior of a state-of-the-art car. The driver dummy is not considered here and hence excluded from the simulations. The model is simulated in LS-DYNA. One simulation lasts over 45 hours on 36 CPUs.

Four parameters characterizing the restraint system are regarded as uncertain. The parameter ranges for this study are intentionally extended to yield more variability than in real projects. So, the outputs shown in Fig. 13 and 14 do not match actual development quantities. All parameters are uniformly distributed and normalized to $[0, 1]$ with MinMaxScaler (3.27). The first parameter, x_1, calibrates the passenger airbag to be soft, $x_1 = 0$, or hard, $x_1 = 1$. The second parameter, x_2, repeats this for the knee airbag. The third parameter, x_3, controls the time-to-fire of the restraint systems. For low values, e.g. $x_3 = 0$, the systems are triggered early. For high values, e.g. $x_3 = 1$, the systems react late. The fourth parameter, x_4, regulates the belt settings — from soft, $x_4 = 0$, to hard, $x_4 = 1$.

The occupant model is simulated $n_{\text{train}} + n_{\text{val}} = 100 + 40 = 140$ times using different parameter configurations created by Sobol' sequence sampling. The work of the Master thesis [LN21] is continued in terms of considering prescribed target values. Therefore, two

key results are exemplarily addressed: the head injury criterion (HIC_{15}) (3.18) extracted from the resultant head acceleration time history of the dummy as well as the so-called cumulative 3 ms value ($\mathrm{CA}_{3\mathrm{ms}}$) of the chest acceleration curve. The expensive finite element model is replaced by metamodels to quickly approximate the key results for new parameter values. For this task, two types of metamodels are tested. On the one hand, the finite element model is considered to have the form of history functions, $\mathcal{F}_{\mathrm{SH}}$ (3.16), having $N_{\mathrm{T}} = 1150$ time steps. The key results can then be extracted from these functions — using the respective extraction function, e (3.17). On the other hand, the simulation is directly regarded as the corresponding key result function, $\mathcal{F}_{\mathrm{KR}}$ (3.19).



(a) Histogram for HIC_{15} covers the range from 343.50 to 816.02

(b) Histogram for $\mathrm{CA}_{3\mathrm{ms}}$ spans from 360.58 to 537.48

Figure 14: Histograms of the considered key results from the occupant simulation. Both are constituted of ten equal-width bins.

The histograms of the key results, i.e. HIC_{15} and $\mathrm{CA}_{3\mathrm{ms}}$, created with the 140 respective exact values can be found in Fig. 14. GPR, PCE, and SNN are employed as scalar metamodels, $\mathcal{M}_{\mathrm{KR}}$ (3.36), to approximate these key results. Next to it, the resultant head acceleration curves also studied in [LN21] as well as the resultant chest acceleration time history are emulated by single-target methods (ST-GPR, ST-PCE) and regressor chain-based approaches (RC-GPR, RC-PCE) using GPR and PCE. In addition, multi-target neural networks (MNN) are tested. These metamodels have the form $\mathcal{M}_{\mathrm{SH}}$ (3.46). All 140 exact head and chest acceleration curves that the multi-target regression metamodels have to deal with, respectively, are depicted in Fig. 13 (b) and (c). The key results are then processed by applying the HIC_{15} formula (3.18) to the head acceleration curves or finding the 3 ms value of the chest acceleration time profiles, i.e. $\mathrm{CA}_{3\mathrm{ms}}$. The 3 ms value of a measurement signal is defined as the largest amplitude that is present for 3 milliseconds — either continuous or cumulated — see [COZ+]. Here, the cumulated version is used.

The performance of the different metamodels for the time histories and the extracted key results are compared in Table 3. Four different maximal degrees, $P \in \{1, 2, 3, 4\}$, for PCE models are compared. The SNN models are constituted of two hidden layers with 256 neurons each. They use φ_{relu} as activation functions and are trained for different epochs defining their names. Similarly, the MNN models are equal concerning the choice

of the architecture: three hidden layers with 256, 512, 1024 neurons from left to the right activated by φ_{tanh}. The number of iterations, i.e. epochs, are different and can be read from the title endings.

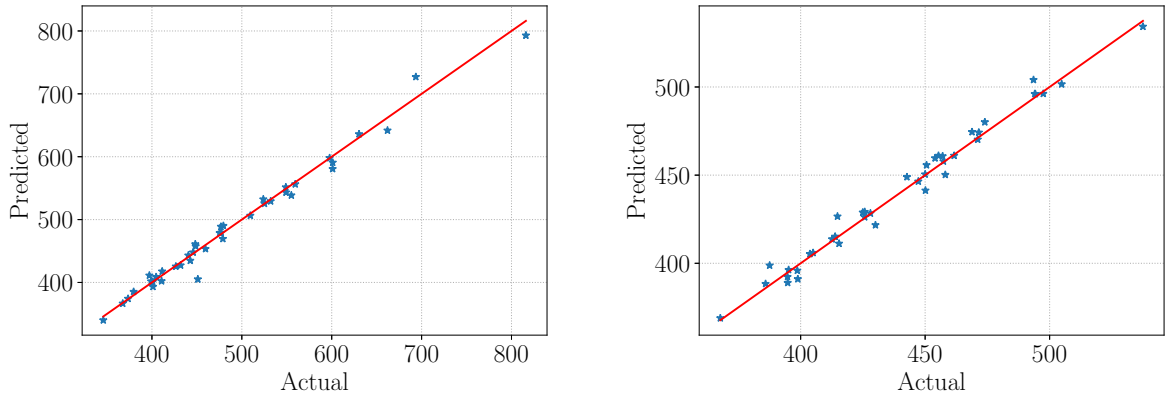
Table 3: Results of different metamodels approximating head as well as chest acceleration time histories and HIC_{15} as well as $\text{CA}_{3\text{ms}}$ key results of the occupant simulation. The mean squared error, MSE_{SH}, shows the averaged MSE of the time histories. The coefficient of determination is calculated for the key results. The best result for each quality measure is marked in bold.

Approach	Head-MSE_{SH}	HIC_{15}-R^2_{KR}	Chest-MSE_{SH}	$\text{CA}_{3\text{ms}}$-R^2_{KR}
GPR	-	0.97340	-	0.97230
PCE-1	-	0.88809	-	**0.98273**
PCE-2	-	0.97562	-	0.97973
PCE-3	-	0.97892	-	0.97195
PCE-4	-	0.92638	-	0.94895
SNN-100	-	0.96811	-	0.97765
SNN-200	-	0.97613	-	0.97493
SNN-300	-	0.97625	-	0.97581
ST-GPR	602.73217	0.96126	86.29035	0.96690
ST-PCE-1	1154.09729	0.94194	92.54107	0.98209
ST-PCE-2	775.56576	0.97853	71.51075	0.98002
ST-PCE-3	787.41539	**0.98367**	90.10494	0.96550
ST-PCE-4	2622.87375	0.95975	326.93971	0.91369
RC-GPR	796.44560	0.93022	160.45546	0.91144
RC-PCE-1	1097.65395	0.91730	95.4902	0.97005
RC-PCE-2	3541.83062	0.95877	236.11060	0.82453
RC-PCE-3	2380.49518	0.92509	330.95230	0.85588
RC-PCE-4	4423.53116	0.67315	845.85581	0.34099
MNN-100	1124.35491	0.93616	91.76897	0.98188
MNN-200	718.02920	0.97468	71.37379	0.97975
MNN-300	**479.14381**	0.95840	**64.75791**	0.97273

By definition, for the regressor chain-based approaches, the output of the previous time step is used as an additional parameter for predicting the output of the current time step. To be consistent with the other parameters, the values of the output are normalized to be between $[0, 1]$. Predicted values that are outside the range $[0, 1]$ are projected to the nearest limit value, i.e. 0 or 1. Otherwise, the use of these values as additional parameters would lead to poor approximation results as they would lie outside the defined parameter interval forcing to solve an extrapolation task.

At first, the key result for the head, HIC_{15}, is studied. According to the R^2_{KR} values in Table 3, the best metamodel for HIC_{15} is ST-PCE-3 — ST-PCE using $P = 3$. Besides its high R^2_{KR}, the actual versus predicted (AvP) plot in Fig. 15 (a) reaffirms the satisfying quality. While ST-PCE-3 provides the best approximated time histories out of the considered approaches to extract the key result from, the time histories created by MNN-300 are

better approximates for the total curve correspondent to the error measures from Table 3. All $n_{\text{val}} = 40$ exact curves are plotted in Fig. 39 together with the approximated curves of ST-PCE-3 and MNN-300.



(a) HIC_{15} (b) $\text{CA}_{3\text{ms}}$

Figure 15: AvP of ST-PCE-3 for HIC_{15} (a) and AvP of PCE-1 for $\text{CA}_{3\text{ms}}$ (b) using the validation sample of size $n_{\text{val}} = 40$.

Due to its definition, HIC_{15} is extracted from the first wider peak of these curves, see also Fig. 13 (b). This first peak is predicted better by ST-PCE-3 than by MNN-300. The second peak, however, can be reproduced more precisely by MNN-300 leading to an overall better approximation result. When MNN is trained for less epochs, e.g. 200, see MNN-200 in Table 3, the predictions of the key result may be better but the overall fit is less strong. This may be due to an overfitting around the first peak when training for too long, e.g. for 300 epochs. Training for 200 epochs does not yield overfitting in the first peak. For the second peak, however, the network MNN-200 is valid but it is not developed as fully as for 300 epochs, cf. MNN-300 in Table 3.

Regarding the HIC_{15} key result, ST-PCE-3 should be used for further analysis. After reviewing the simulation videos, the second peak, if existent, comes from an impact of the dummy head against the instrument panel. This incident must be prevented at all costs and thus reliably predicted. Optimal predictions of the key result are less important than the statement whether the impact will occur or not. This is better accomplished by MNN-300, see Fig. 39. Moreover, appropriate head acceleration curve approximations are also provided by other methods, e.g. ST-GPR and MNN-200, see Table 3. The same holds for the key result approximations where PCE-3, ST-PCE-2, PCE-2, SNN, MNN-200 for example show high R^2_{KR} values.

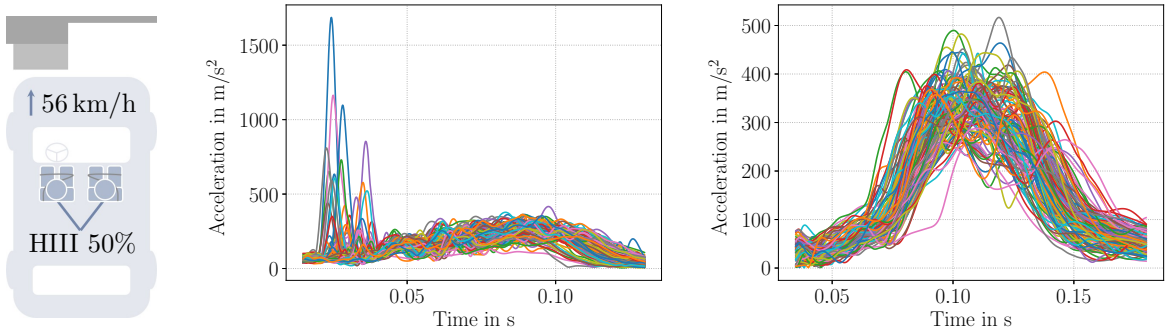
The regressor chain-based approach cannot be recommended here. For all options tried, it downgraded the results compared to its integrated scalar metamodels, e.g. observe the results for GPR, ST-GPR, and RC-GPR in Table 3. This may be interpreted such that the curve regression task here cannot benefit from past information. On the contrary, it lowers the quality, as it has to cope with one more parameter. This does not only apply to the head but also to the chest acceleration curves. Future work should be dedicated to deeply analyze this observation.

Analogous to the head acceleration curves, the same metamodel types are implemented for the resultant chest acceleration curves and their 3 ms values. Again MNN-300 provides the best result for total curve approximation, see Table 3. The single validation time histories — exact curves and predicted profiles produced by MNN-300 — can be observed in Fig. 40. Furthermore, when training the network for shorter epochs, e.g. 100, slightly better results for CA_{3ms} are obtained. Indeed, the best metamodel to approximate CA_{3ms} values is PCE-1, cf. Table 3. Its AvP plot can be viewed in Fig. 15 (c). In conclusion, further in-depth analysis is enabled by replacing the expensive, i.e. 45 hours, finite element model with one of the adequate fast-responding, i.e. within a fraction of a second, metamodels presented. This demonstrates the merit of metamodels for the proposed framework.

Hardware data from the offset deformable barrier frontal impact test

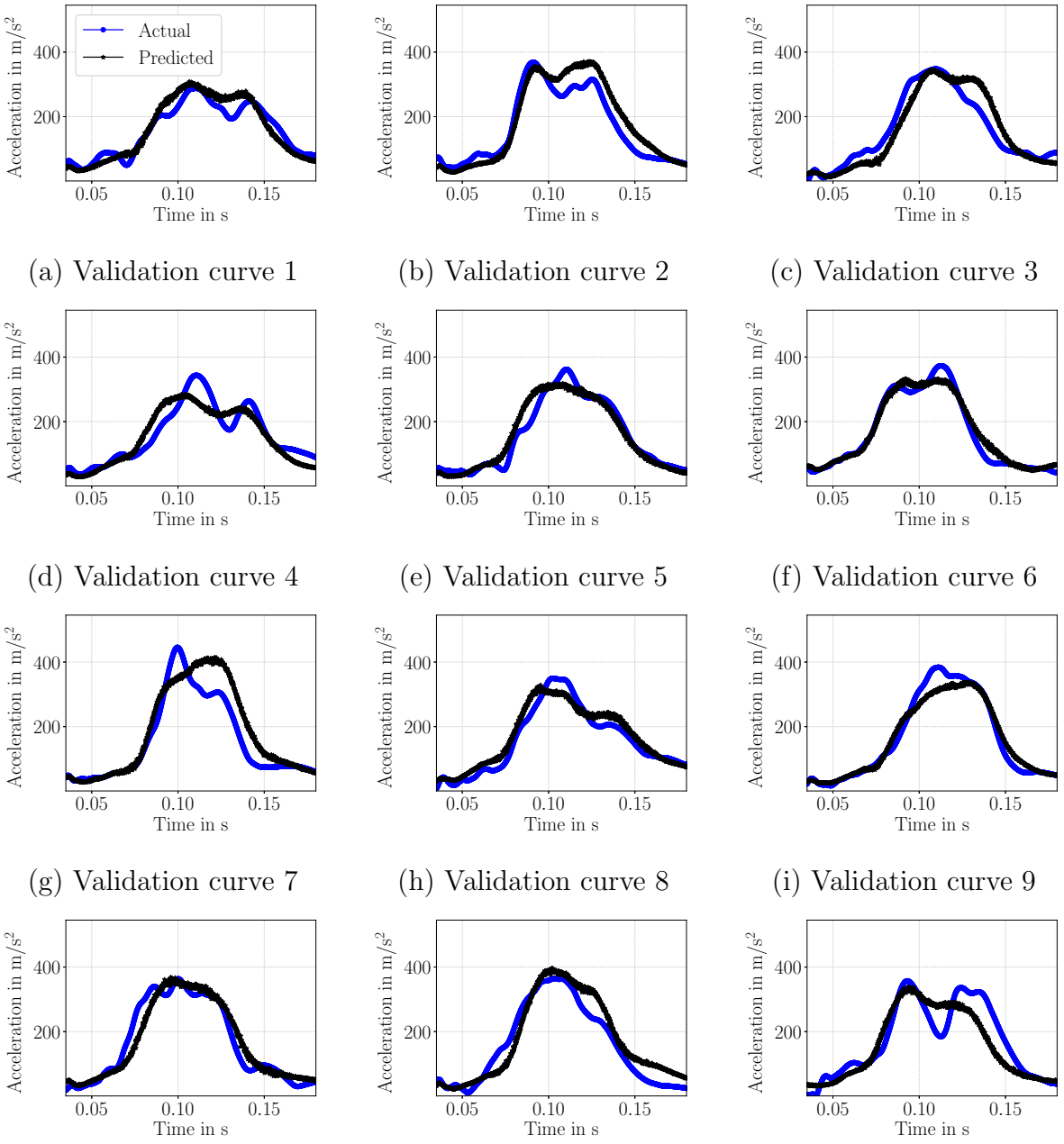
For several years now, the so-called offset deformable barrier (ODB) frontal impact test has been carried out for various consumer protection organizations that are mostly referred to as New Car Assessment Programs (NCAP). The ODB frontal impact is part of e.g. the Euro NCAP, China NCAP, etc., see [car21]. There, the test vehicle contacts a crushable aluminum honeycomb barrier with an overlap of 40% on the driver's side at 56 km/h, see [Uni17]. This is meant to mimic a collision with another vehicle.

To measure the impact on the occupants, two dummies of type Hybrid III 50th percentile male (HIII 50%) are placed on the front seats, see Fig. 16 (a). For this study, the author was provided with data from 112 historical tests from the archive of the BMW Group. These included small, medium, and large cars with different engines, optional equipment, etc., from various phases of the product evolution process, see Fig. 1. Different data, e.g. displacement, velocity, acceleration, force curves, etc., measured by sensors, e.g. the dummies, are available for each test. These quantities are used to assess crash severity. For the NCAPs, some of them are processed to injury criteria that can be converted to rating values, e.g. via injury risk curves or sliding scale functions. Thus, they provide the safety rating of the observed vehicle. This rating then serves as a reference for customers, see [car21].



(a) (b) Crash pulse (c) Head acceleration

Figure 16: Sketch of the ODB frontal impact test (a); 113 hardware curves of the crash pulses (b) and the head accelerations (c).



(a) Validation curve 1 (b) Validation curve 2 (c) Validation curve 3

(d) Validation curve 4 (e) Validation curve 5 (f) Validation curve 6

(g) Validation curve 7 (h) Validation curve 8 (i) Validation curve 9

(j) Validation curve 10 (k) Validation curve 11 (l) Validation curve 12

Figure 17: The predictions results (black) for all $n_{\text{val}} = 12$ individual validation head acceleration curves are compared to the exact hardware trajectories (blue).

It is now attempted to model the relationship between two special data types by a feed-forward deep neural network, i.e. a multi-target regression metamodel, $\mathcal{M}_{\text{SH}}$ (3.46). Crash pulses, time histories of vehicle accelerations, are considered to be the input of the network. They are already recorded in previous early-phase structural tests — with no dummies being involved yet. This means they are available a-priori, i.e. before the dummy measurements. Thus, they can indeed be chosen as feature of the neural network.

Occupant load curves are regarded as the outputs. These load curves are the ingredients to calculate the safety rating of the corresponding NCAP. Here, head acceleration curves of the driver dummy are investigated. Fig. 16 (b) and (c) show all 112 curve pairs, i.e. crash pulses and corresponding head accelerations.

For each curve type, one can observe differences between the trajectories, i.e. the curves are exposed to uncertainties. To train and validate the neural network, the data is randomly separated into $n_{\text{train}} = 100$ training and $n_{\text{val}} = 12$ validation sample curves. A dense network with three layers using 64, 128, 256 neurons each activated by φ_{tanh} activation functions, respectively, is chosen to emulate the relationship. The network is trained for 100 epochs.

The network does not receive any additional information that might improve its quality, i.e. no vehicle mass or measurements, nothing regarding usage or nature of the restraint systems, etc. Furthermore, the curves may exhibit noise due to possible deviations or inconsistencies in measurement processes, see the beginnings of the crash pulses in Fig. 16 (b) for instance. Both curve types are long time series of over thousand time steps — 1150 for the crash pulse curves, 1450 for the head acceleration curves. Each time step is considered to be a feature or an output value, i.e. the input and output spaces of the net are high dimensional. In general, there is little data that can be fed to the net.

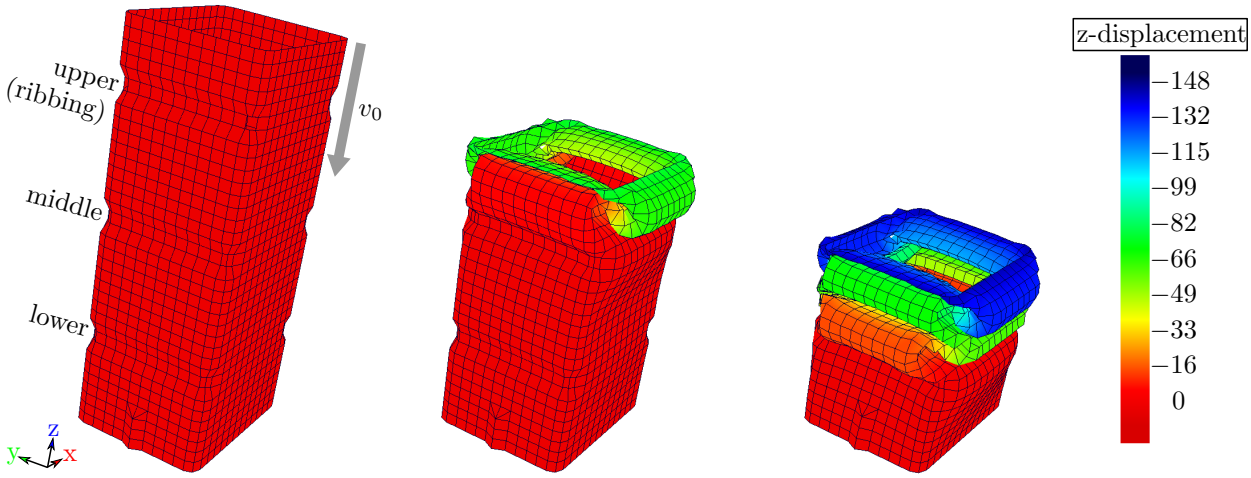
Despite all these obstacles, the network predicts the trends of the validation curves appropriately, see Fig. 17. One the one hand, this result shows the strong relationship between the two observed quantities, the crash pulses and the head accelerations, that is known from previous studies, see [KGE09, Lan21]. On the other hand, it encourages the use of metamodels in the crash domain.

Crashbox deformation simulation

In [JLG21a], the author of this thesis and his colleagues enable uncertainty quantification through a model order reduction metamodel that replaces a crashbox deformation simulation. The model simulated in LS-DYNA [LS-18] can be found on the LS-DYNA example homepage [LS-20] and takes 22 seconds CPU time. It calculates the deformation process of a crashbox that is impacted by a moving plate in negative z-direction, see Fig. 18. The crashbox is a symmetric tube with three pairs of ribbings. It consists of $N_{\text{EN}} = 1925$ nodes. The time histories of the node behaviors are divided into $N_{\text{T}} = 22$ time steps. Five parameters are declared uncertain: x_1 stands for the wall thickness of the crashbox, x_2 represents the initial velocity of the plate, and x_3, x_4, x_5 characterize the depths of the respective opposite upper, middle, and lower ribbings, see Fig. 18, respectively. All parameters are treated as uniformly distributed for the metamodeling task. Bounds for the parameter intervals can be found in [JLG21a].

The work [JLG21a] reconstructs the simulation via a model order reduction metamodel where singular value decomposition (SVD), i.e. basically principal component analysis, is used before training Gaussian process regression (GPR) models. This approach is applied to the x-, y-, z-displacements of all nodes simultaneously which yields $3N_{\text{EN}} = 5775$ rows for the snapshot matrix, S (3.71). Moreover, it divides the training data of size $n_{\text{train}} = 100$ into 20 simulations forming the snapshot matrix, i.e. $S \in \mathbb{R}^{3N_{\text{EN}} \times 20N_{\text{T}}}$, and 80 simulations used to create the metamodels. The data is produced by Latin hypercube sampling. SVD reduces the number of rows of S to $N_{\text{red}} = 8$. On top, a second SVD

is used to decompose the time and the parameter spaces. Then, several metamodels are built for time and parameter instances of the principal components. In this way, the time is regarded as a continuous variable as the times between the N_{T} time steps are interpolated. This approach called 2SVD-8-GPR in the following was implemented in MATLAB [MAT18]. Additionally to 2SVD-8-GPR from [JLG21a], the same strategy is run with more principal components, $N_{\mathrm{red}} = 20$ and $N_{\mathrm{red}} = 30$, denoted as 2SVD-20-GPR and 2SVD-30-GPR.



(a) Time state 1 (b) Time state 11 (c) Time state 22

Figure 18: Crashbox deformation simulation. The figure is taken from [JLG21a].

Another strategy is tested here. Three model order reduction metamodels, $\mathcal{M}^i_{\mathrm{EN}}$ (3.55), are constructed — one for each direction, $i \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$. Behavior directions with relatively little action are otherwise sieved out by the dimensionality reduction and the principal components are formed from the behavior directions with great action. As a consequence, predicting behavior directions with little action might be error-prone. The following procedure is discussed concerning one of the three model order reduction metamodels. Kernel PCA (kPCA) replaces standard SVD to account for nonlinear effects. The polynomial kernel (3.87) is employed. With its three hyperparameters, it is better suited and easier to tune than the Gaussian kernel (3.88) that possesses N_{EN} hyperparameters entailing a difficult optimization problem.

In the new process, the whole training data establishes the snapshot matrices, i.e. $S^i \in \mathbb{R}^{N_{\mathrm{EN}} \times 100 N_{\mathrm{T}}}, i \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$. Likewise, the metamodels are trained with all $n_{\mathrm{train}} = 100$ training points. Furthermore, the time is left discrete. Finite element models themselves do not work with continuous times. Time is therefore not considered as a variable which avoids prediction errors. Instead, each time step means one output entry for each of the $N_{\mathrm{red}}, N_{\mathrm{red}} < N_{\mathrm{EN}}$, chosen principal components. Together, N_{red} functions of the form $\mathcal{F}_{\mathrm{SH}}$ (3.16) must be approximated. This calls for multi-target regression metamodels, see Section 3.2.4. For each of the N_{red} selected principal components from kPCA, a single-target method using Gaussian process regression (ST-GPR) is trained.

The kPCA-ST-GPR metamodel is optimized using the nested approach, see Section 3.2.5 and 3.2.6. A 5-fold cross validation is chosen as cost function, C, in (3.89) to

avoid overfitting. This incident might appear as the hyperparameters of the kPCA-ST-GPR may be trimmed to optimally fit the training values when using the common least-squares formula. Ideal generalization, however, may not be given. Thus, cross validation comes into play. After the cost function is minimized, i.e. the 5-fold cross validation is terminated, the settings for kPCA are installed for the final kPCA-ST-GPR metamodel. The utilized multi-target regression metamodels, i.e. ST-GPR, are fitted for the last time using the whole training set rather than a fraction as for the nested optimization involving cross validation.

Numerically, the nested optimization is executed by the randomized search routine from scikit-learn [PVG+11]. It tests 200 hyperparameter configurations of the reduction method, then trains the corresponding multi-target regression metamodels, and returns the best hyperparameter values that are used for the final model order reduction metamodel. Different numbers of principal components, i.e. N_{red}, are considered and compared — also with the 2SVD-8-GPR approach from [JLG21a]. For the comparison, special forms of the mean squared error and the coefficient of determination for the overall fit are calculated with $n_{\mathrm{val}} = 100$ validation sample points. These points are created by a second Latin hypercube sampling.

The coefficient of determination, cf. (3.32), for the whole simulation is calculated in [JLG21a] using

$$R^2_{\mathrm{EN}}\left(x^1, \ldots, x^{n_{\mathrm{val}}}\right) = 1 - \frac{\sum_{i=1}^{n_{\mathrm{val}}} \sum_{j=1}^{3N_{\mathrm{EN}}} \sum_{k=1}^{N_{\mathrm{T}}} \left(Y^i_{jk} - \hat{Y}^i_{jk}\right)^2}{\sum_{i=1}^{n_{\mathrm{val}}} \sum_{j=1}^{3N_{\mathrm{EN}}} \sum_{k=1}^{N_{\mathrm{T}}} \left(Y^i_{jk} - \sum_{\bar{i}=1}^{n_{\mathrm{val}}} \sum_{\bar{j}=1}^{3N_{\mathrm{EN}}} \sum_{\bar{k}=1}^{N_{\mathrm{T}}} Y^{\bar{i}}_{\bar{j}\bar{k}}\right)^2} \tag{3.97}$$

for the validation parameter vectors $x^1, \ldots x^{n_{\mathrm{val}}} \in \mathcal{X}$, its corresponding exact outputs $Y^1, \ldots, Y^{n_{\mathrm{val}}} \in \mathbb{R}^{3N_{\mathrm{EN}} \times N_{\mathrm{T}}}$ and the approximations $\hat{Y}^1, \ldots, \hat{Y}^{n_{\mathrm{val}}} \in \mathbb{R}^{3N_{\mathrm{EN}} \times N_{\mathrm{T}}}$ produced by the model order reduction metamodel. Here, the mean squared error (3.30),

$$\mathrm{MSE}_{\mathrm{EN}}\left(x^1, \ldots, x^{n_{\mathrm{val}}}\right) = \frac{1}{3n_{\mathrm{val}}N_{\mathrm{EN}}N_{\mathrm{T}}} \sum_{i=1}^{n_{\mathrm{val}}} \sum_{j=1}^{3N_{\mathrm{EN}}} \sum_{k=1}^{N_{\mathrm{T}}} \left(Y^i_{jk} - \hat{Y}^i_{jk}\right)^2, \tag{3.98}$$

is given in addition to R^2_{EN}. To calculate these error measures for the new approach, the outputs of the three model order reduction metamodels are gathered in one matrix, i.e. for kPCA-ST-GPR, it holds

$$\hat{Y} = \begin{pmatrix} \hat{Y}^{\mathrm{x}} \\ \hat{Y}^{\mathrm{y}} \\ \hat{Y}^{\mathrm{z}} \end{pmatrix} \in \mathbb{R}^{3N_{\mathrm{EN}} \times N_{\mathrm{T}}} \tag{3.99}$$

with $\hat{Y}^i = \mathcal{M}^i_{\mathrm{EN}}(x) \in \mathbb{R}^{N_{\mathrm{EN}} \times N_{\mathrm{T}}}, i \in \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$, for an arbitrary but fixed $x \in \mathcal{X}$.

Table 4 shows a comparison of the results. Each metamodel yields appropriate predictions. All listed kPCA-ST-GPR models, i.e. using among others kPCA instead of SVD, outperform the 2SVD-GPR approaches. The best results are provided by the kPCA-10-ST-GPR model — using $N_{\mathrm{red}} = 10$ principal components. Its AvP plot for all nodes

and all time steps shown in Fig. 19 endorses the proper quality. According to Table 4, one might infer that the more principal components used, the better the results are. In [Go21], however, it is shown for analytical applications that the error can start to increase above a certain number of principal components since the last entities derived from them may become difficult to approximate. Therefore, with more principal components, the approximation error may exceed the truncation error leading to worse results than when using fewer principal components.

Table 4: Results of the different model order reduction metamodels for the crashbox deformation simulation determined with the validation sample. The best result is marked in bold.

Approach	R^2_{EN}	MSE_{EN}
2SVD-8-GPR [JLG21b]	0.99980	0.12299
2SVD-20-GPR	0.99986	0.08784
2SVD-30-GPR	0.99986	0.08551
kPCA-4-ST-GPR	0.99987	0.07910
kPCA-5-ST-GPR	0.99991	0.05315
kPCA-6-ST-GPR	0.99997	0.01915
kPCA-7-ST-GPR	0.99996	0.02156
kPCA-8-ST-GPR	0.99996	0.02028
kPCA-9-ST-GPR	0.99997	0.01769
kPCA-10-ST-GPR	**0.99998**	**0.01347**

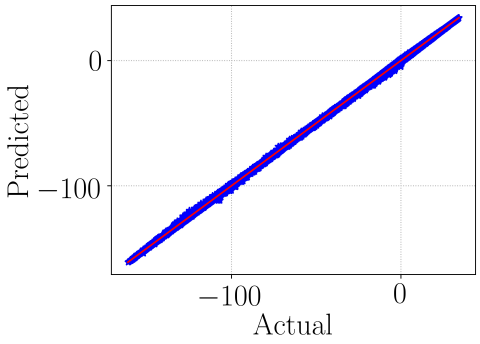


Figure 19: AvP of the best performing model order reduction metamodel, i.e. kPCA-10-ST-GPR, for all entries of the $n_{\text{val}} = 100$ validation matrices.

Furthermore, using less, e.g. $N_{\text{red}} = 4$, principal components means creating less scalar metamodels, e.g. $4N_{\text{T}} = 88$, for the single-target methods inside $\mathcal{M}^i_{\text{EN}}$ — leading to a compacter model. Using $N_{\text{red}} = 10$ principal components, for instance, demands $10N_{\text{T}} = 220$ scalar metamodels to be trained. Training and validation is faster when taking less principal components into account. Indeed, training must be done once and validation is quick — even for models constituted of many principal components. Nonetheless, as a user, one has to find the correct balance between approximation quality and efficiency.

All shown examples considering models for or data from structural components, restraint systems, or complete tests were approximated appropriately by different metamodeling techniques. These include the simplified dummy in a side crash, the occupant simulation of a full frontal test, the hardware data from the ODB frontal impact test, and the crashbox deformation simulation. Without metamodeling, subsequent analysis would not be feasible with respect to time. Their high qualities for the discussed four examples strongly support their usage as chief point in the proposed framework for further applications. Note that they do not only enable sensitivity and uncertainty analysis. Engineers can use them to rapidly perform optimization, reliability analysis, etc.

3.3 || Sensitivity analysis

Two sensitivity approaches are discussed that intend to achieve different goals. Screening is a qualitative approach to identify the relevant inputs that can then be further investigated by quantitative sensitivity approaches, e.g. Sobol' indices. While screening techniques are used without metamodels and can improve approximation quality when metamodels are applied subsequently, variance-based measures are usually calculated on metamodels, see the global framework in Fig. 4.

3.3.1 Distinction between local and global sensitivity analysis

Before these two methods are introduced, a distinction between local and global sensitivity analysis is given. Both methods, screening and variance-based sensitivity analysis, belong to global sensitivity analysis, see [IL15]. This type seeks to find the influence of the full parameter range on the global output uncertainty, see [ZL08a]. Speaking about Sobol' indices, cf. Section 3.3.3, the problem is treated in a probabilistic framework where $\mathscr{X}$ describes the random input vector. Variance-based sensitivity analysis determines the proportion of the impact that the single dimensions of $\mathscr{X}$, i.e. single model parameters, at full range have on

$$\mathrm{Var}(\mathcal{F}(\mathscr{X})) = \mathrm{Var}(\mathscr{Y}). \tag{3.100}$$

By contrast, local sensitivity analysis is the measurement of the influence of local changes in parameters on the output, i.e. the sensitivity is evaluated near a chosen point of interest, see [ZL08b]. Often, local sensitivities are calculated by using gradients,

$$\frac{\partial \mathcal{F}(x)}{\partial x_i}, \quad i \in \{1, \dots, n\}, \tag{3.101}$$

see [CFP02, SRTC05, PBF+16]. In [Sal99], clear drawbacks of local derivative-based methods are reported, e.g. only a fraction of the full parameter range is studied, possible interactions are ignored, etc. The work also recommends to calculate Sobol' measures. Note that local approaches may be extended to global approaches, e.g. via integration over the input space, see [SK10, LIPG13] that also present links to Sobol' indices. This extension makes them attractive but still requires derivatives.

In finite element software, however, several heuristics, detailed modeling, and case distinctions, among others, are involved. Even if the source code would be available, i.e. the simulation would not be regarded as a black box, a derivative-based sensitivity analysis might not be practicable. For this reason, sampling-based screening techniques and variance-based sensitivity analysis are chosen. In the following, the global approaches are meant when sensitivities are to be investigated.

3.3.2 Screening

For all types of analyses that investigate model behavior based on sample points, the more variables there are, the larger the sample size may need to be. The number of variables

translates into the number of model dimensions — the more variables, the higher the number of dimensions and the more difficult the analyses are to perform. In practice, it repeatedly turned out that only the minority of input dimensions contributes significantly to the model output, see [IL15]. Screening methods as qualitative sensitivity analysis techniques identify the relevance of each input implying an input importance ranking, see [IS17]. Hardly or not contributing parameters can be removed from the analysis. In this way, screening can avoid the curse of dimensionality and serve as an alternative to the dimensionality reduction add-on approach for metamodels from Section 3.2.6.

Compared to other sensitivity analysis methods, screening methods work on low budget yet are less profound. In global sensitivity analysis frameworks, it is therefore common to use screening to eliminate irrelevant variables before a more informative but expensive sensitivity approach is applied to the dimension reduced model, see [JLG21b]. A quantitative, i.e. more subtle, sensitivity method after screening may be the calculation of variance-based sensitivity measures introduced in Section 3.3.3. In the following, the elementary effects method is proposed as it is able to identify all types of model complexity, e.g. non-monotonicity and discontinuity, see [IL15].

Radial design-based elementary effects method

A global screening plan was published 1991 by Morris, see [Mor91]. This approach — known as elementary effects method or Morris method — highlights the most important parameters among those of a computer experiment. For each parameter, so-called elementary effects are calculated based on a predefined sampling strategy — consisting of one-factor-at-a-time trajectories in the original approach, see [CCS07]. Estimators for the mean and variance of the elementary effects indicate the contribution of the single parameters to the output uncertainty, see [SRA+08]. According to these quantities, the parameters can be ranked.

An improved version of the Morris method can be found in [CSC11]. It is here referred to as the radial design-based elementary effects method. There, an enhanced estimator for the mean of the elementary effects is presented. In addition, a more efficient sampling strategy is suggested using radial-based setups that also allows to advance the method to calculate total effect Sobol' indices. Our global framework does not only aim for an in-depth sensitivity analysis but also for a metamodel needed for subsequent uncertainty quantification. It is computationally more favorable to later compute Sobol' indices on the approximated response surface than to use the extension of the elementary effects method. The screening method itself, however, can be applied to simplify the work of metamodels by reducing the input dimensionality of the problem. The mathematics behind are briefly summarized based on [CSC11, JLG21b].

Let $x = (x_1, \ldots, x_n)^\top \in [0,1]^n$ be the input of the model, i.e. the system function,

$$\mathcal{F} : [0,1]^n \to \mathbb{R}, \tag{3.102}$$

e.g. $\mathcal{F} = \mathcal{F}_{\mathrm{KR}}$ from Section 3.1, that produces the output

$$y = \mathcal{F}(x) \in \mathbb{R}. \tag{3.103}$$

No requirements are imposed on the system function. Note that the unit hypercube input space, $[0,1]^n$, does not cause a loss of generality since any compact one-dimensional interval can be transformed into $[0,1]$, cf. (3.25).

The method is designed using $r \in \mathbb{N}$ random starting points,

$$\hat{x}^{(j)} = \left(\hat{x}_1^{(j)}, \ldots, \hat{x}_n^{(j)}\right)^\top \in [0,1]^n, \quad j \in \{1, \ldots, r\}. \tag{3.104}$$

Each random starting point asks for extra n points to compute an elementary effect for each dimension. By doing so, the whole procedure results in an overall sample size equal to $r(n+1)$. A non-zero perturbation, $\Delta_i^{(j)} \in (-1,1)\backslash\{0\}$, $i \in \{1, \ldots, n\}$, is added to the i-th component of $\hat{x}^{(j)}, j \in \{1, \ldots, r\}$, yielding the i-th additional point — this is illustrated in Fig. 20. It must hold

$$\delta_i^{(j)} \coloneqq \hat{x}_i^{(j)} + \Delta_i^{(j)} \in [0,1]. \tag{3.105}$$

In [CSC11], the sampling is accomplished via a Sobol' sequence, i.e. a low-discrepancy sequence, to generate the random starting points, $\hat{x}^{(j)}$, and their altered companions,

$$\delta^{(j)} \coloneqq \left(\delta_1^{(j)}, \ldots, \delta_n^{(j)}\right)^\top. \tag{3.106}$$

Note that the perturbation, $\Delta_i^{(j)}$, must be chosen sufficiently large to identify effects due to actual parameter changes rather than e.g. rounding errors, especially for discontinuous computer models.

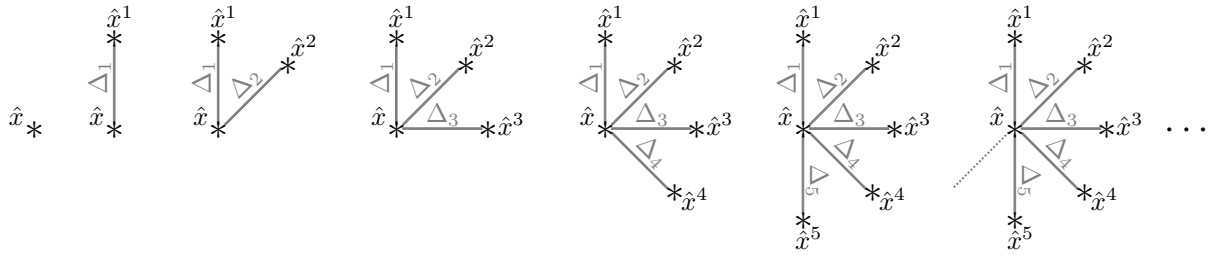


Figure 20: Creation of the radial design sample for an arbitrary but fixed random starting point, $\hat{x} \in \left\{\hat{x}^{(1)}, \ldots, \hat{x}^{(r)}\right\}$.

The elementary effect, $\mathrm{EE}_i^{(j)}$, of $\hat{x}^{(j)}$ for the i-th parameter is then calculated by

$$\mathrm{EE}_i^{(j)} \coloneqq \frac{\mathcal{F}\left(\hat{x}^{(j)} + \Delta_i^{(j)} e_i\right) - \mathcal{F}\left(\hat{x}^{(j)}\right)}{\Delta_i^{(j)}} = \frac{\mathcal{F}\left(\hat{x}_1^{(j)}, \ldots, \delta_i^{(j)}, \ldots, \hat{x}_n^{(j)}\right) - \mathcal{F}\left(\hat{x}^{(j)}\right)}{\delta_i^{(j)} - \hat{x}_i^{(j)}}, \tag{3.107}$$

where $e_i, i \in \{1, \ldots, n\}$, denotes the i-th canonical basis of $\mathbb{R}^n$. These effects can be understood as directional derivatives along the respective dimensions.

Now, estimators for the mean,

$$\mu_i \coloneqq \frac{1}{r} \sum_{j=1}^{r} \mathrm{EE}_i^{(j)}, \tag{3.108}$$

and the variance,

$$\sigma_i \coloneqq \frac{1}{r}\sum_{j=1}^{r}\left(\mathrm{EE}_i^{(j)} - \mu_i\right)^2, \tag{3.109}$$

of the elementary effects, EE_i^j, for all inputs $i \in \{1,\dots,n\}$ can be determined. To avoid that elementary effects annul each other due to opposed signs, see [CCSS05], the conventional mean μ_i is augmented to the quantity,

$$\mu_i^* \coloneqq \frac{1}{r}\sum_{j=1}^{r}\left|\mathrm{EE}_i^{(j)}\right|. \tag{3.110}$$

According to these quantities, μ_i, μ_i^*, σ_i, the relevance of each input can be assessed.

Algorithm 2: Adaptive radial design-based elementary effects algorithm.

Input: System function, $\mathcal{F}$; initial number of iterations, $r \in \mathbb{N}$; threshold, $\varepsilon_{\mathrm{EE}} \in (0, 0.1]$

Output: Estimated means, μ_i and μ_i^*; variances, σ_i, $i \in \{1,\dots,n\}$; algorithm iterations, r_a

forall $j \in \{1,\dots,r\}$ **do**
  draw random starting point $\hat{x}^{(j)}$ from input space and create radial basis sample according to Fig. 20
  calculate elementary effects, $\mathrm{EE}_i^{(j)}$, see (3.107)
calculate Morris quantities, μ_i, μ_i^*, σ_i, using (3.108), (3.110), (3.109), and denote as μ_i^o, $\mu_i^{*,\mathrm{o}}$, σ_i^o, respectively
set flag = true and $r_\mathrm{a} = 0$
while flag **do**
  set $r_\mathrm{a} \leftarrow r_\mathrm{a} + 1$
  draw random starting point, create radial basis sample and calculate elementary effects, $\mathrm{EE}_i^{(r+r_\mathrm{a})}$
  update Morris quantities,

$$\mu_i = \frac{1}{r+r_\mathrm{a}}\sum_{j=1}^{r+r_\mathrm{a}}\mathrm{EE}_i^{(j)}, \quad \mu_i^* = \frac{1}{r+r_\mathrm{a}}\sum_{j=1}^{r+r_\mathrm{a}}\left|\mathrm{EE}_i^{(j)}\right|, \quad \sigma_i = \frac{1}{r+r_\mathrm{a}}\sum_{j=1}^{r+r_\mathrm{a}}\left(\mathrm{EE}_i^{(j)} - \mu_i\right)^2$$

  if $\frac{1}{3n}\sum_{i=1}^{n}\frac{\left|\mu_i-\mu_i^\mathrm{o}\right|}{|\mu_i|} + \frac{\left|\mu_i^*-\mu_i^{*,\mathrm{o}}\right|}{\mu_i^*} + \frac{\left|\sigma_i-\sigma_i^\mathrm{o}\right|}{\sigma_i} < \varepsilon_{\mathrm{EE}}$ **then**
    set flag = false
  else
    continue
  set $\mu_i^\mathrm{o} = \mu_i$, $\mu_i^{*,\mathrm{o}} = \mu_i^*$, $\sigma_i^\mathrm{o} = \sigma_i$

Adaptive algorithm for radial design-based elementary effects method
A novel adaptive algorithm is proposed where the radial design-based elementary effects method is embedded. The initial step of the algorithm is to run the common method introduced above using a small $r \in \mathbb{N}$. First estimates for the quantities, μ_i, μ_i^*, σ_i are obtained that are denoted as μ_i^{o}, $\mu_i^{*,\mathrm{o}}$, and σ_i^{o}, respectively. It is recommended to initialize with $r = 3$ — cf. [STCR04] where the smallest number suggested is equal to 4 for performing the mere Morris method.

For the second step of the algorithm — considered to be the first algorithm iteration — an additional elementary effect for each input is determined. This means another random starting point, $\hat{x}$, is drawn and the procedure is repeated once more. The new elementary effects are then assembled with the pre-existing effects and processed into updated estimates for the means, μ_i and μ_i^*, and the variance, σ_i.

If the stopping criterion,

$$\frac{1}{3n} \sum_{i=1}^{n} \frac{|\mu_i - \mu_i^{\mathrm{o}}|}{|\mu_i|} + \frac{|\mu_i^* - \mu_i^{*,\mathrm{o}}|}{\mu_i^*} + \frac{|\sigma_i - \sigma_i^{\mathrm{o}}|}{\sigma_i} < \varepsilon_{\mathrm{EE}}, \tag{3.111}$$

is fulfilled for a fixed threshold $\varepsilon_{\mathrm{EE}} \in (0, 1)$, the algorithm ends. If not, we set μ_i^{o}, $\mu_i^{*,\mathrm{o}}$, σ_i^{o} equal to the updated quantities, μ_i, μ_i^*, σ_i. In case $\mu_i = 0, \mu_i^* = 0$, or $\sigma_i = 0$ appear, corresponding quotients must be excluded from the sum to avoid dividing through zero. Then, a further random starting point is selected yielding a renewed update of the elementary effects and hence of the aforementioned quantities. The stopping criterion is re-examined. As long as the criterion is not satisfied, the estimates, μ_i, μ_i^*, σ_i, are revised step by step through the use of new random starting points.

The entire adaptive algorithm thus demands an overall sample of size

$$(r + r_{\mathrm{a}})(n + 1) \tag{3.112}$$

to be evaluated where r_{a} denotes the number of algorithm iterations. Note that the stopping criterion in (3.111) is met when an additional elementary effect does not affect the estimates for the means and the variance. The algorithm formulated as pseudo code in Algorithm 2 guarantees reliable results and saves computational effort that could incur as one might involuntarily choose r higher than necessary in the conventional method.

Assessment and ranking
After all, results of the algorithm can be assessed. Estimated means, μ_i and μ_i^*, indicate the influence of the respective parameter, x_i, $i \in \{1, \ldots, n\}$, to the output variability of the system function, $\mathcal{F}$. The higher their values are, the more important is the corresponding parameter. Quantity, σ_i, measures the degree of interactions of each parameter with other variables and identifies possible evoked nonlinear behavior, see [CCS07].

Statements can be made by ranking μ_i, μ_i^*, and σ_i, $i \in \{1, \ldots, n\}$, respectively. Here, the observation of the estimated mean, μ_i, is omitted — only the enhanced version, μ_i^*, is considered. It is common to plot the values for σ_i against the ones for μ_i^* to get a clear overview. If a plotted point is close to the origin, the corresponding parameter is of low relevance for the output and does not interact with other parameters. This is due to the fact that both quantities, μ_i^* and σ_i, must then exhibit low values.

Besides the visual assessment of the input relevance, the formula,

$$\sum_{i=1}^{k} \sigma_i^{\mathrm{h}} \Big/ \sum_{i=1}^{n} \sigma_i \geq \varepsilon_\sigma, \tag{3.113}$$

from [JLG21b], cf. (3.76), is used, where $\sigma_1^{\mathrm{h}} > \cdots > \sigma_k^{\mathrm{h}}$ are the $k \in \mathbb{N}, k < n$, highest estimated variances picked out of $\{\sigma_i | i \in \{1, \ldots, n\}\}$. If this inequality is fulfilled for a given tolerance $\varepsilon_\sigma \in [0.9, 1)$, it is ensured that a percentage over $\epsilon_\sigma \cdot 100\%$ of the elementary effects variance resides in the k highest ranked parameters. Hence, using (3.113), the most important variables — the k highest ranked parameters — are automatically determined. This can be safeguarded by examining the ranking of the estimated mean, μ_i^*. One may as well apply a formula for this quantity analogous to (3.113).

In any case, the irrelevant parameters can be neglected in further investigations. This reduces the number of input dimensions, i.e. the model dimensionality, from n to k and therefore accelerates subsequent analyses which renders the method valuable. Note that the radial design-based elementary effects method is in fact a global approach but its results nonetheless rely on the selection of random starting points.

3.3.3 Variance-based sensitivity analysis

Screening introduced in Section 3.3.2 can identify irrelevant parameters whereby the number of model dimensions is decreased. To better understand the remaining input dimensions, i.e. the relevant variables, variance-based sensitivity measures are calculated. These measures profoundly analyze the relationship between the individual parameters and the observed output as well as interactions among the parameters. In contrast to screening methods, they claim many samples and are therefore computationally intensive. Variance-based sensitivity measures belong to the field of quantitative global sensitivity analysis methods. These measures are chosen as they can cope with nonlinear responses and measure interaction effects in non-additive systems, see [SRA+08].

Theoretical background of Sobol' indices

Originating from the work of Sobol', see [Sob93], variance-based sensitivity measures are also called Sobol' indices. The theoretical foundations of these measures are now briefly introduced by reference to [Sob01, SRA+08, Sal10a]. Let

$$x = (x_1, \ldots, x_k)^\top \in [0, 1]^k, \quad k \in \mathbb{N}, \quad k \leq n, \tag{3.114}$$

without loss of generality be the preserved parameters after the use of screening for the reduced system function,

$$\mathcal{F} : \ [0, 1]^k \ \rightarrow \ \mathbb{R}, \tag{3.115}$$

mapped to the output

$$y = \mathcal{F}(x) \in \mathbb{R}, \tag{3.116}$$

cf. (3.103).

In case no screening method was used or all parameters are relevant according to screening, k is equal to n. It is furthermore assumed that the parameters are independently and uniformly distributed within the unit hypercube, $[0,1]^k$, and $\mathcal{F}$ is integrable as well as square-integrable. Again, assuming hypercubes as input spaces is not a restriction, cf. (3.25). Furthermore, uniform random numbers can be converted into numbers of other distributions, via inversion methods, see [Dev06]. Parameter independence is given in this work, see (3.8). Per parameter, two sensitivity indices are presented in the following that are deduced from the variance decomposition of the output.

The foundation of the variance-based sensitivity indices is given by the fact that $\mathcal{F}$ can be decomposed into

$$\mathcal{F}(x) = \mathcal{F}_0 + \sum_{i=1}^{k} \mathcal{F}_i(x_i) + \sum_{i_1=1}^{k-1} \sum_{i_2=i_1+1}^{k} \mathcal{F}_{i_1,i_2}(x_{i_1}, x_{i_2}) + \ldots + \mathcal{F}_{1,2,\ldots,k}(x_1, x_2, \ldots, x_k), \tag{3.117}$$

where the total number of summands is equal to 2^k. In (3.117), one constant term, $\mathcal{F}_0$, k first-order functions, $\mathcal{F}_i$ of x_i, $\binom{k}{2}$ second-order functions, $\mathcal{F}_{ij}$ of x_i and x_j, etc., appear. To call (3.117) the ANOVA (Analysis of variances) representation of $\mathcal{F}$, the unicity condition,

$$\int_0^1 \mathcal{F}_{i_1,i_2,\ldots,i_s}(x_{i_1}, x_{i_2}, \ldots, x_{i_s})\, dx_{i_w} = 0, \tag{3.118}$$

where $1 \leq i_1 < i_2 < \cdots < i_s \leq k$, $i_w \in \{i_1, i_2, \ldots, i_s\}$, and $s \in \{1, \ldots, k\}$, must additionally hold for the functions in (3.117). This property implies orthogonality of all members in (3.117). Sobol' proved that there exists a unique decomposition (3.117) satisfying (3.118) for any function integrable in $[0,1]^k$, see [Sob93].

Beyond, one can determine expressions for each function in the ANOVA representation of $\mathcal{F}$, i.e.

$$\mathcal{F}_0 = \int_{[0,1]^k} \mathcal{F}(x)\, dx, \tag{3.119}$$

$$\mathcal{F}_i(x_i) = \int_{[0,1]^{k-1}} \mathcal{F}(x) \prod_{w \in \{1,\ldots,k\} \setminus \{i\}} dx_w - \mathcal{F}_0, \quad 1 \leq i \leq k, \tag{3.120}$$

$$\mathcal{F}_{ij}(x_i, x_j) = \int_{[0,1]^{k-2}} \mathcal{F}(x) \prod_{w \in \{1,\ldots,k\} \setminus \{i,j\}} dx_w - \mathcal{F}_0 - \mathcal{F}_i(x_i) - \mathcal{F}_j(x_j), \quad 1 \leq i < j \leq k, \tag{3.121}$$

etc. For the corresponding independent and uniformly distributed random variables $\mathscr{X}_i, i \in \{1, \ldots, k\}$, and the random variable $\mathcal{F}(\mathscr{X}) = \mathscr{Y}$, this translates to

$$\mathcal{F}_0 = \mathrm{E}(\mathcal{F}(\mathscr{X})) = \mathrm{E}(\mathscr{Y}), \tag{3.122}$$

$$\mathcal{F}_i \coloneqq \mathcal{F}_i(\mathscr{X}_i) = \mathrm{E}_{\mathscr{X}_{\sim i}}(\mathscr{Y} | \mathscr{X}_i) - \mathcal{F}_0, \tag{3.123}$$

$$\mathcal{F}_{ij} \coloneqq \mathcal{F}_{ij}(\mathscr{X}_i, \mathscr{X}_j) = \mathrm{E}_{\mathscr{X}_{\sim ij}}(\mathscr{Y} | \mathscr{X}_i, \mathscr{X}_j) - \mathcal{F}_i - \mathcal{F}_j - \mathcal{F}_0, \tag{3.124}$$

etc., where $\mathscr{X}_{\sim i}$ — and $\mathscr{X}_{\sim ij}$ — declare all variables but $\mathscr{X}_i$ — along with $\mathscr{X}_j$. The variance of the output can now be decomposed, i.e.

$$\begin{aligned}\mathrm{Var}(\mathscr{Y}) &= \mathrm{E}\left(\mathscr{Y}^2\right) - \mathrm{E}(\mathscr{Y})^2 = \mathrm{E}\left(\mathcal{F}^2(\mathscr{X})\right) - \mathcal{F}_0^2 \\ &= \sum_{i=1}^{k} \mathscr{V}_i + \sum_{i_1=1}^{k-1} \sum_{i_2=i_1+1}^{k} \mathscr{V}_{ij} + \ldots + \mathscr{V}_{1,2,\ldots,k}\end{aligned} \tag{3.125}$$

with the contributions

$$\mathscr{V}_i \coloneqq \mathrm{Var}_{\mathscr{X}_i}\left(\mathrm{E}_{\mathscr{X}_{\sim i}}(\mathscr{Y}|\mathscr{X}_i)\right), \tag{3.126}$$

$$\mathscr{V}_{ij} \coloneqq \mathrm{Var}_{\mathscr{X}_i,\mathscr{X}_j}\left(\mathrm{E}_{\mathscr{X}_{\sim ij}}(\mathscr{Y}|\mathscr{X}_i,\mathscr{X}_j)\right) - \mathscr{V}_i - \mathscr{V}_j, \tag{3.127}$$

etc. There, the square-integrability of $\mathcal{F}$ was used. Furthermore, for the term $\mathrm{E}\left(\mathcal{F}^2(\mathscr{X})\right)$, the ANOVA representation was squared where several terms vanish after integrating due to the orthogonality of the members. Note that the quantity $\mathscr{V}_i$ is the variance — taken over the i-th parameter — of the output mean conditional on the i-th parameter over all factors but the i-th.

The first order and total effect sensitivity indices are defined as

$$\mathscr{S}_i = \frac{\mathscr{V}_i}{\mathrm{Var}(\mathscr{Y})}, \tag{3.128}$$

$$\mathscr{S}_{\mathrm{T}_i} = 1 - \frac{\mathrm{Var}_{\mathscr{X}_{\sim i}}\left(\mathrm{E}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i})\right)}{\mathrm{Var}(\mathscr{Y})} = \frac{\mathrm{E}_{\mathscr{X}_{\sim i}}(\mathrm{Var}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i}))}{\mathrm{Var}(\mathscr{Y})}, \tag{3.129}$$

respectively. The first order index, $\mathscr{S}_i$, measures the main effect, i.e. first order effect, of the considered parameter on the model output. In formula (3.128), this can be recognized by the term $\mathscr{V}_i$ which signifies the part of the variance that is caused solely and directly by the i-th parameter. All other effects resulting from the remaining parameters or the combination of the considered parameter and other quantities are disregarded there.

By contrast, the total effect sensitivity index, $\mathscr{S}_{\mathrm{T}_i}$, estimates first and higher order effects of the i-th parameter. Interactions involving this parameter are examples of higher order effects in $\mathscr{S}_{\mathrm{T}_i}$. This interpretation is confirmed by the fact that $\mathrm{Var}_{\mathscr{X}_{\sim i}}\left(\mathrm{E}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i})\right.$ involves the variance parts influenced by all parameters except the i-th and its interactions, i.e. it contains the first order effects of $\mathscr{X}_{\sim i}$. Thus, $\mathrm{Var}(\mathscr{Y}) - \mathrm{Var}_{\mathscr{X}_{\sim i}}\left(\mathrm{E}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i})\right)$ is the variance part that is composed of all terms where the i-th parameter provides an order-independent contribution. In (3.129), the equality

$$\mathrm{Var}(\mathscr{Y}) = \mathrm{E}_{\mathscr{X}_{\sim i}}(\mathrm{Var}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i})) + \mathrm{Var}_{\mathscr{X}_{\sim i}}\left(\mathrm{E}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i})\right) \tag{3.130}$$

was exploited, see [MGB74]. The quantity $\mathrm{E}_{\mathscr{X}_{\sim i}}(\mathrm{Var}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i}))$ moreover denotes the mean — taken over all factors except the i-th — of the output variance conditional on all factors but the i-th over the i-th parameter.

Whether higher order effects exist in the model, can hence be identified by calculating respective differences of $\mathscr{S}_{\mathrm{T}_i}$ and $\mathscr{S}_i$ — by definition, it holds $\mathscr{S}_{\mathrm{T}_i} \geq \mathscr{S}_i$. Moreover, by sorting $\mathscr{S}_i$, an importance ranking of the parameters is established. The output variance

can, on average, be reduced by $\mathscr{S}_i$ in case the i-th parameter would be fixed. If, for a fixed $i \in \{1, \ldots, k\}$, it holds that $\mathscr{S}_{\mathrm{T}_i} = 0$, the i-th parameter is a non-influential factor and can be set to any constant value within its range. For perfectly additive models, the sum of all $\mathscr{S}_i$ equals one, i.e.

$$\sum_{i=1}^{k} \mathscr{S}_i = 1. \tag{3.131}$$

If it is less than one, the model at hand is non-additive. Similarly, the sum of all $\mathscr{S}_{\mathrm{T}_i}$ is equal to one if the model is additive. Otherwise, the sum is greater than one and the model is non-additive.

Numerical calculation of Sobol' indices

In practice, the first order and total effect sensitivity indices are realized as described in [JLG21b]. To compute them, a radial design setup is again used, cf. Section 3.3.2. For the sake of consistency, we choose the same terms as for screening. This yields an alternative notation which underlines the similarity between Morris and Sobol' indices. Common notations are written down in [Sal10a]. Compared to the Morris method, the function has to be evaluated at $n_{\mathrm{Sob}} \in \mathbb{N}$ random starting points, $\hat{x}^{(j)} \in [0,1]^k$, and their altered companions,

$$\hat{x}_{\delta_i}^{(j)} \coloneqq \left(\hat{x}_1^{(j)}, \ldots, \delta_i^{(j)}, \ldots, \hat{x}_k^{(j)}\right)^\top \in [0,1]^k, j \in \{1, \ldots, N\}. \tag{3.132}$$

In addition and unlike the Morris method, the outputs of the mutations,

$$\delta^{(j)} = \left(\delta_1^{(j)}, \ldots, \delta_k^{(j)}\right) \in [0,1]^k, \quad j \in \{1, \ldots, N\}, \tag{3.133}$$

are needed.

The sensitivity indices can now be computed. In literature, one can find different ways to calculate them, see e.g. [Sob93, HS96, Jan96]. Saltelli et al. discuss several variants in [Sal10a]. According to their suggestion, the formula

$$s_i \coloneqq \frac{\frac{1}{n_{\mathrm{Sob}}} \sum_{j=1}^{n_{\mathrm{Sob}}} \mathcal{F}\left(\delta^{(j)}\right) \left(\mathcal{F}\left(\hat{x}_{\delta_i}^{(j)}\right) - \mathcal{F}\left(\hat{x}^{(j)}\right)\right)}{\frac{1}{2}\hat{\sigma}_{\hat{x},\delta}} \tag{3.134}$$

is chosen to estimate the first order sensitivity index whereas the so-called Jansen's estimator,

$$s_{\mathrm{T}_i} \coloneqq \frac{\frac{1}{2n_{\mathrm{Sob}}} \sum_{j=1}^{n_{\mathrm{Sob}}} \left(\mathcal{F}\left(\hat{x}^{(j)}\right) - \mathcal{F}\left(\hat{x}_{\delta_i}^{(j)}\right)\right)^2}{\frac{1}{2}\hat{\sigma}_{\hat{x},\delta}} \tag{3.135}$$

is used for the total effect sensitivity index for the single parameters, $x_i, i \in \{1, \dots, k\}$. These formulas serve as numerical representatives of $\mathscr{S}_i$ and $\mathscr{S}_{\mathrm{T}_i}$, see (3.128) and (3.129), since, on the one hand, the coincident denominators of s_i and $s_{\mathrm{T}_i}, i \in \{1, \dots, k\}$, in (3.134) and (3.135) are estimators for the variance of the output,

$$\hat{\sigma}_{\hat{x},\delta} \coloneqq \frac{1}{n_{\mathrm{Sob}}} \sum_{j=1}^{n_{\mathrm{Sob}}} \left(\mathcal{F}\left(\hat{x}^{(j)}\right)^2 + \mathcal{F}\left(\delta^{(j)}\right)^2 \right) - \left(\frac{1}{n_{\mathrm{Sob}}} \sum_{j=1}^{n_{\mathrm{Sob}}} \mathcal{F}\left(\hat{x}^{(j)}\right) \right)^2 - \left(\frac{1}{n_{\mathrm{Sob}}} \sum_{j=1}^{n_{\mathrm{Sob}}} \mathcal{F}\left(\delta^{(j)}\right) \right)^2 . \tag{3.136}$$

On the other hand, the numerator of s_i estimates $\mathscr{V}_i$, see (3.126), that is the numerator of $\mathscr{S}_i$ and the numerator of s_{T_i} gauges $\mathrm{E}_{\mathscr{X}_{\sim i}}(\mathrm{Var}_{\mathscr{X}_i}(\mathscr{Y}|\mathscr{X}_{\sim i}))$ that is the numerator of $\mathscr{S}_{\mathrm{T}_i}$.

The calculation effort to determine all Sobol' indices, s_i and s_{T_i}, amounts to

$$n_{\mathrm{Sob}}(k+2). \tag{3.137}$$

To gain accurate results, a high n_{Sob} is requisite. Here, it is proposed to use $n_{\mathrm{Sob}} = 10\,000$. This entails heavy computational costs for the method. Variance-based sensitivity analysis for time-consuming models is therefore infeasible with respect to computational time, e.g. for explicit finite element simulations of crash events with contact issues.

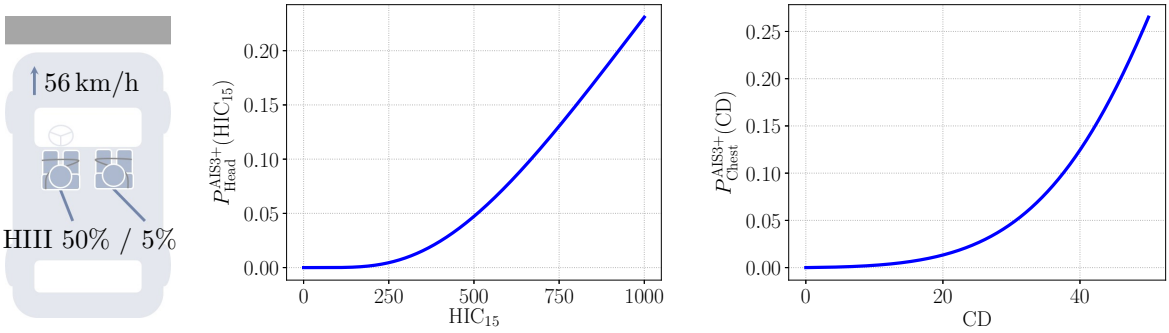
To allow this kind of analysis, it is typical to fit metamodels, e.g. from Section 3.2, to laborious system functions, $\mathcal{F}$, by feeding them a narrow size of training points, see [IL15]. Then, instead of evaluating the exact system function, the quickly responding metamodels are utilized to approximate the required outputs. In this way, the indices, s_i and s_{T_i}, can be computed within seconds. There exist even metamodels from that these indices can be derived analytically, i.e. calculating the estimators (3.134) and (3.135) can be omitted. Examples are polynomial chaos expansion in [Sud08] or Gaussian process regression in [MILR09]. The formulas, (3.134) and (3.135), however, can generally be applied to any model that can be translated into the form (3.115). An open question is whether sensitivities are fully preserved due to approximation by metamodels. Does a small approximation error lead to a small or yet larger noise in sensitivities? This question should be answered in future research.

3.3.4 Exemplary usage of sensitivity analysis

One mission of the National Highway Traffic Safety Administration (NHTSA), the U.S. federal agency for road and vehicle safety, is to define consumer protection programs. These programs consist of crash tests that automobiles may have to undergo. The NHTSA is free to choose which model it evaluates. As a rule, frequently sold cars are drawn. The selected car earns points for each test set by the program. These points sum up to an overall score called the safety rating. The current flagship program of the NHTSA is called the U.S. New Car Assessment Program (U.S. NCAP), see [car21]. Depending on the performance, it awards more or less stars. One star stands for the worst and five stars for the best rating.

The U.S. NCAP carries out a frontal, two side, and a rollover test, see [U.S08]. Here, the frontal crash test is considered where the car frontally impacts a rigid barrier with full

width at a speed of 56 km/h, see Fig. 21 (a). Two dummies are placed on the front seats. They measure different signals that are filtered and processed to scalar crash performance values — also referred to as injury criteria. Six injury criteria are used to assess the full frontal test: the head injury criterion (HIC_{15}), see (3.18), as well as the chest deflection (CD), the femur force (FF), neck compression (NC), the neck tension (NT), and the so-called N_{ij}, a quantity extracted and processed from neck signals. HIC_{15} and N_{ij} are unitless. CD is measured in millimeters. FF, NC, plus NT are given in kilonewtons.



(a) (b) $P^{\text{AIS }3+}_{\text{Head}}$-risk curve (c) $P^{\text{AIS }3+}_{\text{Chest}}$-risk curve

Figure 21: Sketch of the U.S. NCAP full frontal test (a); Risk curves for the head injury criterion (b) and the chest deflection (c).

The injury criteria are then passed through medically empirical functions using the Abbreviated Injury Scale (AIS), see [Sta69]. These functions calculate probabilities for injuries of certain severity, e.g. AIS 3+, at specific body regions. For the dummy, a Hybrid III 50th percentile male (HIII 50%) ATD, on the driver's seat, the probability of suffering a head injury is calculated by the risk curve

$$P^{\text{AIS }3+}_{\text{Head}} \coloneqq P^{\text{AIS }3+}_{\text{Head}}(\mathrm{HIC}_{15}) = \Phi\left(\frac{\ln(\mathrm{HIC}_{15}) - 7.45231}{0.73998}\right) \tag{3.138}$$

where Φ is the cumulative distribution function of the standard normal distribution. Similarly, probabilities of chest, femur, and neck injuries are obtained by the functions

$$P^{\text{AIS }3+}_{\text{Chest}} \coloneqq P^{\text{AIS }3+}_{\text{Chest}}(\mathrm{CD}) = \frac{1}{1+\exp\left(10.5456 - 1.568 \cdot \mathrm{CD}^{0.4612}\right)}, \tag{3.139}$$

$$P^{\text{AIS }2+}_{\text{Femur}} \coloneqq P^{\text{AIS }2+}_{\text{Femur}}(\mathrm{FF}) = \frac{1}{1+\exp(5.795 - 0.5196 \cdot \mathrm{FF})}, \tag{3.140}$$

$$P^{\text{AIS }3+}_{\text{Neck}} \coloneqq P^{\text{AIS }3+}_{\text{Neck}}\left(P^1_{\text{Neck}}, P^2_{\text{Neck}}, P^3_{\text{Neck}}\right) = \max(P^1_{\text{Neck}}, P^2_{\text{Neck}}, P^3_{\text{Neck}}) \tag{3.141}$$

where the arguments of $P^{\text{AIS }3+}_{\text{neck}}$ come from

$$P^1_{\text{Neck}} \coloneqq P^1_{\text{Neck}}(\mathrm{NC}) = \frac{1}{1+\exp(10.9745 - 2.375 \cdot \mathrm{NC})}, \tag{3.142}$$

$$P^2_{\text{Neck}} \coloneqq P^2_{\text{Neck}}(\text{NT}) = \frac{1}{1+\exp(10.9745 - 2.375 \cdot \text{NT})}, \tag{3.143}$$

$$P^3_{\text{Neck}} \coloneqq P^3_{\text{Neck}}(\text{N}_{\text{ij}}) = \frac{1}{1+\exp(3.2269 - 1.9688 \cdot \text{N}_{\text{ij}})}, \tag{3.144}$$

see [car21]. Risk curves of the head and the chest injury criteria are depicted in Fig. 21 (b) and (c), respectively.

The joint probability of overall injury is determined by

$$\begin{aligned} P_{\text{Joint}} &\coloneqq P_{\text{Joint}}\left(P^{\text{AIS 3+}}_{\text{Head}}, P^{\text{AIS 3+}}_{\text{Chest}}, P^{\text{AIS 2+}}_{\text{Femur}}, P^{\text{AIS 3+}}_{\text{Neck}}\right) \\ &= 1 - \left(1 - P^{\text{AIS 3+}}_{\text{Head}}\right)\left(1 - P^{\text{AIS 3+}}_{\text{Chest}}\right)\left(1 - P^{\text{AIS 2+}}_{\text{Femur}}\right)\left(1 - P^{\text{AIS 3+}}_{\text{Neck}}\right). \end{aligned} \tag{3.145}$$

The final score for the driver in the considered test called relative risk (RR) is computed via the expression

$$\text{RR} = \frac{P_{\text{Joint}}}{0.15} \tag{3.146}$$

where 0.15 is taken as the baseline risk. RR can then be interpreted as stars. It is now considered as a function of the injury criteria, i.e.

$$\text{RR}\left(\text{HIC}_{15}, \text{CD}, \text{FF}, \text{NC}, \text{NT}, \text{N}_{\text{ij}}\right). \tag{3.147}$$

Exemplarily, the two sensitivity methods are demonstrated on this function. For this purpose, intervals for the injury criteria are specified, see Table 5. Here, the independent injury criteria are assumed to be uniformly distributed.

Table 5: Chosen intervals of the injury criteria for the sensitivity analyses.

Injury criteria	Interval
HIC_{15}	$[250, 400]$
CD	$[15, 28]$
FF	$[3, 5]$
NC	$[2.8, 3.6]$
NT	$[2.5, 3.6]$
N_{ij}	$[0.05, 0.15]$

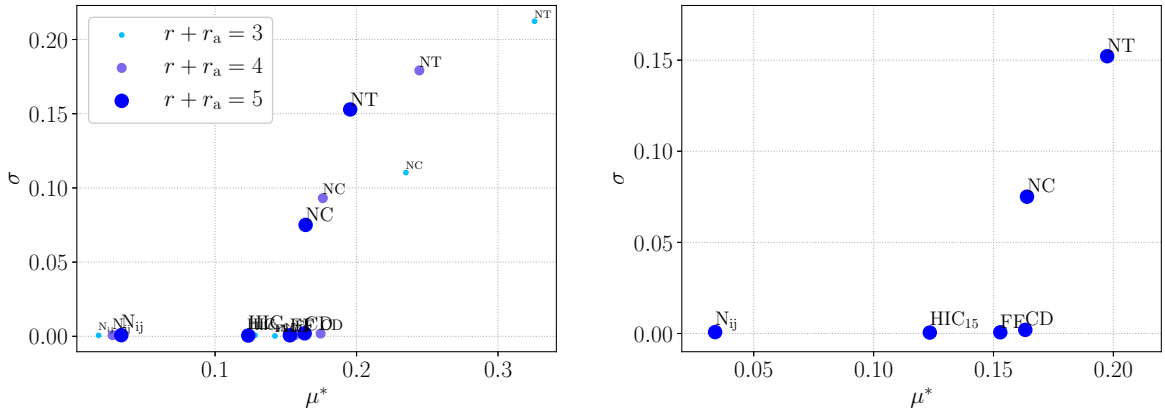
Elementary effects method

The adaptive algorithm of the Morris method is used. There are $n = 6$ parameters — injury criteria. The stopping value is set to $\varepsilon_{\text{EE}} = 0.15$. Algorithm 2 is started with $r = 3$. After two iterations, i.e. at $r_{\text{a}} = 2$, the left-hand side of the formula (3.111) is equal to 0.11740. So, the stopping criterion is already fulfilled. For this, a total of

$$(r + r_{\text{a}})(n + 1) = (3 + 2)(6 + 1) = 35 \tag{3.148}$$

functional evaluations were required.

The results shown in Fig. 22 indicate that NT, NC, CD, FF, and HIC_{15} have all significant influence on the RR value due to the high μ^* scores when the intervals of the injury criteria are defined as in Table 5. These five lead to over 99% of the elementary effects variance according to formula (3.113). So, the missing injury criterion, $\mathrm{N_{ij}}$, is hardly relevant and might be discarded for further analysis. Furthermore, NC and NT have high σ values. These are due to the interactions residing in $P_{\mathrm{Neck}}^{\mathrm{AIS}\,3+}$ (3.141) where the maximum of $P_{\mathrm{Neck}}^{1}(\mathrm{NC}), P_{\mathrm{Neck}}^{2}(\mathrm{NT}), P_{\mathrm{Neck}}^{3}(\mathrm{N_{ij}})$ is taken.



(a) Development of the Morris indices (b) Final Morris indices, i.e. $r + r_a = 5$

Figure 22: Screening results of the RR function from the U.S. NCAP rating show the relevance of injury criteria.

Sobol' indices

Variance-based sensitivity analysis confirms the results of screening, cf. Fig. 22 with 23. The first order and total effect Sobol' indices, see (3.134) and (3.135), were calculated using $n_{\mathrm{Sob}} = 10\,000$. They attest that all injury criteria except $\mathrm{N_{ij}}$ are important for the RR value. The sum of the first order as well as the sum of the total effect indices,

$$\sum_{i=1}^{n} s_i = 0.96259, \quad \sum_{i=1}^{n} s_{\mathrm{T}_i} = 1.03138, \tag{3.149}$$

are different. Thus, the function is non-additive. First order and total effect indices clearly deviate for NC and NT, respectively, cf. σ values of the Morris method. This can again be comprehended by analyzing formula (3.141): the maximum injury probability of the neck quantities is taken to calculate the probability of overall injury as well as the RR value. According to this, NC interacts with NT coinciding with their Sobol' indices.

Note that importance rankings that may be derived are not consistent between Morris and Sobol' indices, cf. the indices for NT in Fig. 22 (b) and Fig. 23. The elementary effects method used a small number of random starting points, i.e. a high stopping value, $\varepsilon_{\mathrm{EE}}$, that cannot yield a reliable ranking. To improve, one would have to increase the number of random starting points. However, the small Morris sample is able to classify whether parameters are relevant or irrelevant. The Sobol' indices calculated based on a large sample are profound measures and can be used to provide a solid ranking.

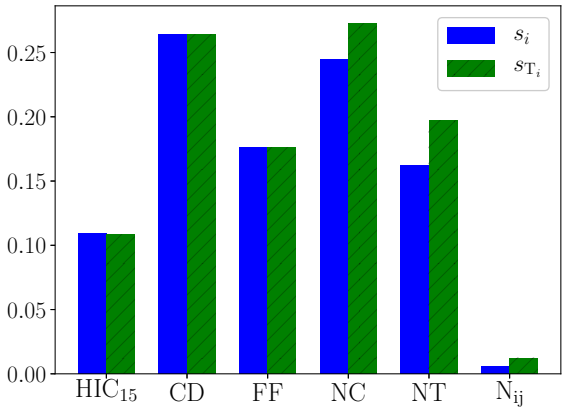


Figure 23: Sobol' indices for the parameters of the RR function with chosen intervals from Table 5.

3.4 || Uncertainty quantification

As discussed in Section 3.3, sensitivity analysis identifies the relevance of each parameter to the considered output. It moreover shows possible interaction effects between parameters. Thus, sensitivity analyses primarily explore the input space. For details about the output, an uncertainty quantification is performed afterwards. Thereby, the characteristics and the constitution of the output are determined. The results of an uncertainty quantification include how likely certain outputs are to occur.

3.4.1 Distinction of forward and backward uncertainty quantification

Uncertainty quantification separates two tasks — forward and backward analyses, see [GHO17]. Forward quantification is also called uncertainty propagation. It sends the input, $\mathscr{X}$, following a fixed probability distribution, $p_{\mathscr{X}}$, e.g. a continuous density function or a discrete probability allocation, through the model, $\mathcal{F}$. The goal is to identify the unknown mean, variance, or probability distribution, $p_{\mathscr{Y}}$, of the output, $\mathscr{Y} = \mathcal{F}(\mathscr{X})$, see [LC09]. For this purpose, the model is evaluated in forward fashion. Parameter distributions must be chosen carefully as inaccuracies falsify results due to error propagation.

Backward or inverse uncertainty analysis intends to solve the opposite problem. The probability distribution, $p_{\mathscr{Y}}$, of the output is known while distributions, $p_{\mathscr{X}}$, of the parameters are objects of desire. It is common to use a Bayesian framework to calibrate these distributions, see [Nag19]. Parameter identification, see [HPR17], and data assimilation, see [LG07], also belong to inverse uncertainty quantification. Backward quantification has to cope with the so-called identifiability issue: different combinations of parameters can produce identical results leading to decision difficulties, see [ACA11]. In addition, forward and backward uncertainty analysis may suffer the curse of dimensionality that may be addressed by the sensitivity analysis methods from Section 3.3.

In this thesis, the problem at hand is to identify the probability distribution of the output given uncertain parameters. Therefore, forward quantification is used to propagate the parameters through the system. This will be done by the Dempster-Shafer evidence theory that is discussed in the next section. As a possible extension of the framework, future research may explore whether backward quantification can be performed after forward analysis to validate results.

3.4.2 Uncertainty propagation through the system via Dempster-Shafer evidence theory

In 1976, the mathematician Shafer further developed the work of Dempster, see [Dem67], to the evidence theory (ET). Other sources also refer to ET as Dempster-Shafer (evidence) theory or as theory of the belief functions. Conventional ET merges data from multiple perspectives into one comprehensive assertion. Each of these perspectives features a probability allocation that is integrated into the calculation. The basic construct of ET can be extended to perform uncertainty propagation, see [NGS04, SPM09, ES09, ALRL10]. Uncertain parameters are herein defined as intervals. The model is evaluated with respect to the uncertainties in a way that lower and upper limits for the cumulative distribution function (CDF) of the output are determined. These limits are called belief and plausibility curve, respectively.

Mathematics behind the evidence theory
The following concept orientates at [JLG21a, JLG21b]. ET can be applied to a general input-output model,

$$y = \mathcal{F}(x) \in \mathbb{R}, \tag{3.150}$$

where $\mathcal{F}$ is the system function for the input $x \in \mathbb{R}^d, d \in \mathbb{N}, d \leq k \leq n$. The dimension, d, can be equal to the initial input dimension, n, see (3.10), if no other method was used before. According to our framework, see Fig. 4 and 5, specific methodologies can reduce this number: if a screening approach was applied a-priori, d can be set equal to k, see (3.114). Should the variance-based sensitivity analysis from Section 3.3.3 identify further irrelevant parameters, d can be even smaller than k.

Each uncertain, independent parameter, x_i, is supposed to lie within a connected interval, $I_i = [a_i, b_i] \subset \mathbb{R}, a_i, b_i \in \mathbb{R}, i \in \{1, \ldots, d\}$. These intervals may cover, for instance, the support of assumed probability density functions of random variables, $\mathscr{X}_i$, corresponding to x_i, see (3.9). Moreover, no further knowledge about $\mathcal{F}$ is required to enable the theory, i.e. $\mathcal{F}$ can also be a black box model. This makes ET applicable for finite element crash simulations. The ET procedure is shown in six steps beginning with an expert assessment of the overall intervals, I_i, $i \in \{1, \ldots, d\}$. This assessment defines the probability density function of each component. Each density function has the form of a step function. The expert assessment is explained in two steps:

1. The intervals, I_i, are divided into $\kappa_i \in \mathbb{N}$ subintervals called focal elements,

$$I_i^j \subset I_i, \qquad j \in \{1, \ldots, \kappa_i\}, \tag{3.151}$$

for each input dimension, $i \in \{1, \ldots, d\}$. Their union establishes a new input set,

$$\hat{I}_i := \bigcup_{j=1}^{\kappa_i} I_i^j \subset I_i, \tag{3.152}$$

for x_i. For each dimension, i.e. an arbitrary but fixed $i \in \{1, \ldots, d\}$, it is possible that these subintervals, $I_i^j, j \in \{1, \ldots, \kappa_i\}$, are overlapping, connected, or disjoint. Note that in the case of disjoint focal elements, the resulting input set, $\hat{I}_i$, can contain gaps, i.e. it can be disconnected.

2. Subsequently, the experts associate the focal elements, I_i^j, with probabilities in a so-called basic probability assignment (BPA). This is done using a BPA operator, m_i, corresponding to x_i that is defined by

$$m_i(J) \begin{cases} \in [0,1], & \text{for } J \in \mathcal{I}_i := \left\{ I_i^j \middle| j \in \{1, \ldots, \kappa_i\} \right\}, \\ = 0, & \text{for } J \in \mathcal{B}(\mathbb{R}) \backslash \mathcal{I}_i, \end{cases} \tag{3.153}$$

where $\mathcal{B}(\mathbb{R})$ is the Borel σ-algebra of the real numbers. A necessary condition of each operator, m_i, is that the sum of probabilities assigned to the focal elements must equal one,

$$\sum_{J \in \mathcal{I}_i} m_i(J) = 1, \qquad i \in \{1, \ldots, d\}. \tag{3.154}$$

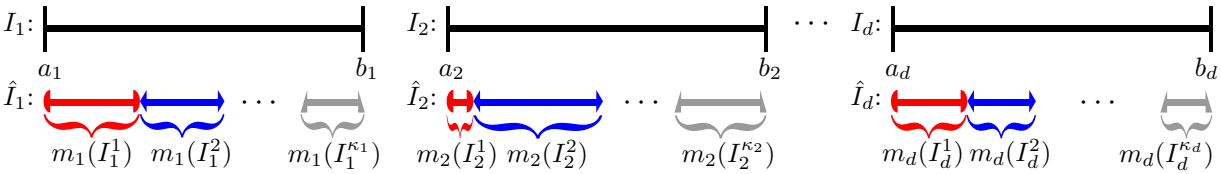


Figure 24: Illustration of the two expert assessment steps.

The expert assessment depicted in Figure 24 is not required to reach unanimity between all experts in the assessment. In case there are differing opinions, they can be combined and weighted. The following procedures, however, stays the same.

$$\begin{aligned} C^1 &= I_1^1 \times I_2^1 \times \cdots \times I_d^1 = \bullet \times \bullet \times \cdots \times \bullet \\ C^2 &= I_1^1 \times I_2^1 \times \cdots \times I_d^2 = \bullet \times \bullet \times \cdots \times \blacklozenge \\ &\vdots \\ C^{\kappa} &= I_1^{\kappa_1} \times I_2^{\kappa_2} \times \cdots \times I_d^{\kappa_d} = \blacktriangle \times \blacktriangle \times \cdots \times \blacktriangle \end{aligned}$$

Figure 25: Processing the focal elements to interval cells.

The uncertainties assessed by the experts, i.e. the created data, are first processed as shown in Figure 25 and then propagated through the model. This procedure is delineated in the next four steps:

3. All possible focal element combinations between input dimensions are formed yielding d-dimensional spaces also known as interval cells. In total $\kappa := \prod_{i=1}^{d} \kappa_i$ interval cells are obtained. These interval cells are denoted as

$$C^{j_1,j_2,\ldots,j_d} := I_1^{j_1} \times I_2^{j_2} \times \cdots \times I_d^{j_d}, \tag{3.155}$$

for $j_i \in \{1, \ldots, \kappa_i\}$ and $i \in \{1, \ldots, d\}$. Instead of writing $C^{j_1,j_2,\ldots,j_d}$, $j_i \in \{1, \ldots, \kappa_i\}$, respectively, the notation is simplified by C^l, $l \in \{1, \ldots, \kappa\}$. This is admissible since there is a one-to-one mapping between $\{1, \ldots, \kappa\}$ and $\prod_{i=1}^{d}\{1, \ldots, \kappa_i\}$. The set of resulting interval cells is henceforth declared as $\mathcal{C}$.

4. For each C^l, the composite BPA, p^l, is calculated by

$$p^l := p^{j_1,\ldots,j_d} := \prod_{i=1}^{d} m_i\left(I_i^{j_i}\right). \tag{3.156}$$

exploiting the independence of probability variables, $\mathscr{X}_i, i \in \{1, \ldots, d\}$. Basically, the single BPAs belonging to the focal elements that define the interval cell, C^l, are multiplied together. Adding up all composite BPAs, p^l, must equal one, i.e.

$$\sum_{l=1}^{\kappa} p^l = 1. \tag{3.157}$$

5. Lower and upper limits for the CDF of the output are found by propagating every interval cell, C^l, through the model. In mathematical terms, two optimization problems,

$$\overline{y}^l = \sup_{x \in C^l} \mathcal{F}(x), \tag{3.158}$$

$$\underline{y}^l = \inf_{x \in C^l} \mathcal{F}(x), \tag{3.159}$$

must be solved for each interval cell — that makes 2κ optimization problems in total. Note that there are also other possibilities to obtain $\overline{y}^l$ and $\underline{y}^l$, e.g. by applying sampling strategies, see [HJS06, TdWL+18]. Their accuracy strongly depends on the size of the sample.

6. As final step, the ET curves are formulated. The belief curve reads

$$\mathrm{Bel}(y) = \sum_{\left\{l \middle| y \geq \overline{y}^l\right\}} p^l. \tag{3.160}$$

The plausibility curve is defined as

$$\mathrm{Pl}(y) = \sum_{\left\{l \middle| y \geq \underline{y}^l\right\}} p^l. \tag{3.161}$$

The formulas contain the supremum and infimum values, $\overline{y}^l$ and $\underline{y}^l$, as well as their composite BPAs, p^l. Practically, one sorts the supremum and infimum values in ascending order, respectively. Then, their probabilities are cumulated. This produces CDF-like curves which can serve as limits of the unknown CDF of the output. To provide a probabilistic interpretation, note that $y = \mathcal{F}(x)$ is a realization of a random variable, $\mathscr{Y}$, since x is the one of $\mathscr{X}$. For an arbitrary but fixed $\tilde{y}$, the belief curve is viewed to be a lower limit of the CDF of $\mathscr{Y}$, i.e. it holds

$$\mathrm{Bel}(\tilde{y}) \leq \mathscr{P}(\mathscr{Y} \leq \tilde{y}), \tag{3.162}$$

where $\mathscr{P}(A)$ symbolizes the probability of event A. Analogously, the plausibility curve is taken as an upper limit, i.e.

$$\mathscr{P}(\mathscr{Y} \leq \tilde{y}) \leq \mathrm{Pl}(\tilde{y}). \tag{3.163}$$

A bottleneck of ET may arise if the number of parameters or focal elements is too high. Because then, there are numerous interval cells and accordingly many optimization problems that need to be solved. This can cause large computational efforts. For example, the first application of Section 3.2.7 has $n = 36$ parameters. In [JLG21b], where the same example is considered, it is discussed that $2\kappa = 2 \cdot 3^{36} > 10^{17}$ optimization problems would have to be solved in case each parameter interval is divided into three subintervals. This huge amount of data would have to be stored. Therefore, our framework in Fig. 4 suggests to narrow down parameters by sensitivity analysis. This can break the bottleneck.

Further, the use of ET becomes prohibitively expensive when optimizing the system function is costly, e.g. owing to nonlinearities or its black-box nature. This can be overcome by applying metamodels — see Section 3.2 — to approximate the system function and thus accelerate the optimization. Other works also take these remedies, see the multipoint approximation method in [BGC04], stochastic expansion methods in [EST11], the model order reduction approach in [JLG21a], or kriging in [JLG21b] for instance.

Differing expert assessments

The parameter intervals, I_i, can be associated with uncertainty by several experts. Their opinions may vary as sources can be diverse, e.g. assessments can come from human knowledge, existing related data or information, etc. Thus, different BPAs can occur for one parameter. These BPAs can be bundled for each parameter. Greater weight can be given to more trustworthy sources. There exist different methods to combine BPAs, see [Lim08]. Common approaches are Dempster's rule of combination or Yager's rule. According to [Lim08], the method of weighted mixture is supposed to be even more robust than the other two.

Reasons for using the evidence theory

Among multiple uncertainty propagation methods, see Section 1.2, the choice fell on the Dempster-Shafer evidence theory. This is due to diverse reasons. ET is flexible regarding uncertainty assignment. It allows different characteristics to be tested, e.g. fine or rough division into subintervals. The assignment is designed in a simple and comprehensible way.

Most uncertainty allocations in the field of crash are based on expert opinions as there are no standard universally accepted classifications for the parameters to be considered. ET offers the ideal platform for this.

Although intended for epistemic uncertainties, ET can be used for aleatory uncertainties enabling mixed uncertainty propagation, see [ES09, EST11]. Probability density functions of aleatory variables can therefore be sufficiently well approximated by histograms. These histograms can be converted to ET structures, see [JLG21a].

Assessments can differ between engineers or other sources. Therefore, ET is capable of weighing these opinions and propagating the resulting summarized uncertainties through the system — another plus point in terms of flexibility. Besides, the outcome of ET, a corridor for the CDF of the output, is easy to interpret. Based on this visualization, decisions can be made, e.g. whether revising or keeping the current design. In summary, by enabling the evidence theory through the proposed framework in Fig. 5 with its single components, the research question from Section 1.2 is successfully answered, especially for finite element crash simulations.

3.4.3 Exemplary usage of evidence theory

The U.S. NCAP rating from Section 3.3.4 is addressed here again to demonstrate the procedure of ET. Therefore, the analytical relative risk function, RR (3.147), that considers the injury criteria as arguments is taken as the system function, $\mathcal{F}$, of ET. The six steps of ET from Section 3.4.2 are passed through. A fictive expert makes the uncertainty assessment, i.e. BPAs, for the injury criteria, see Fig. 26.

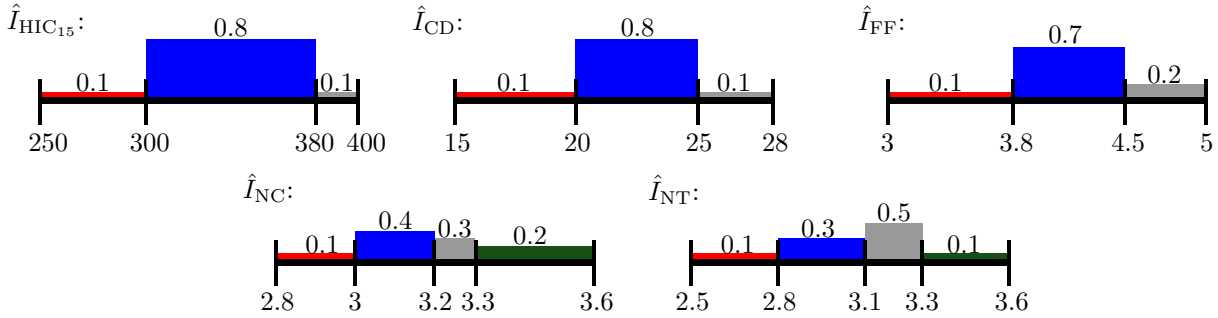


Figure 26: Fictive BPA allocations of the five injury criteria, the arguments of the U.S. NCAP RR function, that contribute in the considered constellation.

First, the expert decides to use the same ranges, $\hat{I}_{\mathrm{HIC}_{15}}, \hat{I}_{\mathrm{CD}}, \hat{I}_{\mathrm{FF}}, \hat{I}_{\mathrm{NC}}, \hat{I}_{\mathrm{NT}}$, for the injury criteria as considered in Table 5 of Section 3.3.4 when sensitivity analyses were applied. The expert knows the results of the sensitivity analyses. Therefore, he does not consider $\mathrm{N_{ij}}$ as parameter for uncertainty propagation as $\mathrm{N_{ij}}$ hardly influences the rating, see the results of Section 3.3.4. Hence, $\mathrm{N_{ij}}$ is fixed to its highest, i.e. most penetrating, interval value — equal to 0.15. The expert divides three of the remaining five intervals into three subintervals, respectively. The other two are segregated into four subintervals each.

In total $3^3 \cdot 4^2 = 432$ interval cells are created, i.e. $2 \cdot 432 = 864$ optimization problems must be solved. The optimization for this function is straightforward. Since all probability

functions of injuries are monotone, the joint probability of overall injuries, P_{Joint}, and the RR function are also monotone. On this account, each minimum value of RR is located on the lower limits of the subintervals, respectively. On the contrary, each maximum value can be found at the upper limits of the subintervals, respectively.

The result of the uncertainty propagation via ET shown in Fig. 27 tells that three discrete rating results are possible — in particular, three, four, or five stars. The regions of the stars are separated by gray solid vertical lines. ET provides two curves, the plausibility and the belief curve, from which probabilities for specific results can be derived. In this application, the plausibility curve is the best case curve whereas the belief curve represents the worst case. The actual CDF of the relative risk lies between the two.

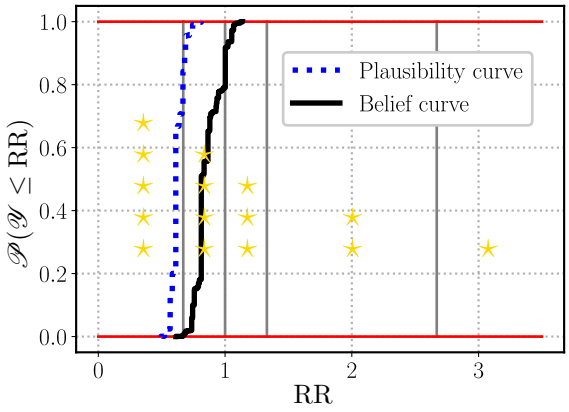


Figure 27: ET results of RR for the BPA allocations from Fig. 26.

The plausibility curve, Pl (3.161), indicates that it is very likely or rather plausible — over 84% — that five stars will be achieved. Four stars are reached even to 100% according to Pl. On the contrary, the belief curve, Bel (3.160), warns engineers. It says that it is only less than 0.6% believable that five stars will be earned. If the goal is to gain these five stars, engineers should think about revising the current design to remove probabilities of ending up in the three or four star region. They can also take the risk and be rewarded with at least four stars if the actual CDF is close to Pl. In this case, however, they should not be surprised if they fall back to three stars when the CDF tends toward Bel since the belief curve can only exclude three stars by about 79%. So, according to the few remaining percentages, it could also come to three stars.

Another practical application where ET is combined with an model order reduction metamodel can be found in [JLG21a]. There, the author of this thesis and his colleagues consider the maximum deformation of a crashbox as uncertain due to epistemic and aleatory parameters, cf. the fourth application of Section 3.2.7. The belief and the plausibility curve are then calculated to limit the actual CDF of the maximum crashbox deformation. Furthermore, ET bounds for the CDF of the rib deflection of the simplified dummy are established in [JLG21b], cf. the first application of Section 3.2.7. There, the importance of metamodels for finite element simulations is emphasized.

Application of the framework to a real-world project

4.1 Introduction of the real-world project

The framework is now demonstrated on a real world application. Uncertainties of a full vehicle crash simulation are therefore defined and analyzed. The finite element model simulates a side pole test — that is part of the European New Car Assessment Programme (Euro NCAP), the customer protection of Europe for vehicle safety — of a state-of-the-art car. This LS-DYNA simulation is calculated on 168 CPUs via high performance computing. The model is not simplified, i.e. it remains a full vehicle simulation. Each simulation takes over 32 hours to be finished. This large simulation was intentionally chosen to challenge the framework to the greatest extent. However, engineers may be able to reduce the computation time through sophisticated simplifications, geometry revisions, etc. The same holds for the parameters. Their ranges are extended to yield more variability. The considered quantities therefore differ from actual development magnitudes.

In the side pole test, the vehicle is pushed sideways at 32 km/h against a fixed, narrow pole, see [Uni16]. The vehicle is positioned at an angle of 75 degrees to the pole. These conditions impose high demands on the vehicle. Since the impact acts only on a limited area of the vehicle, large deformation forces are generated. This may allow the pole to penetrate deep into the interior. Engineers must therefore prescribe precise controls of passive safety concepts to best protect occupants. The Euro NCAP program places a WorldSID 50th percentile adult male ATC (WS 50%) on the front seat to assess the occupant safety of vehicles, see Fig. 28.

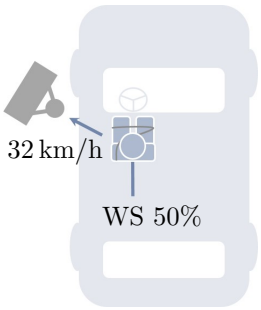


Figure 28: Drawing of the Euro NCAP side pole test.

Table 6: Limits for the Euro NCAP rating functions, see [car21].

Injury criteria	Lower limit l_{IC}	Upper limit u_{IC}
HIC_{15} (-)	500	700
HA_{3ms} (g)	72	80
CD (mm)	28	50
AD (mm)	47	65
PF (kN)	1.7	2.8

The side pole test is rated by points. For different body segments, injury criteria (IC) are defined similar to the U.S. NCAP rating, see Section 3.3.4. The impacts on the

head are expressed via the HIC_{15} (3.18) and the so-called 3 ms value ($\mathrm{HA}_{3\mathrm{ms}}$). The latter declares the maximum acceleration measured in multiples of the gravitational acceleration that lasts for at least 3 ms, cf. Section 3.2.7. The maximum chest deflection (CD), the maximum abdomen deflection (AD), and the maximum pubic force (PF) are quantities symbolizing the dummy torso from top to bottom. Sensors are installed on the dummy that record the time histories of these quantities. For the rating, the maximum of these time histories are taken, respectively.

The measured values of the ICs are processed to rating points. The rating functions for the ICs are all of the form

$$\mathrm{SS}_{\mathrm{IC}}(\zeta) = \begin{cases} 4, & \text{for } \zeta < l_{\mathrm{IC}}, \\ \frac{4(\zeta - u_{\mathrm{IC}})}{l_{\mathrm{IC}} - u_{\mathrm{IC}}}, & \text{for } l_{\mathrm{IC}} \leq \zeta \leq u_{\mathrm{IC}}, \\ 0, & \text{for } \zeta > u_{\mathrm{IC}}, \end{cases} \tag{4.1}$$

where ζ is the value for the chosen IC. Due to its shape, the function is also referred to as sliding scale (SS) system, see [Eur20]. Fig. 29 depicts the sliding form for the injury criterion CD. The values for the thresholds, l_{IC} and u_{IC}, are specific for each IC and can be found in Table 6. The highest score for each IC is to obtain four points. The lowest score is zero points. The smaller the value of the IC is, the better score is yielded.

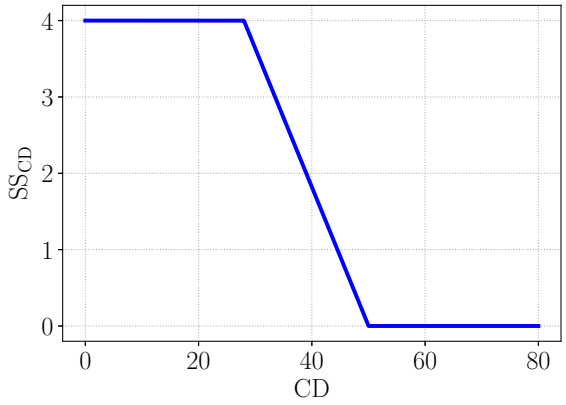


Figure 29: Sliding scale ($\mathrm{SS}_{\mathrm{CD}}$) function for the chest deflection (CD).

The points coming from these functions are summed up to an overall rating. For the deflection ICs — AD and CD — there is a peculiarity. AD is measured at two locations, the lower and the upper rib of the abdomen, AD_{l} and AD_{u}. The higher value, $\max\{\mathrm{AD}_{\mathrm{l}}, \mathrm{AD}_{\mathrm{u}}\}$, of both is then used for the corresponding rating function. For CD, it is analogous except that there exist three locations, the lower, middle, and the upper thorax rib deflection, i.e. CD_{l}, CD_{m}, and CD_{u}. Besides these points for the injury criteria, the overall rating can lose score if so-called modifiers are violated, e.g. the airbag deploys incorrect, the door opens, etc. These modifiers are neglected here. Indeed, they are not infringed in this model.

Values for the injury criteria are obtained by evaluating the full vehicle simulation for the side pole impact. Due to uncertainty, the resulting values are scattered. Here,

$n = 9$ uncertain parameters are examined. They involve uncertainties of both classes. Aleatory irregularities are e.g. caused by deviating test executions, see Fig. 3. In this study, aleatory factors are the varying barrier position, the differing velocity, diverging car mass and mass ratio, and the scattering impact angle. Next to them, epistemic uncertainties are considered. Engineers need to calibrate different design specification. Here, the firing times of four restraint systems are investigated: the time-to-fire (TTF) of the curtain airbag, the side airbag, the load limiter, and the belt retractor. Table 7 shows an overview of the parameters along with their properties such as ranges, category, etc.

Table 7: Uncertain input parameters for the full vehicle finite element model of the side pole test.

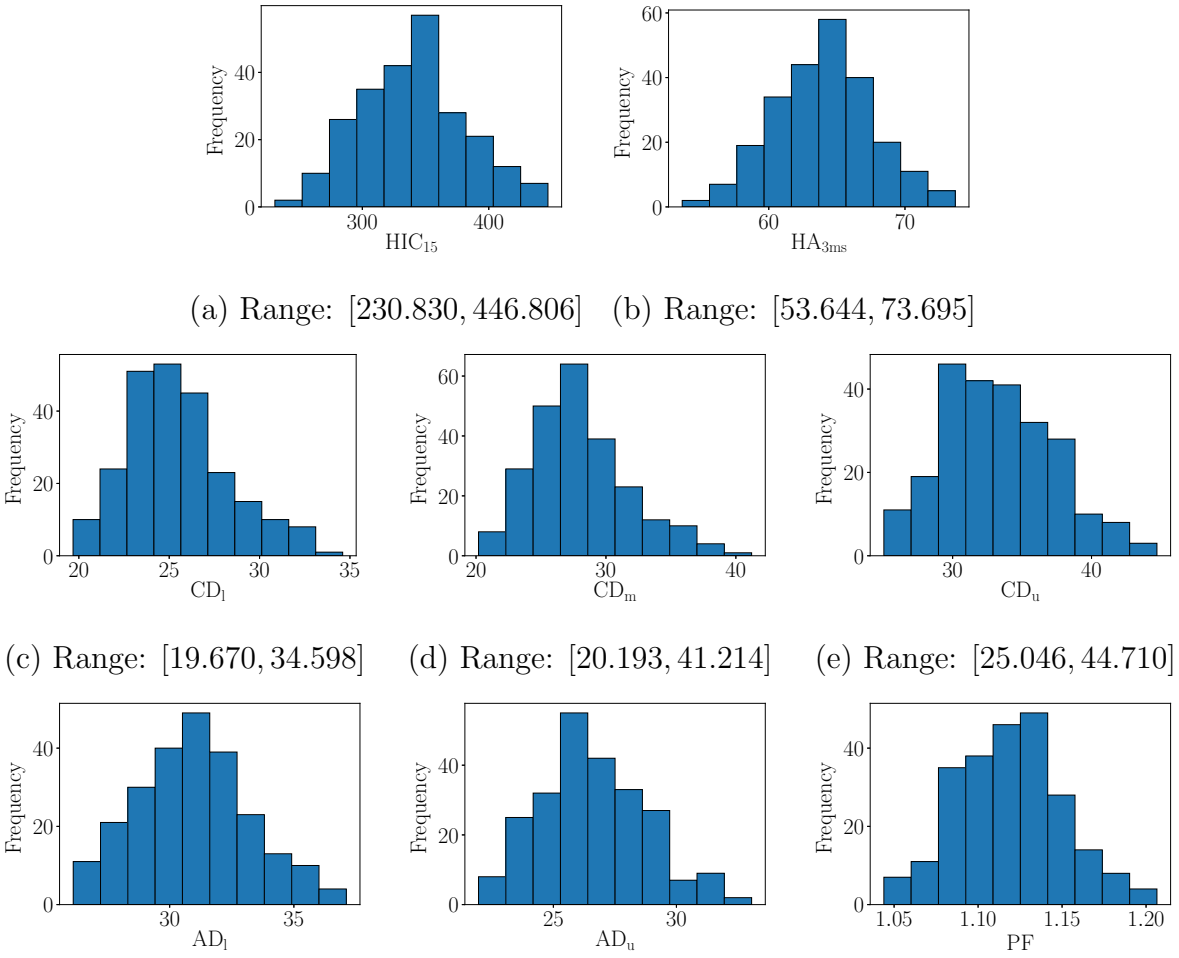
Parameter	Description	Category	Nominal value	Range	Unit
x_1	Delta barrier x-position	Aleatory	0	$[-15, 15]$	mm
x_2	Delta velocity	Aleatory	0	$[-1, 1]$	km/h
x_3	Delta mass	Aleatory	0	$[-25, 25]$	kg
x_4	Delta mass ratio	Aleatory	0	$[0, 0.02]$	/
x_5	TTF curtain airbag	Epistemic	8	$[5, 10]$	ms
x_6	TTF side airbag	Epistemic	8	$[5, 10]$	ms
x_7	TTF belt load limiter	Epistemic	8	$[5, 10]$	ms
x_8	TTF belt retractor	Epistemic	8	$[5, 10]$	ms
x_9	Delta impact angle	Aleatory	0	$[-1, 1]$	°

Thanks to defining the parameters, the simulation can be regarded as black box functions of different complexity, $\mathcal{F}_{\text{EN}}, \mathcal{F}_{\text{CN}}, \mathcal{F}_{\text{SH}}, \mathcal{F}_{\text{KR}}$, see Section 3.1.2. Particularly, the Euro NCAP asks for the injury criteria, i.e. scalar key results, that can be modeled by $\mathcal{F}_{\text{KR}}$. Note that Section 3.1.2 covers how the different black box functions relate to each other, i.e. key results may be extracted from the outputs of $\mathcal{F}_{\text{EN}}, \mathcal{F}_{\text{CN}}$, and $\mathcal{F}_{\text{SH}}$. For the FE simulation and consequently these mappings, the time is discretized into $N_{\text{T}} = 1200$ time steps. To accelerate the slow FE simulations, metamodels are used as approximations. For their creation, training data must be generated. Therefore, a total of 240 simulations were run. Their parameter values were generated by Sobol' sequence sampling. Practically, this was done by using Python's library SALib [HU17].

Before the framework components are applied, the results of the 240 simulations are reviewed. Since the points of the Sobol' sequence sample span the entire input space uniformly, rough ranges of the key results, i.e. injury criteria, as well as their approximate distributions in form of histograms can be analyzed, see Fig. 30. A first observation states that there is clear uncertainty in each IC caused by the parameters. Nevertheless, for some ICs, it is good-natured scattering. That is, uncertainty is present but it causes relatively minor harm to the dummy. This can be deduced from the safety rating that, among others, is used by engineers and the consumer protection authority to assess systems.

For this reason, before starting the planned analysis, it is now determined how many points are lost for single ICs. The bounds for the simulated key results from Fig. 30 are thus compared with the limits of the sliding scale system from Table 6 as between these limits the change in points is made out. Then, from that, the resulting point range of the

safety rating can be interpreted. The results from Table 8 indicate that only two types of ICs lead to deduction in rating scores: $\mathrm{HA_{3ms}}$ and the chest ICs, CD_i, $i \in \{\mathrm{l}, \mathrm{m}, \mathrm{u}\}$.



(a) Range: $[230.830, 446.806]$ (b) Range: $[53.644, 73.695]$

(c) Range: $[19.670, 34.598]$ (d) Range: $[20.193, 41.214]$ (e) Range: $[25.046, 44.710]$

(f) Range: $[26.117, 37.111]$ (g) Range: $[21.960, 33.057]$ (h) Range: $[1.044, 1.206]$

Figure 30: Histograms of the ICs ((a), (b): head-related IC; (c), (d), (e): chest-related ICs; (f), (g), (h): abdomen- and pubic-related ICs). All histograms are constituted of ten equal-width bins. The ranges they cover are given under the respective plots. Units can be found in Table 6.

According to Table 8, $\mathrm{CD_u}$ can cause the biggest possible loss regarding points followed by $\mathrm{CD_m}$ and $\mathrm{CD_l}$. Recall that these ICs are combined to one point supplier, $\mathrm{CD} = \max\{\mathrm{CD_l}, \mathrm{CD_m}, \mathrm{CD_u}\}$. Although $\mathrm{CD_u}$ shows the highest values, all three should be considered as, for instance, it could happen that it holds $\mathrm{CD_u} < \mathrm{CD}_i, i \in \{\mathrm{m}, \mathrm{l}\}$, for some parameter constellation. In the following, $\mathrm{CD_l}$, $\mathrm{CD_m}$, and $\mathrm{CD_u}$ are thus analyzed in more detail and subjected to the framework. The smaller uncertainty caused by $\mathrm{HA_{3ms}}$ is not studied further. Theoretically, however, it can be studied in the same way — this holds for the other ICs as well.

Note that the findings established by Table 8 may also be obtained by e.g. calculation of the Spearman's rank correlation coefficients, see [Dod08]. The function to be observed

may be defined as a merged rating function that sums up the single SS_{IC} functions (4.1). The Spearman's rank correlation can be applied as this function would be monotone. Correlation coefficients between an arbitrary but fixed IC and the corresponding output of the merged rating function could be calculated. Coefficients close to 1 or -1 indicate that the correlation is high, i.e. the considered IC changes the outcome and should be considered for further analysis. Coefficients near zero suggest the opposite. However, Table 8 is given here because it remained manageable and overseeable with the few ICs.

Table 8: Comparison of the obtained bounds for the simulated key results and the limits of the sliding scale system.

Injury criteria	Obtained IC bounds ∩ Limits SS_{IC}	Point range
HIC_{15} (-)	$[230.830, 446.806] \cap [500, 700] = \{\}$	$\{4\}$
HA_{3ms} (g)	$[53.644, 73.695] \cap [72, 80] = [72, 73.695]$	$[3.153, 4]$
CD_u (mm)	$[25.046, 44.710] \cap [28, 50] = [28, 44.710]$	$[0.962, 4]$
CD_m (mm)	$[20.193, 41.214] \cap [28, 50] = [28, 41.214]$	$[1.597, 4]$
CD_l (mm)	$[19.670, 34.598] \cap [28, 50] = [28, 34.598]$	$[2.800, 4]$
AD_u (mm)	$[21.960, 33.057] \cap [47, 65] = \{\}$	$\{4\}$
AD_l (mm)	$[26.117, 37.111] \cap [47, 65] = \{\}$	$\{4\}$
PF (kN)	$[1.044, 1.206] \cap [1.7, 2.8] = \{\}$	$\{4\}$

4.2 | Metamodeling

As in Section 3.2.7, different metamodels are trained, validated, and compared. Therefore, the 240 simulations are divided into a training sample of size $n_{\text{train}} = 200$ and a validation sample of size $n_{\text{val}} = 40$. The considered key results, CD_l, CD_m, and CD_u, are not straightforward regarding their postprocessing mapping, g (3.14). This postprocessing is performed in black box manner by commercial software code. Therefore, the finite element simulation is only considered in two forms: as history function, $\mathcal{F}_{SH}$ (3.16), and as key result function, $\mathcal{F}_{KR}$ (3.19). Consequently, two types of metamodels — scalar metamodels, $\mathcal{M}_{KR}$ (3.36), and multi-target regression metamodels, $\mathcal{M}_{SH}$ (3.46), from Sections 3.2.3 and 3.2.4 — can be utilized. These metamodels are and are not combined with the elementary effects method from Section 3.3.2.

For simplicity, the key result notations

$$y = CD \in \mathbb{R} \tag{4.2}$$

for the Euro NCAP point supplier,

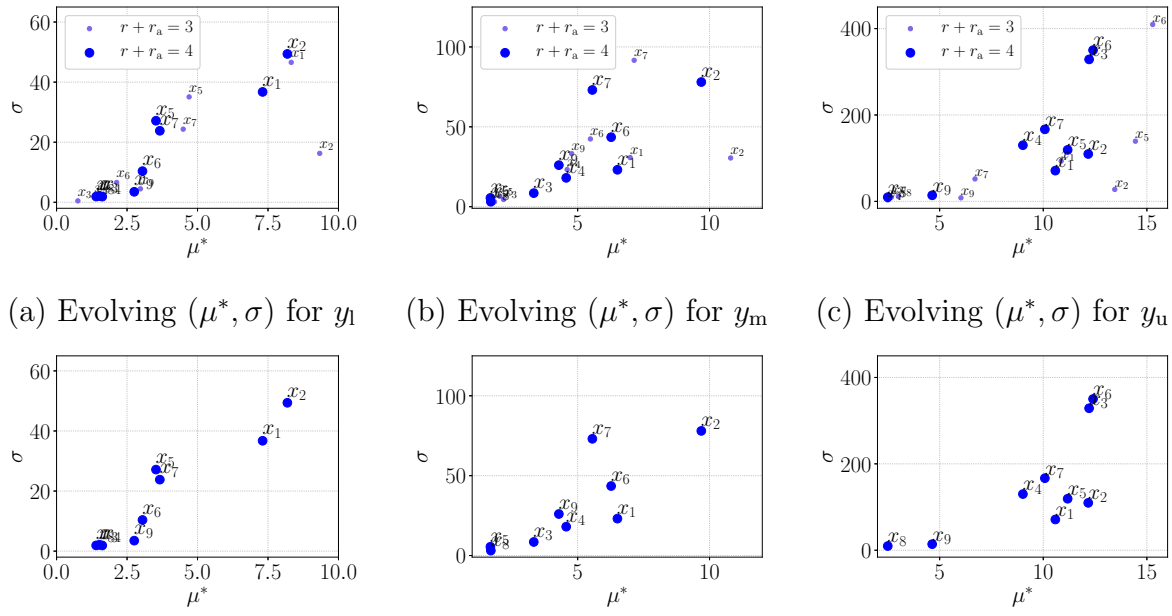
$$y_l = CD_l \in \mathbb{R}, \quad y_m = CD_m \in \mathbb{R}, \quad y_u = CD_u \in \mathbb{R} \tag{4.3}$$

for the lower, middle, and upper maximal chest deflections, respectively, are introduced. From now on, the time histories of these key results are denoted as $y_L, y_M, y_U \in \mathbb{R}^{1 \times N_T}$, respectively. These quantities have the relationship

$$y = \max\{y_l, y_m, y_u\} = \max\{\max\{y_L\}, \max\{y_M\}, \max\{y_U\}\}. \tag{4.4}$$

4.2.1 Elementary effects method

The number of simulations required for screening are subtracted from the $n_{\text{train}} = 200$ training points for subsequent metamodel creation. To keep the budget low, the radial design-based elementary effects (EE) method is executed for overall 4 random starting points, i.e. the adaptive algorithm, see Section 3.3.2, is only run for $r_a = 1$ iteration ignoring the stopping criterion. In total, $(r + r_a)(n + 1) = 4 \cdot 10 = 40$ simulations are needed. This leaves 160 points for the creation of metamodels that are combined with the EE method. As the method here is meant to mitigate the curse of dimensionality and to boost metamodels, it must find clearly irrelevant parameters instead of providing e.g. an importance ranking that it is also capable of. Therefore, $r+r_a = 4$ is considered sufficient. Results of the EE method for $y_{\text{l}}, y_{\text{m}}$, and y_{u} are illustrated in Fig. 31, respectively.



(a) Evolving (μ^*, σ) for y_{l} (b) Evolving (μ^*, σ) for y_{m} (c) Evolving (μ^*, σ) for y_{u}

(d) Final (μ^*, σ) for y_{l} (e) Final (μ^*, σ) for y_{m} (f) Final (μ^*, σ) for y_{u}

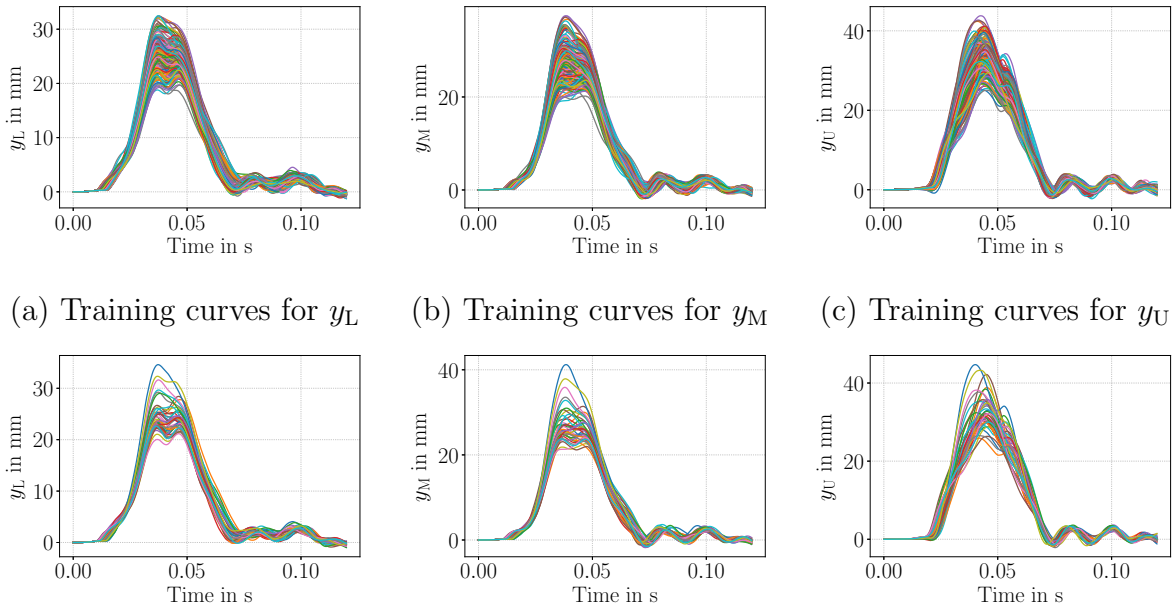
Figure 31: Screening results for the key results y_{l} ((a) and (d)), y_{m} ((b) and (e)), and y_{u} ((c) and (f)). The top row shows the development of the Morris indices while the bottom row shows the final indices for $r + r_a = 4$.

The Morris plots in Fig. 31 and consequently the screening results are different between the three key results, $y_{\text{l}}, y_{\text{m}}$, and y_{u}. Each key result is therefore assessed separately. For y_{l}, Fig. 31 (d) suggests to keep the parameters x_1, x_2, x_5, x_6, x_7. Formula (3.113) states that a percentage greater than 94% of the elementary effects variance is preserved using these $k = 5$ parameters which is considered sufficient. The metamodels for y_{l} coupled with screening thus operate with x_1, x_2, x_5, x_6, x_7. For y_{m}, screening, cf. Fig. 31 (e), identifies $x_1, x_2, x_4, x_6, x_7, x_9$, i.e. $k = 6$, maintaining 93% of the elementary effects variance. Finally, 98% are obtained when taking $x_1, x_2, x_3, x_4, x_5, x_6, x_7$, i.e. $k = 7$, into account for y_{u}, cf. Fig. 31 (f). In particular, the parameters x_1, x_2, x_6, x_7 seem to be relevant for each key result according to the shallow EE method being applied.

4.2.2 Scalar and multi-target regression metamodels

Several metamodels are created and compared. The main settings for the metamodels to be tested are the same as used in Section 3.2.7: anisotropic Matérn 3/2 kernels for Gaussian process regressions (GPR) and Legendre polynomial basis functions with maximal degree, $P \in \mathbb{N}$, chosen by the common truncation set (3.93) for polynomial chaos expansions (PCE). Nonetheless, individual configurations are provided specially, e.g. for scalar or multi-target neural networks (SNN, MNN). The abbreviations for the metamodels are defined in the same way as in Section 3.2.7 except for neural networks that are additionally numbered serially starting with one due to different architectures. Likewise, the numerical implementation is realized with the same routines from there.

Scalar metamodels and multi-target regression metamodels, see Sections 3.2.3 and 3.2.4, are trained with $n_{\text{train}} = 200$ Sobol' sequence sampling points. When they are combined with the EE method from Section 4.2.1, the training data shrinks to a sample of 160 points for a fair comparison. These metamodels can be recognized by their prefix, EE. The histogram of the three key results, $y_{\text{l}}, y_{\text{m}}$, and y_{u}, that scalar metamodels work with can be found, among others, in Fig. 30. The curves, $y_{\text{L}}, y_{\text{M}}$, and y_{U}, that the multi-target regression metamodels have to face are depicted in Fig. 32 — divided into $n_{\text{train}} = 200$ training curves and $n_{\text{val}} = 40$ validation curves.



(a) Training curves for y_{L} (b) Training curves for y_{M} (c) Training curves for y_{U}

(d) Validation curves for y_{L} (e) Validation curves for y_{M} (f) Validation curves for y_{U}

Figure 32: Time histories of the thorax rib deflections, y_{L} ((a) and (d)), y_{M} ((b) and (e)), and y_{U} ((c) and (f)), are illustrated. The top row shows all $n_{\text{train}} = 200$ training curves while the bottom row depicts the $n_{\text{val}} = 40$ validation curves.

Table 9 shows a comparison of results produced by different metamodels. All chosen metamodels, are run with and without the EE method. In combination with the EE method, the metamodels operate on less — 5, 6, 7 — parameters depending on the

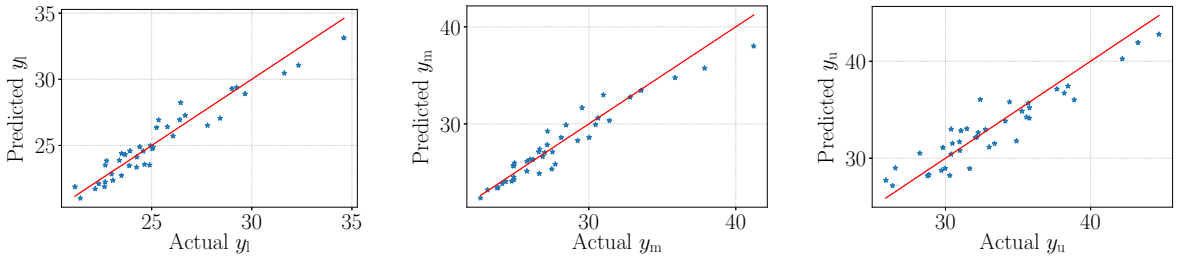
screening results for $y_\mathrm{l}, y_\mathrm{m}$, and y_u, respectively, see Section 4.2.1. As scalar metamodels, GPR, PCE with different maximal degrees, and SNN models are utilized. SNN-1-100 and EE-SNN-1-100 consist of one hidden layer with 512 neurons and the φ_tanh activation function. They are trained for 100 epochs characterized by their suffixes. SNN-2-120 and EE-SNN-2-120 comprise three hidden layers with 32, 64, 32 units from left to the right that are activated by φ_relu. SNN-3-60 and EE-SNN-3-60 are composed of five hidden layers made up of 32, 64, 128, 64, 32 neurons and φ_relu activation functions.

Table 9: Metamodel qualities for the key results $y_\mathrm{l}, y_\mathrm{m}$, and y_u. The mean squared errors (MSE) assess the approximations of the whole curves, i.e. the single histories (SH). The coefficients of determination (R^2) are calculated on the scalar key results (KR).

Approach	y_L-MSE_SH	y_l-R^2_KR	y_M-MSE_SH	y_m-R^2_KR	y_U-MSE_SH	y_u-R^2_KR
GPR	-	0.88420	-	0.87875	-	0.74825
PCE-1	-	0.90145	-	0.89285	-	0.82267
PCE-2	-	0.89532	-	0.90799	-	0.78179
PCE-3	-	< 0	-	0.37287	-	< 0
EE-GPR	-	0.80089	-	0.87032	-	0.60736
EE-PCE-1	-	0.89763	-	0.89664	-	0.80581
EE-PCE-2	-	0.91353	-	0.91669	-	0.76889
EE-PCE-3	-	0.89053	-	0.88370	-	< 0
SNN-1-100	-	0.90547	-	0.89696	-	0.82481
SNN-2-120	-	0.90829	-	0.91132	-	0.84907
SNN-3-60	-	0.89859	-	0.89353	-	**0.86211**
EE-SNN-1-100	-	0.89968	-	0.89880	-	0.80725
EE-SNN-2-120	-	0.88815	-	0.89635	-	0.77994
EE-SNN-3-60	-	0.89042	-	0.91201	-	0.77392
ST-GPR	1.33958	0.38244	1.71177	0.40355	2.57741	0.54306
EE-ST-GPR	1.15879	0.798212	0.94973	0.85808	2.88671	0.50491
ST-PCE-1	0.51037	0.90385	0.54651	0.90247	0.91411	0.83289
ST-PCE-2	0.57228	0.89347	0.61764	0.91287	1.08959	0.78494
ST-PCE-3	24.09076	< 0	35.08040	< 0	62.92448	< 0
EE-ST-PCE-1	0.54181	0.89865	0.56599	0.90364	1.01265	0.81531
EE-ST-PCE-2	0.51926	0.91201	0.56016	0.91507	1.19627	0.77605
EE-ST-PCE-3	0.71400	0.86909	0.88064	0.88258	7.69683	< 0
MNN-1-290	**0.49168**	**0.91988**	**0.53185**	**0.91702**	0.94979	0.82507
MNN-2-110	0.50260	0.91070	0.53429	0.90838	**0.90791**	0.83791
MNN-3-140	0.50524	0.91225	0.53641	0.90780	0.90911	0.83587
EE-MNN-1-290	0.51220	0.90231	0.54115	0.90782	1.07546	0.80652
EE-MNN-2-110	0.53792	0.90675	0.56227	0.90932	1.00814	0.82382
EE-MNN-3-140	0.53662	0.90412	0.56157	0.90789	1.00670	0.82388

Multi-target regression metamodels examined are single-target methods with GPR and PCE (ST-GPR, ST-PCE) and multi-target feed-forward neural networks (MNN).

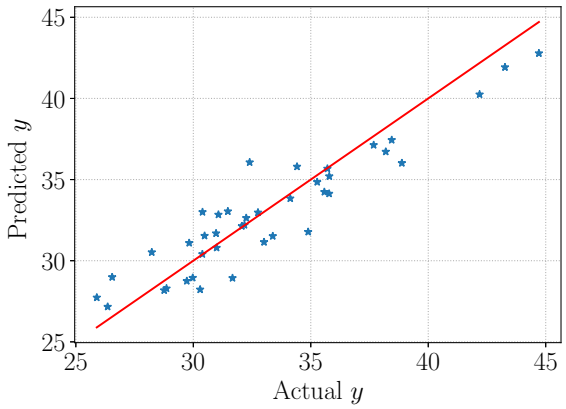
Three MNNs are trained. MNN-1-290 and EE-MNN-1-290 hold one hidden layer with 256 neurons. They use φ_{relu} as activation functions. MNN-2-110 and EE-MNN-2-110 assemble three hidden layer each featuring 128 neurons that are activated by φ_{id} activation functions. MNN-3-140 and EE-MNN-3-140 also include three hidden layer with φ_{id} activation functions but they are equipped with 32, 64, and 128 neurons from left to right. The strategies of the scalar and the multi-target regression metamodels to approximate the key results are different. Scalar metamodels directly predict the maximum values of the corresponding time histories. So, purely theoretically, they must indirectly anticipate the times of the maxima. Multi-target regression metamodels recreate the time histories and do not actually care about the times of the maxima. They are in fact read off afterwards. Thus, one might think that the role of multi-target regression metamodels is simpler and hence, the approximations may be better than for scalar metamodels.



(a) AvP plot of MNN-1-290 for y_{l}

(b) AvP plot of MNN-1-290 for y_{m}

(c) AvP plot of SNN-3-60 for y_{u}



(d) AvP plot of the fusion metamodel for y

Figure 33: AvP plots of MNN-1-290, SNN-3-60, and their fusion for $y_{\text{l}}, y_{\text{m}}, y_{\text{u}}$, and y.

Indeed, the scarcely best metamodel for y_{l}, y_{m}, and simultaneously y_{L}, y_{M} is MNN-1-290 regarding the highest R^2_{KR} values and the lowest MSE_{SH} — a multi-target regression metamodel, see Table 9. However, the scalar metamodel SNN-3-60 exhibits best results for y_{u}, i.e. it refutes the speculation that multi-target regression metamodels generally perform better. Similar observations can be made in Section 3.2.7 on the occupant simulation of a full frontal test. Moreover, the highest prediction score for the time histories y_{U} are obtained by MNN-2-110. All $n_{\text{val}} = 40$ exact validation time histories for y_{L}, y_{M},

and y_{U} are plotted in Fig. 41, 42, 43. In each single plot within these three figures, the approximated profile — corresponding to the shown exact curve — produced by the best multi-target regression metamodel, i.e. MNN-1-290 for y_{L}, y_{M}, and MNN-2-110 for y_{U}, is depicted on top. This provides visual comparisons of actual and predicted quantities that truly promise appropriate approximations.

AvP plots for the extracted key results, i.e. processed from MNN-1-290 for y_{l} and y_{m}, as well as for the directly computed values, i.e. evaluating SNN-3-60 for y_{u}, can be found in Fig. 33. In addition, the same figure features the AvP plot of the comprising quantity y. Its values are composed of the predictions of the three best performing metamodels. Mathematically, as it holds $y = \max\{y_{\mathrm{l}}, y_{\mathrm{m}}, y_{\mathrm{u}}\}$, one has $\hat{y} = \max\{\hat{y}_{\mathrm{l}}, \hat{y}_{\mathrm{m}}, \hat{y}_{\mathrm{u}}\}$ where $\hat{y}_{\mathrm{l}}, \hat{y}_{\mathrm{m}}$ come from the respective MNN-1-290 model and $\hat{y}_{\mathrm{u}}$ is derived by SNN-3-60. In the following, i.e. for sensitivity and uncertainty analysis, $\hat{y}$ resulting from $\hat{y}_{\mathrm{l}}, \hat{y}_{\mathrm{m}}, \hat{y}_{\mathrm{u}}$ produced by the two MNN-1-290 models and SNN-3-60 is meant when speaking about the approximation of y. The corresponding metamodel for $\hat{y}$ is called the fusion of the three neural networks. All aforementioned AvP plots confirm the high R^2 values and indicate suitable approximation qualities of the metamodels enabling further analyses. The deviations between the actual and predicted values, however, are higher than e.g. observed in the applications of Section 3.2.7. These incidents are later addressed in Section 4.4.1.

It can be concluded that neural networks should definitely be considered for crash applications. For the real-world project, they yield the best fitting metamodels among all possibilities. However, it is not guaranteed that they will consistently produce better results for general applications than other metamodels, cf. the results obtained in Section 3.2.7 for an empirical rebuttal. Moreover, the performance of GPR could possibly increase if another kernel — perhaps more suited for the problem – would be used. PCE could become better if e.g. the truncation strategy would be changed. Obviously, it is not feasible to try out everything imaginable. Nevertheless, several options should be defined, considered, and compared. The best option should then be utilized for the next steps. Note that each application may require a different metamodel regarding the variant, the method, and the configuration to optimally fit the data and enable further investigations.

4.3 || Sensitivity analysis

The best performing metamodels, MNN-1-290 and SNN-3-60, for $y_{\mathrm{u}}, y_{\mathrm{l}}, y_{\mathrm{m}}$ from Section 4.2.2 and their fusion to find $y = \max\{y_{\mathrm{l}}, y_{\mathrm{m}}, y_{\mathrm{u}}\}$ are now used to analyze sensitivities of the $n = 9$ parameters, see Table 7. Therefore, variance-based sensitivity measures from Section 3.3.3 are calculated. Using $n_{\mathrm{Sob}} = 10\,000$, first order and total effect indices, s_i (3.134) and s_{T_i} (3.135), provide profound relevance classifications of parameters for considered key results, see [SRA+08, IL15]. The respective metamodels must be evaluated

$$n_{\mathrm{Sob}}(n+2) = 110\,000 \tag{4.5}$$

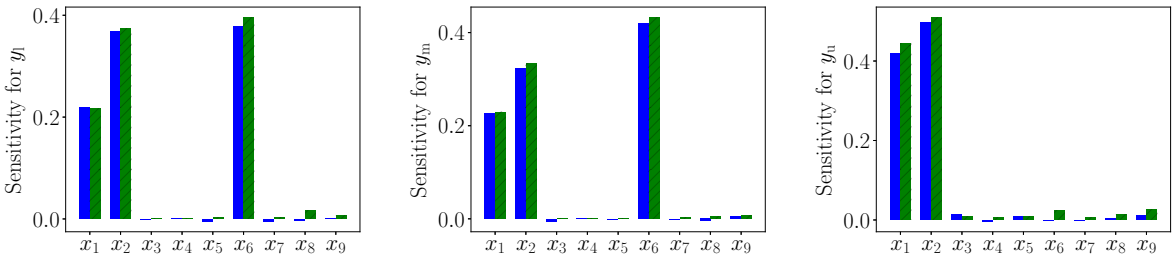
times. So, this analysis would not be temporally feasible using the slow finite element model itself. The uniformly distributed Sobol' sequence numbers for the parameters are not converted to other distributions, thus the parameters follow uniform distributions. This practice is further explained in Section 4.5.

The bar plot of the Sobol' indices for each key result is shown in Fig. 34. On the one hand, it can be noticed that the bar graphs for y_l and y_m look similar. Parameter x_6, the time-to-fire of the side airbag, is the most contributing parameter for y_l and y_m. From a technical point of view, this is reasonable as the side airbag is located in the door panel below the window and is hence able to protect the lower part of the upper body. Furthermore, the aleatory parameters x_1 and x_2 are also relevant and decide on the severity of the deflections. Other parameters hardly influence the key results and can be neglected. The input space for y_l and y_m thus can be shrunken to be three-dimensional. The small differences between the indices, s_i and s_{T_i}, for x_1, x_2, x_6 indicate little higher order effects in the system.

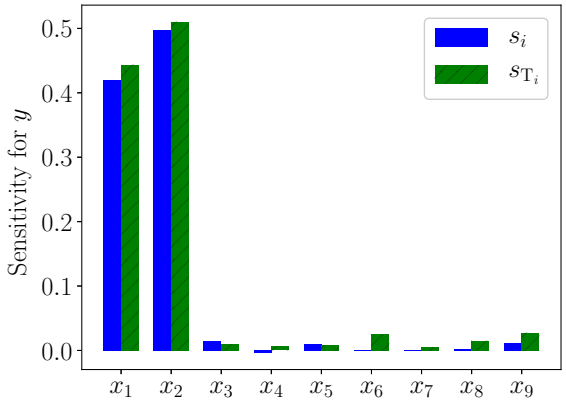
On the other hand, Fig. 34 (c) and (d) depict that the bar charts for y_u and y are virtually identical. This is due to the fact that y is dominated by y_u because of its definition to be the maximum of y_l, y_m, and y_u where it holds

$$y_\mathrm{u} \geq y_\mathrm{m} \geq y_\mathrm{l} \tag{4.6}$$

in most cases, cf. the ranges of Fig. 30 and the time history heights in Fig. 32. For the same reason, the AvP plots of y_u and y are practically equal, see Fig. 33 (c) and (d).



(a) Sobol' indices for y_l (b) Sobol' indices for y_m (c) Sobol' indices for y_u



(d) Sobol' indices for y

Figure 34: Bar charts of the Sobol' indices for the parameters mapped to $y_\mathrm{l}, y_\mathrm{m}, y_\mathrm{u}$, and y. The best performing metamodels from Section 4.2.2 are utilized to determine these indices.

It is conspicuous that parameter x_6 does not affect y_u and y. An engineering explanation may be that the side airbag in this application sits too low to mitigate deflections at the upper rib of the chest. One could expect that the curtain airbag whose ignition time is regulated by x_5 stands in for the side airbag. Effectively, this does not apply. The curtain airbag does not intervene upper chest rib injuries as it apparently does not deploy extensively enough. In the end, it turns out that only parameters x_1 and x_2 are significant for y_u and y. Nevertheless, x_6 is retained for subsequent uncertainty analysis to define a consistent uncertainty assignment for all key results, $y_\mathrm{l}, y_\mathrm{m}, y_\mathrm{u}$ and y. Summarizing, the variance-based sensitivity measures reduces the number of parameters to $d = 3$, i.e. only x_1, x_2, and x_6 are considered to be uncertain. Other parameters are set to their nominal values. Screening from Section 4.2.1 could not limit the input space to the same extent but its anticipation that x_1, x_2, and x_6 are relevant was correct.

4.4 || Uncertainty propagation

Uncertainties are allocated to the remaining $d = 3$ parameters, x_1, x_2, and x_6, of the real-world application. The previously assumed uniform distributions are replaced by problem-specific Dempster-Shafer uncertainty structures, see Section 3.4.2. Speaking in ET terms, the parameter intervals from Table 7 are split into focal elements and undergo basic probability assignments (BPA). In other words, they are divided into subintervals to which probabilities are ascribed, see the first two steps in Section 3.4.2. This is done fictively which is illustrated in Fig. 35.

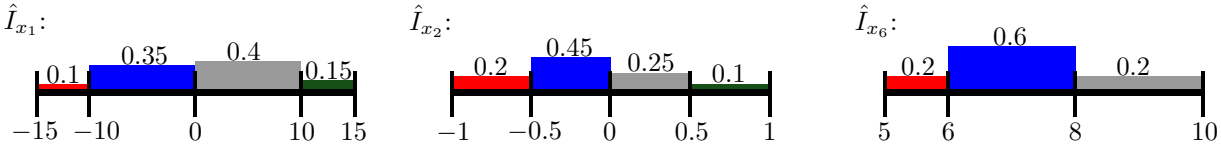


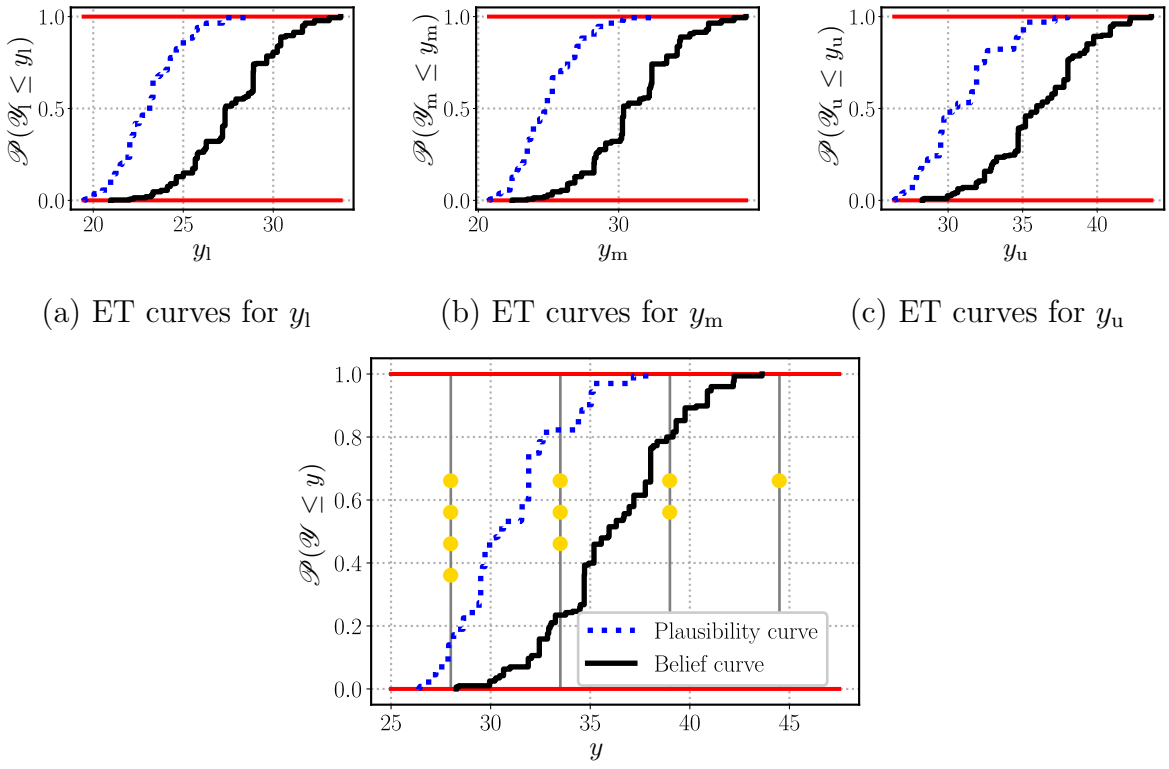
Figure 35: Fictive focal elements and basic probability assignments for the three retained parameters, x_1, x_2, and x_6.

For x_1 and x_2, four focal elements are considered. The interval of parameter x_6 is divided into three subintervals. So, $\kappa = 4^2 \cdot 3 = 48$ interval cells are established, for each of which the composite BPA has to be calculated as well as the minimum and maximum of the system function must be found, see steps three to five of Section 3.4.2. To enable this optimization, the metamodels of Section 4.2.2 are used as system function — analogously to Section 4.3 — here mapping x_1, x_2, and x_6 to $y_\mathrm{l}, y_\mathrm{m}, y_\mathrm{u}$, and y, respectively. The particle swarm routine from the Python library Scikit-opt [Guo21] determines the minima and maxima. For each optimization, the particle swarm algorithm requires hundreds or even thousands of metamodel evaluations. For this reason, ET cannot be performed with the full-scale model — analogous to variance-based sensitivity analysis. After the 2κ optimization problems are solved, the ET curves can be plotted. For each of the four key results, $y_\mathrm{l}, y_\mathrm{m}, y_\mathrm{u}$, and y, the belief and plausibility curves are depicted in separate graphs, see Fig. 36.

The belief and plausibility curves envelop the actual cumulative distribution function for the key results based on the Dempster-Shafer uncertainty allocations illustrated in

Fig. 35. Specific statements are exemplarily made using the graph for $y = \mathrm{CD}$, i.e. Fig. 36 (d). This Euro NCAP injury criterion can be translated to rating points, see Section 4.1. The integer boundaries of these points are delineated in the bottom graph by solid gray vertical lines with golden round dots attached to them that characterize the number of points. Basically, the beginning of the plausibility curves and the end of the belief curve indicate that chest deflections between about 26 mm and 44 mm are possible with certain probabilities of two different measures. From this, it can be deduced that Euro NCAP points between 1.09 and 4 are achievable.

According to the plausibility curve, the maximum chest deflection is smaller than or equal to 33.5 mm leading to at least three rating points by plausible 82%. As antagonist, the belief curve warns and states that this is only 23% believable. Considering at least 2 points or a deflection of less than or equal to 39 mm, the plausibility curve guarantees 100% probability of occurrence while the belief curve is at 80%. Such conclusions can therefore be drawn from these curves. Engineers have to decide whether these risk should be taken or concepts have to revised. Therefore, they must judge by hand whether the actual cumulative distribution function is closer to the plausibility or belief curve.



(a) ET curves for y_l (b) ET curves for y_m (c) ET curves for y_u

(d) ET curves for y

Figure 36: Plausibility and belief curves for $y_\mathrm{l}, y_\mathrm{m}, y_\mathrm{u}$, and y based on the assignments prepared in Fig. 35.

Note that the ET curves slip tighter together when the focal elements are more finely defined, i.e. a greater number of subintervals is selected. This narrows the gap between the two curves which allows the actual cumulative distribution function to be better

identified. For quasi-continuous assignments, the curves would almost lie on top of each other. This would make the ET procedure, especially its optimization, more complicated, as the number of interval cells would increase drastically. In the field of vehicle crash, however, it is difficult to agree on clear continuous distributions due to sparse data and limited knowledge about the parameters. Therefore, the imprecise distributions of the Dempster-Shafer evidence theory are predestined for uncertainty propagation.

4.4.1 Error awareness and conservative evidence theory curves

The AvP plots in Fig. 33 and the R^2 scores from Table 9 indicate that the neural networks used for the approximation of the finite element simulation have a sufficient accuracy. However, slight deviations between actual and predicted values — higher than in the applications of Section 3.2.7 — can be detected in the AvP plots. Among others, this is due to the complex finite element model and its long computational times on many CPUs from diverse computing machines. The distribution of the calculation jobs to various CPUs can lead to differing results for the same initial parameters when using different computers, e.g. because of disparate rounding techniques. These and other numerical effects imply that the finite element model or rather its gathered data, i.e. the training and validation sample, is mathematically speaking noisy or rather not fully well-defined. The accuracy of metamodels is limited by this circumstance. Therefore, precise results cannot be expected but the correct detection and forecast of trends. This is definitely fulfilled as can be recognized e.g. from the curve approximations in Fig. 41, 42, and 43.

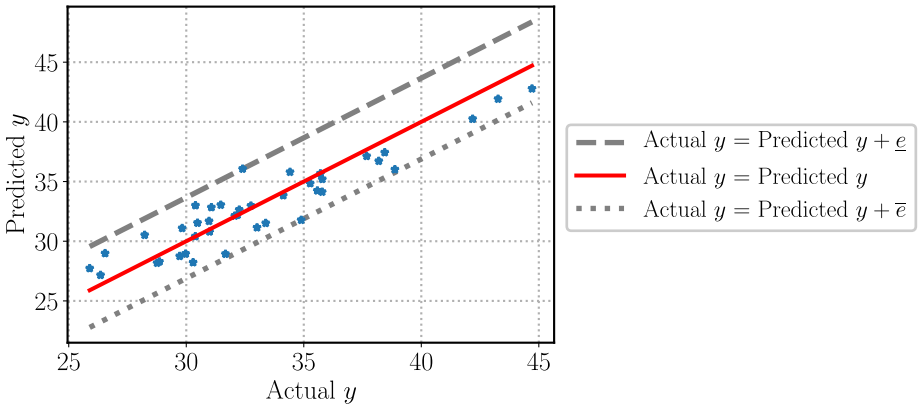


Figure 37: AvP plot of y with conservative error bounds established by the largest metamodeling underestimate and overestimate.

Nevertheless, deviations are present and should not be ignored. For this reason, the differences between the $n_{\text{val}} = 40$ validation points and their approximations are analyzed. Exemplarily, the Euro NCAP key result, $y = \max\{y_{\text{l}}, y_{\text{m}}, y_{\text{u}}\}$, is considered, i.e. the fusion of the MNN-1-290 metamodels and SNN-3-60 is investigated. The biggest underestimation of a validation sample point is equal to $\overline{e} = 3.11273\,\text{mm}$ — found at $y = 34.88977$ and $\hat{y} = 31.77704$, cf. Fig. 37. The largest overestimation is $\underline{e} = -3.66406\,\text{mm}$ for

$y = 32.39803$ or $\hat{y} = 36.06209$. The overestimate and the underestimate are used to calculate conservative ET curves.

On the one hand, the system function to find the conservative maximum value for each interval cell is augmented by adding the largest underestimate, i.e. the optimization procedure in (3.158) changes to

$$\overline{y}^l = \sup_{x \in C^l} \mathcal{F}(x) + \overline{e}. \tag{4.7}$$

On the other hand, optimizations for the conservative minimum values read

$$\underline{y}^l = \inf_{x \in C^l} \mathcal{F}(x) + \underline{e}, \tag{4.8}$$

cf. (3.159). Actually, the conservative optimal values thus must not be determined again. They can also be calculated by adding these overestimation and underestimation values to the already found optimal values from Section 4.4, i.e. by shifting the plausibility curve towards left and moving the belief curve towards right in the bottom plot of Fig. 36. This can be done as the constant terms added to the original system function do not influence the optimal arguments. The resulting conservative plausibility and belief curves can be observed in Fig. 38. They are indeed conservative as the largest deviation — from one specific location — is incorporated whereas the error around other parameter regions might be less.

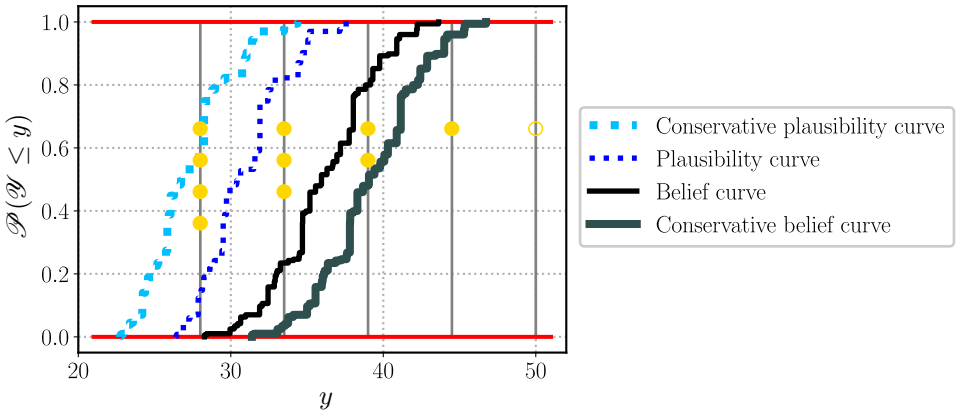


Figure 38: Evidence theory plots including original and conservative plausibility and belief curves for y.

The conservative ET curves envelop the original plausibility and belief curves. They permit more latitude for the actual cumulative distribution function. On one side, the conservative plausibility curve suggest higher chances to obtain four points: it is, for instance, 59% conservatively plausible while originally, this is only 14% plausible. On the other side, the original belief curve excludes scores below one point. The conservative belief curve, however, displays a small probability that the score can fall below one. The conservative ET curves thus take a more cautious approach to the assessment of uncertainties. Metamodeling errors caused by approximating the finite element simulation are taken into account.

4.5 || Recalling the results and discussion

The real-world project was passed through the uncertainty management framework depicted in Fig. 5. The large finite element simulation of a side pole test was first assessed in an engineering manner. It was worked out that the maximum chest deflection, y, is the injury criterion most likely to lose points in the Euro NCAP rating. The framework thus concentrated on this key result and its suppliers — lower, middle, and upper chest rib deflection, i.e. $y_\mathrm{l}, y_\mathrm{m}$, and y_u. The elementary effects method filtered out irrelevant parameters with a deliberately small budget, see Fig. 31. For the single key results, different numbers of parameters were identified as important. These information could but must not necessarily be fed to the next stage: metamodeling.

Different metamodels were studied. Many of them provided appropriate approximation results, see Table 9. The slightly best quality was reached by variants of neural networks — MNN-1-290 architectures for $y_\mathrm{l}, y_\mathrm{m}$, and SNN-3-60 for y_u. The Euro NCAP key result, y, was then extracted from the fusion of these three metamodels. Instead of awaiting a computational time more than 32 hours on 168 CPUs for a new parameter configuration, the neural networks could be evaluated within fractions of a second. This relief on calculation efforts enabled the use of profound sensitivity analysis and uncertainty propagation. In addition, reducing the number of parameters through sensitivity analysis facilitated the evidence theory.

As with the metamodels, uniformly distributed parameters were assumed for the Sobol' sensitivity analysis. In the case of the metamodels, the entire input space should be approximated as thoroughly as possible which is why the samples are drawn in a way that their points are uniformly distributed in the space. This ensures a fair chance of approximating the system in each input region at the same level. Sensitivity analysis intends to treat each spot of the parameter intervals as equally important to especially account for parameter outliers that can be crucial for the system. Furthermore, expert opinions are provided later in the development stage. In earlier phases, e.g. for sensitivity analysis, ET uncertainty assignments are not known. This situation is also described in [JLG21b]. In case the assignments are available, sensitivity analysis can already be run with the uncertainty allocation of ET instead of using uniformly distributed parameters. Note that this may change Sobol' indices. Here, however, parameters were considered to be uniformly distributed for sensitivity analysis. Before applying uncertainty propagation, the uniform distributions were substituted by ET structures: experts assessed the remaining relevant parameters in-depth and allocated probabilities of occurrence.

Sensitivity analysis identified that the aleatory parameters x_1 and x_2 significantly contribute to all key results. They stand for the barrier x-position and the velocity, respectively, see Table 7. Moreover, the time-to-fire of the side airbag — denoted as x_6 — is additionally crucial for y_l and y_m. For this reason, it is also considered for the subsequent uncertainty quantification together with x_1 and x_2. The Dempster-Shafer evidence theory is utilized. Basic probability assignments were realized fictitiously. Exemplary statements were made using the plausibility and belief curve depicted in Fig. 36. These statements, however, were critically edited by taking into account the metamodeling error. For this purpose, conservative ET curves were defined. They are a rigorous way to incorporate the error made by the metamodel into the ET curves. Future research should examine

how to process this error in a more sophisticated manner. Besides, the metamodeling error may also be considered in the sensitivity analysis, see [SWV+22]. In fact, the small error that is present here has no significant influence on the Sobol' indices, especially not on the importance of the parameters. Another open question is how sensitive the basic probability assignments are to the Dempster-Shafer evidence theory curves. Future work should investigate this matter.

Summarizing, the framework was successfully applied. It provided insight into the relevance of the parameters considered and the uncertainties inherent in the key results. The parameters, x_1 and x_2, that are relevant to the main key result, y, are aleatory, i.e. they cannot be controlled and reduced. Engineers therefore cannot improve the behavior of their car in this load case by configuring the chosen epistemic parameters. For this specific constellation, they must accept the risk of possibly losing around three Euro NCAP rating points in the worst case. In practice, to improve the performance nonetheless, one should introduce and address further epistemic parameters to make the car and its systems more crash-proof.

Critical reflection and outlook

The compact uncertainty management framework for finite element simulations from Fig. 5, i.e. representative methods for its components, was kept as broad as possible. Not every single exceptional case could be treated. Individual methods for the framework components might be adjusted for specific applications. Nonetheless, the general uncertainty management framework from Fig. 4 provides a structured procedure on how to approach.

For example, the parameters were considered to be continuous quantities intrinsic to the model code and represented hardware settings. This thesis did not specifically address how to deal with discrete parameters or hardware-unrelated instances, e.g. numerical uncertainties or approximation inadequacies. Parameters of class numerical uncertainties are e.g. different discrete numbers of CPUs for solving the finite element model, the length of the solver time step, or the fineness of the finite element mesh. These parameters may be varied to assess the numerical robustness of the simulation. If metamodels are intended to be used for this purpose, methods capable of processing discrete or mixed input quantities may be required, see [XCY16, Bar15].

For model inadequacy, i.e. the discrepancy between simulation and experiment, reference is made to the field of verification and validation that attempts to cope with and minimize this type of uncertainty, see [OT02, RKW15, Sar13]. In [WM19], the epistemic uncertainty of a crashbox is reduced by using verification and validation tools to calibrate the numerical finite element model to experimental investigations. Adequacy of the applications was presumed throughout this thesis. Still, numerical effects leading to noisy data had to be faced for the real-world project, see Section 4.4.1. The metamodels thus could not be fully accurate. In future applications, it is recommended to quantify, assess, and, if necessary, reduce these numerical robustness issues before actually applying the framework to the finite element model.

While the scope of this thesis does not extend to the prevention of these numerical issues, the following thoughts should be considered in future research. Note that the practicability of these recommendations need to be verified. To begin with, the solver version should not be changed among the simulations. It is recommended to solve all finite element simulations on a fixed number of CPUs and, if possible, on the same hardware. Rounding inconsistencies, for instance, which would result from other implementation settings, can thus be circumvented.

The simulations in this thesis were calculated on different machines as the single jobs were distributed through a high-performance computing cluster. Divergent rounding and interpolation procedures of the various machines yielded deviating results in some cases. This was not optimal with respect to precision but by doing so, the required samples were established faster than using a single machine where all but one simulation pause in a queue. Besides, quantification of the numerical uncertainties by simulating the model for

several solver time steps and mesh sizes can bring insights about the numerical stability. If the performance is insufficient, i.e. the outputs highly deviate, the initial conditions, the CAD geometry, etc. may require adjustments.

Moreover, the proposed metamodels presented in Section 3.2 emulate continuous responses. In fact, continuity was given in the applications of this thesis. The approximation quality hence was appropriate. Crash phenomena, however, might feature discontinuous behavior, non-differentiabilities, or bifurcations. Especially, for structural crash events, these situations might occur. A crashbox, for instance, kinks to the right for one parameter configuration and to the left for the other — similar to the well-known Euler's column, see [Joh83]. Likewise, this can happen for dummy signals. The motion-related time history of the dummy head might have either a positive or a negative slope when the head moves up or down. Both examples exhibit jumps, i.e. discontinuities, that the metamodels used may struggle with.

To handle this, classification algorithms, see [Kot07, SHG20, OAA+17], could be employed to label the discontinuity types before training a metamodel for each of these types. Alternatively, there exist metamodels that can theoretically model jumps. A Master thesis, see [Dek20], tutored by the author of this dissertation investigates multi-variate B-spline regression. By defining multiple knots for B-spline basis functions, discontinuities at the location of these knots can be introduced. The Master thesis pursues a free knot selection to automatically determine the optimal knot locations. To accomplish this by a gradient-based optimization, new numerically efficient derivative expressions of the B-splines with respect to the knots are determined. However, this promising approach quickly suffers the curse of dimensionality since the hyperparameters of a variety of basis functions must be optimized — even for small input dimensions. Future research should look into ways to make the B-spline regression sparse.

Aside from that, additional metamodels could be tested and compared for the real-world project in Chapter 4. This does not only refer to methods or variants but also to the considered key results. The fused quantity, y, used for the Euro NCAP rating is extracted from three neural networks emulating $y_\mathrm{l}, y_\mathrm{m}$, and y_u. Metamodels could be established for y directly. Vehicle safety engineers are also interested in the Euro NCAP rating points that are deduced from y. One could thus regard these points as key result and let metamodels map from the input space to the rating points. The considered strategies in Section 4.2.2, however, yield appropriate results. On top of that, engineers want to trace the origin of the rating points back to the single sources, $y_\mathrm{l}, y_\mathrm{m}$, and y_u. For these reasons and the sake of clarity, additional key results are omitted.

Beyond, further multi-target regression metamodels should be studied. The regressor chain-based method could not satisfy author's expectations, cf. results for the occupant model in Section 3.2.7. Recurrent neural networks are designed for long time sequences but do not bring any benefit to the task at hand, i.e. mapping independent parameters to output curves, see Section 3.2.4. Hence, temporal dependencies of output time histories could not be exploited using the metamodels of this thesis. Future studies should be devoted to incorporate these dependencies and thus improving quality of multi-target regression metamodels. For example, neural networks that use loops — similar to recurrent neural networks but only for output quantities — could be examined. Possibly, the regressor chain-based method might be enhanced to effectively improve approximations.

Last but not least, recent research explores how additional available technical information, e.g. physical laws, scientific principles, or constraints, can be integrated into metamodeling approaches or machine learning algorithms, see [KKL+21]. In so-called physics-informed or physics-inspired neural networks, known or assumed underlying physical equations, e.g. ordinary or partial differential equations, are incorporated into the loss function to boost model training and to strengthen approximations, see [RPK19, YP19, ZLS20]. These ideas are promising regarding the advancement of metamodels. Particularly for crash applications, future research must elaborate what information or data of required type are available and which of these can actually be incorporated and ultimately be useful. If this eventually yields an improvement, physics-informed approaches should be included in the presented framework, see Fig. 5.

Conclusion

This thesis proposes a general framework to manage uncertainties in automotive crash applications. The main challenges that are tackled by this framework are resource inefficiency of models for passive safety scenarios and their high number of parameters from several sources that crucially impact results and subsequently safety of vehicles. The framework therefore considers different paths to enable the final stage uncertainty propagation. Which route to take depends on the type of the object under study. Resource efficient models can be directly subjected to sensitivity and uncertainty analysis. Resource inefficient models might or might not be simplified by engineering knowledge. The number of parameters of the pending model decide on how to proceed. All presented options amount to fast-responding approximated response surfaces that replace the expensive model. Many parameters may impede the creation of high quality approximated response surfaces. It is proposed to use screening to rectify this situation by identifying clearly irrelevant parameters and reducing the input space to the contributing dimensions. After approximated response surfaces are established, sensitivity analysis profoundly explores the input space. Finally, uncertainty quantification can be performed to measure the impact of the uncertain parameters on the output. The entire framework is summarized in Fig. 4.

Finite element simulations are the models most frequently investigated by engineers. For this reason, special attention was paid to these simulations, i.e. the framework was customized to them, see Fig. 5. First, these black box-like objects were put into a novel mathematical form and decomposed to mappings of different levels of information. Second, variants of metamodels approximating these mappings were discussed and applied. Third, screening via the Morris method and variance-based sensitivity analysis were explained and prepared for numerical implementation. Last, the Dempster-Shafer evidence theory was presented for uncertainty propagation. These methods collectively — structured via the proposed framework — overcome the mentioned obstacles, e.g. the large number of parameters and resource inefficiency. They provide a fast, flexible, and visualizable usage to support engineers, thus the framework offers a satisfactory answer to the research question posed in Section 1.2.

To substantiate the theoretical procedure, the framework is tested practically in a real-world project. Therefore, the finite element simulation of a side pole test from the European New Car Assessment Programme was subjected to the components of the framework. Different metamodels approximating scalar crash test quantities of interest are compared with multi-target regression metamodels that emulate the time histories of these quantities before scalar values are extracted. The best metamodel was used for next analyses, i.e. sensitivity analysis and uncertainty propagation. Sobol' indices indicated that only two out of nine initial parameters are relevant for the key result, here the maximum chest deflection. These two important parameters are both aleatory uncertainties.

Thus, engineers cannot change or improve the situation in this set-up, e.g. by configuring epistemic parameters. They have to cope with the corridor for the actual cumulative distribution function — the corridor that is defined by the plausibility and belief curve of the Dempster-Shafer evidence theory.

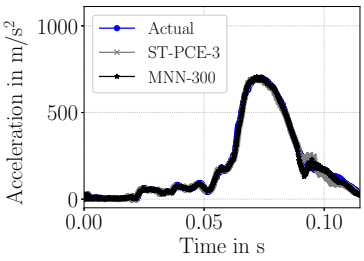
The application of the framework was successful and its outcome guides engineers in decision making. Nonetheless, opportunities for further research were devised to enhance the framework. Proposed metamodels must be substituted if discrete parameters or discontinuous outputs are encountered. Furthermore, model inadequacy as special uncertainty type is not considered in this thesis. To address this type, methods from the field of verification and validation should be examined and coupled with the presented framework. In addition to the investigated metamodels, other methodologies may be utilized or existing approaches may be refined. For example, one might study how to profitably include the temporal dependencies of the time history outputs or further available technical information into the metamodel. Notwithstanding these potential updates, this work represents an important step forward in tackling uncertainties that appear in applications of vehicle safety. Even for resource intensive models with several parameters, engineers receive information about the uncertainties they must address in vehicle development to build safer cars.

Appendix

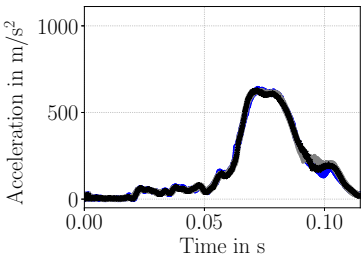
To improve the flow of reading, this appendix includes figures that would have occupied excessive space in the previous chapters. Subplots of figures are enumerated using numbers — not letters as before — due to their high number.

A.1 || Time histories for the occupant simulation of a full frontal test

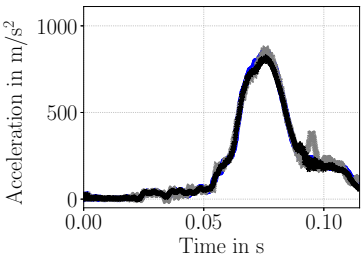
A.1.1 Head acceleration curves



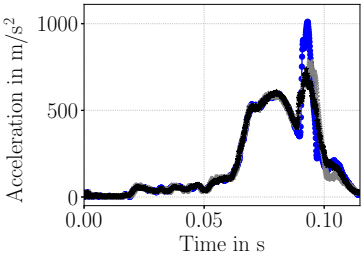
(1) Validation curve 1



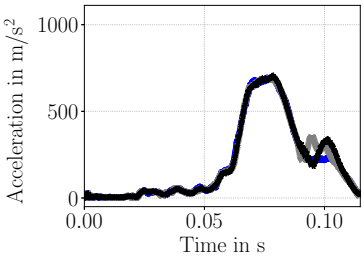
(2) Validation curve 2



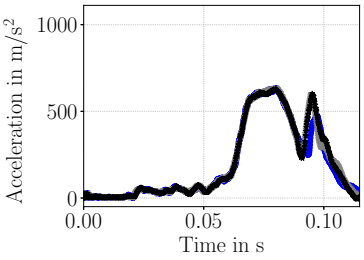
(3) Validation curve 3



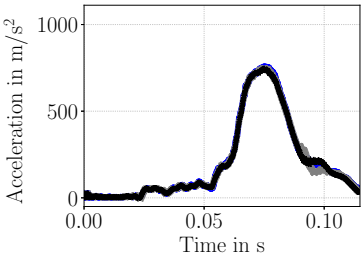
(4) Validation curve 4



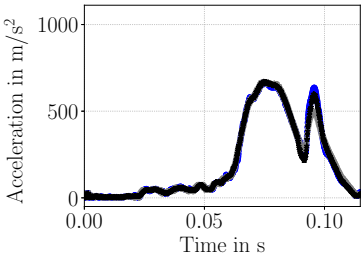
(5) Validation curve 5



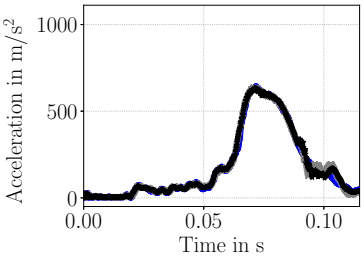
(6) Validation curve 6



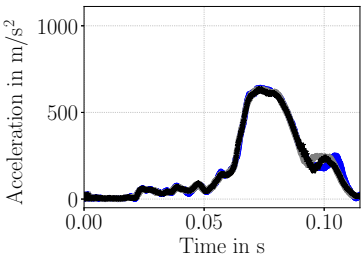
(7) Validation curve 7



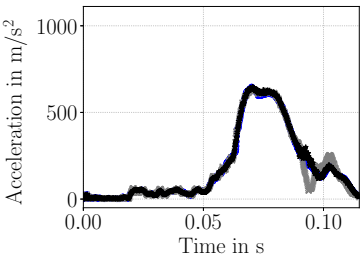
(8) Validation curve 8



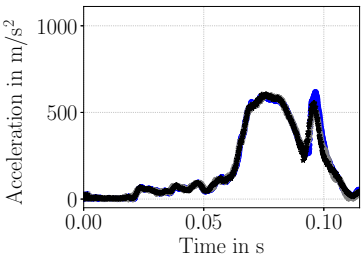
(9) Validation curve 9



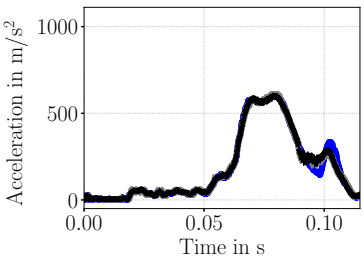
(10) Validation curve 10



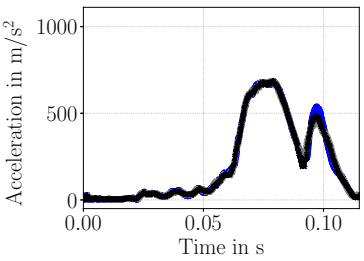
(11) Validation curve 11



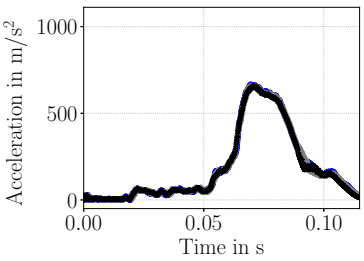
(12) Validation curve 12



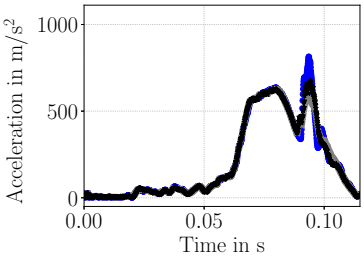
(13) Validation curve 13



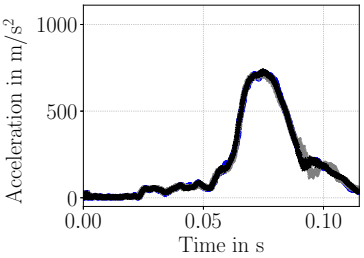
(14) Validation curve 14



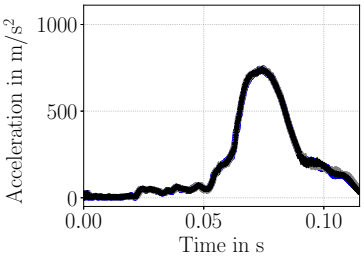
(15) Validation curve 15



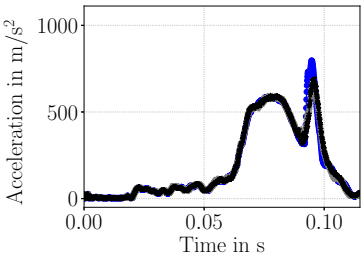
(16) Validation curve 16



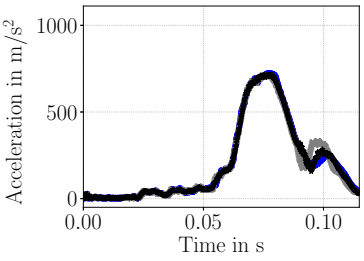
(17) Validation curve 17



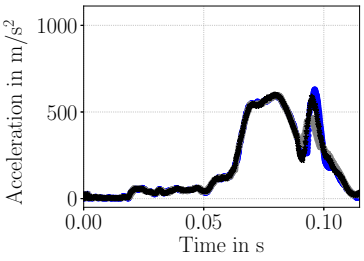
(18) Validation curve 18



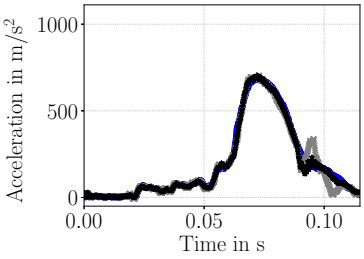
(19) Validation curve 19



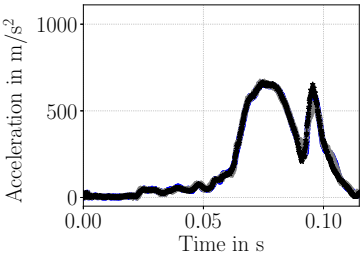
(20) Validation curve 20



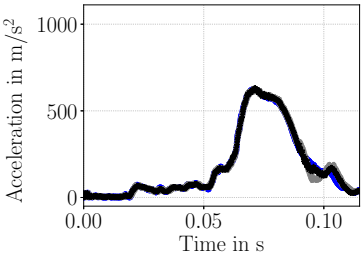
(21) Validation curve 21



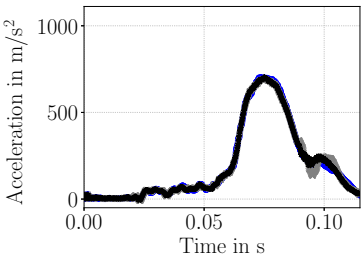
(22) Validation curve 22



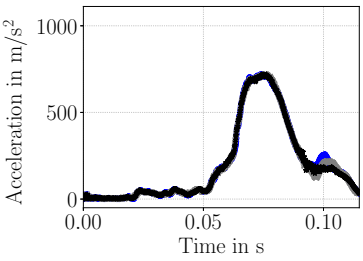
(23) Validation curve 23



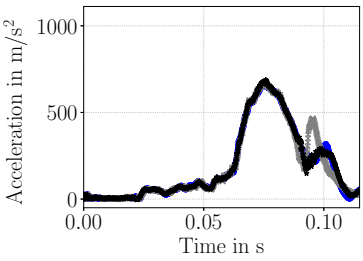
(24) Validation curve 24



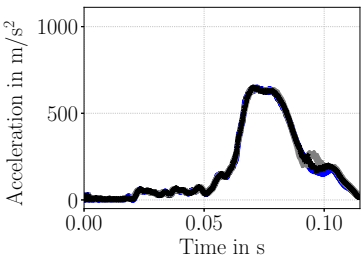
(25) Validation curve 25



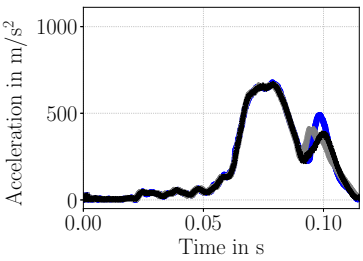
(26) Validation curve 26



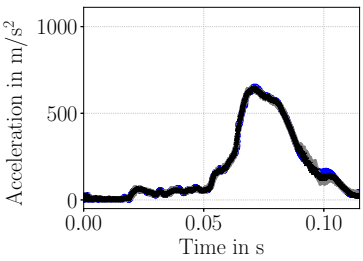
(27) Validation curve 27



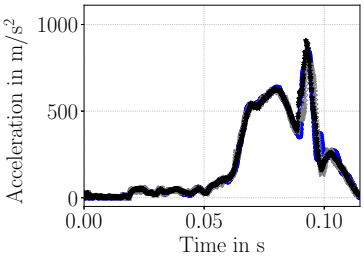
(28) Validation curve 28



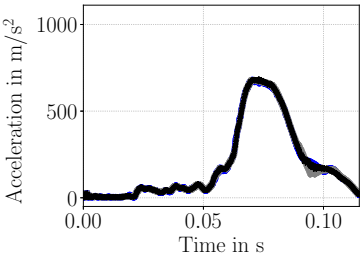
(29) Validation curve 29



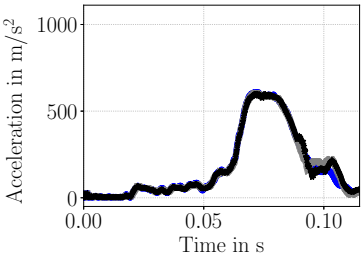
(30) Validation curve 30



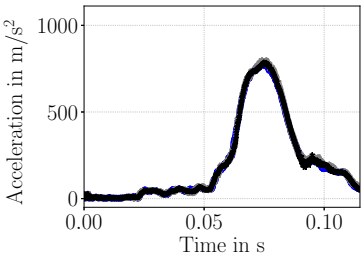
(31) Validation curve 31



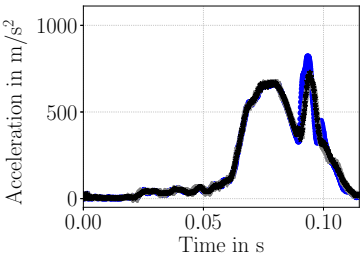
(32) Validation curve 32



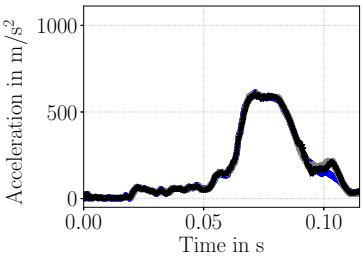
(33) Validation curve 33



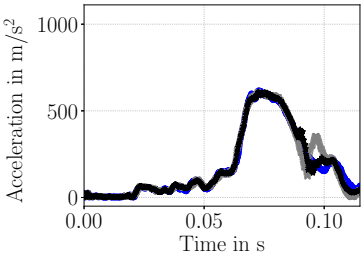
(34) Validation curve 34



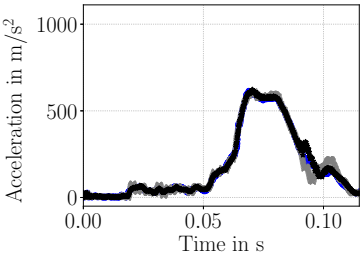
(35) Validation curve 35



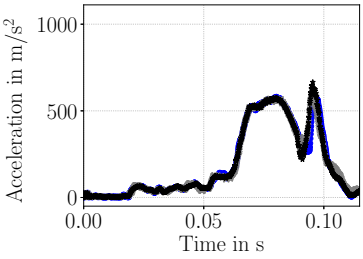
(36) Validation curve 36



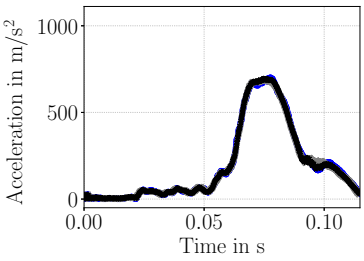
(37) Validation curve 37



(38) Validation curve 38



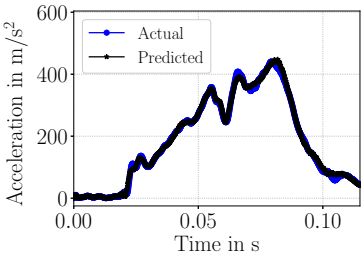
(39) Validation curve 39



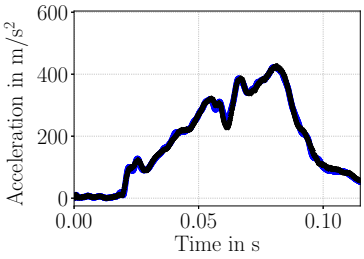
(40) Validation curve 40

Figure 39: Actual and predicted time histories of the $n_{\text{val}} = 40$ validation head acceleration curves of the occupant simulation. The predictions are from ST-PCE-3 and MNN-300.

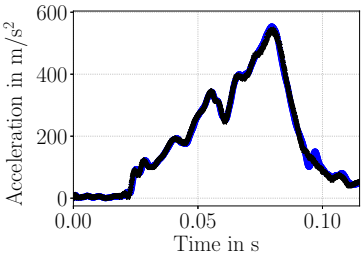
A.1.2 Chest acceleration curves



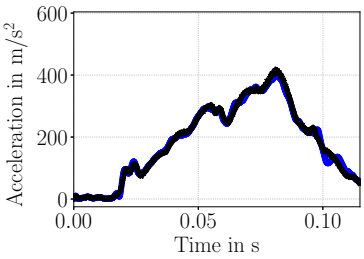
(1) Validation curve 1



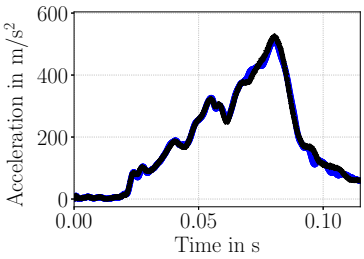
(2) Validation curve 2



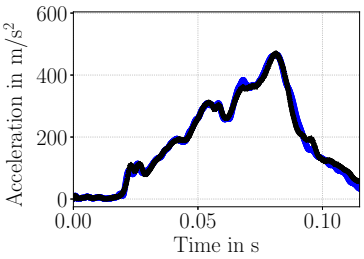
(3) Validation curve 3



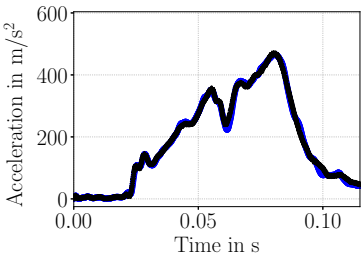
(4) Validation curve 4



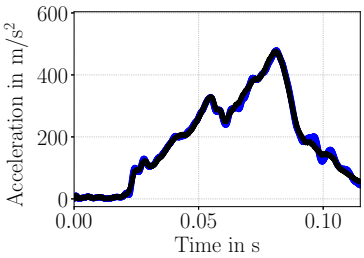
(5) Validation curve 5



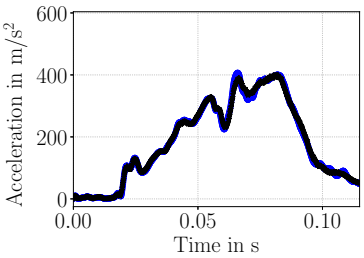
(6) Validation curve 6



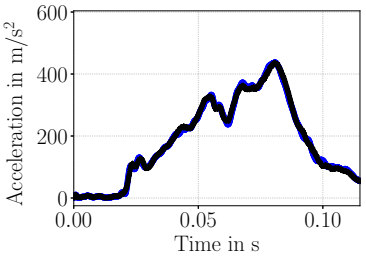
(7) Validation curve 7



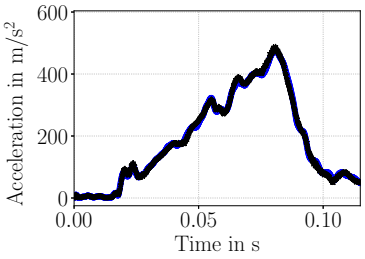
(8) Validation curve 8



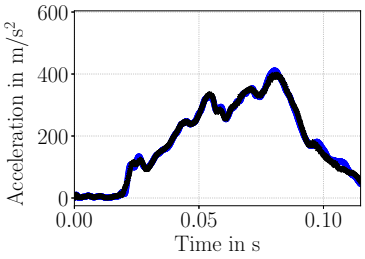
(9) Validation curve 9



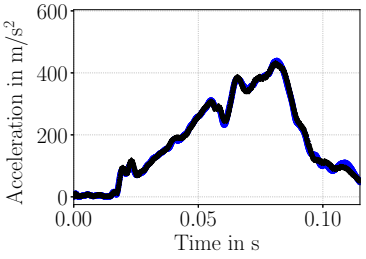
(10) Validation curve 10



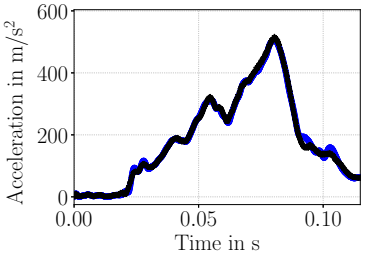
(11) Validation curve 11



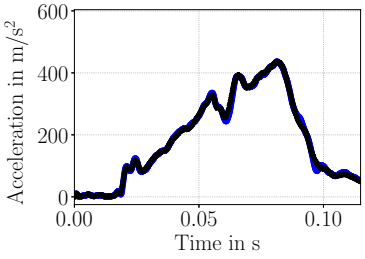
(12) Validation curve 12



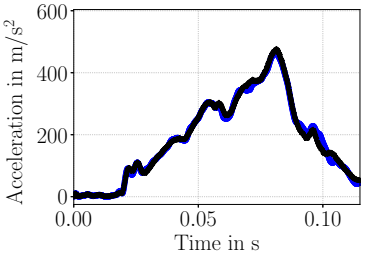
(13) Validation curve 13



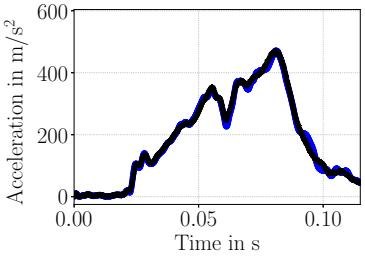
(14) Validation curve 14



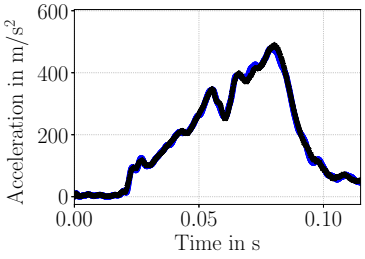
(15) Validation curve 15



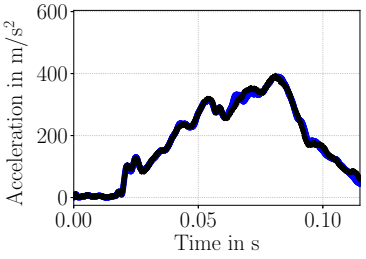
(16) Validation curve 16



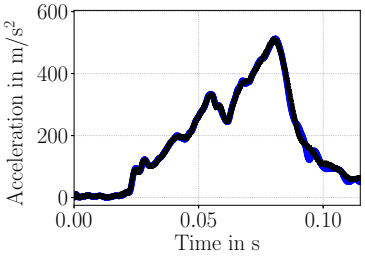
(17) Validation curve 17



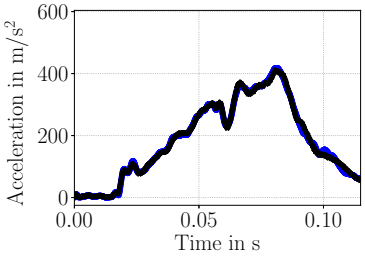
(18) Validation curve 18



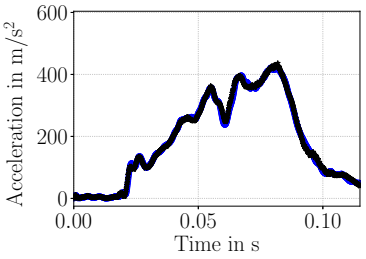
(19) Validation curve 19



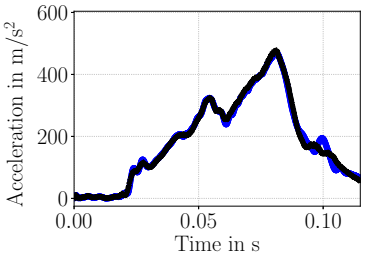
(20) Validation curve 20



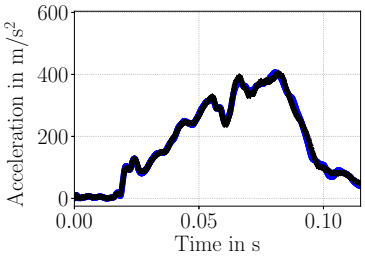
(21) Validation curve 21



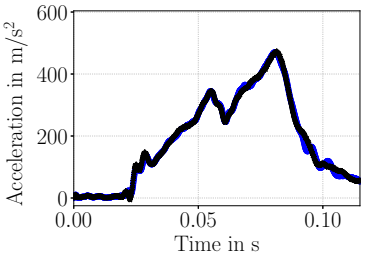
(22) Validation curve 22



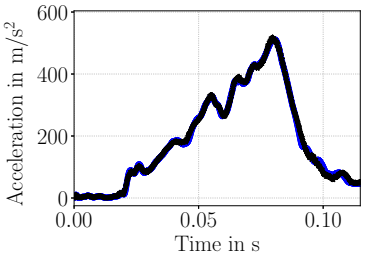
(23) Validation curve 23



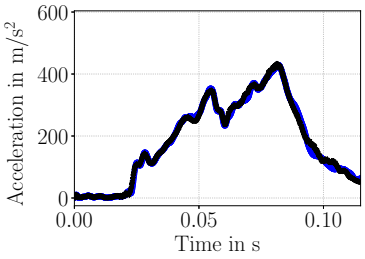
(24) Validation curve 24



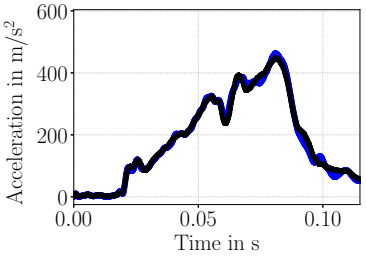
(25) Validation curve 25



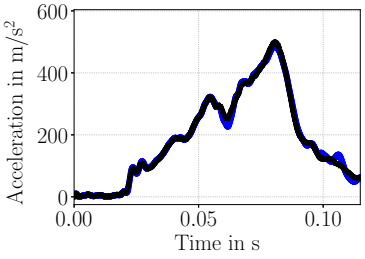
(26) Validation curve 26



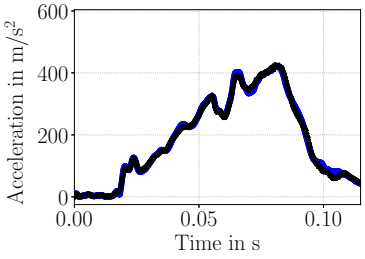
(27) Validation curve 27



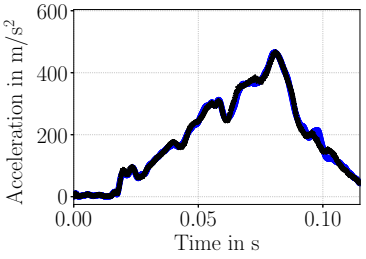
(28) Validation curve 28



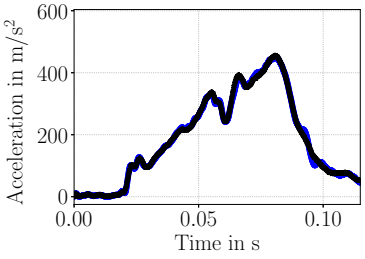
(29) Validation curve 29



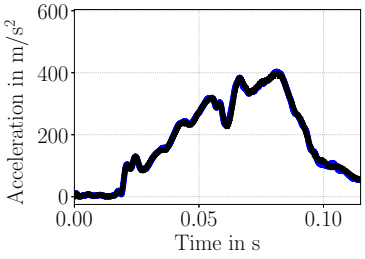
(30) Validation curve 30



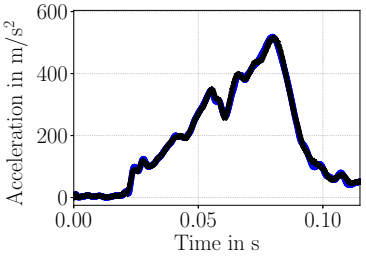
(31) Validation curve 31



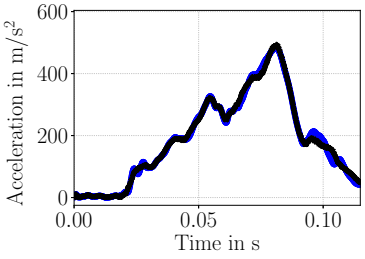
(32) Validation curve 32



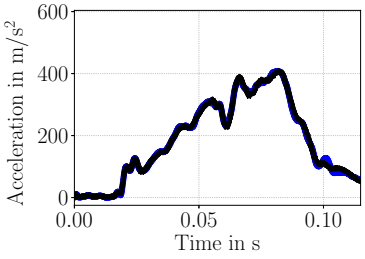
(33) Validation curve 33



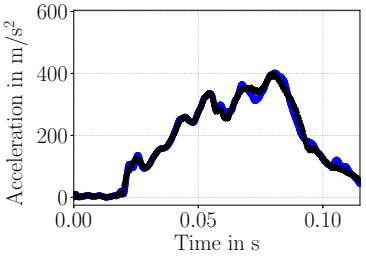
(34) Validation curve 34



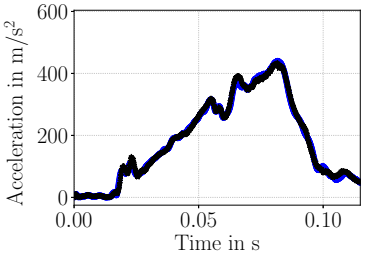
(35) Validation curve 35



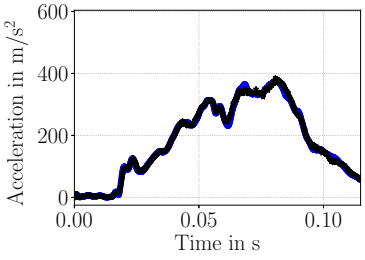
(36) Validation curve 36



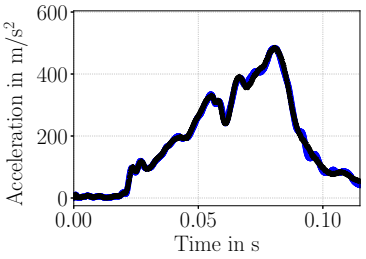
(37) Validation curve 37



(38) Validation curve 38



(39) Validation curve 39

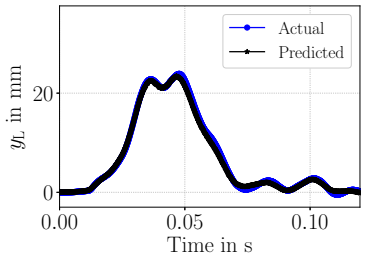


(40) Validation curve 40

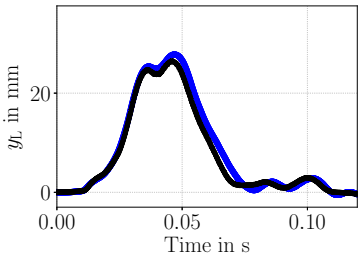
Figure 40: Actual and predicted time histories of the $n_{\text{val}} = 40$ validation chest acceleration curves of the occupant simulation. The predictions are produced by MNN-300.

A.2 || Time histories for the chest deflection of the real-world application

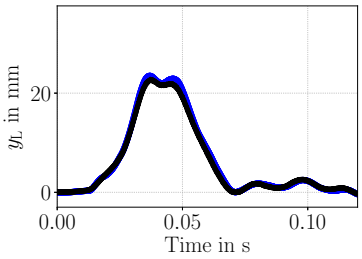
A.2.1 Lower chest deflection curves



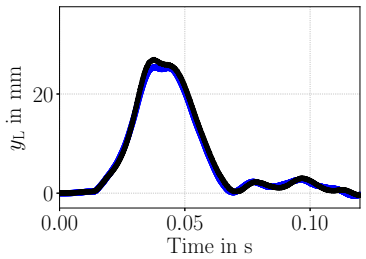
(1) Validation curve 1



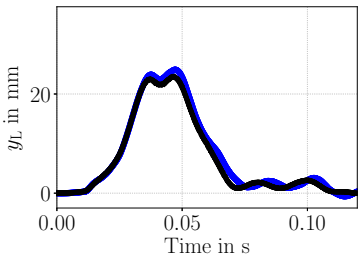
(2) Validation curve 2



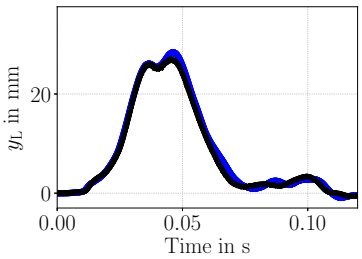
(3) Validation curve 3



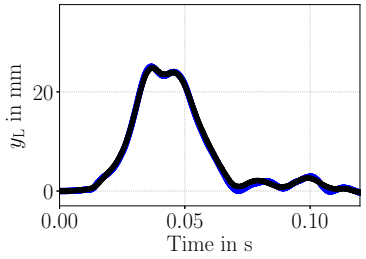
(4) Validation curve 4



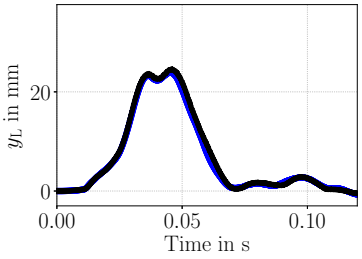
(5) Validation curve 5



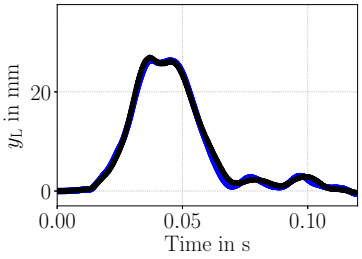
(6) Validation curve 6



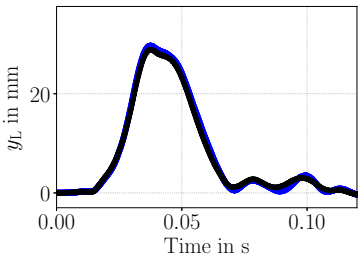
(7) Validation curve 7



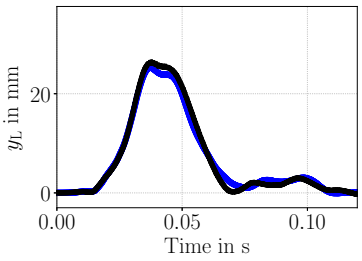
(8) Validation curve 8



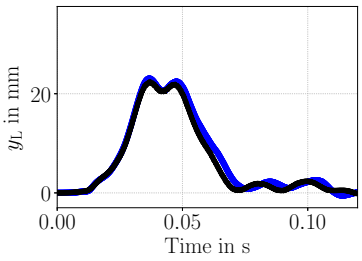
(9) Validation curve 9



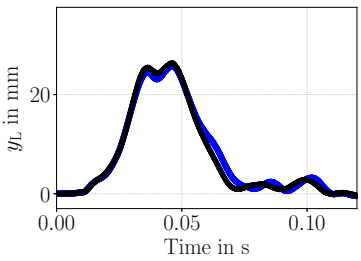
(10) Validation curve 10



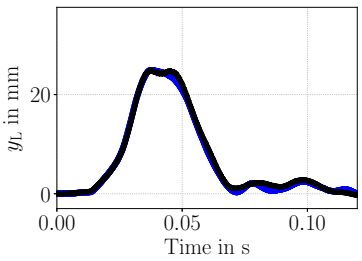
(11) Validation curve 11



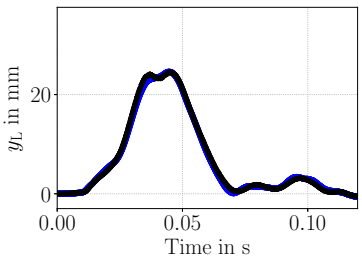
(12) Validation curve 12



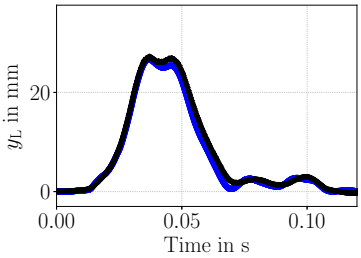
(13) Validation curve 13



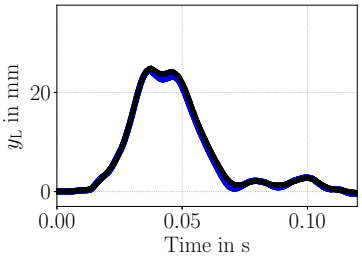
(14) Validation curve 14



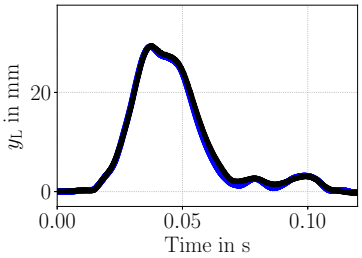
(15) Validation curve 15



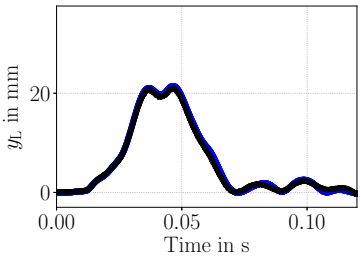
(16) Validation curve 16



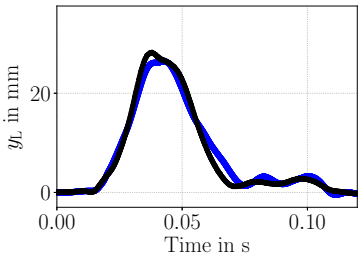
(17) Validation curve 17



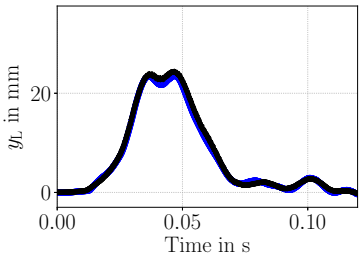
(18) Validation curve 18



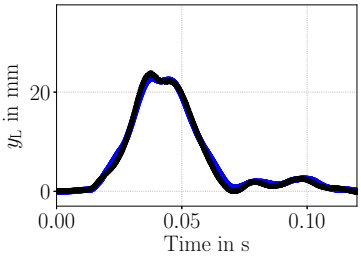
(19) Validation curve 19



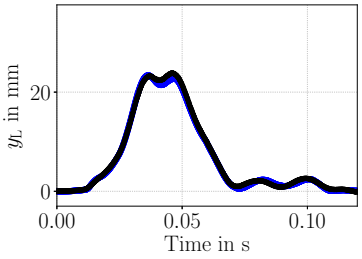
(20) Validation curve 20



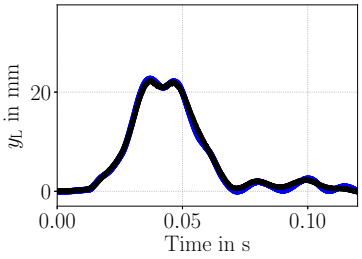
(21) Validation curve 21



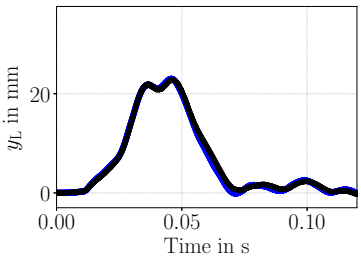
(22) Validation curve 22



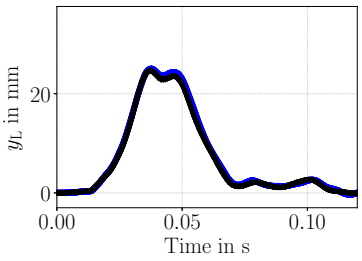
(23) Validation curve 23



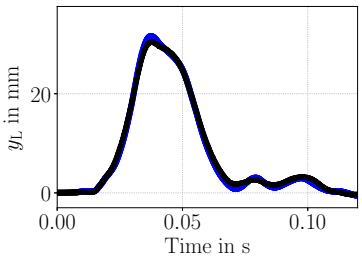
(24) Validation curve 24



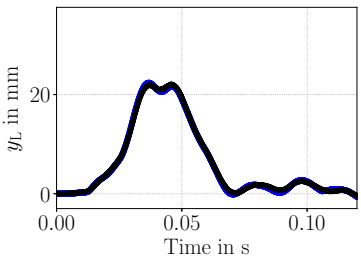
(25) Validation curve 25



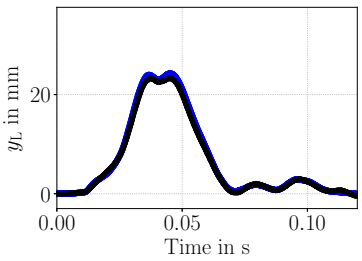
(26) Validation curve 26



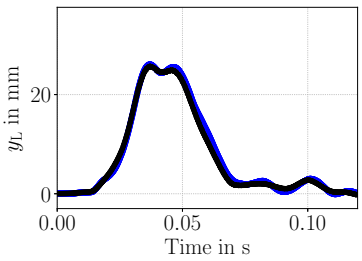
(27) Validation curve 27



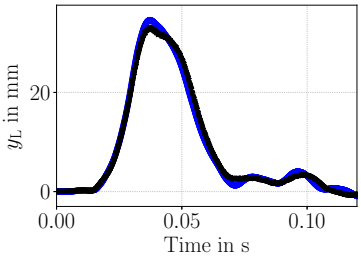
(28) Validation curve 28



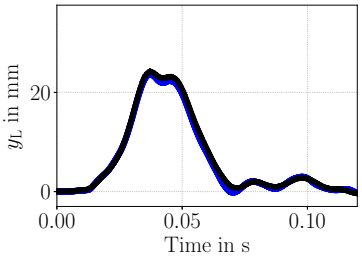
(29) Validation curve 29



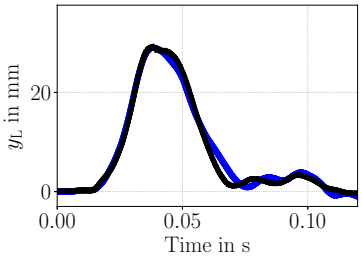
(30) Validation curve 30



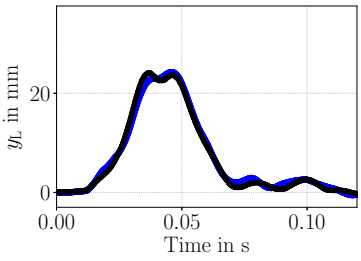
(31) Validation curve 31



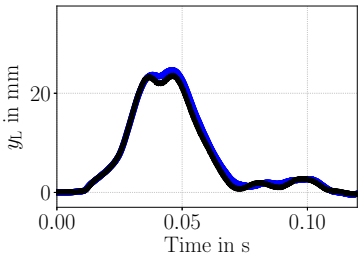
(32) Validation curve 32



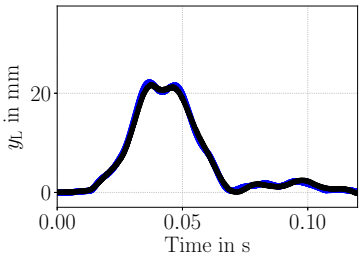
(33) Validation curve 33



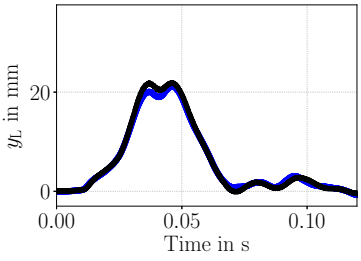
(34) Validation curve 34



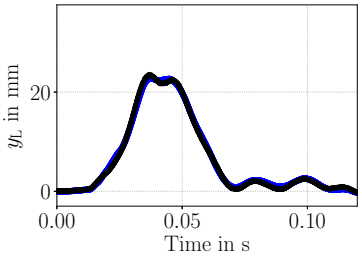
(35) Validation curve 35



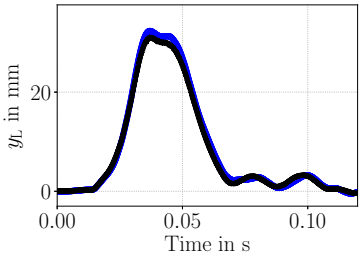
(36) Validation curve 36



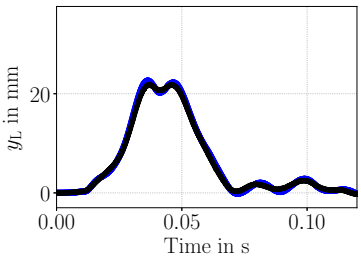
(37) Validation curve 37



(38) Validation curve 38



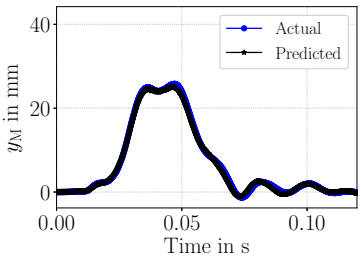
(39) Validation curve 39



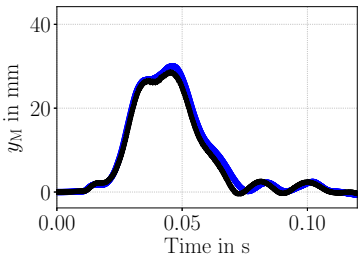
(40) Validation curve 40

Figure 41: Actual and predicted time histories of the $n_{\text{val}} = 40$ lower chest deflection curves, y_L, of the real-world project. The predictions are produced by MNN-1-290.

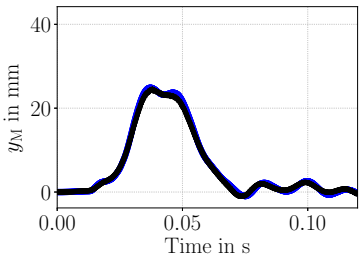
A.2.2 Middle chest deflection curves



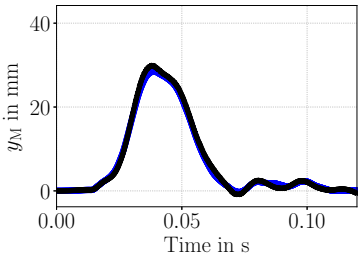
(1) Validation curve 1



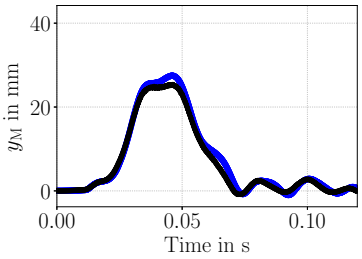
(2) Validation curve 2



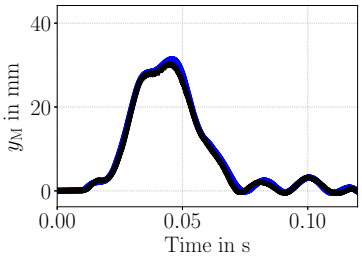
(3) Validation curve 3



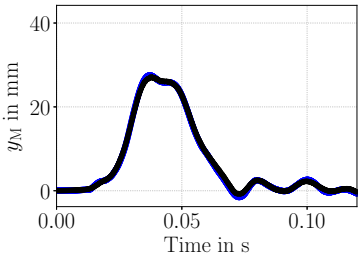
(4) Validation curve 4



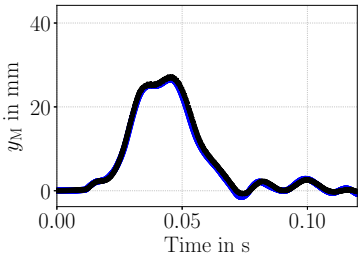
(5) Validation curve 5



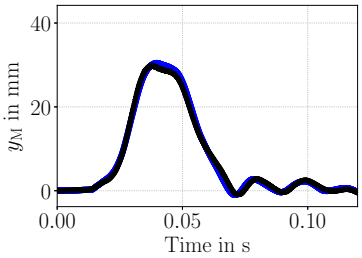
(6) Validation curve 6



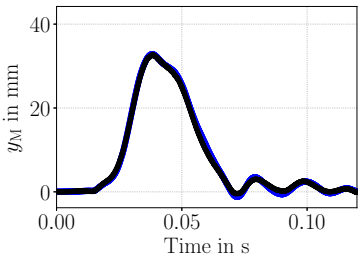
(7) Validation curve 7



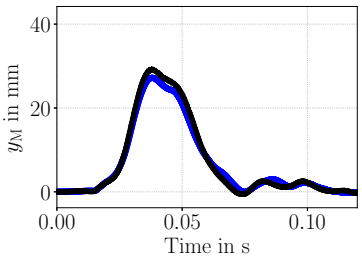
(8) Validation curve 8



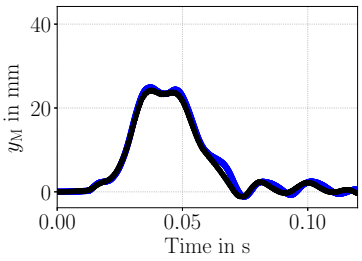
(9) Validation curve 9



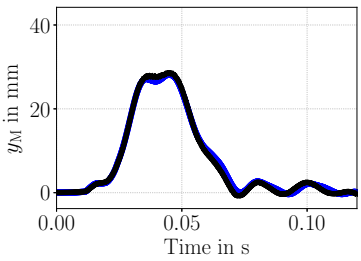
(10) Validation curve 10



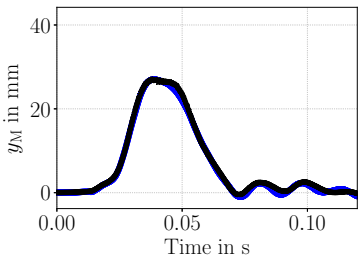
(11) Validation curve 11



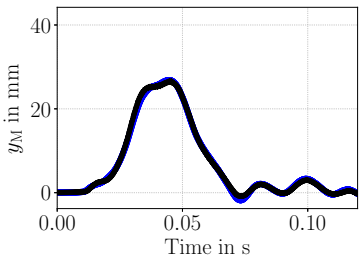
(12) Validation curve 12



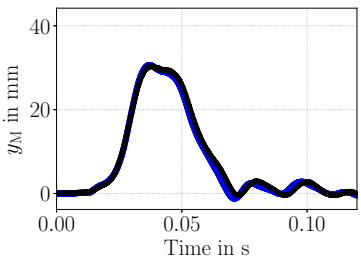
(13) Validation curve 13



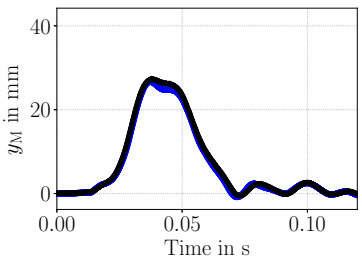
(14) Validation curve 14



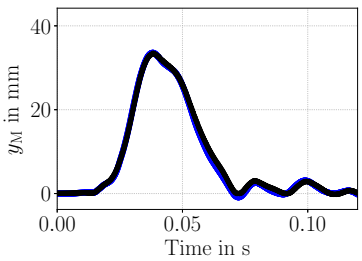
(15) Validation curve 15



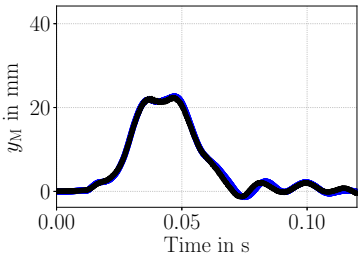
(16) Validation curve 16



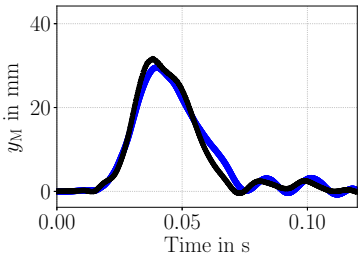
(17) Validation curve 17



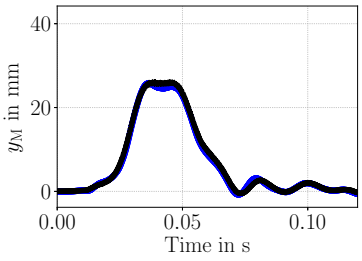
(18) Validation curve 18



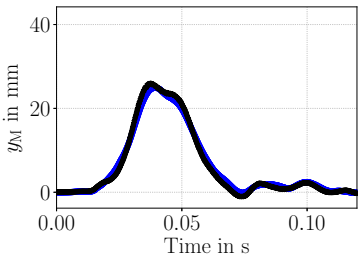
(19) Validation curve 19



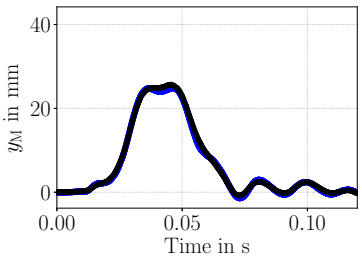
(20) Validation curve 20



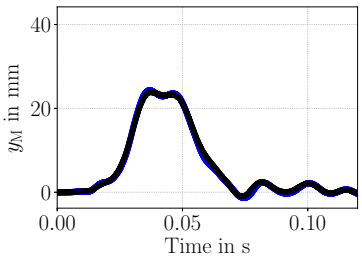
(21) Validation curve 21



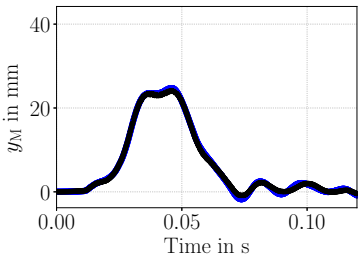
(22) Validation curve 22



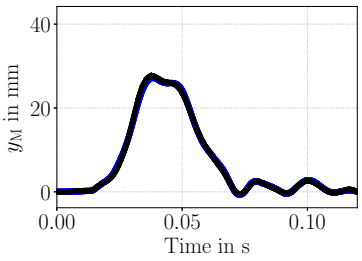
(23) Validation curve 23



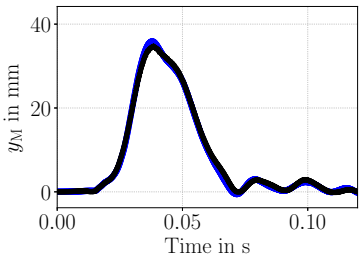
(24) Validation curve 24



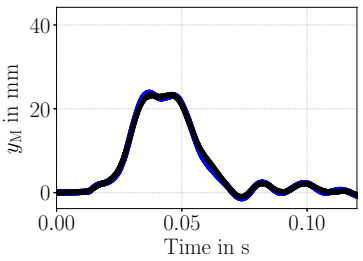
(25) Validation curve 25



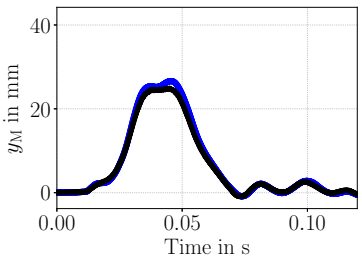
(26) Validation curve 26



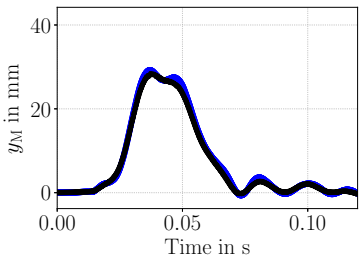
(27) Validation curve 27



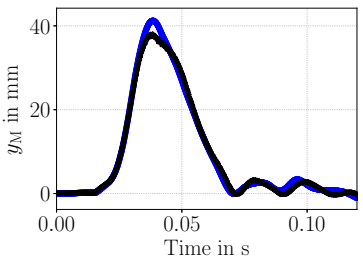
(28) Validation curve 28



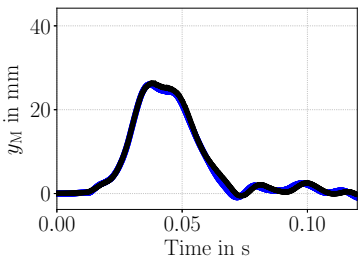
(29) Validation curve 29



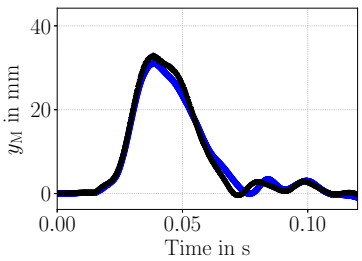
(30) Validation curve 30



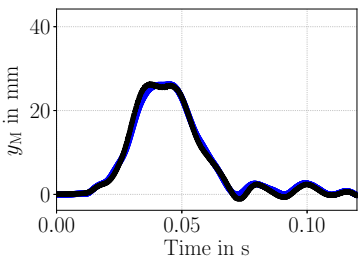
(31) Validation curve 31



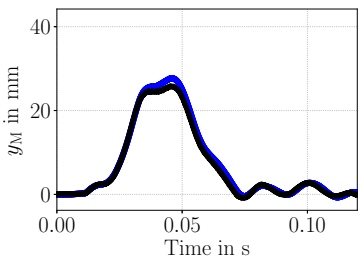
(32) Validation curve 32



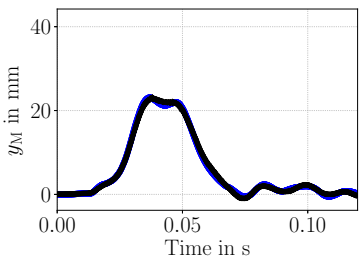
(33) Validation curve 33



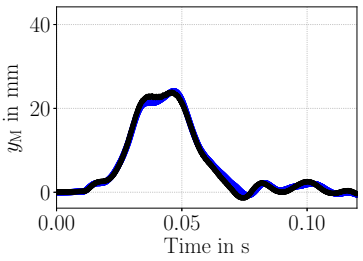
(34) Validation curve 34



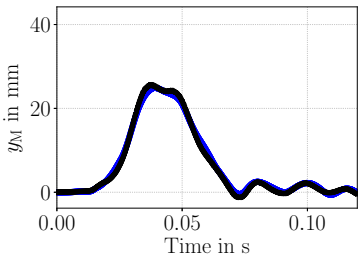
(35) Validation curve 35



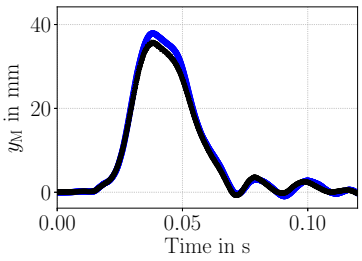
(36) Validation curve 36



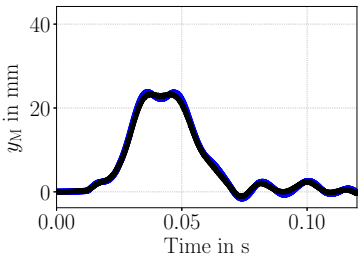
(37) Validation curve 37



(38) Validation curve 38



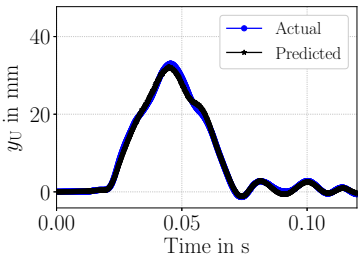
(39) Validation curve 39



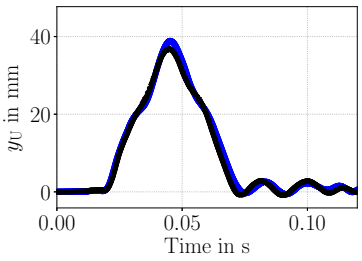
(40) Validation curve 40

Figure 42: Actual and predicted time histories of the $n_{val} = 40$ middle chest deflection curves, y_M, of the real-world project. The predictions are produced by MNN-1-290.

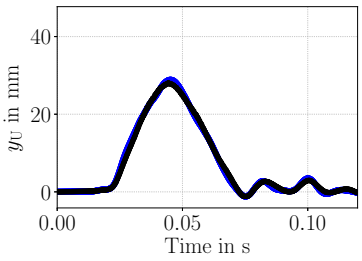
A.2.3 Upper chest deflection curves



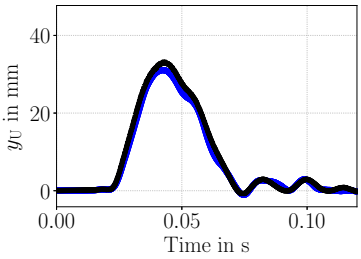
(1) Validation curve 1



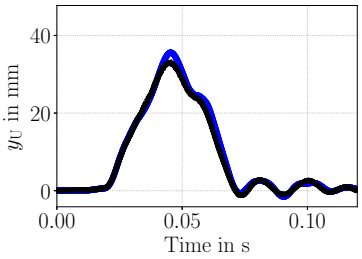
(2) Validation curve 2



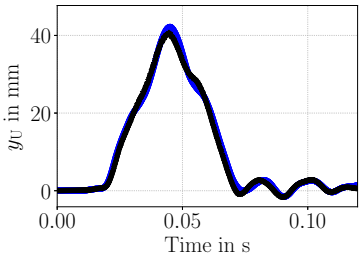
(3) Validation curve 3



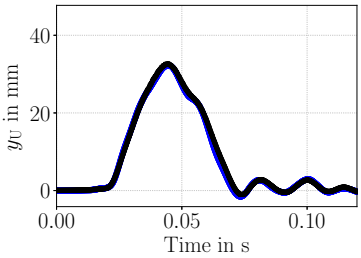
(4) Validation curve 4



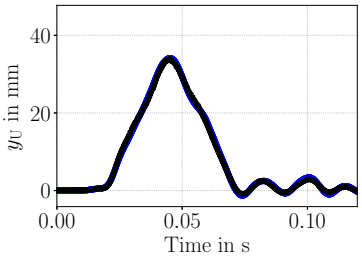
(5) Validation curve 5



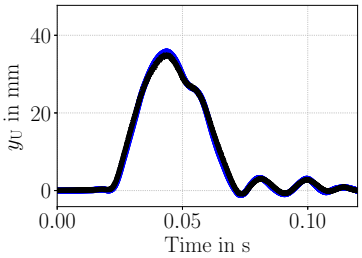
(6) Validation curve 6



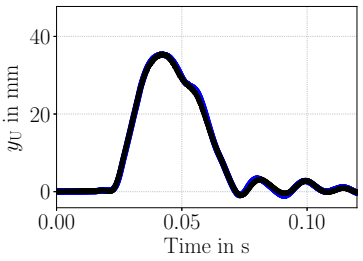
(7) Validation curve 7



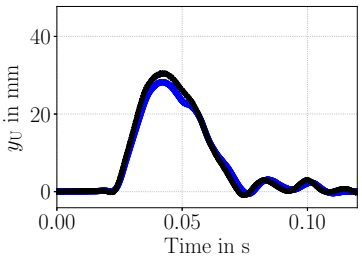
(8) Validation curve 8



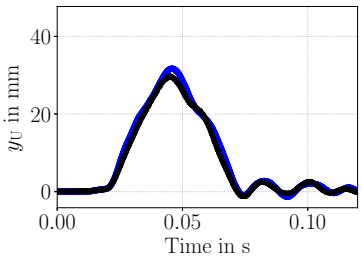
(9) Validation curve 9



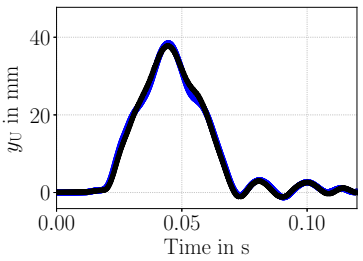
(10) Validation curve 10



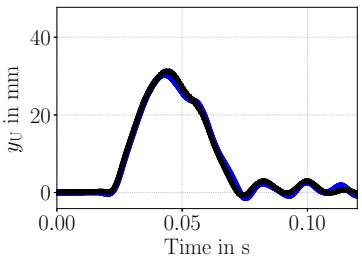
(11) Validation curve 11



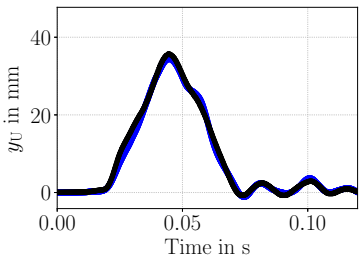
(12) Validation curve 12



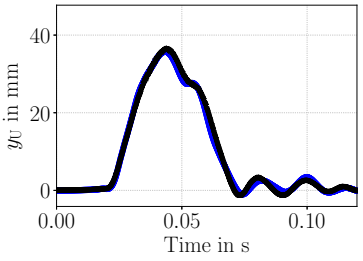
(13) Validation curve 13



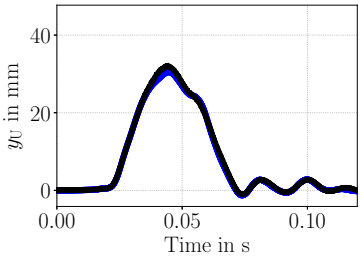
(14) Validation curve 14



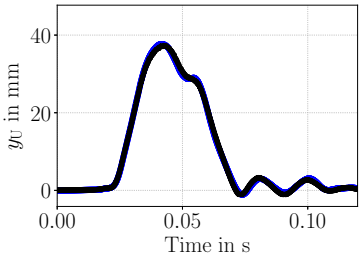
(15) Validation curve 15



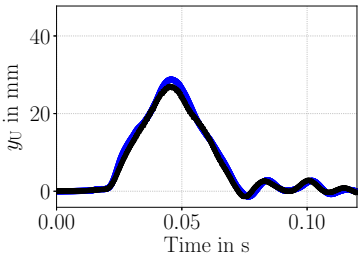
(16) Validation curve 16



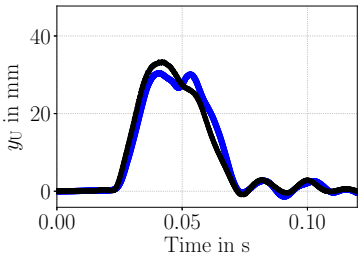
(17) Validation curve 17



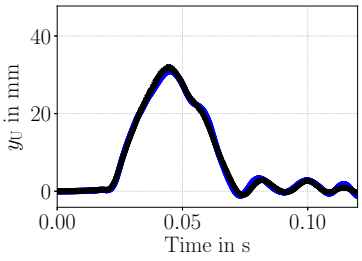
(18) Validation curve 18



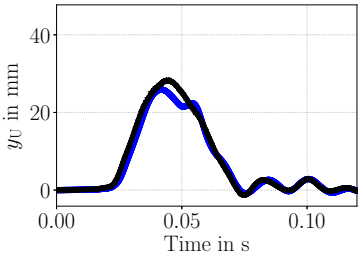
(19) Validation curve 19



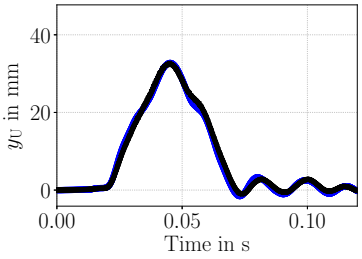
(20) Validation curve 20



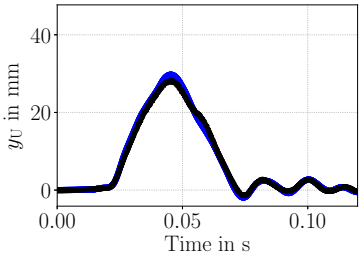
(21) Validation curve 21



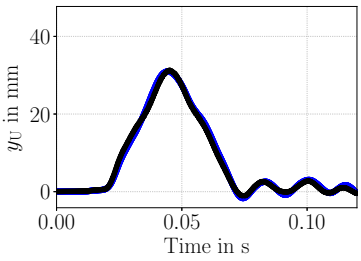
(22) Validation curve 22



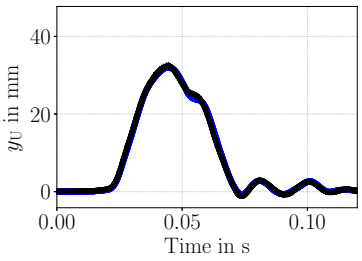
(23) Validation curve 23



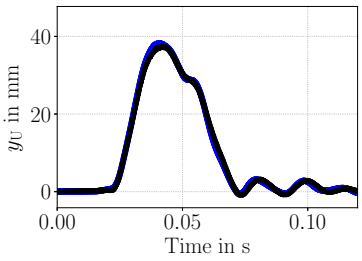
(24) Validation curve 24



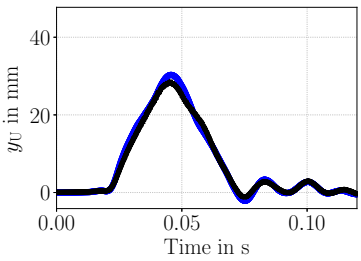
(25) Validation curve 25



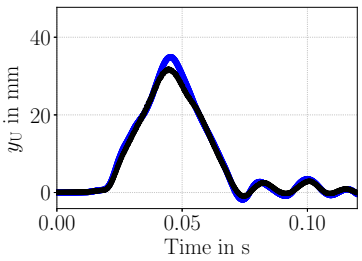
(26) Validation curve 26



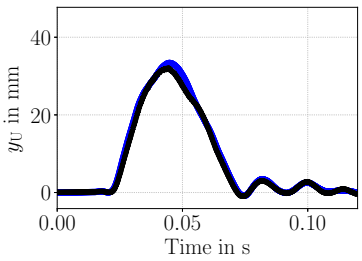
(27) Validation curve 27



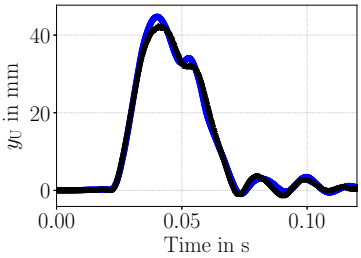
(28) Validation curve 28



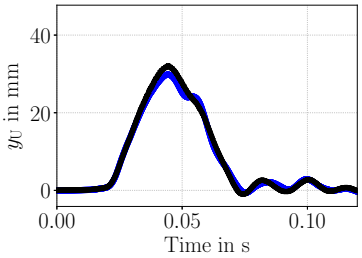
(29) Validation curve 29



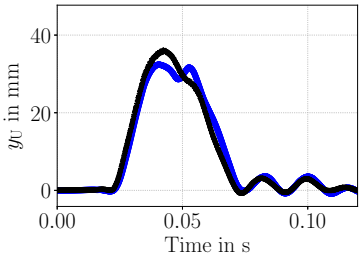
(30) Validation curve 30



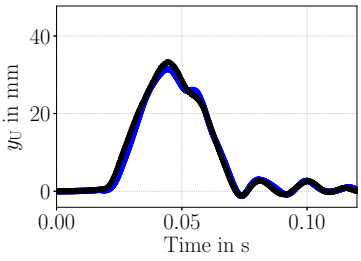
(31) Validation curve 31



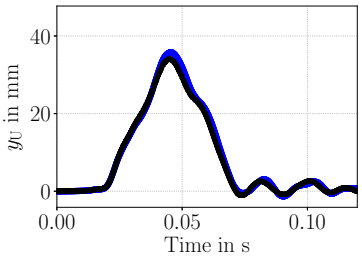
(32) Validation curve 32



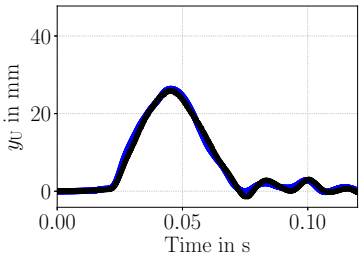
(33) Validation curve 33



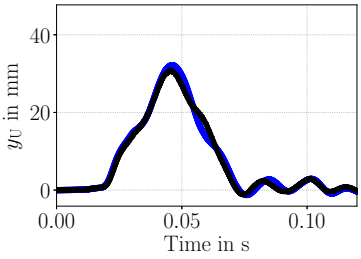
(34) Validation curve 34



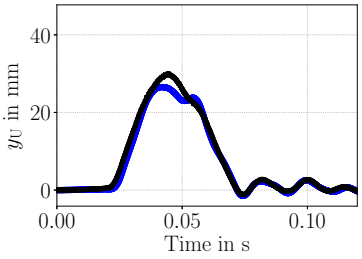
(35) Validation curve 35



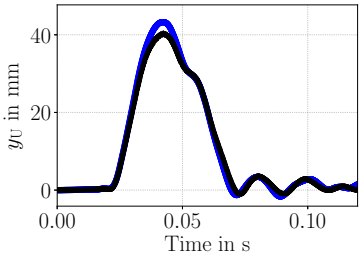
(36) Validation curve 36



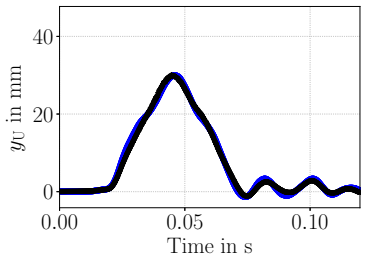
(37) Validation curve 37



(38) Validation curve 38



(39) Validation curve 39



(40) Validation curve 40

Figure 43: Actual and predicted time histories of the $n_{\text{val}} = 40$ upper chest deflection curves, y_{U}, of the real-world project. The predictions are produced by MNN-2-110.

References

[AAB+15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`, 2015. Online; accessed 26-August-2021.

[ABH+17] M. Azmeer, M. H. Basha, M. F. Hamid, M. T. A. Rahman, and M. S. M. Hashim. Design optimization of rear uprights for UniMAP automotive racing team Formula SAE racing car. In *Journal of physics: conference series*, volume 908, page 012051. IOP Publishing, 2017.

[Abr05] A. Abraham. Artificial neural networks. In P. H. Sydenham and R. Thorn, editors, *Handbook of measuring system design*, chapter 129. John Wiley & Sons, 2005.

[ACA11] P. D. Arendt, W. Chen, and D. Apley. Improving identifiability in model calibration using multiple responses. In *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE 2011*, pages 1213–1222, 2011.

[AK06] B. M. Ayyub and G. J. Klir. *Uncertainty modeling and analysis in engineering and the sciences*. CRC Press, 2006.

[AK15] N. Altman and M. Krzywinski. Points of significance: simple linear regression. *Nature methods*, 12(11), 2015.

[AKVS03] A. Abusam, K. J. Keesman, and G. Van Straten. Forward and backward uncertainty propagation: an oxidation ditch modelling example. *Water Research*, 37(2):429–435, 2003.

[AL08] M. A. Alvarez and N. D. Lawrence. Sparse convolved Gaussian processes for multi-output regression. In *NIPS*, volume 21, pages 57–64, 2008.

[ALRL10] E. Auer, W. Luther, G. Rebner, and P. Limbourg. A verified MATLAB toolbox for the Dempster-Shafer theory. *Proceedings of the Workshop on the Theory of Belief Functions*, 2010.

[Ash08] R. B. Ash. *Basic probability theory*. Courier Corporation, 2008.

[AZI16] H. Anysz, A. Zbiciak, and N. Ibadov. The influence of input data standardization method on prediction accuracy of artificial neural networks. *Procedia Engineering*, 153:66–70, 2016.

[Bac19] C. Bach. *Data-driven model order reduction for nonlinear crash and impact simulations*. PhD thesis, Technical University of Munich, 2019.

[Bar15] R. R. Barton. Tutorial: simulation metamodeling. In *2015 Winter simulation conference (WSC)*, pages 1765–1779. IEEE, 2015.

[Bau98] H. Bauer. *Fahrsicherheitssysteme*. Vieweg+Teubner Verlag, Wiesbaden, Hessen, 1998.

[BBOM16] M. A. Bouhlel, N. Bartoli, A. Otsmane, and J. Morlier. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53(5):935–952, 2016.

[BBV17] S. Banholzer, D. Beermann, and S. Volkwein. POD-based error control for reduced-order bicriterial PDE-constrained optimization. *Annual Reviews in Control*, 44:226–237, 2017.

[Ber19] D. Berrar. Cross-validation. In S. Ranganathan, K. Nakai, and C. Schönbach, editors, *Encyclopedia of bioinformatics and computational biology: ABC of bioinformatics*, volume 1, pages 542–545. Elsevier, 2019.

[BGC04] H.-R. Bae, R. V. Grandhi, and R. A. Canfield. An approximation approach for uncertainty quantification using evidence theory. *Reliability Engineering & System Safety*, 86(3):215–225, 2004.

[BJH11] S. Burhenne, D. Jacob, and G. P. Henze. Sampling based on Sobol' sequences for Monte Carlo techniques applied to building simulations. In *Proceedings of the international conference building simulation*, pages 1816–1823, 2011.

[BL10] R. G. Brereton and G. R. Lloyd. Support vector machines for classification and regression. *Analyst*, 135(2):230–267, 2010.

[BMM18] D. Betancourt, R. L. Muhanna, and R. L. Mullen. Interval field for spatially and temporally dependent uncertainty–machine learning approach. In *Proceedings of the 8th international workshop on reliable engineering computing*, volume 8, 2018.

[BMNHG18] E. Borgomeo, M. Mortazavi-Naeini, J. W. Hall, and B. P. Guillod. Risk, robustness and water resources planning under uncertainty. *Earth's Future*, 6(3):468–487, 2018.

[BMW21] BMW Group. Online catalog of new cars. `https://www.bmw.de/de/neufahrzeuge.html`, 2021. Online; accessed 10-June-2021.

[Bor07] E. Borgonovo. A new uncertainty importance measure. *Reliability Engineering & System Safety*, 92(6):771–784, 2007.

[Bot91] L. Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8):12, 1991.

[Bra17] S. Brams. Entwicklung eines Dummy-Ersatzmodells zur Lösungsraumidentifikation und Optimierung der Rückhaltesysteme im Seitencrash. Master's thesis, Technische Hochschule Ingolstadt, 2017.

[BS07] H.-G. Beyer and B. Sendhoff. Robust optimization – a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, 2007.

[BS08] G. Blatman and B. Sudret. Sparse polynomial chaos expansions and adaptive stochastic finite elements using a regression approach. *Comptes Rendus Mécanique*, 336(6):518–523, 2008.

[BS10] G. Blatman and B. Sudret. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, 25(2):183–197, 2010.

[BS11] G. Blatman and B. Sudret. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of computational Physics*, 230(6):2345–2367, 2011.

[BSV14] P. Benner, E. Sachs, and S. Volkwein. Model order reduction for PDE constrained optimization. In G. Leugering, P. Benner, S. Engell, A. Griewank, H. Harbrecht, M. Hinze, R. Rannacher, and S. Ulbrich, editors, *Trends in PDE constrained optimization*, pages 303–326. Springer, 2014.

[BVBL15] H. Borchani, G. Varando, C. Bielza, and P. Larranaga. A survey on multi-output regression. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(5):216–233, 2015.

[BW19] S. Bock and M. Weiß. A proof of local convergence for the ADAM optimizer. In *2019 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2019.

[CAE11] M. Carvalho, J. Ambrósio, and P. Eberhard. Identification of validated multibody vehicle models for crash analysis using a hybrid optimization procedure. *Structural and Multidisciplinary Optimization*, 44(1):85–97, 2011.

[car21] carhs gmbh. SafetyCompanion 2021, 2021.

[CCKL19] J. Choi, D. Chun, H. Kim, and H.-J. Lee. Gaussian YOLOv3: an accurate and fast object detector using localization uncertainty for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 502–511, 2019.

[CCS07] F. Campolongo, J. Cariboni, and A. Saltelli. An effective screening design for sensitivity analysis of large models. *Environmental Modelling & Software*, 22:1509–1518, 2007.

[CCSS05] F. Campolongo, J. Cariboni, A. Saltelli, and W. Schoutens. Enhancing the Morris method. *Proceedings of the 4th International Conference on Sensitivity Analysis of Model Output (SAMO 2004)*, pages 369–379, 2005.

[CD14] T. Chai and R. R. Draxler. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014.

[CDW14] P. G. Constantine, E. Dow, and Q. Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *SIAM Journal on Scientific Computing*, 36(4):A1500–A1524, 2014.

[CEW15] P. G. Constantine, A. Eftekhari, and M. B. Wakin. Computing active subspaces efficiently with gradient sketching. In *2015 IEEE 6th international workshop on computational advances in multi-sensor adaptive processing (CAMSAP)*, pages 353–356. IEEE, 2015.

[CFP02] H. Christopher Frey and S. R. Patil. Identification and review of sensitivity analysis methods. *Risk analysis*, 22(3):553–578, 2002.

[CGEO15] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson. MPDM: multipolicy decision-making in dynamic, uncertain environments for autonomous driving. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 1670–1677. IEEE, 2015.

[CHB09] J. A. Cottrell, T. J. R. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[Cho15] F. Chollet. Keras. `https://keras.io`, 2015. Online; accessed 26-August-2021.

[CI12] T. Chantrasmi and G. Iaccarino. Forward and backward uncertainty propagation for discontinuous system response using the Pade-Legendre method. *International Journal for Uncertainty Quantification*, 2(2), 2012.

[Cot79] S. C. Cotter. A screening design for factorial experiments with interactions. *Biometrika*, 66(2):317–320, 1979.

[COZ+] D. Cichos, M. Otto, S. Zölsch, S. Clausnitzer, D. D. Vetter, and G. Pfeiffer. Crash Analyse Kriterien, Version 2.4. Technical report, Arbeitskreis Messdatenverarbeitung Fahrzeugsicherheit MDVFS (TÜV Rheinland, National Instruments, IAT, Porsche, Task Force ISO TS 13499, Ford, Delphi), Arbeitsgruppe Algorithmen. `http://mdvfs.org/download/crash-analyse-kriterien-2-4/`, 2015. Online; accessed 01-November-2021.

[CSC11] F. Campolongo, A. Saltelli, and J. Cariboni. From screening to quantitative sensitivity analysis. A unified approach. *Computer Physics Communications*, 182:978–988, 2011.

[CWYH19] L. Chen, H. Wang, F. Ye, and W. Hu. Comparative study of HDMRs and other popular metamodeling techniques for high dimensional problems. *Structural and Multidisciplinary Optimization*, 59(1):21–42, 2019.

[Dek20] G. Dekermenjian. B-spline regression as a metamodel for applications in vehicle safety. Master's thesis, Technical University of Munich, 2020.

[Dem67] A. P. Dempster. Upper and lower probabilities induced by a multivalued mapping. *The Annals of Mathematical Statistics*, 38(2):325–339, 1967.

[Dev06] L. Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 13:83–121, 2006.

[DKD09] A. Der Kiureghian and O. Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 32(2):105–112, 2009.

[DMGLGR16] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi. Mean absolute percentage error for regression models. *Neurocomputing*, 192:38–48, 2016.

[DMS21] M. Daub, S. Marelli, and B. Sudret. On constrained distribution-free p-boxes and their propagation. *9th International Workshop on Reliable Engineering Computing (REC2021)*, 2021.

[Dod08] Yadolah Dodge. *The concise encyclopedia of statistics*. Springer Science & Business Media, 2008.

[Don00] D. L. Donoho. High-dimensional data analysis: the curses and blessings of dimensionality. *AMS math challenges lecture*, 1(2000):1–32, 2000.

[DP01] D. Dubois and H. Prade. Possibility theory, probability theory and multiple-valued logics: a clarification. *Annals of mathematics and Artificial Intelligence*, 32(1):35–66, 2001.

[Dub83] O. Dubrule. Cross validation of kriging in a unique neighborhood. *Journal of the International Association for Mathematical Geology*, 15(6):687–699, 1983.

[Dur19] R. Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[DW15] F. Duddeck and E. Wehrle. Recent advances on surrogate modeling for robustness assessment of structures with respect to crashworthiness requirements. In *10th European LS-DYNA conference*, 2015.

[ES09] M. S. Eldred and L. P. Swiler. Efficient algorithms for mixed aleatory-epistemic uncertainty quantification with application to radiation-hardened electronics, part I : algorithms and benchmark results, SAND2009-5805. Technical report, Sandia National Laboratories, 2009.

[EST11] M. S. Eldred, L. P. Swiler, and G. Tang. Mixed aleatory-epistemic uncertainty quantification with stochastic expansions and optimization-based interval estimation. *Reliability Engineering & System Safety*, 96(9):1092–1113, 2011.

[Eur20] European New Car Assessment Programme. Assessment protocol – adult occupant protection, version 9.1.2, 2020.

[FCJR17] R. R.-M. Fonseca, B. Chen, J. D. Jansen, and A. Reynolds. A stochastic simplex approximate gradient (StoSAG) for optimization under uncertainty. *International Journal for Numerical Methods in Engineering*, 109(13):1756–1776, 2017.

[FEA15] Abaqus FEA. *Theory guide*. Dassault Systèmes, Vélizy-Villacoublay, Île-de-France, 2015.

[FEA16] Abaqus FEA. *Version 2016*. Dassault Systèmes, Vélizy-Villacoublay, Île-de-France, 2016.

[Fen13] J. Fender. *Solution spaces for vehicle crash design*. PhD thesis, Technical University of Munich, 2013.

[FHK+16] L. Fahrmeir, C. Heumann, R. Künstler, I. Pigeot, and G. Tutz. *Statistik: der Weg zur Datenanalyse*. Springer, 2016.

[FKG+15] S. Ferson, V. Kreinovich, L. Grinzburg, D. Myers, and K. Sentz. Constructing probability boxes and Dempster-Shafer structures. Technical report, Sandia National Laboratories, 2015.

[FL15] J. Feinberg and H. P. Langtangen. Chaospy: an open source tool for designing methods of uncertainty quantification. *Journal of Computational Science*, 11:46–57, 2015.

[GB16] M. Gu and J. O. Berger. Parallel partial Gaussian process emulation for computer models with massive output. *The Annals of Applied Statistics*, 10(3):1317–1347, 2016.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[GCW+20] N. Gray, D. Calleja, A. Wimbush, E. Miralles-Dolz, A. Gray, M. De Angelis, E. Derrer-Merk, B. U. Oparaji, V. Stepanov, L. Clearkin, and S. Ferson. Is no test better than a bad test: impact of diagnostic uncertainty in mass testing on the spread of COVID-19. *PloS one*, 15(10):e0240775, 2020.

[Ger11] M. Gerdts. *Optimal control of ODEs and DAEs*. Walter de Gruyter, 2011.

[GH18] M. Guo and J. S. Hesthaven. Reduced order modeling for nonlinear structural analysis using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 341:807–826, 2018.

[GH19] M. Guo and J. S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer Methods in Applied Mechanics and Engineering*, 345:75–99, 2019.

[GHO17] R. Ghanem, D. Higdon, and H. Owhadi. *Handbook of uncertainty quantification*, volume 6. Springer, 2017.

[GHV20] C. Gräßle, M. Hinze, and S. Volkwein. Model order reduction by proper orthogonal decomposition. In P. Benner, S. Grivet-Talocia, A. Quarteroni, G. Rozza, W. Schilders, and L. M. Silveira, editors, *Model order reduction volume 2: snapshot-based methods and algorithms*, pages 47–96. De Gruyter, 2020.

[GJP13] A. Gutermuth, U. Jung, and M. Pitzer. Komponenten-Berechnungsmodelle von Pkw-Karosserien. *ATZ-Automobiltechnische Zeitschrift*, 115(9):722–729, 2013.

[Go21] L. S. Go. Non-intrusive model order reduction approaches for crash applications. Master's thesis, Technical University of Munich, 2021.

[Gro16] M. Gross. A planet with two billion cars. *Current Biology*, 26(8):R307–R310, 2016.

[Guo21] Guofei. Scikit-opt. `https://github.com/guofei9987/scikit-opt`, 2021. Online; accessed 26-August-2021.

[GV17] M. Gubisch and S. Volkwein. Proper orthogonal decomposition for linear-quadratic optimal control. *Model reduction and approximation: theory and algorithms*, 5:66, 2017.

[Hay10] S. Haykin. *Neural networks and learning machines*. Pearson Education India, 3 edition, 2010.

[HD03] J. C. Helton and F. J. Davis. Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems. *Reliability Engineering & System Safety*, 81(1):23–69, 2003.

[HDY+12] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, Jaitly N., A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 2012.

[Her21] M. Herold. Improvement of non-intrusive surrogate modeling by combining random feature expansion with dimensionality reduction into an end-to-end optimization problem. Master's thesis, Technical University of Munich, 2021.

[HFRD13] T. Helldin, G. Falkman, M. Riveiro, and S. Davidsson. Presenting system uncertainty in automotive UIs for supporting trust calibration in autonomous driving. In *Proceedings of the 5th international conference on automotive user interfaces and interactive vehicular applications*, pages 210–217, 2013.

[HHB94] J. S. Hammonds, F. O. Hoffman, and S. M. Bartell. An introductory guide to uncertainty analysis in environmental and health risk assessment. Technical report, Senes Oak Ridge, Inc., 1994.

[HHMR08] J. Halgrin, G. Haugou, E. Markiewicz, and L. Rota. Integrated simplified crash modelling approach dedicated to pre-design stage: evaluation on a front car part. *International Journal of Vehicle Safety*, 3(1):91–115, 2008.

[HJS06] J. C. Helton, W. L. Johnson, J. D. Oberkampf, and C. B. Storlie. A sampling-based computational strategy for the representation of epistemic uncertainty in model predictions with evidence theory, SAND2006-5557. Technical report, Sandia National Laboratories, 2006.

[HK06] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International journal of forecasting*, 22(4):679–688, 2006.

[HL98] Y. Hua and W. Liu. Generalized Karhunen-Loève transform. *IEEE Signal Processing Letters*, 5(6):141–142, 1998.

[HP02] Y.-C. Ho and D. L. Pepyne. Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570, 2002.

[HPR17] L. E. Hale, M. Patil, and C. J. Roy. Aerodynamic parameter identification and uncertainty quantification for small unmanned aircraft. *Journal of Guidance, Control, and Dynamics*, 40(3):680–691, 2017.

[HQZ+15] Z. Huang, H. Qiu, M. Zhao, X. Cai, and L. Gao. An adaptive SVR-HDMR model for approximating high dimensional problems. *Engineering Computations*, 2015.

[HS96] T. Homma and A. Saltelli. Importance measures in global sensitivity analysis of nonlinear models. *Reliability Engineering and System Safety*, 52:1–17, 1996.

[HS97] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSS08] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel methods in machine learning. *The annals of statistics*, 36(3):1171–1220, 2008.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[HU17] J. Herman and W. Usher. SALib: an open-source Python library for sensitivity analysis. *Journal of Open Source Software*, 2(9):97, 2017.

[HU18] J. S. Hesthaven and S. Ubbiali. Non-intrusive reduced order modeling of nonlinear problems using neural networks. *Journal of Computational Physics*, 363:55–78, 2018.

[IL15] B. Iooss and P. Lemaître. A review on global sensitivity analysis methods. In G. Dellino and C. Meloni, editors, *Uncertainty management in simulation-optimization of complex systems: algorithms and applications*, pages 101–122. Springer, New York City, 2015.

[IS15] S. Ioffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[IS17] B. Iooss and A. Saltelli. Introduction to sensitivity analysis. In R. Ghanem, D. Higdon, and H. Owhadi, editors, *Handbook of uncertainty quantification*, pages 1103–1122. Springer, Cham, 2017.

[Jac12] D. Jackson. *Fourier series and orthogonal polynomials*. Courier Corporation, 2012.

[Jan96] M. J. W. Jansen. Analysis of variance designs for model output. *Computer Physics Communications*, 117:35–43, 1996.

[JBB+20] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun. Hands-on Bayesian neural networks-a tutorial for deep learning users. *ACM Computing Surveys*, 1(1), 2020.

[JKDW01] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. Interval analysis. In *Applied interval analysis*, pages 11–43. Springer, 2001.

[JLG21a] J. S. Jehle, V. A. Lange, and M. Gerdts. Enabling the evidence theory through non-intrusive parametric model order reduction for crash simulations. *9th International Workshop on Reliable Engineering Computing (REC2021)*, 2021.

[JLG21b] J. S. Jehle, V. A. Lange, and M. Gerdts. Proposing an uncertainty management framework to implement the evidence theory for vehicle crash applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part B: Mechanical Engineering*, 2021. Accepted for publication.

[JNR05] A. Jain, K. Nandakumar, and A. Ross. Score normalization in multimodal biometric systems. *Pattern recognition*, 38(12):2270–2285, 2005.

[Joe15] Grus Joel. *Data science from scratch*. O'Reilly Media, 2015.

[Joh83] B. G. Johnston. Column buckling theory: historic highlights. *Journal of Structural Engineering*, 109(9):2086–2096, 1983.

[Jol02] I. T. Jolliffe. *Principal component analysis*. Springer, 2 edition, 2002.

[Jos06] V. R. Joseph. Limit kriging. *Technometrics*, 48(4), 2006.

[KB20] A. Kratsios and I. Bilokopytov. Non-euclidean universal approximation. In *NeurIPS Proceedings, Advances in Neural Information Processing Systems*, volume 33, 2020.

[KBTB14] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev. Why the Monte Carlo method is so important today. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):386–392, 2014.

[KE19] J. N. Kani and A. H. Elsheikh. Reduced-order modeling of subsurface multi-phase flow models using deep residual recurrent neural networks. *Transport in Porous Media*, 126(3):713–741, 2019.

[KGD10] K. Kompass, C. Gruber, and C. Domsch. Der Beitrag von Fahrerassistenzsystemen zur aktiven und passiven Sicherheit – die integrale Sicherheit als Antwort auf die wachsenden Anforderungen an die Fahrzeugsicherheit. In *4. Tagung Sicherheit durch Fahrerassistenz, TÜV SÜD und TU München*, 2010.

[KGE09] L. Kübler, S. Gargallo, and K. Elsäßer. Frontal crash pulse assessment with application to occupant safety. *ATZ worldwide*, 111(6):12–17, 2009.

[KGH20] M. Kast, M. Guo, and J. S. Hesthaven. A non-intrusive multifidelity method for the reduced order modeling of nonlinear problems. *Computer Methods in Applied Mechanics and Engineering*, 364:112947, 2020.

[KGVB05] G. Kerschen, J.-C. Golinval, A. F. Vakakis, and L. A. Bergman. The method of proper orthogonal decomposition for dynamical characterization and order reduction of mechanical systems: an overview. *Nonlinear dynamics*, 41(1):147–169, 2005.

[KH09] K. Kompass and W. Huber. Integrale Sicherheit – Effektive Wertsteigerung in der Fahrzeugsicherheit/Integral safety-effective added value in vehicle safety. In *Fahrzeugsicherheit und Elektronik, Umwelt und Energie. 11. Technischer Kongress des VDA, 25. und 26. März 2009, Volkswagen Wolfsburg. Tagungsband*, 2009.

[KJ13] M. Kuhn and K. Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

[KKL+21] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[KKP06] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas. Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117, 2006.

[KL80] V. Klema and A. Laub. The singular value decomposition: its computation and some applications. *IEEE Transactions on automatic control*, 25(2):164–176, 1980.

[Kle13] A. Klenke. *Probability theory: a comprehensive course*. Springer Science & Business Media, 2013.

[Kli99] G. J. Klir. On fuzzy-set interpretation of possibility theory. *Fuzzy sets and systems*, 108(3):263–273, 1999.

[Kli13] H. Klie. Unlocking fast reservoir predictions via non-intrusive reduced order models. In *SPE Reservoir Simulation Symposium*. OnePetro, 2013.

[KMV18] W. Keiper, A. Milde, and S. Volkwein. *Reduced-order modeling (ROM) for simulation and optimization: powerful algorithms as key enablers for scientific computing*. Springer, 2018.

[KO01] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, 2001.

[Kot07] S. B. Kotsiantis. Supervised machine learning: a review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[Kri51] D. G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139, 1951.

[KTA12] S. Kucherenko, S. Tarantola, and P. Annoni. Estimation of global sensitivity indices for models with dependent variables. *Computer physics communications*, 183(4):937–946, 2012.

[KVR16] S. Kavitha, S. Varuna, and R. Ramya. A comparative analysis on linear regression and support vector regression. In *2016 Online international conference on green engineering and technologies (IC-GET)*, pages 1–5. IEEE, 2016.

[Lan21] V. A. Lange. *Identification of coupled solution spaces for vehicle crash structure and restraint system design*. PhD thesis, Technical University of Munich, 2021.

[LC09] S. H. Lee and W. Chen. A comparative study of uncertainty propagation methods for black-box-type problems. *Structural and multidisciplinary optimization*, 37(239), 2009.

[LCO17] H. Liu, J. Cai, and Y.-S. Ong. An adaptive sampling approach for kriging metamodeling by maximizing expected prediction error. *Computers & Chemical Engineering*, 106:171–182, 2017.

[Lei20] L. Leidinger. *Explicit isogeometric B-rep analysis for nonlinear dynamic crash simulations*. PhD thesis, Technical University of Munich, 2020.

[LFSD18] V. A. Lange, J. Fender, L. Song, and F. Duddeck. Early phase modeling of frontal impacts for crashworthiness: from lumped mass–spring models to deformation space models. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 233(12):3000–3015, 2018.

[LG07] Y. Liu and H. V. Gupta. Uncertainty in hydrologic modeling: toward an integrated data assimilation framework. *Water resources research*, 43(7), 2007.

[Lim08] P. Limbourg. *Dependability modelling under uncertainty*, chapter Representation and propagation of uncertainty using the Dempster-Shafer theory of evidence. Springer, 2008.

[LIPG13] M. Lamboni, B. Iooss, A.-L. Popelin, and F. Gamboa. Derivative-based global sensitivity measures: general links with Sobol' indices and numerical tests. *Mathematics and Computers in Simulation*, 87:45–54, 2013.

[LL11] W. Li and Z. Liu. A method of svm with normalization in intrusion detection. *Procedia Environmental Sciences*, 11:256–262, 2011.

[LMS20] C. Lataniotis, S. Marelli, and B. Sudret. Extending classical surrogate modeling to high dimensions through supervised dimensionality reduction: a data-driven approach. *International Journal for Uncertainty Quantification*, 10(1), 2020.

[LMS21] N. Lüthen, S. Marelli, and B. Sudret. Sparse polynomial chaos expansions: literature survey and benchmark. *SIAM/ASA Journal on Uncertainty Quantification*, 9(2):593–649, 2021.

[LN21] M. Lahoz Navarro. Design and implementation of deep learning models for applications in vehicle safety. Master's thesis, Universidad Politécnica de Madrid, 2021.

[LOC18] H. Liu, Y.-S. Ong, and J. Cai. A survey of adaptive sampling for global metamodeling in support of simulation-based complex engineering design. *Structural and Multidisciplinary Optimization*, 57(1):393–416, 2018.

[Log16] D. L. Logan. *A first course in the finite element method*. Cengage Learning, 6 edition, 2016.

[LS-06] LS-DYNA. *Theory manual*. Livermore Software Technology Corporation, 2006.

[LS-18] LS-DYNA. *Version R9.3*. Livermore Software Technology Corporation, Livermore, California, 2018.

[LS-20] LS-DYNA. Examples homepage. `https://www.dynaexamples.com`, 2020. Online; accessed 26-August-2021.

[LV13] O. Lass and S. Volkwein. POD Galerkin schemes for nonlinear elliptic-parabolic systems. *SIAM Journal on Scientific Computing*, 35(3):A1271–A1298, 2013.

[LWR00] G. Li, S.-W. Wang, and H. Rabitz. High dimensional model representations (HDMR): concepts and applications. In *Proceedings of the Institute of Mathematics and its applicatons workshop on atmospheric modeling*, pages 15–19. Citeseer, 2000.

[LXW16] H. Liu, S. Xu, and X. Wang. Sampling strategies and metamodeling techniques for engineering design: comparison and application. In *ASME turbo expo 2016: turbomachinery technical conference and exposition*. American Society of Mechanical Engineers Digital Collection, 2016.

[MAT18] MATLAB. *Version 9.5.0.944444 (R2018b)*. The MathWorks Inc., Natick, Massachusetts, 2018.

[MBC00] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[MBGS18] M. Moustapha, J.-M. Bourinet, B. Guillaume, and B. Sudret. Comparative study of kriging and support vector regression for structural engineering applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 4(2), 2018.

[MDMDV11] D. Moens, M. De Munck, W. Desmet, and D. Vandepitte. Numerical dynamic analysis of uncertain mechanical structures based on interval fields. In *IUTAM symposium on the vibration analysis of structures with uncertainties*, pages 71–83. Springer, 2011.

[MGB74] A. M. Mood, F. A. Graybill, and D. C. Boes. *Introduction to the theory of statistics*. McGraw-Hill, 3 edition, 1974.

[Mil05] J. Miles. R-squared, adjusted r-squared. In *Encyclopedia of Statistics in Behavioral Science*. John Wiley & Sons, 2005.

[MILR09] A. Marrel, B. Iooss, B. Laurent, and O. Roustant. Calculation of Sobol' indices for the Gaussian process metamodel. *Reliability Engineering and System Safety*, 94(3):742–751, 2009.

[MLBB08] A. A. Miranda, Y.-A. Le Borgne, and G. Bontempi. New routes from minimal approximation error to principal components. *Neural Processing Letters*, 27(3):197–207, 2008.

[Mor91] M. D. Morris. Factorial sampling plans for preliminary computational experiments. *Technometrics*, 33:161–174, 1991.

[MP21] Z. Ma and W. Pan. Data-driven nonintrusive reduced order modeling for dynamical systems with moving boundaries using Gaussian process regression. *Computer Methods in Applied Mechanics and Engineering*, 373:113495, 2021.

[MPV21] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 6 edition, 2021.

[MS18] S. Marelli and B. Sudret. An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis. *Structural Safety*, 75:67–74, 2018.

[MSBG14] M. Moustapha, B. Sudret, J.-M. Bourinet, and B. Guillaume. Metamodeling for crashworthiness design: comparative study of kriging and support vector regression. *Proceedings of the 2nd International Symposium on Uncertainty Quantification and Stochastic Modeling*, 2014.

[MSBG15] M. Moustapha, B. Sudret, J.-M. Bourinet, and B. Guillaume. Adaptive kriging reliability-based design optimization of an automotive body structure under crashworthiness constraints. *Proceedings of the 12th International Conference on Applications of Statistics and Probability in Civil Engineering (ICASP12)*, 2015.

[Mur12] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[Nag19] J. B. Nagel. *Bayesian techniques for inverse uncertainty quantification*. PhD thesis, ETH Zurich, 2019.

[Nat19] National Center for Statistics and Analysis (National Highway Traffic Safety Administration). 2018 fatal motor vehicle crashes: overview. Traffic safety facts - research note. Report no. DOT HS 812 826. `https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812826`, 2019. Online; accessed 26-August-2021.

[NDLT08] M. H. Nguyen and F. De La Torre. Robust kernel principal component analysis. In *Advances in NIPS 21*. Citeseer, 2008.

[NGS04] E. Nikolaidis, D. M. Ghiocel, and S. Singhal. *Engineering design reliability handbook*. CRC Press, 2004.

[NJ18] R. E. Neapolitan and X. Jiang. *Artificial intelligence: with an introduction to machine learning*. CRC Press, 2018.

[NPM12] V. Novák, I. Perfilieva, and J. Mockor. *Mathematical principles of fuzzy logic*, volume 517. Springer Science & Business Media, 2012.

[OAA+17] F. Y. Osisanwo, J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi, and J. Akinjobi. Supervised machine learning algorithms: classification and comparison. *International Journal of Computer Trends and Technology (IJCTT)*, 48(3):128–138, 2017.

[OHJ+85] W. L. Oberkampf, J. C. Helton, C. A. Joslyn, S. F. Wojtkiewicz, and S. Ferson. Challenge problems: uncertainty in system response given uncertain parameters. *Reliability Engineering & System Safety*, 2004(1-3):11–19, 85.

[OK12] P. O'Connor and A. Kleyner. *Practical reliability engineering*. John Wiley & Sons, 2012.

[OSD03] A. Olsson, G. Sandberg, and O. Dahlblom. On Latin hypercube sampling for structural reliability analysis. *Structural safety*, 25(1):47–68, 2003.

[OT02] W. L. Oberkampf and T. G. Trucano. Verification and validation in computational fluid dynamics. *Progress in aerospace sciences*, 38(3):209–272, 2002.

[PBF+16] F. Pianosi, K. Beven, J. Freer, J. W. Hall, J. Rougier, D. B. Stephenson, and T. Wagener. Sensitivity analysis of environmental models: a systematic review with practical workflow. *Environmental Modelling & Software*, 79:214–232, 2016.

[PGL06] S. L. Padula, C. R. Gumbert, and W. Li. Aerospace applications of optimization under uncertainty. *Optimization and Engineering*, 7(3):317–328, 2006.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[PWG18] B. Peherstorfer, K. Willcox, and M. Gunzburger. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review*, 60(3):550–591, 2018.

[Ras03] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[Ray14] M. Rayamajhi. *Efficient methods for robust shape optimisation for crashworthiness*. PhD thesis, Technical University of Munich, 2014.

[Red19] J. N. Reddy. *Introduction to the finite element method*. McGraw-Hill Education, 4 edition, 2019.

[RGGZ+21] M. Rocas, A. García-González, S. Zlotnik, X. Larráyoz, and P. Díez. Non-intrusive uncertainty quantification for automotive crash problems with VPS/Pamcrash. *Finite Elements in Analysis and Design*, 193:103556, 2021.

[RKW15] W. J. Rider, J. R. Kamm, and V. G. Weirs. Verification, validation and uncertainty quantification for CGS. Technical report, Sandia National Laboratories, 2015.

[RN10] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 3 edition, 2010.

[Roh15] J. Rohmer. Boosting kernel-based dimension reduction for jointly propagating spatial variability and parameter uncertainty in long-running flow simulators. *Mathematical Geosciences*, 47(2):227–246, 2015.

[RPK19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[RW06] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, Cambridge, Massachusetts, 2006.

[Saa11] Yousef Saad. *Numerical methods for large eigenvalue problems: revised edition*. SIAM, 2011.

[Sal99] A. Saltelli. Sensitivity analysis: could better methods be used? *Journal of Geophysical Research: Atmospheres*, 104(D3):3789–3793, 1999.

[Sal10a] A. Saltelli. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Computer Physics Communications*, 181:259–270, 2010.

[SAL10b] D. J. Seidel, C. O. Ao, and K. Li. Estimating climatological planetary boundary layer heights from radiosonde observations: comparison of methods and uncertainty analysis. *Journal of Geophysical Research: Atmospheres*, 115(D16), 2010.

[Sar13] R. G. Sargent. Verification and validation of simulation models. *Journal of simulation*, 7(1):12–24, 2013.

[SB19] G. Scheidacker and S. Brams. Simplified model for the WorldSID50 dummy model. Technical report, BMW Group, Total Vehicle Department, 2019.

[Sch15] J. Schmidhuber. Deep learning in neural networks: an overview. *Neural networks*, 61:85–117, 2015.

[SG07] L. P. Swiler and A. A. Giunta. Aleatory and epistemic uncertainty quantification for engineering applications. *Proceedings of the Joint Statistical Meetings, American Statistical Association*, 2007.

[SG10] D. Sperling and D. Gordon. *Two billion cars: driving toward sustainability.* Oxford University Press, 2010.

[SHG20] P. C. Sen, M. Hajra, and M. Ghosh. Supervised classification algorithms in machine learning: a survey and review. In *Emerging technology in modelling and graphics*, pages 99–111. Springer, 2020.

[SK10] I. M. Sobol' and S. Kucherenko. Derivative based global sensitivity measures. *Procedia-Social and Behavioral Sciences*, 2(6):7745–7746, 2010.

[SK19] A. Sagheer and M. Kotb. Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*, 323:203–213, 2019.

[SMA19] O. San, R. Maulik, and M. Ahmed. An artificial neural network framework for reduced order modeling of transient flows. *Communications in Nonlinear Science and Numerical Simulation*, 77:271–287, 2019.

[Sob67] I. M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[Sob76] I. M. Sobol. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*, 16(5):236–242, 1976.

[Sob93] I. M. Sobol'. Sensitivity analysis for non-linear mathematical models. *Mathematical Modelling and Computational Experiment*, 1:407–414, 1993.

[Sob01] I. M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55:271–280, 2001.

[Sou10] C. R. Souza. Kernel functions for machine learning applications. *Creative Commons Attribution-Noncommercial-Share Alike*, 3:29, 2010.

[Spe91] D-F. Specht. A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576, 1991.

[SPM09] L. P. Swiler, T. L. Paez, and R. L. Mayes. Epistemic uncertainty quantification tutorial. *Proceedings of the IMAC-XXVII*, 2009.

[SRA+08] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola. *Global sensitivity analysis. The primer*. Wiley, Chichester, 2008.

[Sri02] R. Srinivasan. *Importance sampling: applications in communications and detection*. Springer Science & Business Media, 2002.

[SRTC05] A. Saltelli, M. Ratto, S. Tarantola, and F. Campolongo. Sensitivity analysis for chemical models. *Chemical reviews*, 105(7):2811–2828, 2005.

[SS17] S. Sharma and S. Sharma. Activation functions in neural networks. *Towards Data Science*, 6(12):310–316, 2017.

[SSM97] B. Schölkopf, A. Smola, and K.-R. Müller. Kernel principal component analysis. In *International conference on artificial neural networks*, pages 583–588. Springer, 1997.

[SSM16] R. Schöbi, B. Sudret, and S. Marelli. Rare event estimation using polynomial-chaos kriging. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 3(2), 2016.

[SSW15] R. Schöbi, B. Sudret, and J. Wiart. Polynomial-chaos-based kriging. *International Journal for Uncertainty Quantification*, 5(2), 2015.

[Sta69] J. D. States. The abbreviated and the comprehensive research injury scales. *SAE Transactions*, pages 2625–2634, 1969.

[STCR04] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto. *Sensitivity analysis in practice: a guide to assessing scientific models*. Wiley, Chichester, West Sussex, 2004.

[Sud08] B. Sudret. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering and System Safety*, 93:964–979, 2008.

[SW10] S. Shan and G. G. Wang. Metamodeling for high dimensional simulation-based design problems. *Journal of Mechanical Design*, 132(5), 2010.

[SWMW89] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–435, 1989.

[SWN03] T. J. Santner, B. J. Williams, and W. I. Notz. *The design and analysis of computer experiments*, volume 1. Springer, New York City, 2003.

[SWV+22] J. Song, P. Wei, M. A. Valdebenito, M. Faes, and M. Beer. Data-driven and active learning of variance-based sensitivity indices with Bayesian probabilistic integration. *Mechanical Systems and Signal Processing*, 163:108106, 2022.

[Tah16] H. Taherdoost. Sampling methods in research methodology; how to choose a sampling technique for research. *International Journal of Academic Research in Management*, 2016.

[TdWL+18] W. Tian, P. de Wilde, Z. Li, J. Song, and B. Yin. Uncertainty and sensitivity analysis of energy assessment for office buildings based on Dempster-Shafer theory. *Energy Conversion and Management*, 174:705–718, 2018.

[TMES19] E. Torre, S. Marelli, P. Embrechts, and B. Sudret. Data-driven polynomial chaos expansion for machine learning regression. *Journal of Computational Physics*, 388:601–623, 2019.

[Tru79] G. V. Trunk. A problem of dimensionality: a simple example. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):306–307, 1979.

[UCD+16] S. Ulaganathan, I. Couckuyt, T. Dhaene, J. Degroote, and E. Laermans. Performance study of gradient enhanced kriging. *Engineering with computers*, 32(1):15–34, 2016.

[Uni16] United Nations Economic Commission for Europe. Agreement concerning the adoption of uniform technical prescriptions for wheeled vehicles, equipment and parts which can be fitted and/or be used on wheeled vehicles and the conditions for reciprocal recognition of approvals granted on the basis of these prescriptions, addendum 134: regulation no. 135, revision 1. `https://unece.org/fileadmin/DAM/trans/main/wp29/wp29regs/2016/R135r1e.pdf`, 2016. Online; accessed 23-June-2021.

[Uni17] United Nations Economic Commission for Europe. Agreement concerning the adoption of uniform technical prescriptions for wheeled vehicles, equipment and parts which can be fitted and/or be used on wheeled vehicles and the conditions for reciprocal recognition of approvals granted on the basis of these prescriptions, addendum 93: regulation no. 94, revision 3. `https://unece.org/fileadmin/DAM/trans/main/wp29/wp29regs/2017/R094r3e.pdf`, 2017. Online; accessed 23-March-2021.

[U.S08] U.S. Department of Transportation. Consumer information; new car assessment program. `https://www.nhtsa.gov/sites/nhtsa.gov/files/nhtsa-2006-26555-0114.pdf`, 2008. Online; accessed 25-June-2021.

[U.S21] U.S. Department of Transportation. Federal motor vehicle safety standards: standard no. 208; occupant crash protection, 11 C.F.R. §571.208. electronic code of federal regulations. `https://www.ecfr.gov/cgi-bin/text-idx?node=se49.6.571_1208`, 2021. Online; accessed 10-June-2021.

[vdMPvdH09] L. van der Maaten, E. Postma, and J. van den Herik. Dimensionality reduction: a comparative review. *Journal of machine learning research*, 10(66-71):13, 2009.

[VF05] M. Verleysen and D. François. The curse of dimensionality in data mining and time series prediction. In *International work-conference on artificial neural networks*, pages 758–770. Springer, 2005.

[VMQ+13] M. Vasile, E. Minisci, D. Quagliarella, M. Guénot, I. Lepot, C. Sainvitu, J. Goblet, and R. F. Coelho. Adaptive sampling strategies for non-intrusive POD-based surrogates. *Engineering computations*, 2013.

[VRD09] G. Van Rossum and F. L. Drake. *Python 3 reference manual*. CreateSpace, Scotts Valley, California, 2009.

[Wan19] X. Wan. Influence of feature scaling on convergence of gradient iterative algorithm. *Journal of Physics: Conference Series*, 1213(3), 2019.

[WB09] Z. Wang and A. C. Bovik. Mean squared error: love it or leave it? A new look at signal fidelity measures. *IEEE signal processing magazine*, 26(1):98–117, 2009.

[WC91] I. Wendt and C. Carl. The statistical distribution of the mean squared weighted deviation. *Chemical Geology: Isotope Geoscience Section*, 86(4):275–285, 1991.

[Web09] Julian Weber. *Automotive development processes: processes for successful customer oriented vehicle development*. Springer Science & Business Media, 2009.

[Wil98] C. K. I. Williams. Prediction with Gaussian processes: from linear regression to linear prediction and beyond. In *Learning in graphical models*, pages 599–621. Springer, 1998.

[WM19] P. Wellkamp and M. Meywerk. Reduction of epistemic uncertainty of a crash box model-experimental and numerical investigations. *Latin American Journal of Solids and Structures*, 16, 2019.

[Wol02] D. H. Wolpert. The supervised learning no-free-lunch theorems. *Soft computing and industry*, pages 25–42, 2002.

[XCY16] H. Xu, C.-H. Chuang, and R.-J. Yang. Towards optimization of multi-material structure: metamodeling of mixed-variable problems. *SAE International Journal of Materials and Manufacturing*, 9(2):400–409, 2016.

[XK02] D. Xiu and G. E. Karniadakis. The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM journal on scientific computing*, 24(2):619–644, 2002.

[XWVA05] M. Xu, P. Watanachaturaporn, P. K. Varshney, and M. K. Arora. Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, 97(3):322–336, 2005.

[YHHZ20] H. Yong, J. Huang, X. Hua, and L. Zhang. Gradient centralization: a new optimization technique for deep neural networks. In *European conference on computer vision*, pages 635–652. Springer, 2020.

[YLL+14] I. E.-H. Yen, T.-W. Lin, S.-D. Lin, P. K. Ravikumar, and I. S. Dhillon. Sparse random feature algorithm as coordinate descent in hilbert space. In *Advances in neural information processing systems*, pages 2456–2464, 2014.

[YP19] Y. Yang and P. Perdikaris. Adversarial uncertainty quantification in physics-informed neural networks. *Journal of Computational Physics*, 394:136–152, 2019.

[Zan02] T. A. Zang. *Needs and opportunities for uncertainty-based multidisciplinary design methods for aerospace vehicles*. National Aeronautics and Space Administration, Langley Research Center, 2002.

[Zel94] A. Zell. *Simulation neuronaler Netze*, volume 1. Addison-Wesley Bonn, 1994.

[ZL08a] X. Zhou and H. Lin. Global sensitivity analysis. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*, pages 408–409. Springer, Boston, Massachusetts, 2008.

[ZL08b] X. Zhou and H. Lin. Local sensitivity analysis. In S. Shekhar and H. Xiong, editors, *Encyclopedia of GIS*, page 616. Springer, Boston, Massachusetts, 2008.

[ZLS20] R. Zhang, Y. Liu, and H. Sun. Physics-informed multi-LSTM networks for metamodeling of nonlinear structures. *Computer Methods in Applied Mechanics and Engineering*, 369:113226, 2020.

[ZO99] Y.-G. Zhao and T. Ono. A general procedure for first/second-order reliability method (FORM/SORM). *Structural safety*, 21(2):95–112, 1999.

[ZTZ05] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The finite element method: its basis and fundamentals*. Elsevier, 6 edition, 2005.

[ZY21] Y. Zhang and Q. Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[ZZC11] P. Zhu, Y. Zhang, and G. Chen. Metamodeling development for reliability-based design optimization of automotive body structure. *Computers in Industry*, 62(7):729–741, 2011.