UNIVERSITY OF WUPPERTAL

# Process for Extraction of Knowledge from Crash Simulations by means of Dimensionality Reduction and Rule Mining

*Dissertation for obtaining a doctorate submitted by:*
Constantin Diez

*Supervisors:*
Prof. Dr.-Ing. Axel Schumacher
Privatdozent Dr.-Ing. Ioannis Doltsinis

*in the*

Faculty for Mechanical and Safety Engineering
University of Wuppertal

Waldenbuch
February 16, 2019

Die Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20190313-111615-6
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20190313-111615-6]

*"The purpose of life is acquiring skill through learning,*
*the meaning of life is serving others with these skills."*

University of Wuppertal

# *Abstract*

Faculty for Mechanical and Safety Engineering
Chair for Optimization of Mechanical Structures

Dissertation of Constantin Diez

**Process for Extraction of Knowledge from Crash Simulations by means of Dimensionality Reduction and Rule Mining**

This thesis proposes an efficient process flow for analyzing an ensemble of vehicle crash simulations. The process has two goals. The first goal is to algorithmically detect the largest and most notable types of deformation. The second goal is to find out how to avoid or trigger a user-specified deformation behavior.

The first goal of deformation behavior segmentation, is approached with a novel dimensionality reduction technique. This dimensionality reduction technique not only makes it possible to derive a lightweight, intermediate, and mesh-free representation of the simulation results, but also makes it possible to compute a normalized simulation similarity. These simulation similarities can be used to find groups of similar deformation behaviors by using clustering algorithms and low-dimensional embeddings. An engineer then has to decide which types of deformation are acceptable, and which are not.

Having chosen the desired types of deformation, it is then possible to find out how they can be achieved by using rule mining. The rule mining algorithm returns multiple safe design spaces, in which the engineer's demand is fulfilled. Because the rules might be unsafe at their boundaries, an optimization can limit the probability of a rule being wrong to within a specified limit.

Bergische Universität Wuppertal

# *Kurzfassung*

Fakultät für Maschinenbau und Sicherheitstechnik
Lehrstuhl für Optimierung mechanischer Strukturen

Dissertation von Constantin Diez

**Prozess zur effizienten Wissensgewinnung aus Crashsimulationen mithilfe von Dimensionsreduktion und Regelextraktion**

Diese Doktorarbeit schlägt einen Prozessfluss vor, welcher eine effiziente Analyse von einer Vielzahl an Crashsimulationen ermöglicht. Der gesamte Prozessfluss hat zwei zentrale Ziele. Das erste Ziel ist die algorithmische Detektion der größten und wichtigsten Deformationsmoden einer Struktur unter Crashbelastung. Das zweite Ziel ist es herauszufinden, wie diese Deformationsmoden gezielt ausgelöst oder vermieden werden können.

Das erste Ziel der Deformationsmodensegmentierung wird in dieser Arbeit mittels einer neuartige Dimensionsreduktion angegangen. Diese Dimensionsreduktion erzeugt nicht nur eine speicherarme und netzunabhängige Darstellung der Simulationsergebnisse, sondern ermöglicht auch die Berechnung einer prozentualen Ähnlichkeitskennzahl. Infolgedessen können mithilfe niederdimensionaler Einbettung und Clustering Gruppen an Simulationen identifiziert werden, welche untereinander ein ähnliches Deformationsverhalten aufweisen. Ein Ingenieur kann zu diesem Zeitpunkt selbst entscheiden, welche Deformationsmoden er als akzeptabel befindet, und welche er vermeiden möchte.

Nachdem das bevorzugte Deformationsverhalten klargestellte wurde, kann durch Rule Mining herausgefunden werden, wie das bevorzugte Deformationsverhalten erreicht werden kann. Der Rule Mining Algorithmus schlägt dabei multiple Design-Bereiche vor, in welchen das Ziel des Ingenieurs erfüllt wird. Weil diese Regeln an ihren Grenzen unsicher sein können, kann über die Optimierung der Regelgrenzen eine spezifizierte Zuverlässigkeit erreicht werden, um damit eine sichere Nutzung zu gewährleisten.

# *Acknowledgements*

After such a long journey through life, I want to thank especially my mother and my father,

*Barbara and Reinhard Diez,*

from the deepest bottom of my heart. They always supported me and used a lot of resources, time and effort, just that I could become what Iam now. They always understood my inner desire for more knowledge and cultivation of my skills, so that they decided to put my needs before theirs in a humble manner for many times. For being another pillar of my life, I also want to thank my grandparents for their support and inspiring presence, having achieved great things out of very rough times. I honor hereby:

*Lieselotte and Siegfried Gagstatter,*
*Hilde and Kurt Diez.*

Since I always count my closest friends as part of my family, also many thanks to them. They always relieved me of stress, by giving me the feeling, that I always was and will be a piece of them and their life. They always watch over me, provide me a safe haven and complement my lacks, inabilities, deficiencies and weaknesses. They are indeed my complement in many ways. Thank you for accompanying me all these years:

*Michael, Natalie, Alexander & Marianne Mohenweiser*
*Philipp Gebhardt & Marion Lütz*
*Katja Müller & Jens Heine*
*Charlotte Gunzinam*
*Michael Pflüger*

Beyond friends, there are always those people unmentioned, which support you or keep inspiring you in many ways. I also want to honor:

*Christiane & Hubert Lehr*
*Tang Li Long (唐理龙)*
*Mohammed Lamrabet*
*Klaus Lenhart*

Professionally I want to thank the awesome team at Opel for making this thesis possible, and also enduring my creativity. It is not very common for a doctoral candidate to get such an intensive and thorough care. Also the thesis wouldn't have this quality, if the team hadn't relieved me from daily tasks. I highly value this supervision and it encouraged me to put all of my effort and skill into my work. Special thanks to:

*C. Wieser, L. Harzheim, S. Frik, J. Morawski, B. Lauterbach, M. Stolzenburg, N. Sygusch*

On the academic side, I want to thank my academic supervisors for their assessment and guidance. Their feedback kept me on track and they always had time for me when I needed their opinion, even under busy circumstances. Thanks to:

*A. Schumacher, I. Doltsinis*

# Contents

# Notation

| notation | example | description |
|---|---|---|
| vectors | $\vec{a}$ or $a_i$ | vectors are either noted by an arrow or a single, lowercase index. |
| matrices | $\underline{\underline{a}}$ or $a_{ij}$ | matrices are either noted by two underscores or two, lowercase indices. |
| pseudo inverse | $\underline{\underline{a}}^{+}$ | The pseudo inverse of a matrix is noted by a superset $+$. |
| variables | $X$<br>*var_name* | This thesis uses the convention, that variable symbols are either written in capital letters or in the specified, special font. |
| variable realizations | $x$ | While variables are written in capital letters, realizations with assigned values are written in lowercase. This also applies to matrices and vectors. |
| variable info | $X^{(info)}$ | An uppercase text, embraced in round brackets, contains additional textual information. This must not be confused with an index. |
| design variables | $X_4$<br>$x_4 = 3.7$ | Design variables for simulation runs are indicated by the letter $X$. |

| notation | example | description |
|---|---|---|
| responses | $Y_1$<br>$y_1 = 0.3$ | Simulation responses are noted by the letter $Y$. |
| deformation classes | $C_2$<br>$c_2 = True$ | Deformation classes are noted by the letter $C$. A realization of a deformation class can only have a Boolean value, which states, whether the realization belongs to this deformation class or not. |
| sets | $\{1, 2, 3\}$ | Sets are described by curly brackets. |
| set size | $\lvert\{4, 5\}\rvert = 2$ | The absolute value of a set returns the number of elements within the set. |
| inner product | $\left\langle \vec{a} \middle\vert \vec{b} \right\rangle$ | An inner product or dot product between two vectors or functions is indicated by the Dirac-notation. |
| erf | $erf(0.5)$ | The error function $erf(\cdot)$ is a sigmoid-like function and is defined as $erf(x) = \frac{1}{\pi} \int_{-x}^{x} e^{-t^2} dt$. |

# List of Abbreviations

| | |
|---|---|
| **ANN** | **A**rtificial **N**eural **N**etwork |
| **ANOVA** | **AN**alysis **O**f **VA**riance |
| **CAE** | **C**omputer **A**ided **E**ngineering |
| **CART** | **C**lassification **A**nd **R**egression **T**rees |
| **CFD** | **C**omputational **F**luid **D**ynamics |
| **DOE** | **D**esign **O**f **E**xperiments |
| **DPCA** | **D**ifferential **P**rincipal **C**omponent **A**nalysis |
| **DT** | **D**ecision **T**ree |
| **FEM** | **F**inite **E**lement **M**ethod |
| **HPC** | **H**igh **P**erformance **C**omputing |
| **KDD** | **K**nowledge**D**iscovery in **D**atabases |
| **KDE** | **K**ernel **D**ensity **E**stimation |
| **KLE** | **K**arhunen-**L**oève **E**xpansion |
| **kNN** | **k** **N**earest **N**eighbors |
| **LBO** | **L**aplace **B**eltrami **O**perator |
| **LLE** | **L**ocally **L**inear **E**mbedding |
| **MDS** | **M**ulti **D**imensional **S**caling |
| **ML** | **M**achine **L**earning |
| **NCAP** | **N**ew **C**ar **A**ssessment **P**rogramme |
| **PCA** | **P**rincipal **C**omponent **A**nalysis |
| **PDF** | **P**robability **D**ensity **F**unction |
| **PML** | **P**rincipal **M**anifold **L**earning |
| **RBF** | **R**adial **B**asis **F**unction |
| **RF** | **R**andom **F**orest |
| **RT** | **R**andom **T**ree |
| **SoS** | **S**tatistics **o**n **S**tructures |
| **SVM** | **S**upport **V**ector **M**achine |
| **t-SNE** | **t**-distributed **S**tochastic **N**eighbor **E**mbedding |

# Chapter 1

# Introduction

Evaluating and extracting knowledge from a large amount of data is a central challenge in today's data-driven world. In many situations, the amount of data is simply too large to be analyzed by humans properly. This challenge is also slowly arising in the field of Computer Aided Engineering (CAE), where for certain investigation types, the number of simulations simply exceeds human capabilities. This general necessity to automate human tasks gave rise to Machine Learning (ML). Machine learning is an emerging field, which gives computers the ability to detect patterns and dependencies through advanced statistical methods [5]. It is frequently used in other disciplines, such as marketing [38], financial services [56], geography [37] and many others, in order to attain new insights. The field of CAE is still in the early transformation phase of adopting this new branch of technology. This thesis will propose a new way of embedding and more importantly, using some of these techniques to reduce the workload on engineers, and get new insights from his or her simulation data. In order to understand where and how to use machine learning specifically, an introduction to crash simulation, as a branch of CAE, will be given in this chapter. This thesis will only cover crash simulation, and will therefore address this field of work specifically, even though sometimes generalizations may be possible.

## 1.1 Vehicle Crash Simulation

Crash simulation uses the Finite Element Method (FEM) to approximate the analytical solution of the mechanical equations. The abstract goal of vehicle crash simulation is to control the energy absorption in such a way that occupants and pedestrians remain unharmed reliably.

The following list enumerates the challenges of crash simulation modeling:

- contact and friction modeling
- nonlinear material behavior
- strain-rate dependency
- bifurcations

Before simulations can be performed, a respective model is built, and is calibrated first at specimen-level and thereafter at component-level. The resulting crash simulation model serves as a basis for additional investigations. After working out a final design, at least one more test is done for validation and certification purposes. Structural simulations have shown to be a powerful tool, enabling advanced analysis of structural systems. Modern cars owe their high safety to simulations, since a simulation can yield many insights which would be impossible for a test.

## 1.2 Variational Studies

In the past, crash simulations were performed in a nominal way, which means that nominal values were chosen as inputs, and a single simulation was carried out to evaluate the performance. Nonetheless, in order to account for physical uncertainties, variational studies are performed. The increased knowledge about a design makes it possible to narrow down the safety factors , and thus achieve for example, further weight reduction. There are two reasons for performing multiple simulations with variations in the input data:

- Check a design for the influence of tolerances (Robustness Analysis)
- Search a better design (Design Space Exploration)

Robustness Analysis focuses on the surrounding of a specific design. First of all, it is very obvious, that every manufactured component is unique due to its tolerances. Furthermore, the testing conditions may also vary within a specified range, according to specifications and regulations. As a result, every physical test will always differ in some way, and thus might have a slightly different outcome. The

digital world adds another influence, which is the numerical round-off error, due to load balancing along multiple computers. These factors must not influence the virtual crash performance of a car significantly. In order to investigate a car's performance with respect to tolerances, one could simply perform and analyze multiple simulations. A mechanical system, in this thesis, can be considered to be robust if small input variations do not influence the outcome of the simulation severely, in respect of a user-specific metric.

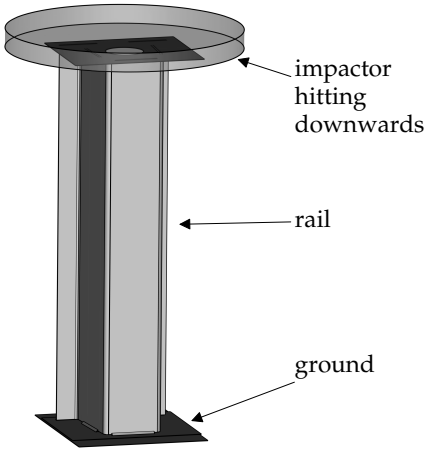
Design Space Exploration in contrast, tries to identify new potentials regarding crash performance. In these investigations, the structural design variables are varied in a much larger interval than mere tolerances. Doing so can lead to different deformation patterns of energy absorption, and gives an engineer an opportunity to identify an improved design.

Both approaches, robustness analysis and design space exploration, do not have to be strictly separated, but may also be blended, as in Robust Design Optimization. Similarly, this thesis performs design space exploration and robustness analysis in a blended manner, but with a main focus on design space exploration. In consequence, the algorithms and the process flow do not fit robustness analysis as well as they fit design space exploration.

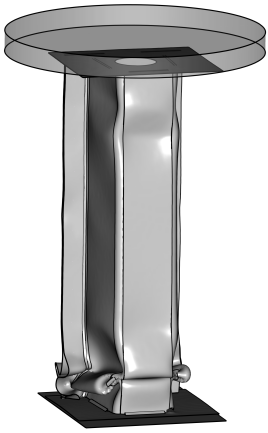
## 1.3 The Key Questions of Structural Behavior Analysis

Analyzing a large amount of simulations is the major challenge tackled by this thesis. The task will be illustrated in the following example:

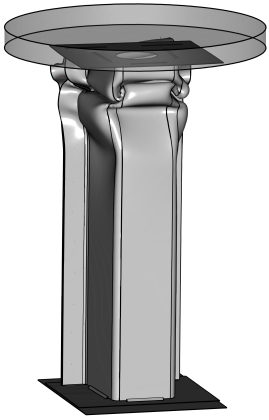
The model in figure 1.1 is a longitudinal rail, which is being crushed by an impactor at the upper end, and as a consequence starts to buckle (see Appendix A). The model is a simplification of a car crash, where the crash absorbing structure is folded by the barrier. The abstract goal is to control the location of the buckling. In order to investigate the structure's behavior under uncertainty, it was parametrized with 173 variables, involving material properties, geometric sizes, connection modeling and the load conditions. The parametrized model was then used to create 1000 different variants. The variables were sampled from a normal distribution, in combination with a Latin-Hypercube-Design [68]. Two deformed samples are shown on the lower part of figure 1.1, which have been randomly selected from the ensemble. They show, that the initiation of buckling can also occur at the lower




(A) Simulation model of the rail.



(B) Buckling bottom.



(C) Buckling top.

FIGURE 1.1: Model of a virtual drop-tower test. An impactor is dropped onto the rail and triggers different buckling modes, depending on the input variation.

end of the rail. The real issue at this point is, that one does not know how the rail is deforming in all other result files, and it is humanly unpractical to investigate this manually.

This challenge of comparing large numbers of result files can be generalized in a certain way. A specific deformation pattern, such as buckling bottom or buckling top in figure 1.1, will be named in the following as an effect. An effect will be defined in this thesis as a set of simulations, which show similar results of any kind. This can either be in terms of responses or deformation behavior.

An engineer analyzing an ensemble of simulations is now confronted with the following central questions:

1 What effects occur?

2 What influences an effect?

3 How to avoid or trigger an effect?

Question 1 tries to find out what major effects take place throughout all simulations. Question 1 already poses a major difficulty when faced with many simulations. The core issue is that it is humanly impracticable to analyze all of these simulations, and categorize them in terms of a deformation mode by hand. The engineer simply does not know how frequently a certain deformation mode occurs. Whether these are all modes is also unknown. Question 1 can be approached by comparing deformation modes algorithmically, but this topic is quite complex. The simplest approach is to measure certain properties of the model, such as the displacement of a few nodes or the intrusion, but there are two reasons why this will always be incomplete. Firstly, the deformation behavior does not necessarily correlate well with measurable properties, such as the intrusion. Secondly, observing only a few nodes leaves out a lot of information from the rest of the mesh. For example, a neighboring node might be a much better choice, but one simply doesn't know about it. This problem is very similar to facial recognition, where one could also analyze a face from a few hand-selected pixels. However, a human's emotional expression is determined by many fine details across the whole face, thus a more complete view is required and will yield better results. In conclusion, a combination of both is necessary: the analysis of responses, in combination with deformation modes.

Question 2 is a soft formulation of question 3. If one can distinguish, for example, the deformation modes discovered by question 1,

then in the event of undesired behavior, one is interested in the variables which possibly triggered the deformation mode. This task is comparable to correlation studies. The danger though, is that correlation does not imply causation, and thus an engineer usually has to confirm the relationship as being physical, later on. Therefore, the answers to question 2 should be seen as hints, rather than facts.

Question 3 tries to answer how an effect can be avoided or enforced, such as a certain deformation pattern. The challenge is to find out what all simulations with a specific deformation pattern have in common. Therefore an engineer usually opens up multiple models and starts comparing them, but as was stated previously, this is not practicable in the motivation example. The approach of this thesis is to let an algorithm discover what the simulations have in common. Therefore, it is merely fed the input variables, and asked how a specific deformation pattern is prevented or triggered. It is very important to note that variables might trigger instabilities somewhere in the model, but may not be the real cause of the problem. In conclusion, the answers to question 3 are also hints which need human confirmation. The good thing is, however, that it is much easier and faster for a human to simply check the facts highlighted by an algorithm, than to search the entire data oneself.

## 1.4 Side Challenges

The central questions of structural behavior analysis are accompanied by many side challenges, which make any solution approach difficult. The major side challenges are:

(a) Handling of different meshing

(b) Handling of geometric differences

(c) Dealing with highly nonlinear data

(d) Low sample counts

(e) Efficient data reduction

(f) Keeping the process manageable

First of all, it is very clear that one major difficulty is how to handle different meshes. One cannot rely on the fact that a part has an

identical mesh (a), thus results mapping might be necessary. Unfortunately, mapping also has its limits when considering topological changes, since there is no clean solution if one mesh has additional holes or ribs somewhere (b). This becomes even more problematic in the event that entire parts might have been added or removed. After data alignment, the core challenge shifts to the data analysis itself. A crash simulation can be seen as a nonlinear data cloud evolving over time. Understanding nonlinear data with meta-models of any kind, will always need higher sample counts compared to mainly linear data (c). The advantage of virtual analysis is in general, that new samples can be generated, but this can only be done in a very limited way for crash simulations, since computation still takes a long time, and thus it is desirable to require as few samples as possible (d). This especially opposes (c) where higher sample counts to account for nonlinearity. The complexities of (a) to (d) finally channel into (e) and (f). Accounting for all of these challenges requires many mathematical tricks to work well. Despite the fact that computers have become quite fast, every operation on the entire mesh, such as results mapping, takes a lot of time and thus it is not easy to be efficient (e). All the mathematical tricks also need proper configuration. If a user has to determine multiple settings for every new investigation, then this lowers the usability significantly (f).

## 1.5 Goal of this Thesis

The goal of this thesis is to provide a process which guides a user to the answers to the key questions of structural behavior analysis (sec. 1.3), while taking into account the constraints of its side challenges (sec 1.4). The first question about effect detection, can be answered by means of dimensionality reduction, which enables the visualization and clustering of an entire simulation ensemble at once. The second question about the influences of certain effects, can be answered with a variable importance ranking. The last question about how to avoid or trigger certain effects, will be addressed by a proprietary rule mining algorithm. Rule mining tries to find all the possible major and simple solutions. A major focus of this thesis was not only its algorithmic fragments, but also the layout of the entire process as a whole.

# Chapter 2

# Related Work

This section discusses how the central questions of postprocessing with their side challenges were approached previously.

**Dimensionality Reduction for Effect Detection** The question, how can different deformation modes be identified and distinguished, can be answered algorithmically by means of dimensionality reduction. Dimensionality reduction sees the result of a single FEM simulation as a high dimensional data vector, and tries to reduce its length without losing significant information. This compression scheme is usually applied to all simulation results at once, so that the algorithm knows which data patterns throughout the ensemble are the largest. The goal is usually to reduce the amount of data for further processing, and thus obtain a smaller and more easily inspectable representation. If the data is reduced to 2D or 3D, an embedding can be plotted, and a scientist can search for patterns, such as clusters, in the plot himself.

The most common technique for dimensionality reduction is Principal Component Analysis (PCA) [53, 31], which tries to preserve linear dimensions with largest variance. Due to it being the simplest approach, it has already been applied in the context of crash simulation previously. In [1] it was used to correlate principal components of the deformation field with the input variables of a design of experiments study. PCA was used in [47] to simultaneously analyze and reduce the scatter between crash simulations. Later on, in [67], the algorithm was also used to investigate crash simulation branching.

PCA is an entirely linear technique since it performs a linear transformation on the data. In consequence if the data has a nonlinear structure such as a semi-circle, PCA is inherently unable to find a coordinate system which describes the data optimally (for the semi-circle this is simply a line in the embedded space.) Therefore, PCA is not

expected to perform best in general. In [6] three methods were compared in terms of their accuracy: PCA, principal manifold learning (PML) [20] on sparse grids and local linear embedding (LLE) based on tangent space estimation [70]. The errors showed that nonlinear methods did indeed perform better, but the error margin narrows quickly when more embedded dimensions are used.

It is important to note , that almost all of these previous works [47, 67, 6] tried to reconstruct an entire simulation result from an arbitrary point in the low-dimensional space. This can be done quite easily in case of, for example PCA, since a linear transformation is bijective and thus easily invertible. For non-invertible transformations, one may use methods for Pre-Image reconstruction [4]. In contrast to this motivation, this thesis does not try to reconstruct simulations from a low-dimensional embedding. The reason is simply that usually clusters or groups develop in the low-dimensional space, and one may simply pick simulations as representative examples. Firstly, this makes sense because there is a limited motivation to reconstruct simulations if there are many other simulations nearby to interpolate. However, the reconstructed version may show the isolated effect, whereas the representative version may also contain secondary effects. Secondly, [6] also showed that the reconstruction yielded high errors in areas where deformation occurs. This is expected, for the reason that the local deformation is highly nonlinear. Since as engineers, our interest lies especially in areas with strong deformation, it was decided that reconstruction was not worth the mathematical effort.

**Geometry Comparison and Hashing for Effect Detection** The comparison of crash simulations is closely related to the field of geometry comparison. Many recent ideas use the Laplace-Beltrami-Operator (LBO) to achieve a mesh-free surface description. The LBO describes the geometric manifold by its surface curvature. In the work of [59, 60] about Shape-DNA, the eigenvalues of the LBO are interpreted as an isometric shape descriptor. Even though the LBO is quite powerful, it requires a well-defined surface mesh, which is not the case in FEM. FEM-meshes may contain shells and solids at the same time, and also more than two shells may share an edge. The property of isometry is not optimal, since it emphasizes topological changes much strongly than deformations, so that failure contributes to a greater extent than deformations to the Shape-DNA. An operator-based approach, including the LBO, was performed in [33]. The contribution is especially interesting, since a 3D-embedding was done for all the

timesteps of all the simulations, which is computationally very extensive. The embedding made it possible to find out at which timestep a behavior bifurcation occurred. In practice, the specific time of the event may help to identify concatenation of effects during simulation. Nevertheless, bifurcations within the responses are quite rare, and even if they occur, the exact timestep is usually quite uninteresting for design space exploration (not for robustness analysis). Therefore, this thesis does not try to unveil chains of effects automatically, even though it provides the means to do so, and focuses on the influence of the design variables. The dimensionality reduction in this thesis was also not used to create an embedding of all timesteps, because it is computationally very demanding.

Another method for the comparison of geometries are Geometric Spin-Images [48]. This method splices the surface of the geometric object into overlapping snippets and hashes them. Instead of comparing geometries, one has to compare two sets of hashes. The similarity between two sets may be computed with the Jaccard index [34] which is much faster than the original mesh to mesh comparison. Due to the arbitrary snipping, the accuracy is expected to have big variations in the case of repetition . In conclusion, this method can be used to search a database for roughly similar objects, but it is not expected to provide sufficient accuracy when comparing deformed objects in the field of CAE, where minor differences may also be important.

**Inference for Simulation Data** Understanding the connection between simulation inputs and outputs, is a very common task in the field of simulation. Where both the inputs and the outputs are scalar values, influences may be searched with techniques such as Correlation Study [52] or Analysis of Variance (ANOVA) [45]. The backbone of these investigations is usually a meta-model. Because most simulation outputs are continuous variables, regression meta-models, such as Kernel Regression with Radial Basis Functions (RBF) [40] or Gaussian Process Regression [39, 58], are used. As well as continuous outputs, the prediction of discrete variables, such as the failure or non-failure of a component, may also be desired. In order to predict such discrete values, classification methods may be used instead of regression. Common methods are Support Vector Machines (SVM) [8], Artificial Neural Networks [46] or Decision Tree Learning [12].

A new tool to analyze cause and effect in the aspect of robustness analysis, is DIFFCRASH from Sidact GmbH [25]. The tool uses a method called Differential Principal Component Analysis (DPCA) [67,

66] to estimate how the scatter of one component at an early timestep influences the scatter of another component at a later timestep. The tool makes all of its estimates on the displacement field, and requires a user to select the components for DPCA himself, which can be tedious when dealing with an entire car. It has the advantage though, that it does not use parametrized geometry and thus can relate arbitrary components, which is a significant benefit. If one component is identified as the source of scatter, an engineer can stabilize it himself, and then run a validation investigation to check whether the algorithm's hypothesis was correct. What makes this PCA-based algorithm especially powerful, is its independence from model parametrization, in contrast to this thesis which requires a parametrized simulation model. However, a disadvantage is that DIFFCRASH ideally needs identical meshes between the two simulation models. Mere mesh differences can be compensated quite easily by mapping, but if constructive changes are present, then the procedure's quality is expected to deteriorate. It is important to note that, to date, no public investigation which clearly shows how much constructive changes influence the outcome of this PCA-based algorithm has been done.

The software Statistics on Structures (SoS) from Dynardo [24] uses the Karhunen-Loève Expansion (KLE), to decompose the covariance matrix of an arbitrary field into modes. The KLE is almost identical to PCA, which uses the empirical covariance matrix. The KLE in SoS computes the covariance matrix from an autocovariance function, which is simply a (spatial) Gaussian kernel. Because the covariance matrix is computed densely for the entire mesh, the feasible model size is about $10000$ nodes [69]. Larger models need either mesh coarsening or subsampling strategies. The first idea is to use the KLE on the simulation result field. The result field is now only a function of the coefficients of the modes. A field meta-model can be built by connecting mode coefficients and input variables with a meta-model. This field meta-model can now be used for optimization or sensitivity analysis, since one does not have to evaluate a simulation run, but can trigger the field meta-model instead, which is much cheaper. Care must be taken to evaluate the field meta-model only in regions with high prediction quality. A random field may not only be a result field, but also an input field, such as sheet thicknesses derived from measurements. It has proven that much better results are obtained if the input random field is derived from at least a few measurements. The derived modes can be used for the generation of perturbed designs with the target of analyzing the scatter of the simulation results. This results in a more

realistic analysis. A subsequent optimization can help to reduce this scatter in combination with several design parameters.

Mineset [19] is a Machine Learning framework from ESI Group, which also features Association Rule Mining [2]. Association Rule Mining tries to find frequent, if-then conditions between discrete variables. This makes it unsuitable for simulation data, since many input and output variables are continuous. One may transform continuous variables into discrete ones by choosing discrete bins, but if so, one has to face two major issues. Firstly, the choice of the bin thresholds may be suboptimal, either wasting potential or worsening predictions. Secondly choosing too many bins makes it difficult for the engine to search for meaningful rules for the very specific reason that a lot may be found, and it is hard to tell how they connect. Another major problem with classical Association Rule Mining is that it finds a lot of rules regarding certain filtering criteria, thus the actual rule selection has to be done by the user himself. Depending on the number of variables, this may be a lot work for a simulation engineer. Mineset provides an alternative to Association Rule Mining by making it possible to also define rules from Naive-Bayes Classification. This is also a manual approach and does not save a lot of manual effort, because seen abstractly, one has to assemble every rule personally by picking the input variables used for it. Nonetheless, the rule thresholds are much better in the latter case.

An engineer might also be interested in why certain simulations showed a specific deformation behavior. While as stated before, the detection can be done by means of dimensionality reduction, finding a connection to the input variables has proven to be quite a challenge. Previous works [67, 6, 33] searched influences manually, by plotting a low-dimensional embedding and then coloring the simulations according to their input variables or computing the correlation between the principal axes of the embedded space and the input variables. This works quite well, as long as the 'real' principal components of the data are mostly linear. If this is not the case, then the linear principal components do not approximate the data well, which makes the correlations look weak and noisy.

This thesis attempts a new approach in the field of vehicle crash simulation by using rule mining for knowledge extraction. The basic idea is the extraction of rules, where an antecedent Boolean expression $A$ predicts a discrete target class $T$, formally written as: $A \rightarrow T$ (for more see section 4.3). The most notable difference between this

and previous inference approaches, is that rule mining relies on classification and not regression, thus the prediction of a discrete variable and not a continuous one.

Rule mining is a relatively old discipline, originating from about 1970 and having gained notable awareness around 1990 [23]. Originally, rule mining was focused on small datasets with discrete variables, but was quickly extended to also deal with continuous variables. Because simulation data contains a large number of continuous variables, only algorithms can be used which can also deal with continuous data. Most ideas rely on the theory of decision tree learning, which makes it possible to deal quite well, not only with continuous variables, but also with low sample counts. Since every leaf of a decision tree is already a rule, every decision tree learner can therefore indirectly be considered to be a rule mining algorithm. The most well-known mining algorithms are C4.5 [57] and its commercial successor C5, as well as RIPPER [11] and CART [10]. The last one was chosen and further adapted to suit the demands of this thesis.

Rule mining was already done indirectly in the field of Computational Fluid Dynamics (CFD). In the work of [27], rules were extracted manually from a decision tree in order to identify good vehicle concepts, in terms of aerodynamics. The vehicle samples originated from a previous optimization.

# Chapter 3

# Data Analytics Process Flow

This chapter will explain how this thesis approaches the task of post-processing with machine learning. Therefore, this chapter will explain from a high-level view, why certain choices were made, and how these fit to each another. The topics are data generation, effect and pattern recognition and finally, inference of cause and effect. An overview of this section is given in figure 3.1. It will serve as a map for this section and also this thesis.

## 3.1 Data Generation

In order to handle stochastic data, a deterministic FEM-Model is built up and its physical properties are parametrized with a variable vector $X_i$. A realized simulation sample $s$ will be recalled in the following by its variable vector $x_{si}$ (realizations are noted in small print).

**How were the Input Variables Distributed?** The first step of this thesis was to decide how to deal with uncertainty in the input variables. The uncertainty is most usually modeled with probabilistic distribution functions. In the case of robustness analysis, realistic distributions from experiments are preferred, if available. In the case of design space exploration, uniform data distributions within given bounds were chosen, because a large diversity is preferred.

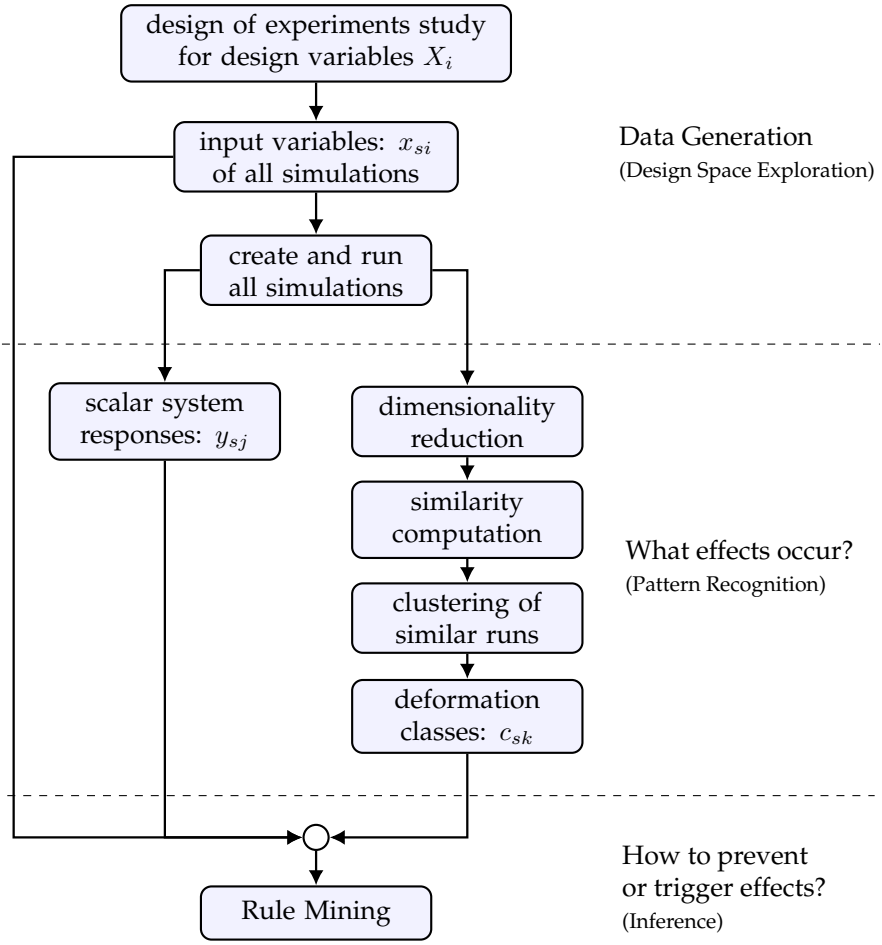



FIGURE 3.1: Abstract data and process flow of this thesis, which serves as a map for this chapter.

**How were Samples created?** Specific realizations can be drawn directly in the case of simple probability distributions, by using a random number generator. Also sampling for other distribution functions, such as a normal distribution, can be derived in such a way, if a transformation to a uniform one exists. In the general case of arbitrary distribution functions resulting from testing, methods such as

the Metropolis-Hastings-Algorithm [30], may be used in order to draw samples.

Since this thesis instead focuses on design space exploration, a uniform or normal distribution is chosen for the input variables. Drawing samples via random number generation can yield very similar runs, due to randomness, and thus wastes computational power. In consequence, the design of the experimental methods was used to enforce a stronger diversity of the designs. All the samples in this thesis were generated using Latin Hypercube Sampling (LHS) [65, 68] in combination with either a uniform or a normal distribution.

**Computing the Simulation Samples** After generating the input variables $x_{si}$ for all simulation runs, the base FEM-Model is adapted for every sample $s$ regarding the variable vector, and is sent to a High Performance Cluster (HPC) for computation. Thereafter, the files resulting from the simulations are saved for further evaluation.

## 3.2 Searching and Detecting Effects

Detection of statistical patterns and thus effects, is the first very important step when analyzing many simulations. The idea is to discover subgroups of simulations, which show similar deformations or system responses. A typical statistical pattern would be, for example, that the rail from section 1.3 only buckles at very specific areas of the specimen, like the bottom or top area. Detecting these subgroups is very important for two specific reasons. Firstly, humans cannot inspect every simulation if the simulation count is large, thus automation is necessary. Secondly the knowledge of which simulation run deformed in a certain way, will be utilized later on to find constructive changes.

The method used for deformation mode segmentation is most usually dimensionality reduction, which will be discussed in section 3.2.2. One may also identify patterns within scalar simulation responses, such as nodal displacement or acceleration, which is covered in section 3.2.1.

### 3.2.1 Analyzing Simulation Responses

The most common way to analyze an ensemble of simulations, is to watch certain system responses. Therefore ,points of measurement are

chosen within the simulation model, at which certain outputs are extracted as time series. Because the rule mining in this thesis works solely on scalar values, the time series have to be converted into scalar values for further analysis. This can be done by taking the maximum value, or averaging over a certain interval. After the evaluation of the time series, an output matrix $y_{sj}$, is obtained which contains all the responses $Y_j$ from all the simulation runs $s$. This matrix may be used directly in order to find statistical patterns within the data. The two approaches are:

- Analyze with statistical metrics
- Use Visualizations

The most common way to analyze the matrix is to use statistical coefficients, such as the standard deviation, in order to find unusual variable scatter. It should be noted here that many statistical coefficients exist in order to characterize a probability distribution, but since every coefficient has its flaws, one should always consider visual methods for additional evaluation. The mean value, for example, is non-descriptive if a probability distribution has two entirely separate peaks.

Visualization very often gives a much better understanding than mere statistical coefficients because humans are still unmatched when it comes to pattern recognition in unstructured data. Very popular visualization methods are:

- Histograms & Probability Distributions
- Scatter Plots
- Parallel Coordinate Plots

Histograms are an approximation of the probability density function and give very good insights, despite being very simplistic. For example, one may discover two separate probability peaks of a response, which reveals a bifurcation somewhere in the simulation. Scatter plots also help to identify patterns such as clusters, but are limited to three dimensions. Parallel coordinate plots make it possible to watch more dimensions at once. The search for the origin of a bifurcation or a pattern, will be performed later on in section 3.3 about inference. At this point, one simply wants to find such unusual patterns in the responses.

### 3.2.2 Deformation Mode Segmentation

A structure usually deforms in a specific way for given variations. Detecting these deformation modes within a set of result files, is a very helpful tool for an engineer, in order to get an overview and understand his structure better. The key technology for analyzing this huge amount of data is dimensionality reduction. Dimensionality Reduction sees a simulation as some kind of high-dimensional vector (e.g. efficient plastic strain of every element) and tries to translate it into low-dimensional space, such as 2D or 3D, so that one may plot a point for every simulation. It is of the utmost importance for every algorithm to lose as little information as possible during this compression phase, so that data points with a high distance in the high-dimensional space also preserve their distance in the low-dimensional space. This automatically leads to the property , that simulations with very similar results ought to be close, whereas very different simulations should have a larger gap between them. Depending on the behavior of the structure, clusters of points may form, which correspond to simulations with very similar deformations.

An embedding for the motivation example of section 1.3 is given in figure 3.2. Instead of plotting a marker for every simulation in the embedded space (here 2D), a respective image of its deformation is placed at its position. Note that the exact coordinate in the embedded space does not matter, what matters is the position of the samples relative to each other. The dimensionality reduction worked well, since similar samples were positioned close to each other.

The goal of this section is to identify clusters of simulations which have similar deformation behavior. This may be done manually in simple cases from an embedding, or in more complex cases, by the usage of clustering algorithms. The output from this step will be a class matrix $c_{sk}$. The class matrix $c_{sk}$ is a Boolean matrix, which denotes whether in simulation $s$ the deformation mode $k$ did or did not occur.

**Which Result Field was Chosen for Analysis?** Before data reduction takes place, a result field has to be selected for compression. The variable choice usually depends on the user's interest. In a crash simulation, there are three major effects to look for:

- movement
- deformation
- failure

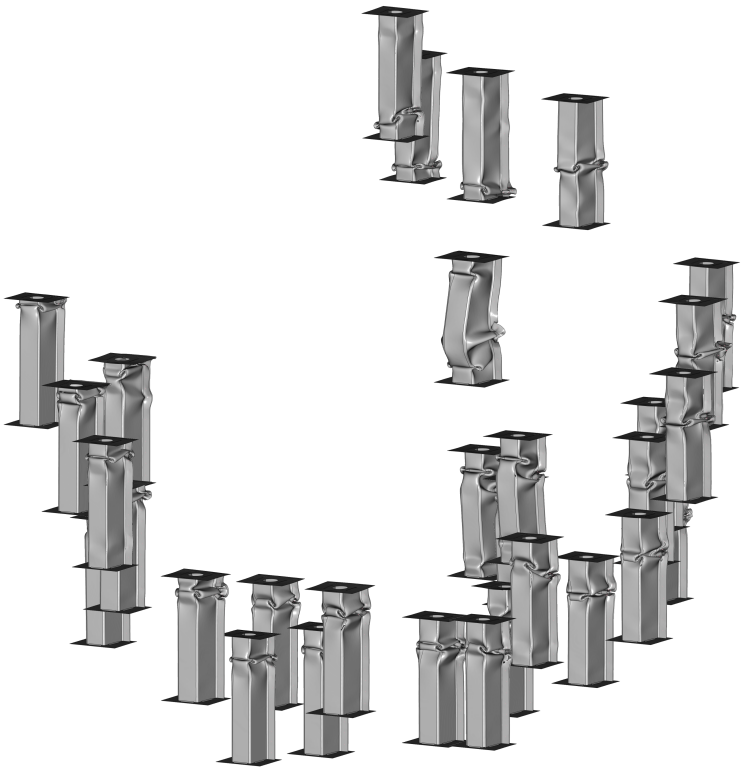


FIGURE 3.2: Detailed 2-dimensional embedding of the longitudinal rail from section 1.3. An image of a deformed simulation is placed at its coordinates in the embedded space.

If the target of interest is movement, then it's obviously best to take the displacement field for the reason that it is strongly dominated by rigid body movement. If on the other hand, deformation is more interesting, then it might be more suitable to take other available results, such as the strain tensor, the efficient plastic strain, the stress tensor, the internal energy or related field variables. There is no final conclusion about which result variable might be most suitable yet. A very appealing approach though, is to use the internal energy of every element, since stress and strain both contribute to it, and it is also

quite robust as an integrative value. In tests, internal energy and efficient plastic strain yielded almost identical embeddings. This makes sense, since in crash simulations the largest contribution to the internal energy is caused by plastic deformation. In this thesis, the efficient plastic strain was chosen as the result field. The last target of interest might be failure, which will not be addressed specifically in this contribution. To analyze the simulation regarding this objective, either the discrete failure index of an element or the element's damage, would be reasonable.

**What is the Idea of Geometry-Based Dimensionality Reduction?** In this thesis, a new approach was chosen to perform dimensionality reduction (for details see sec. 4.1.1). The central idea is to take the element results of a group of parts and convert them into a distribution function. Doing so reduces the effort of mesh-comparison, to merely function-comparison. The intrinsic assumption is that a typical deformation has a typical distribution of the result field. This holds true for asymmetric and complex parts, not though for very symmetric and simple ones.

What distinguishes this proprietary method from all others, is the idea of using crash simulation knowledge by taking the geometric structure into account, and not relying solely on data compression. This results in a rather white-box oriented dimensionality reduction, with a comprehensible data representation, even though the optimality of this approach is still open to discussion. Taking this more specific route yielded multiple advantages, which distinguish the idea greatly from previous methods. Firstly minor topology modifications can be handled much better, since the method relies on geometric simplification with weighted local averaging (see sec. 4.1.1.2). In contrast to this, all other methods require an identical mesh throughout all simulation models. If the meshes are different, mapping has to be performed onto a reference mesh, which proves to be difficult in case of topological modifications, such as a missing rib or an added sheet. It's important to emphasize at this point that any algorithm struggles with large morphings and sizings, including the geometry-based approach of this thesis. It is however possible to extend this algorithm to such capabilities by matching the areas of the simplified geometry correctly, thus a possibility for extension does exist.

## 3.3 Relating Cause and Effect by means of Inference

**How was Inference Done Before?** Having discovered a vast variety of effects in the output data, the question arises how these are actually caused by either input variables or other events. The field of computational inference addresses this problem specifically, and helps humans to find dependencies within large amounts of data. The simplest approach is, for example, to search for correlations between continuous output variables. This is not possible though, if the result variables are discrete deformation classes, as in this thesis. The most notable difference between this thesis and previous works, is thus the type of prediction target. Previous works tried to focus almost entirely on continuous responses, or even the field variables themselves (see sec. 2). The general idea is to fit a meta-model, which in the case of a good prediction quality, understood the connection between inputs and outputs. The meta-model then served as basis for more sophisticated analyses, such as a correlation study or functional analysis of variance.

**What is the Prediction Target in this Thesis?** This thesis takes a very different route in order to determine inference. The final prediction targets will always be discrete values, thus classification instead of regression, will be used. This makes sense, if one wants to predict deformation classes, which are discrete, but continuous results, such as responses, need to be converted into discrete variables first. The most common strategy in association rule mining is to split continuous variable ranges into discrete intervals (bins). For example, one can split the range of intrusion into three categories: low values, medium values and high values. It is much more reasonable though, to split continuous responses at one particular threshold into valid and invalid. The threshold value usually originates from technical specifications or legal regulations. This makes sense, because we want our car design to meet its requirements.

**Which Classification Method was Chosen?** In order to predict discrete target values, any classification algorithm is basically a valid choice. Nonetheless, the algorithms often differ vastly in their inherent properties, and do not fit well to all kinds of data. The most popular ones are Support Vector Machines (SVM) [8], Artificial Neural Networks (ANN) [46], Decision Tree Learning (DT) [57, 10], Naive

Bayes Classification and Adaptive Boosting [22]. Almost all classification models, except for boosting, cannot handle our low sample counts properly, and thus can be ruled out immediately. Adaptive Boosting is a method which relies on other weak classification methods, and combines them to achieve better predictions. In early and also later tests, adaptive boosting with decision tree stumps (AdaBoost-SAMME [71]) did not perform as well as Decision Tree Learning (DT-CART [10]) itself, in terms of prediction accuracy. It always seemed that the low sample count was too extreme, even for boosting. In the end, decision tree learning was chosen as the base technology because it has the following important properties:

- Handling of low sample counts
- White-box model (easy inspectable and communicable)
- Automatic selection of important variables
- Handling of discrete and continuous variables

DTs already satisfy some of the side-challenges of crash simulation from chapter 1.4. What distinguishes DTs most from the other algorithms is the outstanding performance when facing low sample counts. Since low sample counts are in high demand, there is already almost no alternative to DTs. A property which is neglected a lot these days, is the fact that DTs are white-box models. It is very simple to inspect and understand their choices. Furthermore, DTs do two very specific things, which will be very useful later on: they simplify (in the upper area of the tree) and their decisions are easily translatable to humans. This makes DTs especially attractive for human-machine interaction.

**Why was Decision Tree Learning combined with Rule Mining?** Decision tree learning was chosen as the basic classification method, but the technique itself is very unwieldy. An engineer simply never wants to search a decision tree himself for good design recommendations. Therefore, an extension is required, in order to make it more useful. A very good complementary technology is Rule Mining from the field of Knowledge Discovery in Databases (KDD). Rule mining tries to find if-then conditions, where the if-part describes a design space, in which the then-part is predicted (e.g. if $X_5 < 4$ then $Y_4 < 3$). Traditional Association Rule Mining is mainly meant for discrete variables and

search rules with the Apriori algorithm [2]. In contrast to this, the rule search in this thesis is based on DTs, which adapts rule mining to our requirements in a more suitable way. A part of rule mining is also rule filtering. The discipline offers many different criteria in order to rank or reduce the number of mined rules, since one hundred rules or more is serious. This work proposes automatic rule filtering to find the largest and most interesting interactions. The most important rules are passed to the engineer, who may check them. In consequence, a major idea of this thesis is especially the use of automatic knowledge extraction, which should make rule mining attractive for simulation engineers.

**What are Possible Use-Case Scenarios for Rule Mining?** The full extent, of how rules may be utilized, is still open for discussion and research. Nevertheless, in this thesis, the following scenarios are proposed:

- Determine a new valid design
- Use rule bounds as optimization constraint

As stated previously a rule describes a design space, in which a corresponding target is fulfilled, such as triggering a specific deformation mode. An engineer may now choose a specific design himself from this subspace, without having to worry about a violation anymore. In this scenario, rule mining can be seen as a quick tool for getting multiple, simple solution recommendations.

The rules themselves could also be used as constraints for a subsequent optimization. This would prevent undesirable types of deformation or constraint violations. The issue is, that the rule boundaries are simplified in such a way that humans can understand them more easily. An optimization algorithm would simply be better off using the meta-model behind the rules itself, i.e. a decision tree, or even better, a random forest. The optimization algorithm could ask the random forest directly whether the design would be valid or not.

# Chapter 4

# Mathematical Formulations

This chapter will explain the two core algorithms of this thesis. These two algorithms are deformation behavior segmentation and rule mining, which were already introduced in the process flow of figure 3.1. The behavior segmentation uses dimensionality reduction, low-dimensional embedding and clustering, in order to find groups of simulations which deform in a similar manner. The derived deformation classes are then handed over to the rule mining algorithm, which extracts all the relevant knowledge for an engineer.

## 4.1 Deformation Class Segmentation

The first major contribution of this thesis is a process for deformation segmentation. This process originates from the desire to know about all major types of deformation throughout all simulations. An overview of the entire process of deformation class segmentation can be found in figure 4.1. The output of this step is a deformation class matrix $c_{sk}$, denoting in which simulation run $s$ what type of deformation $k$ occurred. This research was published previously in [13, 14].

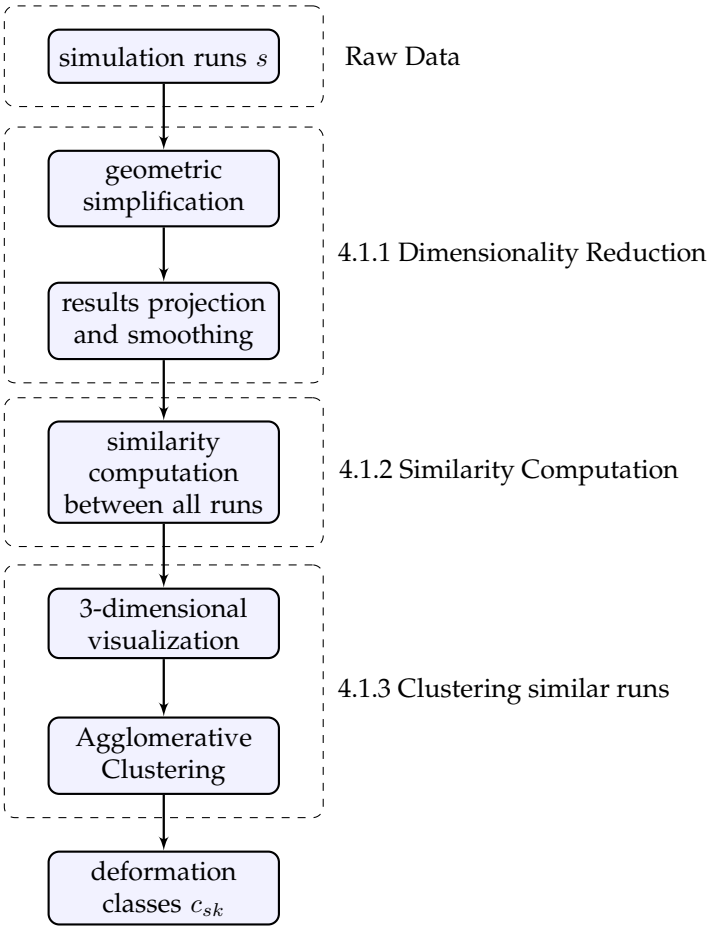



FIGURE 4.1: Roadmap for deriving deformation classes $c_{sk}$ from the raw simulations results. Dimensionality reduction enables the computation of a simulation similarity. These similarities are then used to cluster simulations with similar results. The clusters are then saved as deformation classes.

### 4.1.1 Geometry-Based Dimensionality Reduction

This section will explain the algorithm for geometry-based dimensionality reduction. The input for the procedure is:

- a list of part ids (of the crash-absorbing structure)
- a list of simulation result files
- the polynomial degree for geometry simplification

The output will be a function of reduced plastic strain for every simulation run. This function is a signature for the mesh-based results and has a lower dimensionality. The signature can be stored, accessed and compared much faster than the raw data.

#### 4.1.1.1 Geometric Simplification

Most structural parts in a car allow a simplified representation by a 1D line or a 2D plane since engineers use beams and sheets to guide loads through the structure. This intrinsic assumption is the control of geometry simplification. Firstly, the geometry simplification extracts the position $\vec{p}_s$ of every finite element $e_s$ from the specified group of parts, and performs a parametric Bézier regression through these (either a line or a plane). The output of this step is not the regression itself, but we are interested in finding the closest point of every element $\vec{\tilde{p}}_s$ on the regression. This closest point has a parametric coordinate vector $\vec{u}_s = [u_s, v_s]^T$, which describes the location on the two-dimensional surface. The output will be the parametric coordinates $\vec{u}_s$ belonging to every element. Note that the index $s$ is used in this thesis to index samples in general. In this section, $s$ is not a simulation index, but an element index.

**Why Nonlinear Parametric Bézier Regression?** Parametric nonlinear Bézier regression was chosen for two reasons: the ability to approximate curved structures well, and a very good repeatability. The regression has to be parametric, since the group of parts might be oriented arbitrarily in space and might resemble structures, such as a semi-circle. The algorithm in this thesis was originally based on [51] and generalizes it from solely curves to surfaces as well.

**How is a Parametric Beziér Polynomial Described Mathematically?** A Beziér polynomial in this thesis is a 2-dimensional surface in 3D. It is defined by a 2D-grid of control points, which span the surface. Such a grid is illustrated in figure 4.2. In the case of a line regression, we simply collapse the second dimension $M$, which makes line regression a special case.

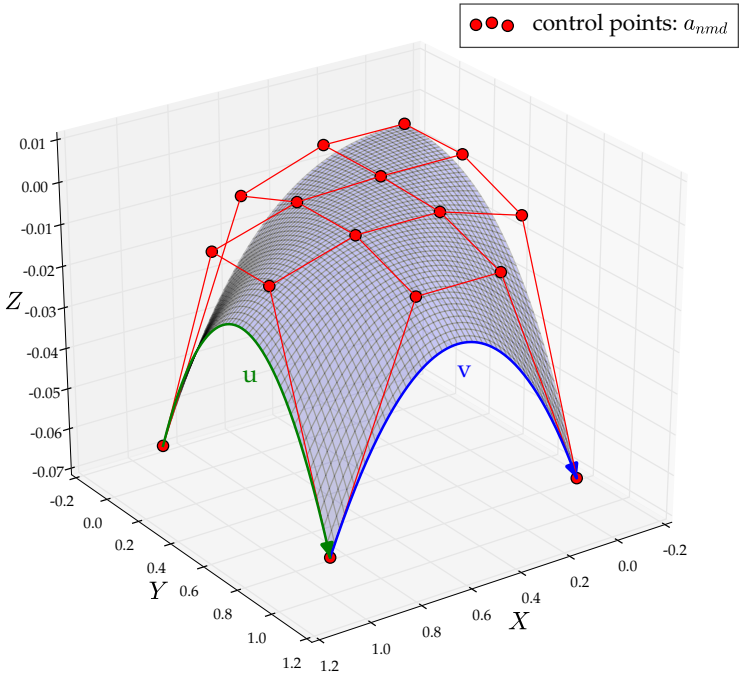



FIGURE 4.2: Example of a Bézier surface, which is spanned up by a grid of four control points in each direction ($N = 4, M = 4$). The endpoints are always on the surface.

The polynomial degree in the first and second parametric dimensions ($u$ and $v$) will be recalled as $N, M \geq 0$. $N$ and $M$ can also be seen as indexes for a 2-dimensional grid of control points, which span up the polynomial. The spatial coordinates of the grid points are saved in a polynomial coefficient tensor $a_{nmd}$, which has the shape $(N, M, 3)$. A point on the surface will be noted by $\widetilde{p}_d$, where $d \in \{1, 2, 3\}$ is the index for x, y and z. This point on the surface can be computed from two parametric coordinates $u$ and $v$,

$$\widetilde{p}_d(u, v) = \sum_{n=0}^{N} \sum_{m=0}^{M} B_n^{(N)}(u) \cdot B_m^{(M)}(v) \cdot a_{nmd}, \tag{4.1}$$

with the Bernstein polynomial,

$$B_n^{(N)}(u) = \binom{N}{n}(1-u)^{N-n}\, u^n. \tag{4.2}$$

The polynomial is always normalized here, so that $u$ and $v$ are bounded by,

$$u, v \in [0, 1]. \tag{4.3}$$

When performing regression, the task is to find the control points $a_{nmd}$ , approximating a 3D cloud of points.

**How High Can the Polynomial Degree Be Selected?** The engineer has to specify the polynomial degree with two parameters: $N, M \geq 0$. The number of control points corresponds to the polynomial degree plus one. Selecting $M = 0$ collapses the second dimension $v$ and performs a line regression, thus the result is just a one-dimensional curve (in 3D). The right choice of the polynomial degree should be made carefully. If the polynomial degree is too low, then the data cannot be approximated well enough. If on the other hand, the polynomial is too high, oscillations will happen and may lead to divergence. The divergence occurs, since the equation systems condition number will automatically get worse with higher polynomial degrees. In practical experiments, our algorithm could handle regression in 1D ($M = 0$) with $N \leq 20$, which is quite good. Regression in 2D proved to be much more sensitive, causing early divergence, usually at $N, M > 3$, which is too low for many practical cases. The automatic choice of the correct polynomial degree is possible through k-fold cross-validation, but since this would increase the computational effort by multiple times, the choice of the correct polynomial degree for every run is left to the user. The engineer usually configures the polynomial degree from one or two samples, before analyzing the entire ensemble. Note that if desired, every simulation may have a different polynomial degree. An illustration of regression is given in figure 4.3.

**What is the Process Flow of the Regression Algorithm?** The regression algorithm needed multiple additions to [51] in order to work well enough. The entire process is outlined in figure 4.4 and will be explained briefly in this paragraph. More detailed explanations can be found in following paragraphs.

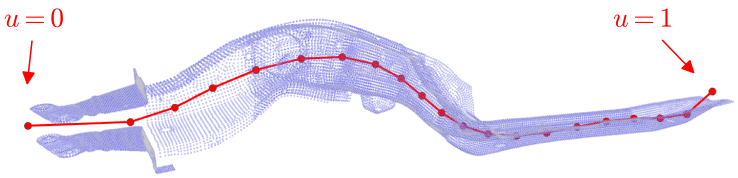



FIGURE 4.3: Example for geometric simplification for the rail of a car. The beam-like structure is simplified to a 1D curve with parametric Bézier regression ($N = 9, M = 0$).

Firstly, a subsampling is done, if the point count is too high. The samples are picked from a uniform distribution. This ensures a constant runtime for larger components. Thereafter, the point cloud density is homogenized. This homogenization simply tries to get rid of dense mesh regions, which can be the result of mesh refinement. This is important, since regression is based on the mean value, which is susceptible to such a concentration of points. A dense mesh region would attract the regression, and in consequence, the regression would no longer run through the geometrical center of the component. Thereafter, one has to estimate the parametric position of every element on the not-yet existent regression. This is the most susceptible step of the regression algorithm, since a bad estimation of $u_s$ and $v_s$ for every subsample $s$ will result in a bad fit or divergence. The algorithm cannot recover well from a wrong estimation. Having estimated the parametric position $u_s$ and $v_s$, the control points' coordinates of the new Bézier polynomial ($a_{nmd}$) can be computed. Since these change the polynomial, one has to correct the parametric position $u_s$and $v_s$ of every subsample $s$ on the regression. This can be understood as a projection of every sample onto the regression with shortest distance. Because we already have a previous estimation of $u_s$ and $v_s$, we simply perform a correction step, which is cheaper than a new computation. If the algorithm converges and subsampling is used, then all other elements also require the computation of the closest position on the regression.

**How Was the Point Cloud Density Homogenized?** For every subsample, the k-nearest-neighbors (kNN) are selected. By default, we

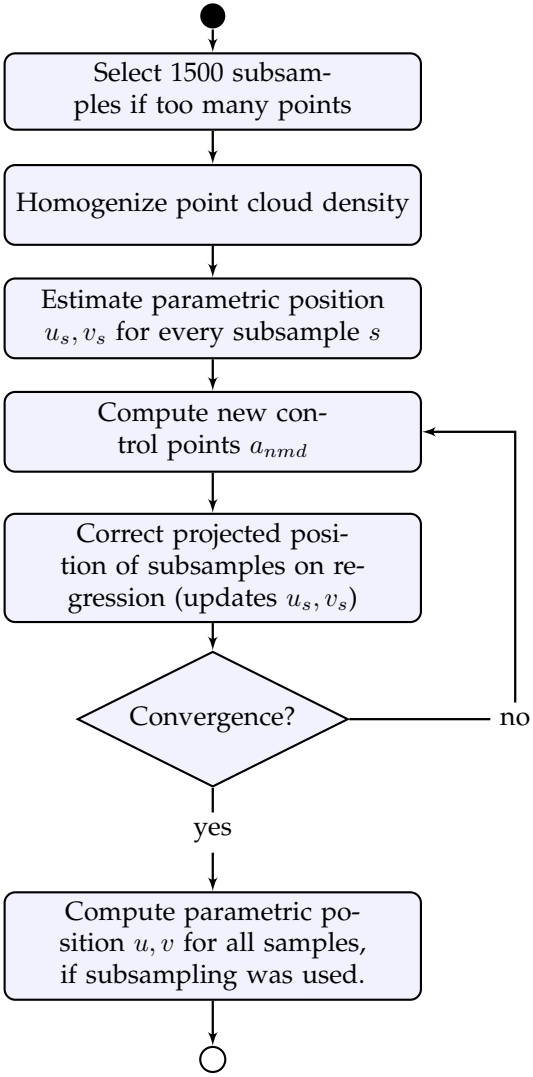



FIGURE 4.4: Algorithm for parametric Bézier regression.

are using 18 neighbors here. We compute an average distance to these neighbors, and assign the average distance to our subsample. At this

point, every subsample $s$ has an average neighbor distance $\overline{d}_s$. Afterwards, we compute the median of the average distances,

$$d^{(target)} = \text{median}(\overline{d}_s), \tag{4.4}$$

which will be the target distance for homogenization. These distances will be crucial in the following computations.

The point density $\rho_s$ around a sample $s$ can be computed by dividing the number of neighboring points $N_p$ by a volume $V$, which is defined by the points average neighbor distance $d_s$,

$$\rho_s = \frac{N_p}{V(\overline{d}_s)}. \tag{4.5}$$

In order to smooth the point cloud density, every point with a density higher than the median density should be assigned a probability to survive a random selection process,

$$\begin{aligned} P^{(survive)}(\rho_s \mid \rho_s < \rho^{(target)}) &= \frac{\rho_s}{\rho^{(target)}} \\ &= \frac{V(\overline{d}_s)}{V(d^{(target)})} \\ &= \frac{2\pi\overline{d}_s^2}{2\pi d^{(target),2}} \\ &= \left(\frac{\overline{d}_s}{d^{(target)}}\right)^2 \end{aligned} \tag{4.6}$$

The number of points gets canceled out in the equation 4.6, since we always used the same amount of neighbors for kNN-computation. The definition of a volume depends on the intrinsic dimension of the component. For solids, a sphere can be assumed, whereas for metal sheets, a circle makes more sense. Since most of the structural components can be expected to be metal sheets, the formula for a circle was selected. The simplest form of the probability can hereby be reduced to a fraction of the distances by the power of two. Interestingly, one can also generalize this formula quite easily to three dimensions,

$$P^{(remove)}(\overline{d}_s \mid \overline{d}_s < d^{(target)}) = \left(\frac{\overline{d}_s}{d^{(target)}}\right)^{\lambda}, \tag{4.7}$$

where the exponent $\lambda$ is a penalty factor, depending on the intrinsic dimension of the component. $\lambda = 2$ shall be used for sheet components, whereas $\lambda = 3$ is used for solid ones. The probability in eq. 4.7 is tested against a random number from a random number generator, which decides whether the point will be selected or not.

**How is the Parametric Position of Every Subsample Estimated?** Before the first regression can be computed, one has to estimate roughly where every sample will be on the regression. This is quite troubling, since no regression exists yet. This estimation is done differently for 1D and 2D regression.

For a 1D regression ($M = 0$), the estimation of $u$ can be done with affine invariant chord-length estimation of point clouds [21, 50]. However, the estimation did not work well in all the test cases. In consequence, a graph-based approach was chosen. This requires the construction of a graph from the subsamples. In order to do so, every subsample is connected with an edge to its six nearest neighbors. Along this graph, we simply search for the shortest path between the most distant points with Dijkstras algorithm [16]. The shortest path is simply a concatenation of linear segments. After having found this shortest path, we can estimate $u_s$ for every subsample $s$, by searching for the closest point on the shortest path to every subsample.

For a 2D regression ($M \geq 1$), the estimation is done by computing the principal components of the point cloud with PCA. The first and second principal components of every subsample will be taken as $u_s$ and $v_s$. Since PCA is a linear method, strongly curved structures will suffer from a bad estimation and thus divergence is more likely.

**How Are the New Polynomial Control Points Computed?** Every subsample $s$ has a corresponding position in space,

$$\vec{p}_s \text{ or } p_{sd}. \tag{4.8}$$

The goal is to find new control points $a_{nmd}$, which fit our Bézier polynomial to our point cloud of subsamples. Since we have an estimation of $u_s$ and $v_s$ for each $\vec{p}_s$ available, we can compute two Bézier

matrices $b_{sn}^{(N)}$ and $b_{sn}^{(M)}$, which simply contain all the coefficients needed by eq. 4.1 in one step,

$$\begin{aligned} b_{sn}^{(N)} &= B_n^{(N)}(u_s) = \binom{N}{n}(1-u_s)^{N-n}\, u_s^n, \\ b_{sm}^{(M)} &= B_m^{(M)}(v_s) = \binom{M}{m}(1-v_s)^{M-m}\, v_s^m. \end{aligned} \tag{4.9}$$

The idea is to minimize the squared error $\epsilon$ between all samples and their closest point on the regression,

$$\begin{aligned} \epsilon &= \sum_s \sum_{d=1}^{3} \left(p_{sd} - \widetilde{p}_{sd}\right)^2 \\ &= \sum_s \sum_{d=1}^{3} \left(p_{sd} - \widetilde{p}_d(u_s, v_s)\right)^2 \\ &= \sum_s \sum_{d=1}^{3} \left(p_{sd} - \sum_n^N \sum_m^M b_{sn}^{(N)} b_{sm}^{(M)} a_{nmd}\right)^2 \\ \epsilon &\to \min \end{aligned} \tag{4.10}$$

As is commonly done, a minimum in error can be found by setting the derivative for the polynomial coefficients to zero.

$$\begin{aligned} \frac{\partial \epsilon}{\partial a_{nmd}} &\stackrel{!}{=} 0 \\ &= 2 \cdot \underbrace{\left(\sum_s p_{sd} - \sum_{n=0}^{N}\sum_{m=0}^{M} b_{sn}^{(N)} b_{sm}^{(M)} a_{nmd}\right)}_{=0} \cdot \underbrace{\left(\sum_s \sum_{n=0}^{N}\sum_{m=0}^{M} -b_{sn}^{(N)} b_{sm}^{(M)}\right)}_{\neq 0} \end{aligned} \tag{4.11}$$

For equation 4.11 to be satisfied, the first underbraced part needs to be zero. Setting it to zero allows the computation of the new control points by using the pseudo inverse,

$$a_{nmd} = \sum_s \left(b_{sn}^{(N)}\right)^+ \cdot \left(b_{sm}^{(M)}\right)^+ \cdot p_{sd}. \tag{4.12}$$

It's important to note, that the pseudo inverse of a matrix transposes its shape from $(S, N)$ to $(N, S)$.

**How is the Closest Position of the Subsamples on the Regression Corrected?** By computing a new polynomial in equation 4.12, the closest point of every subsample on the regression shifts. Because the shift is not expected to be too large, a correction step should suffice. For simplicity, the following explanation is done for a single subsample. Mathematically, we want to minimize the residual $r_d$, which is the distance of our subsample to the regression,

$$r_d(u, v) = p_d - \widetilde{p}_d(u, v) \tag{4.13}$$

The residual can be approximated by a first order Taylor expansion around our previous closest point $\vec{u}_0 = [u_0, v_0]^T$,

$$\begin{aligned} r_d &\approx r_d(u_0, v_0) + \left.\frac{\partial r_d}{\partial \vec{u}}\right|_{u_0, v_0} \cdot \Delta uv_l, \\ &= r_d(u_0, v_0) + \sum_{l=1}^{2} JR_{dl}(u_0, v_0) \cdot \Delta uv_l. \end{aligned} \tag{4.14}$$

The index $l$ corresponds to the parametric dimension by $\vec{u} = [u, v]^T = u_l$. The Jacobian of the residual $JR_{dl}$ and the correction for the parametric position $\Delta uv_l$ have the following form,

$$JR_{dl} = \frac{\partial r_d}{\partial u_l} = \begin{bmatrix} \frac{\partial r_x}{\partial u} & \frac{\partial r_x}{\partial v} \\ \frac{\partial r_y}{\partial u} & \frac{\partial r_y}{\partial v} \\ \frac{\partial r_z}{\partial u} & \frac{\partial r_z}{\partial v} \end{bmatrix}, \tag{4.15}$$

$$\Delta uv_l = \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix}. \tag{4.16}$$

The derivative of the residual regarding $u$ and $v$ can be calculated analytically,

$$\begin{aligned}\left.\frac{\partial r_d}{\partial u}\right|_{u,v} &= \frac{\partial\left(p_d - \widetilde{p}_d(u,v)\right)}{\partial u} \\ &= -\frac{\partial \widetilde{p}(u,v)}{\partial u} \\ &= -\frac{\partial\left(\sum_n \sum_m B_n^{(N)}(u)\, B_m^{(M)}(v)\, a_{nmd}\right)}{\partial u} \\ &= -\sum_{n=0}^{N-1}\sum_{m=0}^{M} N\, B_n^{(N-1)}(u)\, B_m^{(M)}(v)\, \left(a_{(n+1)md} - a_{nmd}\right)\end{aligned} \tag{4.17}$$

$$\left.\frac{\partial r_d}{\partial v}\right|_{u,v} = -\sum_{n=0}^{N}\sum_{m=0}^{M-1} M\, B_n^{(N)}(u)\, B_m^{(M-1)}(v)\, \left(a_{n(m+1)d} - a_{nmd}\right) \tag{4.18}$$

After having assembled the Jacobian $JR_{dl}$, by setting eq. 4.14 to zero one can derive the correction step for the subsamples parametric position,

$$\Delta uv_l = -\sum_{d=1}^{3} \left(JR_{dl}(u_0, v_0)\right)^{+} \cdot r_d(u_0, v_0). \tag{4.19}$$

The subsample's new parametric position can now be derived by performing the update,

$$\begin{aligned} u^{(iter+1)} &= u^{(iter)} + \Delta uv_1 \\ v^{(iter+1)} &= v^{(iter)} + \Delta uv_2 \end{aligned} \tag{4.20}$$

The entire procedure until eq. 4.20 can be generalized to all subsamples $s$ in a vectorized form. Firstly, we compute the residual and also the Jacobian for all subsamples in the ensemble,

$$\begin{aligned} r_{sd} &= r(u_s, v_s) \\ &= x_{sd} - \widetilde{x}_d(u_s, v_s) \\ &= x_{sd} - \sum_n^N \sum_m^M B_n^{(N)}(u_s) B_m^{(M)}(v_s) a_{nmd} \\ &= x_{sd} - \sum_n^N \sum_m^M b_{sn}^{(N)} b_{sm}^{(M)} a_{nmd}, \end{aligned} \tag{4.21}$$

The Jacobian for multiple runs takes the form,

$$\begin{aligned} JR_{sdl} &= JR_{ld}(u_s, v_s) \\ &= \begin{cases} \sum_n^{N-1} \sum_m^M \left( N\, b_{sn}^{(N-1)} b_{sm}^{(M)} \left( a_{(n+1)md} - a_{nmd} \right) \right), & \text{if } l = 1, \\ \sum_n^N \sum_m^{M-1} \left( M\, b_{sn}^{(N)} b_{sm}^{(M-1)} \left( a_{n(m+1)d} - a_{nmd} \right) \right), & \text{else } (l = 2). \end{cases} \end{aligned} \tag{4.22}$$

With these two tensors, one is able to update all samples at once,

$$\Delta uv_{ls} = \sum_{d=1}^{3} (JR_{sdl})^+ \cdot r_{sd}, \tag{4.23}$$

$$\begin{aligned} u_s^{(\text{iter}+1)} &= u_s^{(\text{iter})} - \Delta uv_{1s}, \\ v_s^{(\text{iter}+1)} &= v_s^{(\text{iter})} - \Delta uv_{2s}. \end{aligned} \tag{4.24}$$

Note that the pseudo-inverse of a tensor of third order $(JR_{sdl})^+$ is mathematically undefined. The pseudo-inverse $+$ is broadcast along the last two dimensions $dl$ of the tensor, thus all Jacobian submatrices in the vector will be pseudo-inversed independently, but all at once.

This broadcasting is a single operation in a computer, which is several times faster than computing the inverse for every submatrix independently. At this point, the parametric position of the subsamples remains bounded by normalization,

$$u, v \in [0, 1]. \tag{4.25}$$

#### 4.1.1.2 Results Projection and Smoothing

This step transforms the simulations' results into a simpler, lightweight representation by mapping them onto the simplified geometry. The mapping process consists of projection and smoothing, which allows compensation for different meshes and moderate topology differences. The output from this step will be a distribution function $f(u, v)$, describing the distribution of a result field along the regression.

**Which Result Field was Chosen?** As result field, the equivalent plastic strain $\epsilon_s^{(p,eq)}$ of every element $e_s$ was chosen, since large deformations correlate well with plastic strain in crash simulation . The equivalent plastic strain is derived within the FEM-solver by integration of the equivalent plastic strain rate over time [28],

$$\epsilon^{(p,eq)} = \int_t \dot{\epsilon}^{(p,eq)}\, dt. \tag{4.26}$$

The equivalent plastic strain rate $\dot{\epsilon}^{(p,eq)}$ on the other hand, is computed from the plastic strain rate tensor,

$$\dot{\epsilon}^{(p,eq)} = \sqrt{\frac{2}{3} \sum_i \sum_j \dot{\epsilon}_{ij}^{(p)} \dot{\epsilon}_{ij}^{(p)}}. \tag{4.27}$$

In the following, we will simply refer to equivalent plastic strain $\epsilon^{(p,eq)}$ , as plastic strain $\epsilon^{(p)}$. The displacement was not chosen for the reason that the displacement field consists of two parts: the rigid body movement and the deformation field. In case of large movements, the rigid body fraction will hide the differences in deformation. In consequence, a similarity analysis later on would emphasize the position of the car after the impact, rather than its deformation. It's important to realize that plastic strain makes sense in case of ductile, metal components. It is not reasonable to compare for example, a metallic structure with a fiber reinforced one.

**How is the Smoothing Computed?** The projection of the element results has already been done, as a multitude of elements have already been projected onto the regression. Figure 4.5 shows the projected plastic strain for an example part (left rail sec. 5.2) with only one dimension, $u$. It is very obvious that multiple elements may have the same value of $u$. Also, very specific patterns are already visible, but it is still difficult to compare such scatter plots computationally. Therefore, a smoothing step, utilizing kernel density smoothing, is necessary. The result of the smoothing is a distribution function of plastic strain along the regression, which is also shown in figure 4.5.

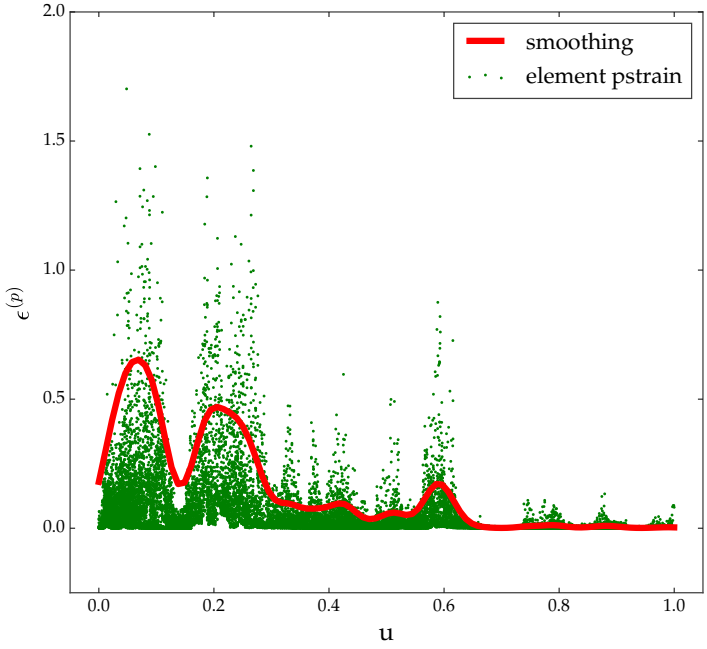



FIGURE 4.5: The plastic strain of every used element can be plotted along the parametric coordinate $u$ of the regression. This plastic strain can be smoothed into a distribution function of plastic strain, which can be compared more easily.

Firstly, we discretize $u, v \in [0, 1]$ with nodal points $\vec{u}_j$. At these nodal points, a value of the smoothed result field will be computed.

Therefore, we sum up the plastic strain of all elements $e_s$, but with an additional weight $w_{sj}$ for every element. This can be understood as positioning a local Gaussian at the position of the nodal point $\vec{u}_j$. The higher the distance between our nodal point and an element, the lower its weight. By doing this, only the plastic strain of elements in the locality of the nodal point contributes to its sum.

$$\begin{aligned} \epsilon_j^{(r)} = \sum_s \epsilon^{(r)}(\vec{u}_j) &= \epsilon_s^{(p)} \cdot w_{sj}, \\ &= \epsilon_s^{(p)} \cdot w(\vec{u}_s - \vec{u}_j), \end{aligned} \tag{4.28}$$

$$w(\vec{u}_s - \vec{u}_j) = \frac{1}{\sqrt{2\pi}b} \exp\left( -\frac{|\vec{u}_s - \vec{u}_j|^2}{2 \cdot b^2} \right). \tag{4.29}$$

The size $b$ of the Gaussian should be chosen to be larger than the element size of the mesh. We found that the following heuristic worked well,

$$b \geq 3 \cdot \text{median}\,(size(e_s)). \tag{4.30}$$

The heuristic in eq. 4.30 ensures that each entry in the smoothing vector $\epsilon_j^{(r)}$ is computed from a width of roughly 6-15 elements. Choosing such a large Gaussian makes the procedure more robust regarding the geometry simplification. If for example, the regression is not exactly the same, then two values $\epsilon_j^{(r)}$ of two different models will not be at the same geometric location. A larger Gaussian can compensate for this flaw and ensures a better repeatability.

The discretized function of reduced plastic strain $\epsilon_j^{(r)}$ can be saved with only a few kilobytes of memory, which is much less than the original FEM results. Since we can evaluate the discretized function of reduced plastic strain anywhere by using interpolation, we will refer to it in the following as distribution function of plastic strain $\epsilon(u, v)$.

Note, that one can see the distribution of plastic strain in figure 4.5 as a random field. As such, it might be characterized not only by the (local) mean value as above, but also additional properties such as the variance.

### 4.1.2 Computation of Similarity

The result from the previous section about dimensionality reduction, is a distribution function of plastic strain $\epsilon(u, v)$ for a single simulation. The goal of this section is to compute a similarity value between two of these functions (e.g. $\epsilon^{(i)}(u, v)$ and $\epsilon^{(j)}(u, v)$), resulting from two different simulations $i$ and $j$. The task of simulation comparison has been whittled down to the task of function comparison.

**What is the Concept of Similarity?** The intrinsic assumption at this point, is, that a typical type of deformation has a typical distribution function of plastic strain. This is true for unsymmetrical components, but not for uniform and symmetric ones, such as a pipe. This is for the reason, that if a pipe buckles to the left or right in exactly the middle, then the amount of plastic strain should be almost identical. From a heuristical point of view, this is not as troublesome as it seems, because the crash absorbing structure of a car is very inhomogeneous , consisting of many different sheets. If the rail of car a for example, buckles to the left or right, then it will have a different pattern.

**How is the Similarity Computed?** Function comparison can be done with multiple metrics, such as feature point alignment [17] or autocorrelation. The simplest approach though is to use an inner product for continuous functions $\langle \cdot | \cdot \rangle$, in order to compare two functions $\epsilon^{(i)}(u, v)$ and $\epsilon^{(j)}(u, v)$,

$$\widetilde{s}_{ij} = \int_{v=0}^{1} \int_{u=0}^{1} \epsilon^{(i)}(u, v) \cdot \epsilon^{(j)}(u, v) \, du \, dv := \left\langle \epsilon^{(i)} \middle| \epsilon^{(j)} \right\rangle . \tag{4.31}$$

By the multiplication of the functions in equation 4.31, their common peaks are strengthened, whereas mismatching peaks are canceled out. The integration of this product yields a value, which describes what both functions have in common. If the geometric simplification is a line, the outer integral along $v$ is neglected. Using normalization enforces a bounded similarity value,

$$s_{ij} = \frac{\left\langle \epsilon^{(i)} | \epsilon^{(j)} \right\rangle}{\sqrt{\left\langle \epsilon^{(i)} | \epsilon^{(i)} \right\rangle \left\langle \epsilon^{(j)} | \epsilon^{(j)} \right\rangle}} \in [0, 1]. \tag{4.32}$$

A simulation similarity of $s_{ij} = 0$ means, that the simulation results are entirely different, whereas $s_{ij} = 1$ means, that they are identical. In the following sections, we will need distances and not similarities. The dissimilarity itself is fortunately already a distance and is easily derivable,

$$d_{ij} = 1 - s_{ij}. \tag{4.33}$$

What distinguishes this procedure specifically from other simulation distance heuristics is the fact that the distance value is well bounded, which is not common. This makes it possible to compare just two simulations and immediately know how similar they are. If the resulting distance were to be bounded arbitrarily, then we could not know whether the distance value was high or not. As a conclusion, arbitrary bounded distances always require consideration of an ensemble of simulations. That the distance value in eq. 4.33 is well bounded, is not however as useful as it seems because in practice, one usually creates a design of experiments study with many simulation runs.

### 4.1.3 Clustering Similar Results

This section will cover the analysis of the simulation distances $d_{ij}$. The simulation distances will be used in order to find groups of simulations with similar behavior, either manually or by means of algorithmic clustering. The output of this section will be a clustering matrix $c_{ij}$. The clustering matrix $c_{ij}$ is a Boolean matrix, where every column $j$ stands for a cluster label and the Boolean entry indicates whether sample $i$ belongs to the cluster or not.

#### 4.1.3.1 Visualization with Low-Dimensional Embeddings

The purpose of a low-dimensional embedding is to visualize high dimensional data or distances, in 2D or 3D. A manual grouping can already be performed from this visualization, if the behavior of the structure is not too complex. The most common techniques for visualization of high-dimensional data are PCA, t-distributed stochastic neighbor embedding (t-SNE) [42] and multidimensional scaling (MDS) [7]. Besides these most common techniques, literally any dimensionality reduction algorithm is a candidate if it reduces the dimension to 2D or 3D.

**Why was Multidimensional Scaling Chosen?** MDS simply converts distances back into coordinates, but the dimensionality of the new coordinate space is specified by the user. For plotting purposes, a 2D or 3D embedding has to be chosen. The conversion is achieved with a dynamic relaxation, which tries to satisfy the originally high-dimensional distances $d_{ij}$ in the low-dimensional space. Doing so preserves the largest structures from the high-dimensional space and makes them visible to us.

**How is Multidimensional Scaling Working?** MDS simply converts distances back into coordinates, but the dimensionality of the new coordinate space is specified by the user. For plotting purposes, a 2D or 3D embedding has to be chosen. The conversion is achieved with a dynamic relaxation, which tries to satisfy the originally high-dimensional distances $d_{ij}$ in the low-dimensional space. Doing so preserves the largest structures from the high-dimensional space and makes them visible to us.

**Embedding for a Crushed Rail** Figure 4.6 shows a MDS visualization of 1000 simulations of the rail from figure 1.1. Every point in the plot is a simulation. The closer the points are, the more similar their simulation results were. The axis is the principal axis of the embedded space. The exact positional value of a point is unimportant in an embedding compared to its relational position regarding all other points.

The structure of the point cloud in figure 4.6 is surprisingly very clear and contains two clusters and two outliers. The representative samples *A* and *B* from both clusters reveal that buckling is occurring either at the top or bottom end. It must be emphasized here, that there is no connection between the clusters, thus there is no transitional behavior between these two states. The outlier *E* simply shows buckling initiation in a much lower region than any other sample. Another investigation revealed that this behavior is extremely rare. The second outlier *D* starts bending to the side, due to simultaneous buckling initiation in the upper and lower area. No similar sample was ever found in any investigation, but the behavior was tested to be reproducible by recomputing the sample multiple times.

It's also very noticeable that cluster *buckling_top* has a very large elongation. To understand the long elongation, six representative samples along a path from sample *B* to *C* are shown in figure 4.7. The

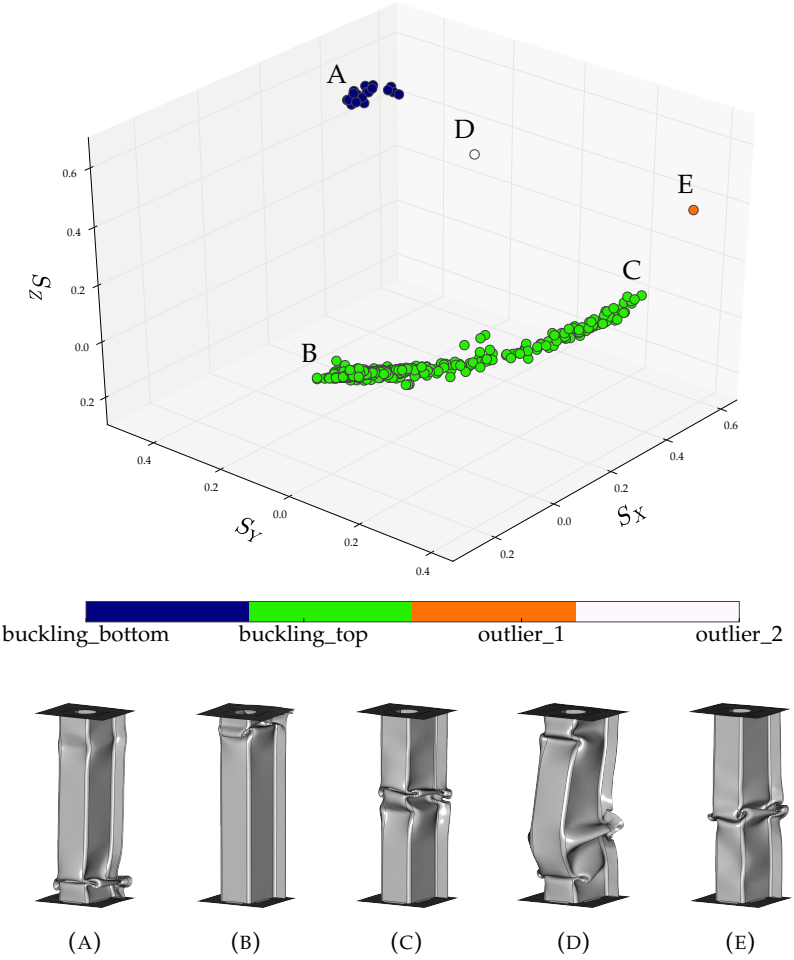



FIGURE 4.6: Low-dimensional embedding of the crashed rail from section 1.3 with MDS. $S_X$, $S_Y$ and $S_Z$ are the principal axis of the embedded similarity space. Simulation runs are picked exemplary in order to understand the clusters.

representatives reveal that buckling is not happening only at the upper end, but also in the middle. Watching such transitions is useful, in order to validate the embedding and also check its quality.

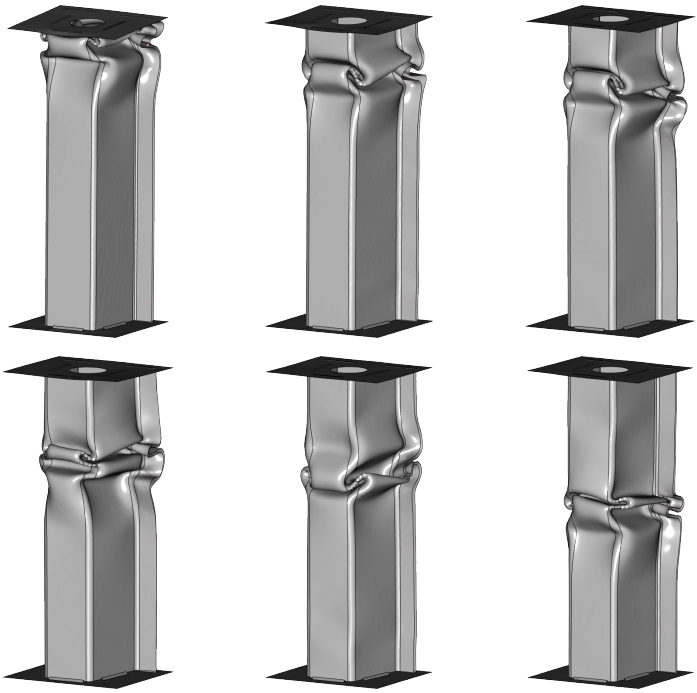


FIGURE 4.7: Choosing six samples in figure 4.6 from the representatives B to C visualizes the transitional behavior.

Analyzing representative samples lets the user build clusters with meaningful names (e.g. *buckling_top*). The separation of the runs in figure 4.6 is clear enough for manual clustering. For its analysis, four clusters were formed: two for the large clusters and two for the outliers. These groups are saved in a Boolean label matrix with shape $(1000, 4)$ in order to investigate possible causes of clusters later on. Technically seen, it is desirable for a rail in a car to fold from the region of impact downwards (here top to down), since firstly deformation shall be kept away from the passenger cell, and secondly the rail shall fold in a stable manner, which is not case if the rail starts buckling near the constrained end.

#### 4.1.3.2 Clustering

With clustering, it is possible to form groups of similar simulations from the distance matrix $d_{ij}$ directly. This makes sense, since the embedding is just an approximation of the distances and thus not reliable. Clustering has proven to be a useful tool, if the cluster boundaries are fuzzy, or the behavior of the structure is complex, thus the embedding is very twisted. The clusters in figure 4.6 are simple enough to be separated by hand, but in more complex cases, the usage of a clustering algorithm has shown to yield better clusters, as shown in section 5.

**What Clustering Algorithm was Chosen?** There are many clustering algorithms available. However, here our choice is limited, because we do not have coordinates, only distances. This already rules out very popular ones, such as k-means clustering [64, 43]. In experiments, Agglomerative Clustering [35] from the scipy package [36] works well with Average-Linkage [63], if the dataset does not contain too many outliers or noise. Another important property leading to the selection of this clustering technique, is its hierarchical nature, making it quite easy to investigate the data at different levels of granularity.

**How does Agglomerative Clustering Work?** Agglomerative clustering starts off by seeing every sample as a cluster. It then joins the closest clusters until there is only one left. The distance computation for the clusters is determined by the linkage. Average-Linkage literally averages the linking-distances between all samples of the two clusters $S_1$ and $S_2$:

$$D^{(average)}(S_1, S_2) = \frac{1}{|S_1|\,|S_2|} \sum_{i \in S_1} \sum_{j \in S_2} d_{ij} \tag{4.34}$$

The absolute value of a cluster or set $|\cdot|$ returns the size of the sets, which is the number of samples in cluster $S_1$ or $S_2$. The joining hierarchy, together with the joining distances, is the key component of the algorithm.

**Example for Agglomerative Clustering.** An example of agglomerative clustering with average-linkage is given in figure 4.8. The dataset is an imaginary dataset made up of three Gaussian distributions, where two Gaussians are very close to each other. The dendrogram of clustering shows the join hierarchy of the algorithm. If two clusters are joined

by the algorithm, the length of the vertical distance describes the join distance. The last join at the top of the tree, is by far the largest one, combining cluster 1 with cluster 2 and 3. Clusters can be derived from the joining hierarchy by simply cutting off the tree at a certain level. Because we used three Gaussians, we chose to cut the tree at a threshold level of $0.35$, retrieving our three original clusters. It's important to realize that determining the number of clusters equals cutting off the dendrogram at a certain join distance level.

**How to Determine the Number of Clusters?** The issue with clustering in general is that a cluster may obviously also contain sub-clusters, and it is difficult to automatically determine the engineer's level of interest, thus at which thresholds the tree should be cut off. Classically, the tree is cut off at one level. In practice, different branches need to be cut off at different levels, because the engineer might be interested in further details of a single cluster. In conclusion, 100% automation of this step is very difficult. Due to this rather philosophical question, the number of clusters and sub-clusters are determined by user input, interactively. Nonetheless this is not a real issue in practice, since interactive clustering doesn't take a lot time, and it is also essential to explore the embedding.

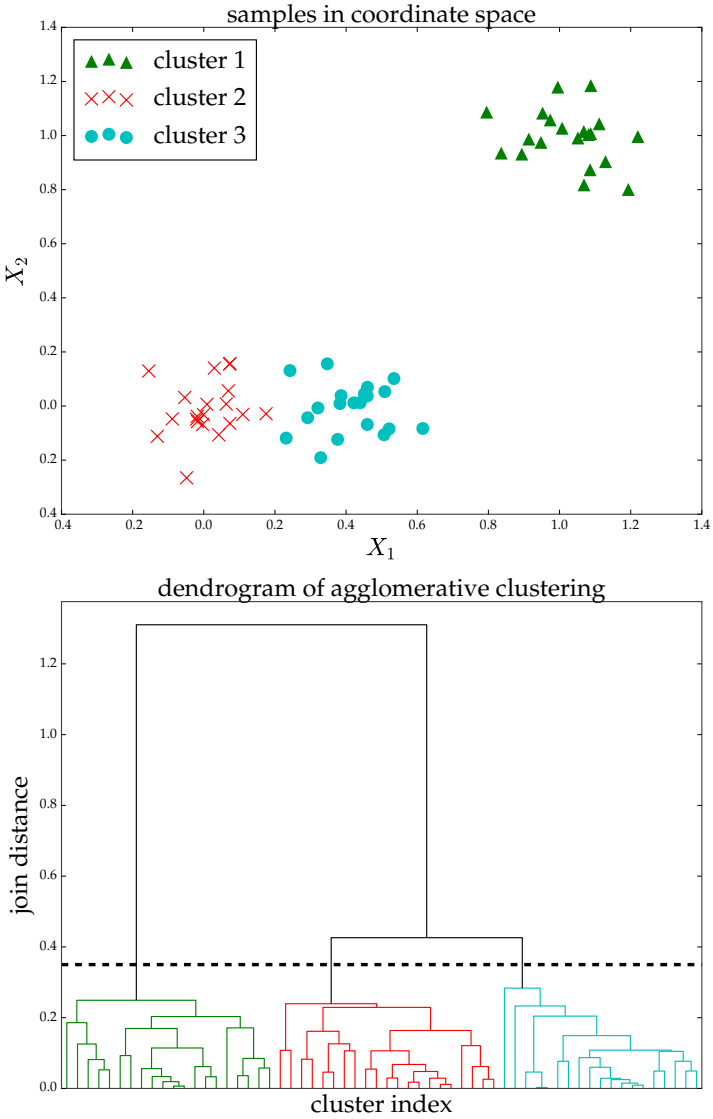



FIGURE 4.8: Visualization of Agglomerative Clustering on an imaginary dataset with two design variables $X_1$ and $X_2$. Cutting off the clustering hierarchy of the dendrogram at a value of $0.35$ collapses the branches into three clusters.

## 4.2 Decision Tree Learning

This section will briefly cover the basics of Decision Tree Learning, since it is the major backbone of this thesis. A very good and complete explanation of DTs can be found in [12], on which this section will also be partially based on.

### 4.2.1 Decision Tree Basics

**What is a Decision Tree?** A decision tree is a classification technique, originally intended for the prediction of discrete data, such as clusters or class labels. A DT is a tree-like structure of Boolean decisions, forwarding samples through its internal nodes to its leaves. Different leaves correspond to different class labels, so that if a new sample is given to the tree, it is forwarded into a leaf and its class will be predicted. The visualization of the decision boundaries of a DT is given in figure 4.9. The data originates from the clustering example in section 4.1.3.2. The DT in the figure classifies all samples correctly, except for one sample from class 2, which ends up in the wrong leaf (we enforced it here for explanatory purposes). The tree could split the impure leaf again to correctly predict all samples, but we forbid splitting if only a single sample would be segregated. If a single sample can be segregated, then the tree could fit itself to noise. Forbidding further splitting equals controlling the depth of a DT, which is usually called pruning. The training algorithm for the creation of the DT in figure 4.9 is called Classification and Regression Tree (DT-CART) [10] and will be explained in the next paragraph. This thesis uses the implementation of the scikit-learn package [54].

**How is a Decision Tree Built Up?** In this paragraph about the creation of DTs, we will focus mainly on DT-CART, which will be used throughout this thesis. The basic question at this point, is how to achieve the 'best' DT from the training data. Unfortunately, this problem was proven to be 'nondeterministic polynomial time complete' (NP-complete) [32], thus it is important to know that training relies heavily on heuristics. The method explained in the following, is also known as greedy search. Greedy search simply means that the algorithm makes the best split locally, and does not consider whether a different choice would be better further down the DT. In consequence, one can never expect to have the very best decision tree, in terms of prediction accuracy.

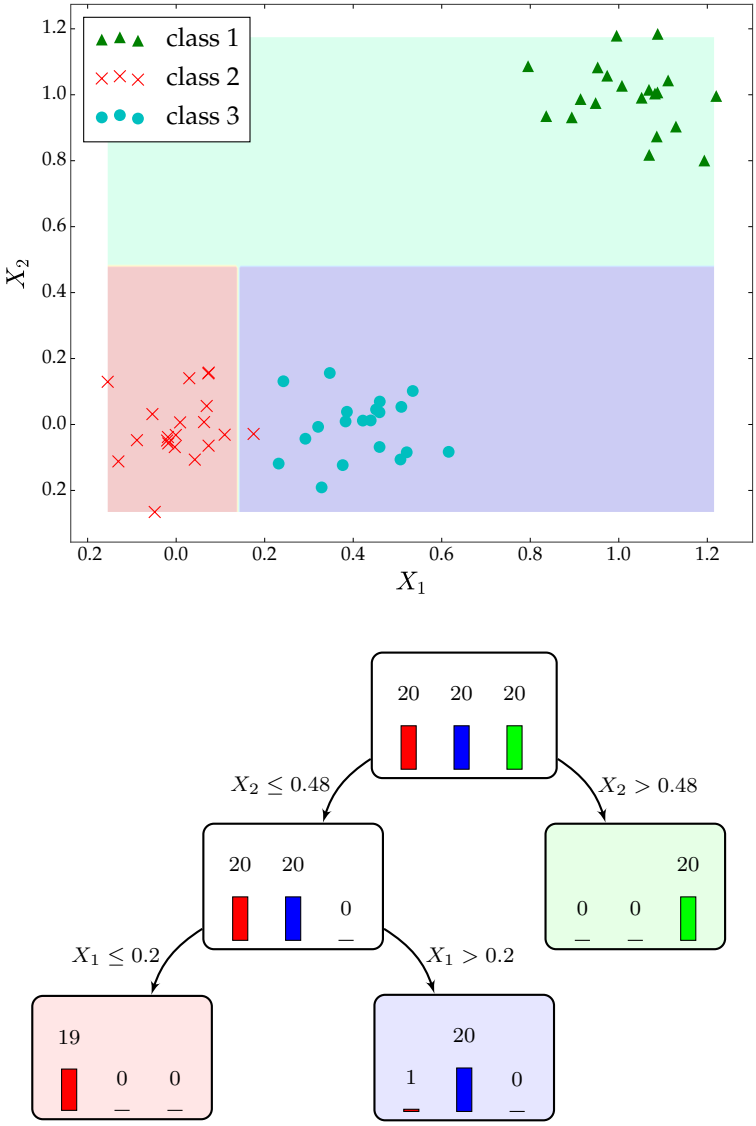



FIGURE 4.9: Visualization of a decision tree with the example dataset from figure 4.8. The tree simply slices the design space axis-aligned into subspaces, in order to isolate the training samples. The subspaces are described by the decisions leading to the leaf in the tree.

Training a decision tree requires the input variable matrix $x_{si}$ of all samples, and a vector of discrete target classes $c_s$. The algorithm now tries to split the set of samples $S = \{s|s \in (1, \ldots, N)\}$ with a new decision into two new subsets $S^{(L)}$ and $S^{(R)}$ recursively. These subsets ought to have a lower impurity than the original set $S$. The impurity of a set $S$ is measured with the Shannon entropy [62],

$$H(S) = -\sum_{C_k} p(C_k\,|S\,) \log_2 \left(p(C_k\,|S\,)\right). \tag{4.35}$$

The probability $p(C_k\,|S\,)$ is the likelihood of the $k$-th class given a set $S$. The probability $p(C_k\,|S\,)$ can be computed easily in a frequentist way, by counting how many samples in the set belong to this class,

$$p(C_k\,|S\,) = \frac{|\{s|s \in S \wedge c_s = C_k\}|}{|S|}. \tag{4.36}$$

The absolute value of a set $|\cdot|$ in eq. 4.36 simply counts the number of samples in a set. An example computation for the entropy of the lowest, impure leaf in figure 4.9 is,

$$\begin{aligned} H(S^{(example)}) &= -\frac{20}{21}\log_2\left(\frac{20}{21}\right) - \frac{1}{21}\log_2\left(\frac{1}{21}\right) \\ &= 0.276 \end{aligned} \tag{4.37}$$

It should be mentioned here, that the impurity of a set is also very frequently measured with the gini impurity $H_G$,

$$H_G(S) = \sum_{C_k} p(C_k\,|S\,) \cdot (1 - p(C_k\,|S\,)), \tag{4.38}$$

but in experiments, the Shannon entropy was found to yield a better classification accuracy throughout all our datasets. Being able to measure the impurity of a set makes it possible to compute the information gain $I$ of a split,

$$I(S, S^{(L)}, S^{(R)}) = H(S) - \left(\frac{|S^{(L)}|}{|S|}H(S^{(L)}) + \frac{|S^{(R)}|}{|S|}H(S^{(R)})\right). \tag{4.39}$$

The information gain can be understood as the difference between the entropy before the split, and the weighted entropy of both subset

leaves after the split. The central question is how to find the best variable with the best split in terms of information gain. Therefore, the algorithm first searches for the best split for every variable, and then chooses the best one from all of them. This can be understood as an optimization for the highest information gain for a single split, which explains the name 'greedy search'.

**What are the Training Settings for Decision Tree Learning?** The major difficulty when training a DT, is to control the depth of the tree, which is called pruning. If one does not limit the depth of the tree, then the tree is at risk of fitting itself to noise, which is called overfitting. The implementation used from the scikit-learn package [54] offers the pruning options enlisted in table 4.1.

| option | default | description |
|---|---|---|
| *max_depth* | None | Maximum tree depth allowed. |
| *min_samples_split* | 2 | The minimum number of samples required for a split. |
| *min_samples_leaf* | 1 | Stops growing the tree further, if a leaf contains the specified number of samples. |
| *max_leaf_nodes* | None | Maximum number of leafs allowed. |
| *min_impurity_decrease* | 0 | Allow only splits, which reduce the entropy by at least the specified value. |

TABLE 4.1: Options for decision tree learning (DT-CART) in the scikit-learn package.

Pruning is crucial for building a DT with a high, overall prediction accuracy, but it is not important for rule mining later on. This

is because the rule mining algorithm inherently focuses on the highest leaves of the tree, which can be considered to be relatively reliable. This removes the heavy burden of controlling pruning, and thus all default arguments are left as specified. The only setting which is changed for rule mining, is the tree depth, which is limited to 20 for higher performance.

**How to do Variable Importance Ranking?** A very interesting fact is that decision trees automatically do variable importance ranking internally, and can simply be asked for it. Every variable split within the tree has a corresponding information gain. Summing up all information gains for the respective variable, tells the user which variable helped most to structure the data. If one wishes to do variable importance ranking, it is recommended to use random forests (RF), for the reason that they generalize better, and thus are more reliable. RFs are explained in section 4.2.2.

**What is Decision Tree Learning Used For?** Decision tree learning will be utilized in the following for rule mining and variable importance ranking. In the case of rule mining, specific leaves of a DT are extracted as rules, and given to the user for better understanding and further usage. The automatic selection of the most important leaves can be difficult, and is explained in section 4.3. Decision tree learning is also used for variable importance ranking, where for a specified target class, the most important input variables should be returned. This is not done with DTs themselves, but with random forests in section 4.2.2.

### 4.2.2 Random Forests

Random forests use multiple decision trees in order to make a prediction. Therefore, a RF builds multiple trees. If a new prediction is requested, every tree will make its own prediction and the most frequent prediction is taken by voting. This would not work properly if every tree was trained in the same way, as explained in section 4.2.1, because all the trees would make the same prediction. Therefore, a RF uses multiple Random Trees (RT) and not classical DTs. A random tree is a decision tree, which is trained in a more random way. Randomness was introduced into the trees with the options explained in table 4.2.

| option | default | description |
|---|---|---|
| *max_features* | $\sqrt{nVariables}$ | Makes only the specified number of variables available for a split. The variables are chosen randomly. |
| *use_bagging* | True | Bagging gives every tree a random subset of samples. |

TABLE 4.2: Options for random forests to introduce randomness into its trees (scikit-learn package).

Reducing the number of variables available at every split, forces a tree to sometimes use lower ranked variables in terms of information gain. This softens the greedy search and helps to find alternative ways to explain a class.

**What is Bagging and How Does it Work?** Bagging [9] is used to improve the classification accuracy of a predictor by randomly generating subsets of samples during training. It was originally intended for DTs, but is used nowadays in combination with other classification methods, such as neural networks. The implementation of bagging used, is allowed to draw $N$ times from a set with originally $N$ samples. Because the selection follows a uniform distribution, a sample may be selected twice, thus the subset will very likely be smaller than the original set. The exact number of unique samples differs every time bagging is done, due to randomness. It was shown in [3], that if this bagging scheme is run a large number of times, then the mean value of unique samples in the subsets converges against $1 - \frac{1}{e} \approx 0.63$.

**Why is Variable Importance Ranking With Random Forests Better Than With Decision Trees?** A trustworthy variable importance ranking requires a classification technique, which understands the data as well as possible. This is different to the rule mining algorithm in this thesis, which requires a decision tree to be only precise in the upper area, and thus roughly. A random forest can generally be expected to have a higher prediction quality than a single decision tree for much better generalizing.

**How is Variable Importance Ranking Done With Random Forests?** Variable importance ranking with random forests does not differ a lot from importance ranking with decision trees. Instead of asking a single tree for the importance of every variable, a random forest returns an importance value for every variable from every tree. In consequence, one can compute the mean and standard deviation for the importance of every variable. In contrast to importance ranking with only a single tree, it is very useful to have the standard deviation, as a kind of error estimation. However, one has to keep in mind that the trees are forced to be random, thus the standard deviation is forced to blow up. Therefore, the standard deviation can be understood as some kind of worst-case estimator for the variation of the variable importances.

**Example for Variable Importance Ranking** In this paragraph, an example for variable importance ranking will be given. The data will once again originate from our crushed rail in figure 1.1. The maximum intrusion of the impactor was measured from every specimen. The idea is to find out what causes high intrusion values. Figure 4.10 shows the histogram of the intrusion. The overall histogram resembles a normal distribution. The only conspicuity is the high variability of the intrusion, ranging from about 50 mm to about 90 mm.

Because importance ranking can only be run on discrete target classes, the continuous response of intrusion needs to be converted into classes. We therefore split the samples in figure 4.10 into two classes: low intrusion and high intrusion. The importance ranking is now run on the target class of high intrusion. The output is shown in table 4.3.

| variable name | mean importance | std. dev. importance |
|---|---|---|
| *initial_velocity* | 0.31 | 0.028 |
| *impact_angle* | 0.12 | 0.029 |
| ... | $< 0.03$ | |

TABLE 4.3: Variable importance ranking for the class of high intrusion.

All mean importances in table 4.3 are normalized, thus sum up to one. The largest importance by far, is the impact velocity, which totally makes sense, due to its quadratic contribution to the kinetic energy.

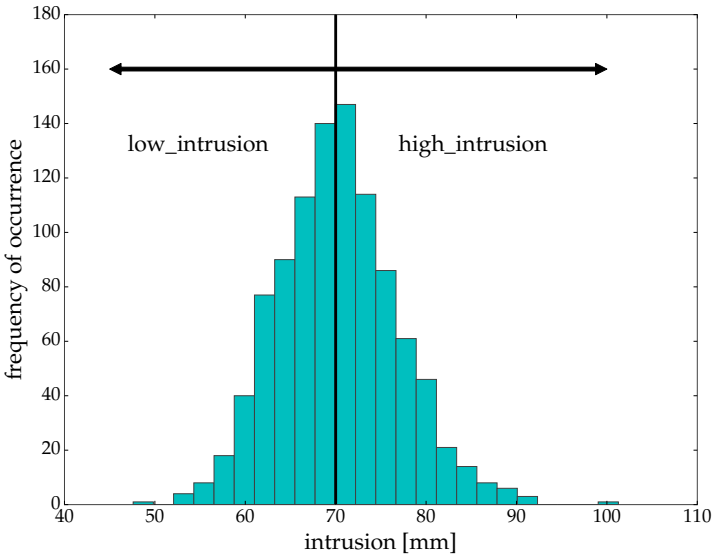



FIGURE 4.10: Histogram of the intrusion for the example of a crashed rail in figure 1.1. The underlying probability distribution resembles a normal distribution. For importance ranking the dataset is split into two classes.

Also, the impact angle seems to play an important role. An explanation would be that the component is geometrically much stiffer in the front, than in the rear. If the impactor is inclined in such a way that it hits the front first, it can be expected to have an impact on the intrusion. The standard deviation of the importances is relatively low, so that the estimates are trustworthy.

## 4.3 Rule Mining for Knowledge Extraction from Simulation Data

The central inference technique of this thesis is Rule Mining. The goal is to find if-then rules connecting input and output variables, unveiling potential physical relations within the data. A rule is an if-then condition

$$A \rightarrow T, \tag{4.40}$$

where the antecedent condition $A$ leads to the conclusion $T$ with a certain accuracy (also known as confidence) regarding the underlying data. An example for a rule is the following expression: $X_1 > 4 \rightarrow \neg C_1$, where the deformation class $C_1$ is prevented (literally 'not $C_1$') by selecting the design variable $X_1$ larger than 4. The target $T$ is usually specified by a user, whereas the mining algorithm searches for multiple, valid $A$. It is quite dangerous to accept such statistical conclusions blindly as facts, thus an engineer has to decide or check, whether the pattern can be accepted as a real physical effect. This thesis will focus on two tasks:

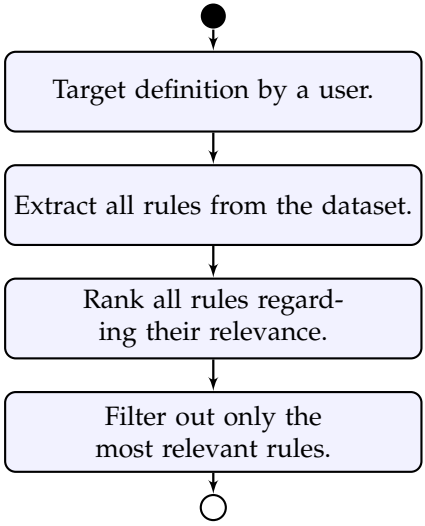



FIGURE 4.11: Process flow for automated rule mining in this thesis.

The algorithm starts off by requiring a target definition from a user. Having specified a target, the algorithm searches all rules it can find, which satisfy the target with a high accuracy. Thereafter the algorithm first ranks all rules regarding their relevance. Determining how relevant a rule is, depends on the metric. The field of rule mining provides many different metrics in order to score the relevance of rules. Since multiple scoring metrics are often used in combination, the first task was to find the metric or metrics, which score rules relating to our engineering demands. The relevance here is a combination of a rule's prediction quality and amount of samples predicted. Having found the most relevant rules leaves the question, how many rules are relevant enough to be shown to the engineer. This specifically extends classical rule mining, where the selection of the most relevant rules is done interactively by a user.

This automatic rule mining is very important, since a CAE-engineer simply does not have either the knowledge or the time to isolate the major dependencies himself. However, since the intent is quite clear, automation can be provided. The algorithm for rule mining was originally published in [41, 15].

### 4.3.1 Example Data

The rule mining will be explained with an imaginary dataset, shown in figure 4.12.

The example has two design variables: $X_1$ and $X_2$. Let's imagine that these two variables entirely determine the mechanical behavior of our structure, thus triggering two different types of deformation (e.g. buckling and no buckling). In such a case, all the samples can be split into two classes according to their deformation type. The function separating these two deformation classes in figure 4.12 is a parabola. A parabola as border is especially interesting, because it is difficult for a decision tree to separate classes with nonlinear boundaries.

### 4.3.2 Target Definition

Before performing rule mining, the targets first need to be specified. The definition of a target condition before rule mining is very similar to the definition of a cost-function before optimization. A target is simply a Boolean expression, separating all samples into two groups: true or false. Thereafter the Rule Mining will investigate what causes the separation of the samples into these two groups. This is achieved by

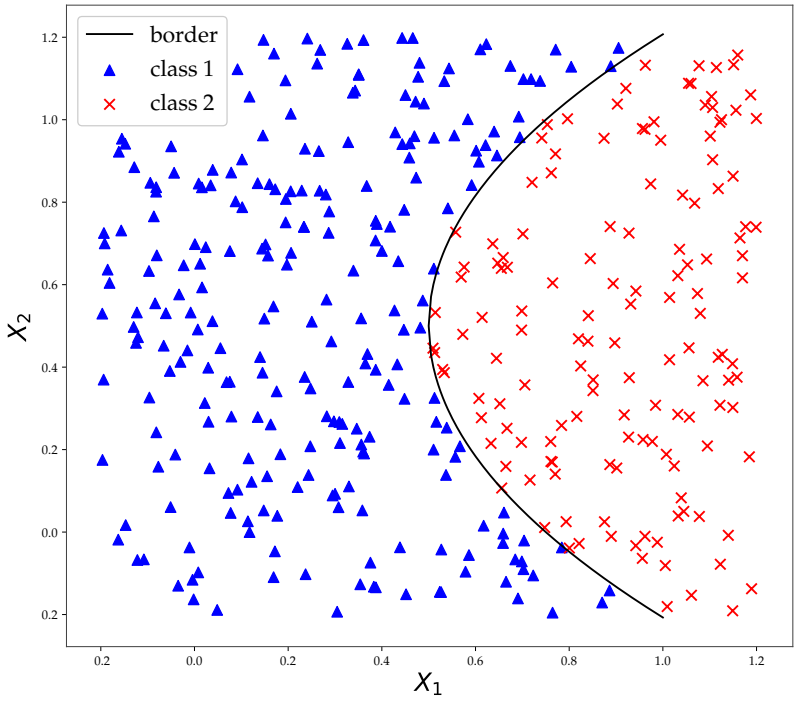



FIGURE 4.12: The example dataset consists of two classes, which are separable by a parabola in the design space $X$.

applying the target $T$ to all simulation samples $s$ in the database. The evaluation result is saved in a Boolean vector $t_s$, describing whether the expression evaluated true or false for a specific sample.

The simplest target definition is to prevent or trigger, a certain deformation mode. By choosing for example $T := \neg C_1$ (literally 'not $C_1$'), the engineer is asking how he can prevent deformation mode $C_1$. In the example case, this is the equivalent of triggering only $C_2$, because we have a two-class problem. In this section, the following target will be used for further explanations,

$$T := \neg C_1 = C_2. \tag{4.41}$$

A target can generally be defined from any variable in the matrix of outputs $y_{sj}$ or classes $c_{sk}$ (see figure 3.1). The only constraint is

that the result of this expression has to be Boolean. The simple target $Y_4 < 10$ for example, demands an output value lower than a specified threshold. While the output $Y_4$ itself is not Boolean, the evaluation of this inequality is. What makes this approach especially interesting, is that one may combine multiple targets arbitrarily (e.g. $Y_4 < 10 \wedge \neg C_1$), making a more complex analysis with multiple targets very easy to specify.

### 4.3.3 Rule Extraction

This section will explain how to attain a set of antecedent conditions $A_i$ leading to a single target $T$. This is also formally known as concept learning [23]. The extraction will be done on a trained decision tree (see section 4.2). This work uses DT-CART [10] from the scikit-learn package [54] to train the decision tree. The matrix of input variables $x_{si}$ and the target evaluation vector $t_s$ , are used for the training process.

**How to Extract Rules From a Decision Tree?** The extraction algorithm simply traverses the entire tree and converts every node into a rule expression, if the accuracy is above the specified threshold (see fig. 4.13). The rule simply consists of the decisions leading to the node itself. Extracted antecedents may be redundant for the reason that also child nodes may satisfy the accuracy limit, but this will be taken care of in section 4.3.4.

**What are the Settings for Rule Extraction?** The rule extraction in this thesis can be limited by an accuracy threshold $\alpha$ for the rules. In the field of rule mining, accuracy is more commonly known as confidence, and is explained in section 4.3.4. A small tolerance negates the effects of wrongly clustered samples, and in consequence heightens the chance of finding more general rules. Therefore, in order to find more general rules, a slightly lower accuracy (e.g. $\alpha = 95\%$) is crucial.

### 4.3.4 Automatic Rule Filtering

The rule filtering takes as input, a set of antecedent conditions $\mathcal{A} = \{A_i\}$ predicting a single target $T$ with a specified, minimal accuracy of $\alpha$.

$$\mathcal{A} := \{A_i \mid \text{conf}\,(A_i \to T) \geq \alpha\} \tag{4.42}$$

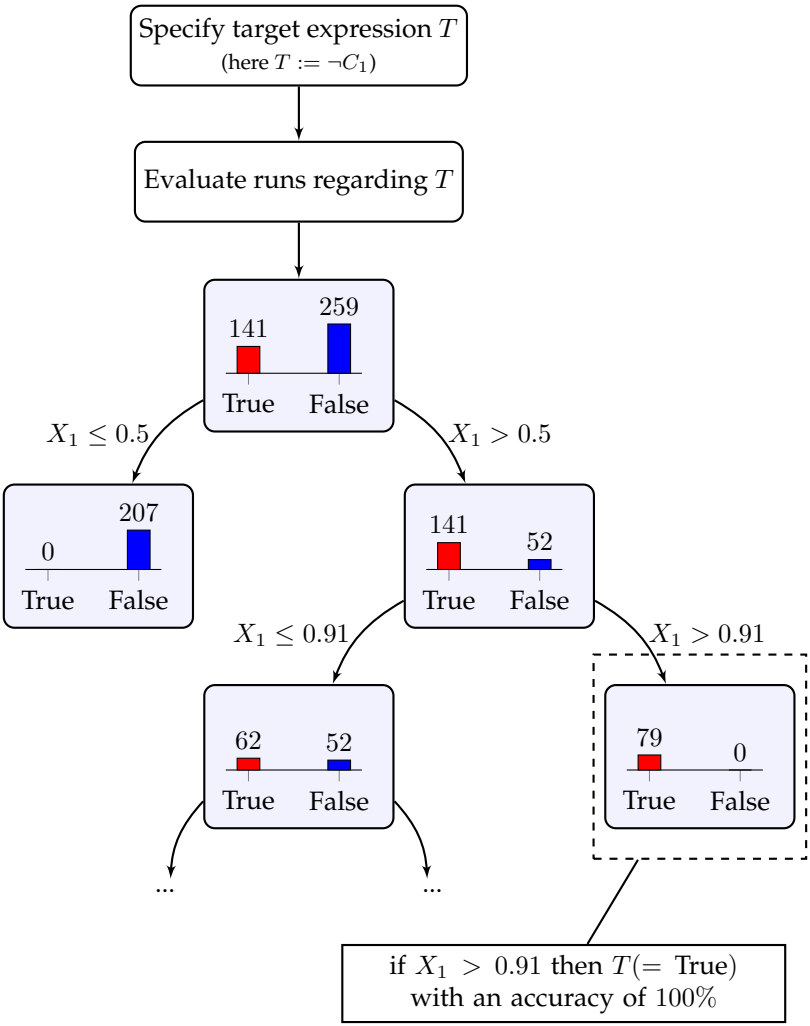



FIGURE 4.13: Rule extraction from a decision tree, which was trained with the labels True or False for a target $T$. The dataset is from the example in fig. 4.12. Every node or leaf is translated into an antecedent for a rule, if the accuracy (confidence) is high enough.

The $\mathrm{conf}$ function simply evaluates the accuracy of a rule regarding the given dataset, and will be explained more detailed later. The goal of this section is to return only the most meaningful $A_i$, therefore we have to apply ranking and filtering.

**What are Relevant Rules?** The question, which rules should be ranked highest, depends on the user's interest. Fortunately, simulation engineers have common interests, which leads to the following filtering targets:

- Large rules, which cover a large amount of the design space, should be ranked higher.
- More accurate rules should be ranked higher.

In first place, engineers prefer to know about the largest interactions within the simulation model, thus rules covering a large fraction of the design space are of major interest. Rough, large rules can be found in the upper area of the decision tree, where it is also relatively reliable in terms of prediction accuracy. Additionally, there is also the requirement that it predicts as many samples as possible, correctly. This is a trade-off, because large rules are rougher, and thus tend to have a lower accuracy. The trade-off was found to be difficult to realize in practice, but could be solved by a specific rule scoring and filtering scheme.

**What is the Major Assumption for Relevant Rules?** An implicit, but very important, assumption is that a rule covering many samples also covers a large design-space. This states to be true for uniform, space-filling sampling techniques, but becomes more problematic in the case of adaptive, local sampling. In such a case, a rule might be ranked high for covering many samples, but describes only a small and thus uninteresting, fraction of the design space.

**How Were Rules Scored?** Association Rule Mining provides a large number of criteria for scoring rules. This paragraph will explain the relevant criteria for this thesis.

---

**Definition 4.3.1.** The support of a rule,

$$\text{supp}\,(A \to T) = \frac{|A \to T|}{|S|} \in [0, 1], \tag{4.43}$$

states, how many samples (in percent) fulfill the rule $A \to T$ regarding a database set $S$.

---

**Definition 4.3.2.** The completeness of a rule,

$$\text{compl}\,(A \to T) = \frac{|A \to T|}{|T|} \in [0, 1], \tag{4.44}$$

denotes the fraction of how many samples showing the desired target $T$ can be predicted by the rule.

---

**Definition 4.3.3.** The confidence,

$$\text{conf}\,(A \to T) = \frac{\text{supp}\,(A \to T)}{\text{supp}\,(A)} \in [0, 1], \tag{4.45}$$

describes the accuracy of a rule. A value of 100% means, that the condition of $A$ has always predicted the target $T$ correctly.

---

**Definition 4.3.4.** The leverage function,

$$\text{lev}\,(A \to T) = \text{supp}\,(A \to T) - \text{supp}(A) \cdot \text{supp}(T) \tag{4.46}$$

from Piatetsky-Shapiro [55] states a rules interestingness. It can be understood as a value describing, how much the prediction of a rule deviates from a statistical random event.

---

**How are Rules Ranked Here?** We found the leverage criterion fit our filtering demands quite well, since it rates rules with a high prediction accuracy and also large sample counts higher. Therefore the set of mined antecedents from equation 4.42 is taken, and ranked according to the leverage function,

$$\begin{aligned}\mathcal{A}^{(ranked)} := \{A_j \mid & \operatorname{conf}(A_j \to T) \geq \alpha \\ & \wedge \operatorname{lev}(A_j \to T) > \operatorname{lev}(A_{j+1} \to T)\}.\end{aligned} \tag{4.47}$$

**How to Remove Redundant Rules?** The antecedents in equation 4.47 are ranked, but may still be redundant. Due to the structure of decision trees, a smaller rule may only be redundant regarding a larger rule, if the smaller rule is entirely contained within the larger one. This makes filtering especially easy. The algorithm simply iterates through $\mathcal{A}^{(ranked)}$ downwards, takes every rule and removes all following rules, which are contained in our higher ranked rule. The result is a set of unique design spaces,

$$\begin{aligned}\mathcal{A}^{(ranked\&unique)} := \{A_j \mid & \operatorname{conf}(A_i \to T) \geq \alpha \\ & \wedge \operatorname{lev}(A_j \to T) > \operatorname{lev}(A_{j+1} \to T) \\ & \wedge A_j \cap A_k = \emptyset \; \forall \; j \neq k\}.\end{aligned} \tag{4.48}$$

The set of ranked and unique rules is visualized for our example dataset in figure 4.14. The algorithm has extracted a total of seven rules. The two largest zones are ranked highest, as desired, but many smaller ones still exist. Note how the rules also do not intersect. Having extracted the seven rules raises the question, how many rules are actually interesting for an engineer.

**How to Determine the Number of Relevant Rules?** Having ranked the rules, leads to the final step of filtering. An algorithm ought to determine how many rules are actually relevant, and thus worthy of being shown to the engineer. In order to determine the number of relevant rules, the completeness $\gamma_j$ for every entry $A_j$ in set $\mathcal{A}^{(ranked\&unique)}$ needs to be computed,

$$\gamma_j = \operatorname{compl}(A_j \to T) \; \forall \; A_j \in \mathcal{A}^{(ranked\&unique)}. \tag{4.49}$$

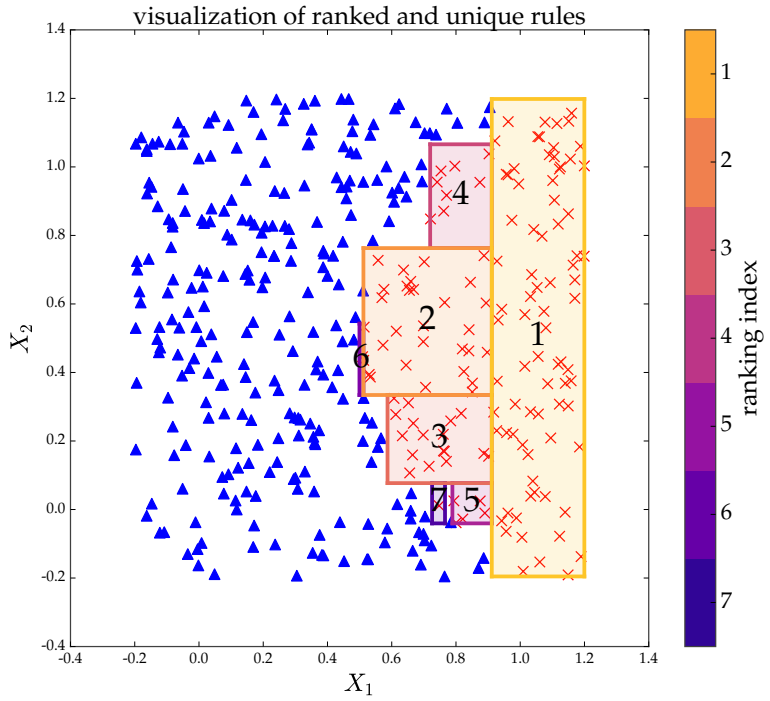



FIGURE 4.14: Visualization of the ranked and unique rules for the example dataset of figure 4.12. The target class (red) can be isolated with seven rules, where the two largest ones are also highest in rank as desired.

For further filtering, we need the cumulative sum of the completeness,

$$\Gamma_j = \sum_{k=1}^{k=j} \gamma_k. \tag{4.50}$$

The cumulative sum $\Gamma_j$ for a ranked and unique rule $j$ sums up the completeness of all higher ranked rules $k$. This simply states how many samples of target $T$ (in percent) are predictable, if one is using the $j$ highest rules. There are two important properties to note for the cumulative sum of completeness: Higher ranked rules (usually) have a larger completeness, and the sum of the completeness will always be

1 (100% target samples covered). In consequence, plotting the cumulative sum resembles a square root function, converging against 1 (see figure 4.15 left).

The major question is, at which point is it not worth using an additional rule. In this thesis, we postulate this point to be the outermost knee point of the cumulative completeness. Searching for the outermost knee, can be done with the 'elbow method', which is often used to determine the number of relevant clusters during clustering [26]. In order to find the largest knee, we simply compute the second derivative $\Delta^2\Gamma_j$ by using forward differentiation,

$$\Delta\Gamma_j = \frac{\Gamma_{j+1} - \Gamma_j}{1}, \tag{4.51}$$

$$\begin{aligned}\Delta^2\Gamma_j &= \frac{\Delta\Gamma_{j+1} - \Delta\Gamma_j}{1} \\ &= \Gamma_{j+2} - 2\Gamma_{j+1} + \Gamma_j,\end{aligned} \tag{4.52}$$

and search for the minimum,

$$\#\text{rules} := \underset{j}{\operatorname{argmin}} \left(\Delta^2\Gamma_j\right). \tag{4.53}$$

This is also illustrated in figure 4.15. By using this automatic filtering, the algorithm determines two major rules for the example dataset. These are shown in figure 4.16.

Figure 4.15 illustrates the automatic rule filtering for the example dataset. The algorithm has determined that only two rules from originally seven (see fig. 4.14) are relevant. The most relevant rule has the bounds,

$$0.91 < X_1 < 1.2 \rightarrow T, \tag{4.54}$$

and the second most relevant rule has the limits,

$$\begin{aligned}0.51 <& X_1 < 0.91 \\ 0.33 <& X_2 < 0.76\end{aligned} \rightarrow T. \tag{4.55}$$

**How many samples are required for rule mining?** This algorithm is inherently tuned for low-sample counts, by using decision tree learning. The number of samples required for mining depends heavily on

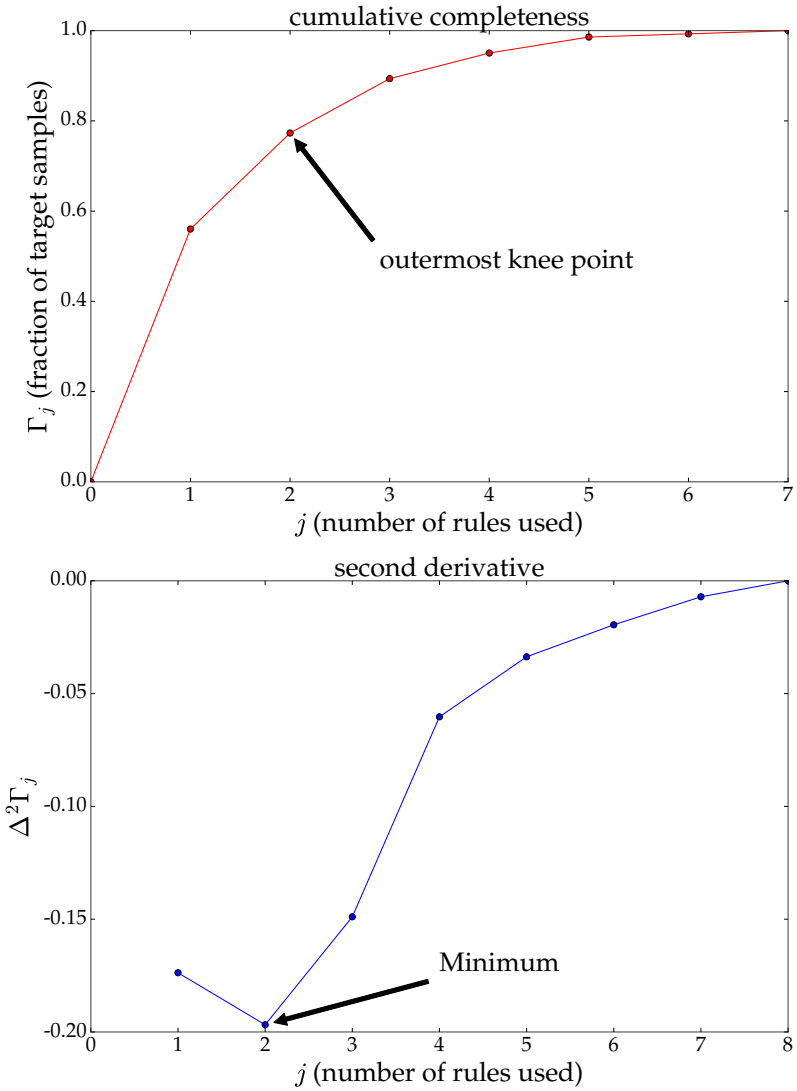



FIGURE 4.15: The cumulative completeness (top) states the fraction of samples covered when using $j$ rules. The number of relevant rules is defined here to be the outermost knee of the function, which is also the largest change in curvature.

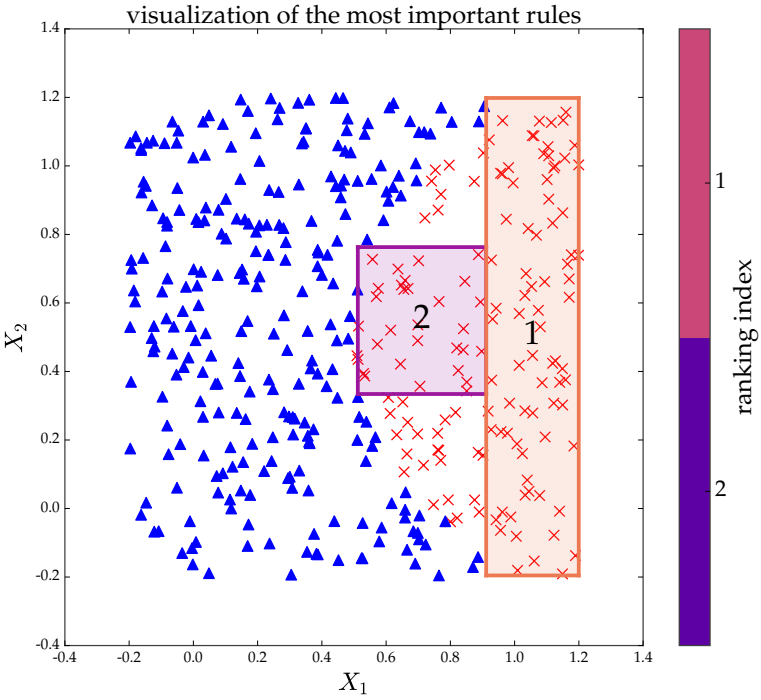



FIGURE 4.16: The most relevant rules determined by automatic rule filtering for the classification dataset from figure 4.12.

the data. It's obvious, that large rules can be found much more easily than smaller ones. In [15], it was shown for the dataset from section 5.2.2.1, that a large rule with a support of $20\%$ (200 samples) could even be found by drawing only 50 subsamples, which was twice as much samples as the number of variables. A small rule with a support of only $5\%$ (50 samples) could no longer be mined reliably, showing the limits of this algorithm. However, it's important to keep in mind here that a lot of tuning for low sample counts can be done, especially testing different decision tree learning algorithms or by using further rule learning techniques. In consequence, the real capabilities of rule mining regarding low sample counts, needs not only more testing but more importantly, also much more research, in order to give a profound statement.

### 4.3.5 Example for Rule Mining

The rule mining algorithm will be demonstrated for the crushed rail dataset from figure 1.1. For this dataset, a 3-dimensional embedding was computed (see fig. 4.6) and two clusters were identified. The cluster *buckling_bottom* showed buckling initiation at the constrained end of the rail, which will be considered undesirable here. Therefore, our rule mining target will be defined as,

$$T := \neg \, buckling\_bottom. \tag{4.56}$$

Even though the dataset consists of 1000 simulations, the cluster *buckling_bottom* contains only 20 samples, which can be considered to be very low, especially because the dataset has 173 input variables. The rule mining finds the following rules for the target in eq. 4.56:

$$0.33 < impact\_angle \rightarrow \neg buckling\_bottom, \tag{4.57}$$

$$-0.40 > impact\_angle \rightarrow \neg buckling\_bottom. \tag{4.58}$$

It seems that all thes amples which show *buckling_bottom*, have an impact angle around zero. Simply avoiding this range entirely prevents the buckling mode. This can also be seen in figure 4.17. An improvement of the structure is therefore possible, by simply inclining the upper end of the component with a slight angle, in order to ensure stable buckling.

### 4.3.6 Making Rules Reliable

By using the previously explained algorithm, it is possible to automatically mine good, rough rules from a decision tree. The issue is though, that the rule bounds might be close to the class border, making them highly unreliable in this area. In order to further increase the quality of the rules, the probability of an antecedent $A$ not predicting the target $T$ correctly should be limited (literally 'the probability of not target given a design space $A$ shall be smaller than a specified limit'),

$$P(\neg T|A^*) \leq P_{limit}. \tag{4.59}$$

**Optimization Formulation** The goal is to find a new design space $A^*$, which satisfies the inequality 4.59. Therefore, a rule antecedent

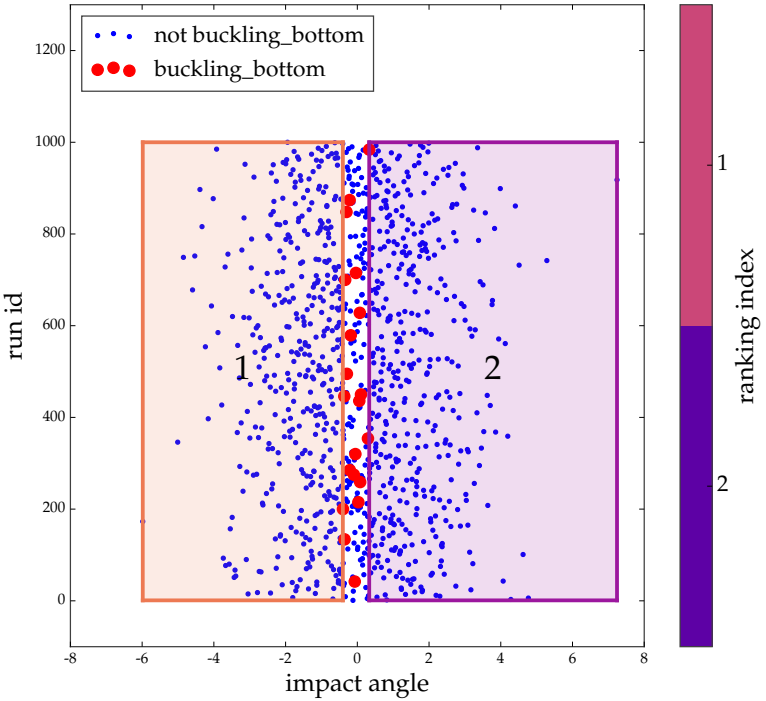



FIGURE 4.17: The two most important rules in order to avoid *buckling_bottom* for the rail dataset from figure 1.1 . The small rule is ranked higher, because it contains more samples, which is a flaw resulting from non-homogenous sampling.

will be seen as a function of its boundary values, which are assembled in a vector $\vec{z}$,

$$A = A(\vec{z}). \tag{4.60}$$

By using $\vec{z}$ as design vector, the entire task can be reformulated as an optimization problem,

$$A^* = A(\vec{z}^*) = \underset{\vec{z}}{\operatorname{argmin}} \left( P(\neg T | A(\vec{z})) - P_{limit} \right)^2 . \tag{4.61}$$

The probability $P(\neg T|A(\vec{z}))$ can be computed from Bayes' theorem,

$$P(\neg T|A) = \frac{P(A|\neg T)P(\neg T)}{P(A)}. \tag{4.62}$$

**Computing all Probability Fragments** At this point, we need to derive all three parts on the right side of eq. 4.62. The probability of not fulfilling the target does not depend on the rule boundaries, and can be computed quite easily,

$$P(\neg T) = \text{supp}(\neg T). \tag{4.63}$$

The other two probabilities need to be estimated. Therefore, a Kernel Density Estimation (KDE) for their Probability Density Function (PDF) is used. The probability function itself will be derived by integrating the PDF

$$P(\vec{w}^{(min)} \leq \vec{w} \leq \vec{w}^{(max)}) = \int_{\vec{w}^{(min)}}^{\vec{w}^{(max)}} pdf(\vec{w}) d\vec{w} \tag{4.64}$$

The vector $\vec{w}$ simply contains all input variables used by the rule, and thus also used by the optimization. Because $\vec{z}$ contains also the lower and upper bounds of the rule at once, it can be rewritten from $\vec{w}$,

$$\vec{z} = \left[\vec{w}^{(min),T}, \vec{w}^{(max),T}\right]^T. \tag{4.65}$$

The KDE is created by using a Gaussian kernel. Therefore, the KDE simply builds the PDF by positioning a small Gaussian at every sample point, and sums them up. In consequence areas with many sample points have a high probability density,

$$pdf(\vec{w}) = \frac{1}{N_{samples}} \sum_{i=0}^{N_{samples}} \frac{1}{\sqrt{det(2\pi \underline{\underline{K}})}} exp\left(-\frac{1}{2} \widetilde{\vec{w}}_i^T \vec{K}^{-1} \widetilde{\vec{w}}_i\right), \tag{4.66}$$

$$\widetilde{\vec{w}}_i = \vec{w} - \vec{w}_i. \tag{4.67}$$

In eq. 4.66, $\underline{\underline{K}}$ is the kernel matrix and $\widetilde{\vec{w}}_i$ is the distance between location $\vec{w}$ and sample $i$ in the rules variable space. The kernel matrix $\underline{\underline{K}}$ has the diagonal form,

$$K_{ij} = \begin{cases} 3.49\hat{\sigma}_j N_{samples}^{-\frac{1}{3}} & ,\forall\ i = j \\ 0 & ,\text{else} \end{cases}, \tag{4.68}$$

and uses Scott's rule of thumb [61] for every variable $j$. The rule of thumb depends on the estimate of the standard deviation $\hat{\sigma}_j$. The optimization requires the computation of the partial derivatives of the probability function eq. 4.64 regarding the rule limits,

$$\frac{\partial P}{\partial \vec{z}} = \left[\frac{\partial P}{\partial \vec{w}^{(min)}}, \frac{\partial P}{\partial \vec{w}^{(max)}}\right]^T. \tag{4.69}$$

These derivatives can be computed numerically, but it is much more efficient to use an analytical derivative. The analytical derivative requires the matrix $\underline{\underline{K}}$ to be diagonal, which means, that the rule variables are treated independently. Using this assumption, the analytical derivatives are:

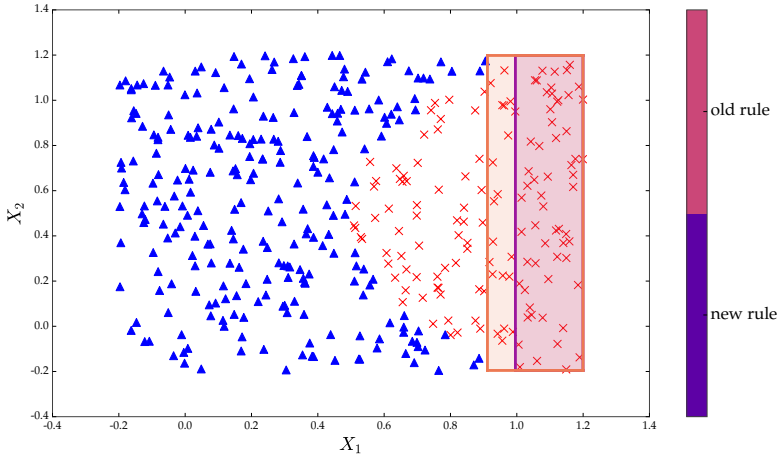
$$\begin{aligned} \frac{\partial P}{\partial w_q^{(min)}} = -\frac{1}{N_{samples}} \sum_{i=0}^{N_{samples}} & \frac{1}{\sqrt{det(2\pi\underline{\underline{K}})}} \\ & \cdot \left( \prod_{j,j\neq q}^{N_{vars}} \frac{\sqrt{2\pi K_{jj}}}{2} \left( erf\left(\sqrt{\frac{1}{2K_{jj}}}\widetilde{w}_{ij}^{(max)}\right) \right.\right. \\ & \qquad \left.\left. -erf\left(\sqrt{\frac{1}{2K_{jj}}}\widetilde{w}_{ij}^{(min)}\right)\right)\right) \\ & \cdot exp\left(-\frac{1}{2K_{qq}}\widetilde{w}_{iq}^{(min)}\right), \end{aligned} \tag{4.70}$$

$$\frac{\partial P}{\partial w_q^{(max)}} = -\frac{1}{N_{samples}} \sum_{i=0}^{N_{samples}} \frac{1}{\sqrt{det(2\pi\underline{\underline{K}})}} \cdot \left( \prod_{j,j\neq q}^{N_{vars}} \frac{\sqrt{2\pi K_{jj}}}{2} \left( erf\left( \sqrt{\frac{1}{2K_{jj}}} \widetilde{w}_{ij}^{(max)} \right) - erf\left( \sqrt{\frac{1}{2K_{jj}}} \widetilde{w}_{ij}^{(min)} \right) \right) \right) \cdot exp\left( -\frac{1}{2K_{qq}} \widetilde{w}_{iq}^{(max)} \right). \tag{4.71}$$

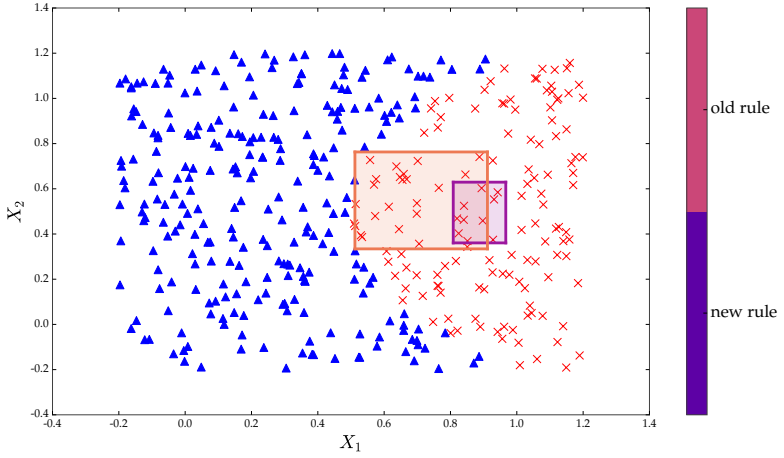
How equations 4.71 and 4.70 are derived is outlined in more detailin [41].

**Example case for Rule Optimization** In order to illustrate the rule optimization, the example dataset's most important rules (see fig. 4.16) will be optimized for higher reliability. For the example dataset, we want to limit the probability of the two most important rules being wrong to $1\%$, which is very low.

Figure 4.18 shows the result of rule optimization. The two base rules, which will be optimized, are from this section's example dataset. The bright rule in every plot is the old base rule, the dark one is always the new, optimized rule. The probability of each rule being wrong shall be limited to $1\%$. The border of the most important rule touches the class border only at the upper and lower end. Therefore, the rule is not reduced in size too much by optimization. The second, most important rule is more volatile, because it shares the entire left edge with the other class. If an engineer were to choose a design exactly at the border, any physical scatter might cause a violation. The rule optimization reduces the size of the rule a lot, and also shifts it a little bit to the right. A small shift may happen, but a large shift is impossible, due to the term $P(A)$ in eq. 4.62. If a large shift were to be necessary, then the optimization would fail, which indicates that the rule is not optimizable for the specified reliability. After optimization, a design can be chosen by the engineer anywhere in the optimized rule, even at the borders, with the assurance that the design will be safe.




(A) Optimization of the most important rule.




(B) Optimization of the second most important rule.

FIGURE 4.18: The probability of rules to be wrong was limited to 1% by optimization of the rule bounds.

# Chapter 5

# Examples

## 5.1 Buckling analysis of a cantilever plate

This example is an impact simulation of a ball onto a sheet of metal and originates from [18]. Since the structure itself is 2-dimensional, it will serve as an example for planar dimensionality reduction. This example is of interest, since it combines an unstable sheet structure with an uncertain, dynamic load, thus being a good example for robustness analysis. Also such sheet impacts occur frequently during the crash simulation of a car. The impact is happening at such an angle, that the deformation may deviate strongly given slightly different initial conditions. As such, this model is expected to yield multiple deformation patterns and thus is more difficult to handle for the dimensionality reduction algorithm in contrast to the rail model from section 1.3.

### 5.1.1 Model Description

The model in figure 5.1 resembles a cantilever beam which is constrained at the left end and receives a traversal load on the right. The load is the upward impact of a ball onto the frame. The Cantilever Plate has a width of 150mm and a height of 100mm and is stabilized by a small frame with a width of 10mm. The impacting ball has a diameter of 30mm. The mesh has an average element-size of 1.5mm which results in 8704 elements for the plate and 4903 elements for the ball. The material model of the plate is defined by the LS-Dyna keyword *MAT_PIECEWISE_LINEAR_PLASTICITY (MAT24) describing a yield curve for isotropic hardening. The element formulation chosen was ELFORM 16, which are fully integrated shells ( four in-plane integration points [29]) in combination with five integration points in thickness direction. The ball has a mass of 0.001175kg. Also as a note,

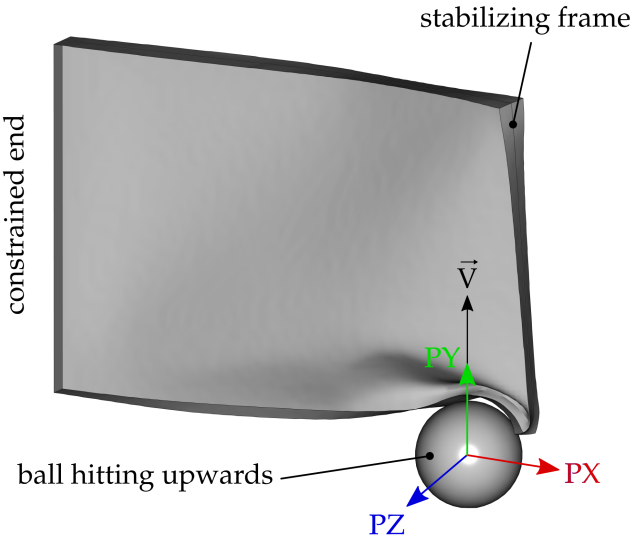



FIGURE 5.1: The cantilever plate is constrained at the left end and is hit on the right side with a ball, which is flying upwards. The balls initial position (*PX*,*PY*,*PZ*) and initial velocity vector ($\vec{V}$) are varied for every simulation.

the time integration schema in LS-Dyna is using a modification of the central difference method [29].

**Input variables** The main purpose of the model was to trigger, detect and relate different buckling modes to the input variables. In total, there are only six input variables: three variables for the variation of the velocity vector, and three for the variation of the ball position. The variables are varied as follows:

The distance of the ball to the plate was only varied numerically by its vertical position *PY*, in order to induce slightly different contact situations in the FEM solver. The variation of *VX* and *VZ* causes the ball not to hit in a vertically perfect manner.

| variable | max. variation |
|---|---|
| $PX$ | $\pm 5mm$ |
| $PY$ | $\pm 0.001mm$ |
| $PZ$ | $\pm 5mm$ |
| $VX$ | $\pm 50\frac{mm}{s}$ |
| $VY$ | $8000 \pm 300\frac{mm}{s}$ |
| $VZ$ | $\pm 50\frac{mm}{s}$ |

**Sampling Scheme** The idea of this model was to test the capabilities of planar dimensionality reduction with a highly unstable structure. To trigger different buckling modes, a uniform LHS design was chosen. In total, 1250 samples were generated and computed.

**Responses and Targets** This simulation model has no responses to keep track of. We are only interested in the variety of buckling modes and how they are caused.

### 5.1.2 Results of the Buckling Analysis

**What Major Deformation Modes Occur?** The embedding for the cantilever plate is shown in figure 5.2. The structure itself is very complex, revealing a highly unstable buckling behavior. In general, three major clusters are visible.

The two outer clusters in the figure are mainly longitudinal, whereas the cluster in the middle is very twisted and thus also hard to visualize. The strong twist is an indication that the entire structure does not have enough freedom to fully unfold itself in a lower dimension. Therefore, it seems better to make an individual embedding for every large clusters later on, to gain further insights.

In order to understand the difference between the clusters, representative simulations are shown. The representative samples are colored in an undeformed state, according to their plastic strain with a maximum of $2\%$ at the end of the simulation. This helps to compare and visualize buckling lines, and also illustrates the view of the dimensionality reduction algorithm. While *large_cluster3* has a very unique buckling style for its curvy buckling line, the other two clusters seem to be much more similar. On closer observation, *large_cluster2* has a strong, central buckling line, whereas in *large_cluster1* the buckling line never runs exactly through the center.

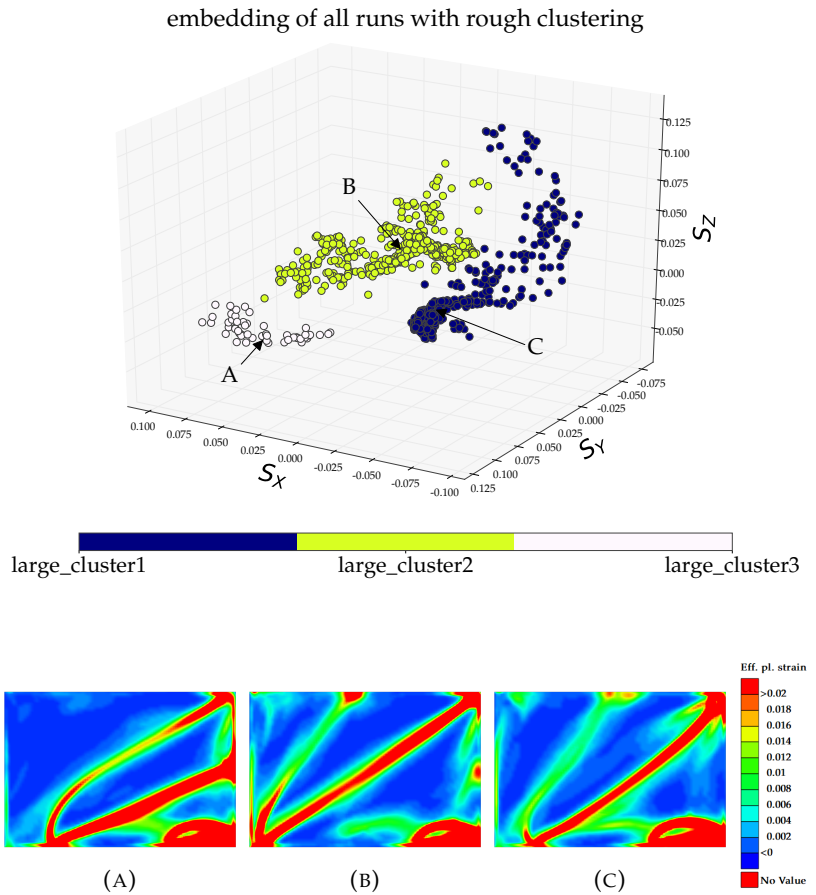



FIGURE 5.2: Embedding of all 1250 runs of the cantilever plate. The embedding can be separated into three major cluster, of which two are mainly longitudinal and one is very twisted.

**What Effects are Happening in *large_cluster1*?** This paragraph will specifically discuss *large_cluster1*, which is shown in figure 5.3. *large_cluster1* has one major, longitudinal dimension, and is separable into three reasonable sub-clusters. *cluster1* has a very high density of samples, and contains in total 498 samples, which are roughly 40%. Several representatives have been picked along the longitudinal dimension to illustrate the transition of states. The transition from *cluster1* to

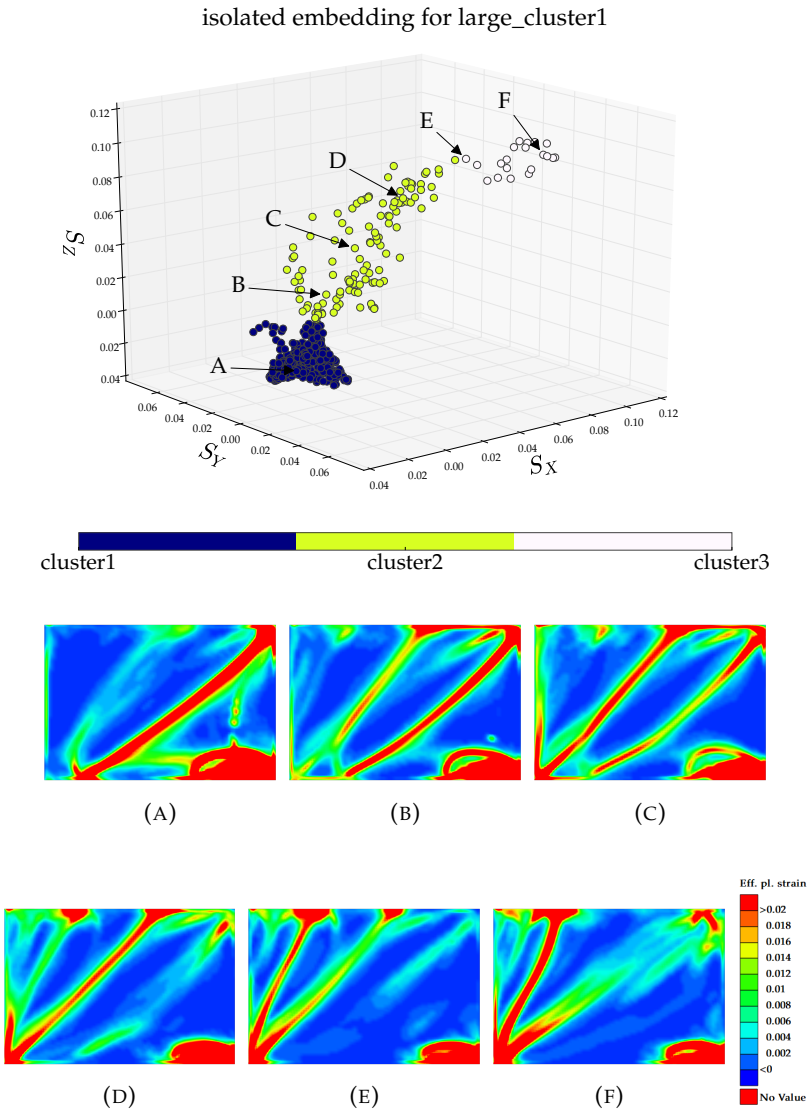



FIGURE 5.3: Embedding for solely *large_cluster1* from figure 5.2. The general structure is longitudinal, but also shows large scatter in the middle.

the middle of *cluster2*, shows that a second buckling line is developing in the top area. As this buckling line becomes stronger, the lower buckling line decreases in strength. Along the path from the middle of *cluster2* to the end of *cluster3*, a third major buckling line is developing in the top left corner. While this third buckling line is increasing, once again the other line simultaneously decreases.

**What Effects are Happening in *large_cluster2*?** In order to gain an understanding of *large_cluster2*, an embedding was created solely for this cluster. The embedding is shown in figure 5.4. Also *large_cluster2* can be split into 6 further sub-clusters. The main structure consists of *cluster4*, *cluster5* and *cluster6*. These three clusters are connected, and thus are expected to have a smooth transition. The difference along the transition is a shift from a buckling line in the lower right (see *cluster4*) to a buckling line in the top left region (see *cluster6*). *cluster7* emerges from *cluster5* and has a decrease in the strength of the central buckling line. The remaining two clusters are a little detached from the others. The representative of *cluster8* looks similar to the neighboring *cluster6*, but with a much weaker buckling line in the top left. By watching additional samples, it can be confirmed that the top left line is decreasing in strength along *cluster8*. The last cluster to discuss is *cluster9*. In contrast to all other runs, it has a very strong buckling line in the lower right, at the location of the impact. Also, the plastic strain in the top left is much more distributed, in comparison to other clusters.

**How to Trigger or Avoid a Specific Cluster?** Following the previous step of effect detection, 10 sub-clusters were detected in total. For this example, it was quite hard to give the clusters sensible names, since their differences were more difficult to describe. The cluster hierarchy previously shown is summarized in table 5.1.

Running feature importance ranking on these clusters reveals that the input variables *PX* and *PZ* are the most important ones, causing more than 85% of entropy reduction in the random forest. *PX* and *PZ* describe the positioning of the ball underneath the plate (see fig. 5.1). Therefore, all samples are plotted according to these two variables in figure 5.5. The colors are the respective cluster labels for each sample.

The clusters can be predicted very well from just these two variables since relatively clear zones and borders exist. Sometimes there are samples from another cluster in another cluster region. These can partially be explained as wrongly clustered ones, but sometimes no

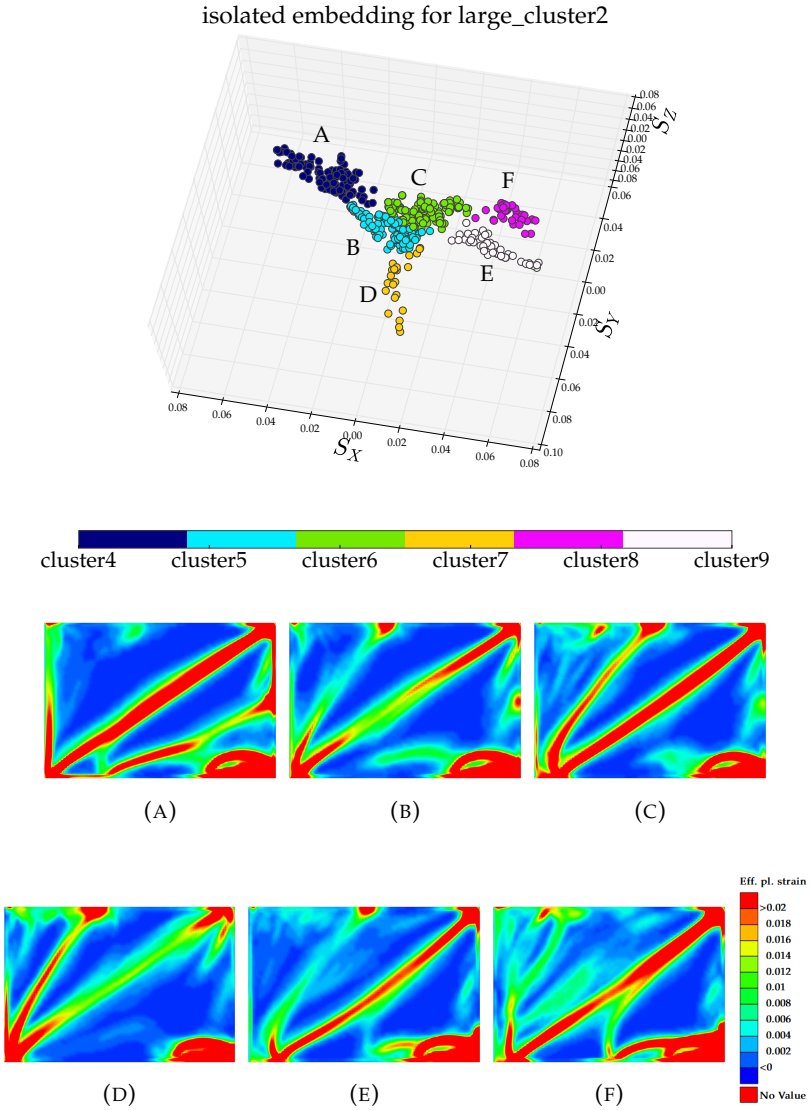



FIGURE 5.4: Computing an individual embedding for *large_cluster2* relieves the MDS-algorithm of a lot of stress, and thus unfolds the twisted structure much better.

| Large Cluster | Sub-Clusters |
|---|---|
| *large_cluster1* | *cluster1* to *cluster3* |
| *large_cluster2* | *cluster4* to *cluster9* |
| *large_cluster3* | *cluster10* |

TABLE 5.1: Summary of the clustering hierarchy for the cantilever plate.

reason can be found. Interestingly, rule mining could be used if desired, but since the regions are easily separable in 2D, it does not make much sense. The rule mining algorithm would create boxes as large as possible for the specified cluster region, and output them as rules.

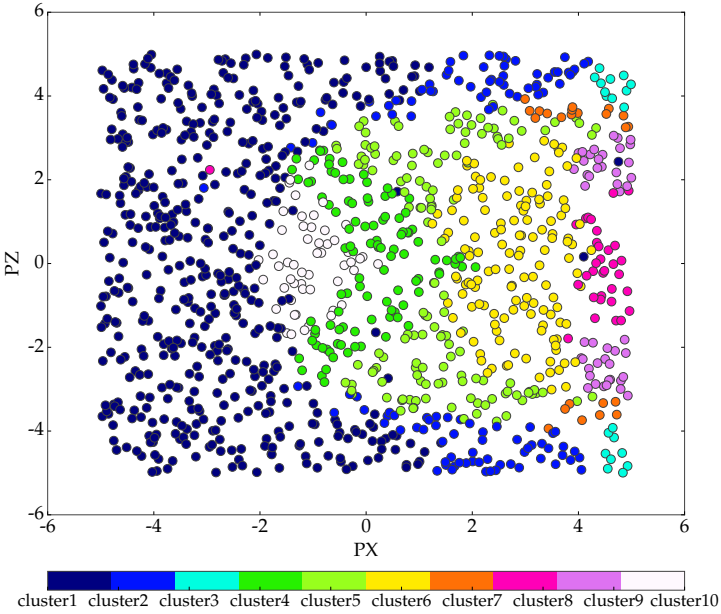



FIGURE 5.5: Plotting every sample regarding their most important input variables (*PX* and *PZ*), and coloring them according to their cluster membership, shows which ball position triggers which buckling mode.

## 5.2 Analysis of a Car Crash

The last example in this thesis is the full-frontal crash of a 2007 Chevrolet Silverado. The FEM-Model was published by the Center for Collision Safety and Analysis (CCSA) at George Mason University [44]. The Silverado is available at the website of the National Highway and Traffic Safety Administration (NHTSA). The main idea of this example is to investigate the deformation behavior of the car, if its sheet thicknesses were to be varied. This is interesting because low thicknesses mean weight savings, and thus a reduction in fuel consumption and CO2 emissions.

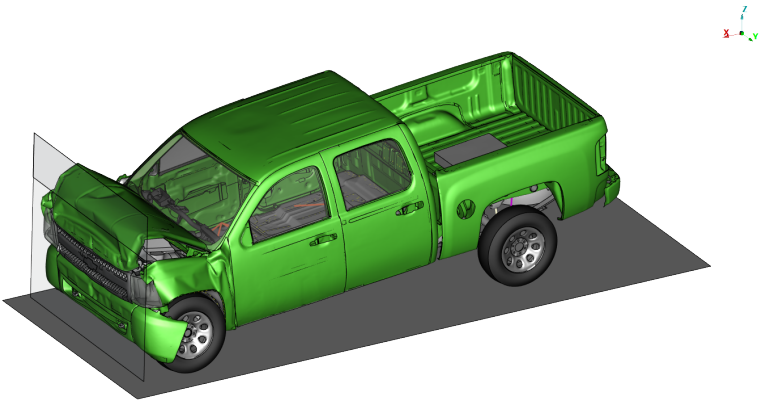


FIGURE 5.6: Simulation of a 2007 Chevrolet Silverados' full frontal crash.

### 5.2.1 Model Description

The simulation model is a car, which crashes with a velocity of $56.32 \frac{km}{h}$ into a rigid barrier. As it has almost 1 million nodes and elements, the model can be considered to be of moderate size.

**Input Variables** The model was not altered topologically in any way, but sheet thicknesses of the crash absorbing structure were chosen as input variables. The selected components are highlighted and labeled in figure 5.7. The nominal value of each thickness was imposed with a variability of $\pm 15\%$. The material of the crash absorbing structure

is steel and was modeled using *MAT_PIECEWISE_LINEAR_PLASTICITY (isotropic hardening), which utilizes a yield curve to describe the elastoplastic behavior.

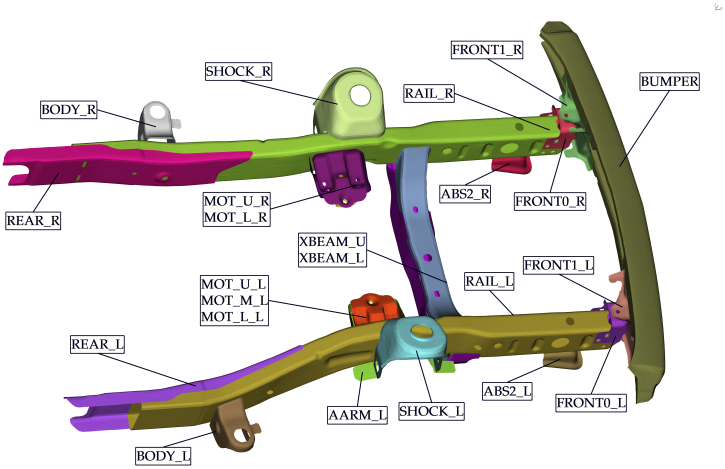



FIGURE 5.7: The sheet thickness of the highlighted components were varied during the investigation. The naming label consists of three parts: the name itself, a location specifier (e.g. U for upper) and then a letter for left or right.

Additionally, the boundary conditions were also imposed with variation. Therefore, the initial velocity of the car and the impact angle onto the barrier were selected. This makes sense for the reason that a structure is usually very susceptible to the way the load enters it. The distance between the car and the barrier was also added to the variable set, but was only varied numerically. In total, the set of variables consists of:

- 24 sheet thicknesses: $\pm 15\%$
- initial velocity (*VELOCITY*): $15645 \frac{mm}{s} \pm 10\%$
- impact angle (*WALL_YAW*): $0 \pm 6\ deg$
- impact distance (*WALL_X*): $\pm 0.001\ mm$

| Variable Name | base thickness [mm] |
|---|---|
| *T_AARM_L* | 2.70 |
| *T_AARM_R* | 2.70 |
| *T_ABS2_L* | 3.50 |
| *T_ABS2_R* | 3.50 |
| *T_BODY_L* | 2.16 |
| *T_BODY_R* | 2.16 |
| *T_BUMPER* | 1.42 |
| *T_FRONT0_L* | 2.56 |
| *T_FRONT0_R* | 2.56 |
| *T_FRONT1_L* | 2.95 |
| *T_FRONT1_R* | 2.95 |
| *T_MOT_LOW_L* | 3.80 |
| *T_MOT_LOW_R* | 4.22 |
| *T_MOT_MID_L* | 3.40 |
| *T_MOT_UP_L* | 3.24 |
| *T_MOT_UP_R* | 3.82 |
| *T_RAIL_L* | 2.55 |
| *T_RAIL_R* | 2.55 |
| *T_REAR_L* | 3.18 |
| *T_REAR_R* | 3.18 |
| *T_SHOCK_L* | 3.24 |
| *T_SHOCK_R* | 3.24 |
| *T_XBEAM_LOW* | 2.46 |
| *T_XBEAM_UP* | 2.16 |

TABLE 5.2: Mean thickness of the crash absorbing sheets shown in figure 5.7.

**Sampling Scheme** This investigation was intended as design space exploration. Therefore, the variables were varied with a LHS design. In total, 1000 simulations were performed.

**Responses and Targets** Since the model has neither a dummy, nor a seat, no occupant criteria are available. Despite this drawback of the model, the intrusion is measured at several points of the frontal wall, which separates the interior and the engine. This measurement reveals whether engine parts violate the cabin space.

Beyond casual responses, it is of major interest to witness the different deformation modes of the crash absorbing structure for two specific reasons. Firstly, it is always interesting to uncover beneficial or undesired types of deformation. Secondly, a response violation, such as a high intrusion, could in the past often be related to inefficient energy absorption beforehand, and thus to deformation classes.

**Partition for Dimensionality Reduction** The dimensionality reduction in this thesis relies on regression of a line through selected components of the crash absorbing structure. The selected structural components are the same components that were used for the input variables (see figure 5.7). These selected components cannot be approximated by a single line, but need a manual decomposition into further groups. In total four groups were created: one for the left rail, one for the right rail, one for the crossbeam and one for the bumper. These groups are shown in figure 5.8. Splitting up the structure into sensible groups and watching each related embedding individually, has proven to yield superior quality for low-dimensional visualization.

## 5.2.2 Results of the Structural Components

The result section discusses the four reduced components individually.

### 5.2.2.1 Analysis of the Bumper

The embedding of the bumper is shown in figure 5.9 and forms three clusters. The three clusters represent buckling on the left, in the middle or on the right of the bumper. The question is, what caused these three deformation types. An importance ranking strongly suggests the impact angle (*WALL_YAW*), with an importance of $70\%$, and the bumper sheet thickness (*T_BUMPER*) with $16\%$.

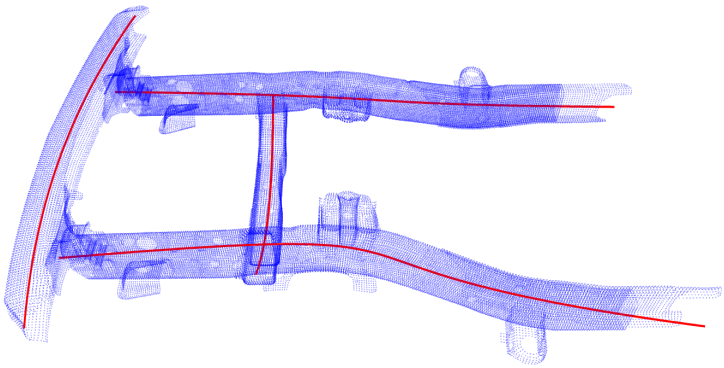


FIGURE 5.8: The crash absorbing structure is divided into four separate groups for dimensionality reduction. The geometric simplification of these four groups are four lines.

Plotting all runs over the two most important variables, and coloring them according to their respective cluster in figure 5.10, reveals that all three buckling modes can be predicted by just using these two variables. A *WALL_YAW* below zero always leads to the cluster *buckling_left*, independent of all other variables. Similarly, in the event of $0 \leq$ *WALL_YAW* $\leq 2.3$, the cluster *buckling_mid* is triggered. For *WALL_YAW* $> 2.3$ the bumper thickness also becomes relevant, making it possible to also trigger *buckling_mid*, if it is high enough. If not, the cluster *buckling_right* is triggered. It makes a lot of sense that these buckling modes are strongly dependent on the *WALL_YAW*. At this point, how relevant these clusters are for the rest of structure is still open for discussion , but since the buckling mode of the bumper controls the distribution of load into the structure, this information may help resolve structural issues.

#### 5.2.2.2 Analysis of the crossbeam

The embedding of the crossbeam is shown in figure 5.11. The structure has a strong elongation, surrounded by a few outliers. The basic deformation behavior is determined by the deformation of the left and the right rail. Depending on how the left or the right rail gets crushed, the crossbeam is either squeezed mainly at the right end (fig. 5.11 middle)

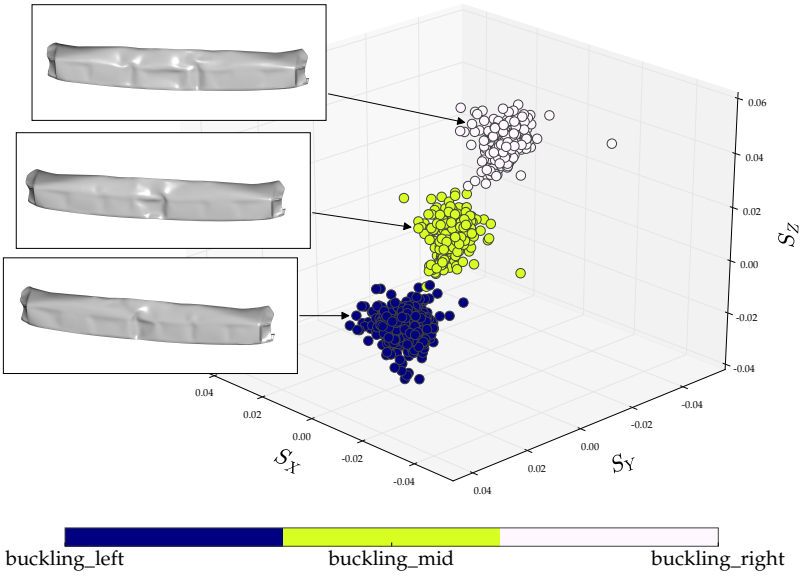



FIGURE 5.9: The 3-dimensional embedding of the bumper has three obvious cluster, which are sparsely connected.

or at both ends at the same time (fig. 5.11 bottom). The representative sample for the outliers (fig. 5.11 top) shows much lower plastic strain at both ends, thus a lower amount of deformation occurs. From a technical perspective it is quite interesting, that one rail almost always puts so much strain on the crossbeam, hinting for a one-sided structural weakness of the structure.

#### 5.2.2.3 Analysis of the left rail

The embedding of the left rail (see fig. 5.12) is the most complex one. The central cluster is *cluster2*, which is directly surrounded by *cluster1* on the left and *cluster3* on the right. Beyond *cluster3* emerges *cluster4*, which is large in size, but also very sparse. It is obvious that *cluster4* is the most interesting one, due to its noisy nature.

**What are the Cluster Differences?** The representative samples for the embedding of the left rail are shown in figure 5.13. Watching the

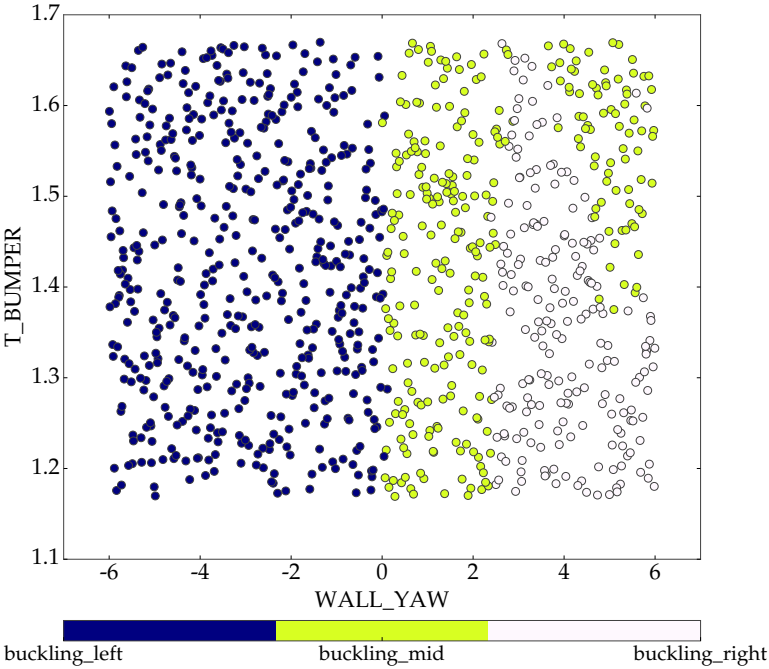



FIGURE 5.10: The buckling modes of the bumper can be predicted from solely the impact angle and its sheet thickness.

representative of *cluster1*, reveals its unique property, that the rail stays quite straight compared to all other clusters. All other clusters buckle more strongly in the middle, which leads to higher bending of the rail. From a technical point of view, it can be considered to be good that the structure remains stable throughout the crash. Therefore *cluster1* is accepted as good. The representatives of *cluster2* and *cluster3* are very similar on first sight. Also in the embedding of fig. 5.12 , *cluster3* and *cluster2* look very similar in terms of their structure. Comparing the representative samples of *cluster2* and *cluster3*, shows that the frontal part of the rail of *cluster3* does not get crushed entirely, in contrast to *cluster2*. This is because the frontal rail of *cluster3* does not stay perfectly straight and bends to the side a little bit. Since this effect between *cluster2* and *cluster3* is still quite small, it is accepted as good here. The most notable difference can be found between *cluster4* and all the other clusters. *cluster4* shows very low deformation in the

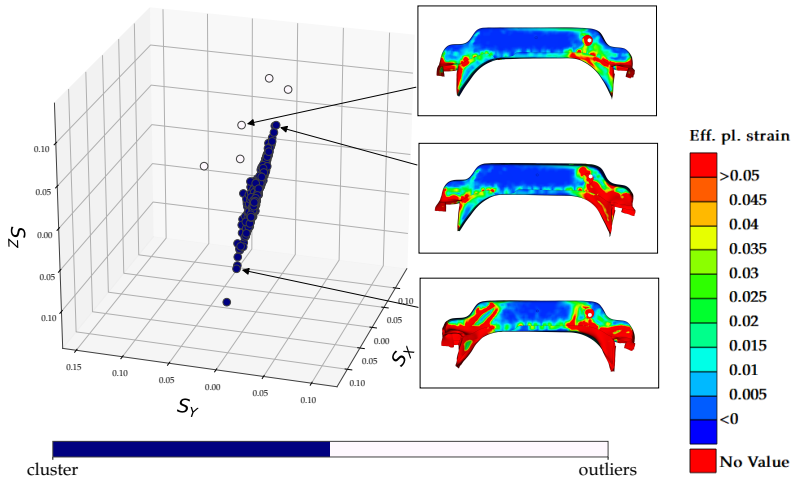



FIGURE 5.11: The embedding of the crossbeam is a single cluster with sparse scatter around it. The representatives are colored according to their respective effective plastic strain with a maximum of 5%.

frontal part of the rail, so that it stays quite straight. Since the major goal of crash simulation is energy absorption through deformation of the rail, the behavior of *cluster4* is highly undesirable, and should be avoided.

**How to prevent *cluster4*?** In the previous section, it was decided that *cluster4* was not acceptable. Therefore, rule mining will be used to prevent this deformation mode. The target will be defined as:

$$T := \neg cluster4. \tag{5.1}$$

The mined antecedents are quite simple:

- $T\_RAIL\_L < 2.65$ (confidence 100%)
- $T\_RAIL\_L > 2.65 \wedge T\_XBEAM\_LOW > 2.30$ (confidence 99.6%)

The first rule makes sense, because a large thickness of the rail also increases the rails stiffness. This could lead to the rail not deforming

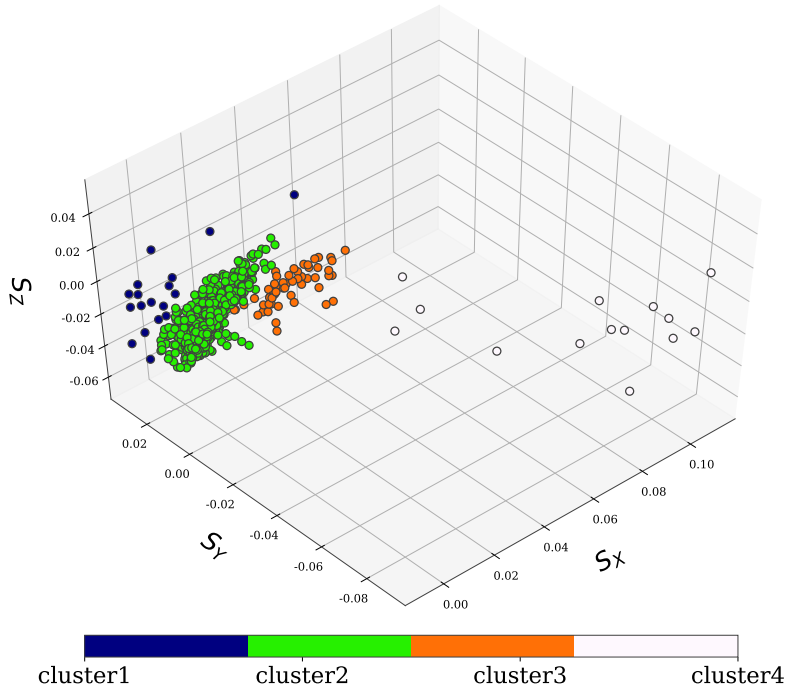
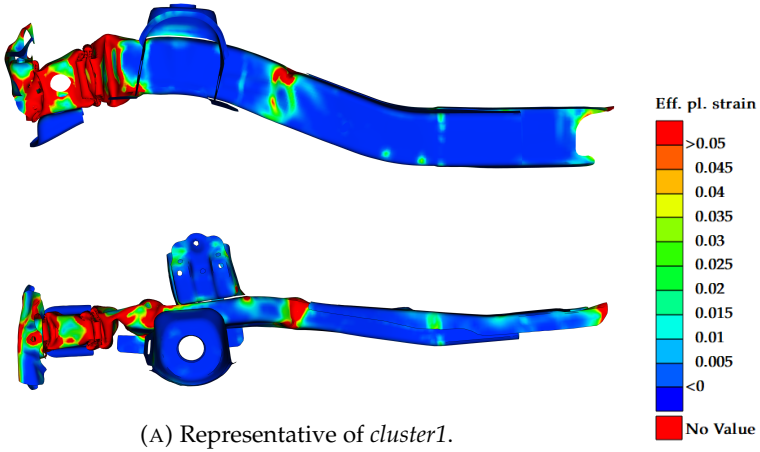


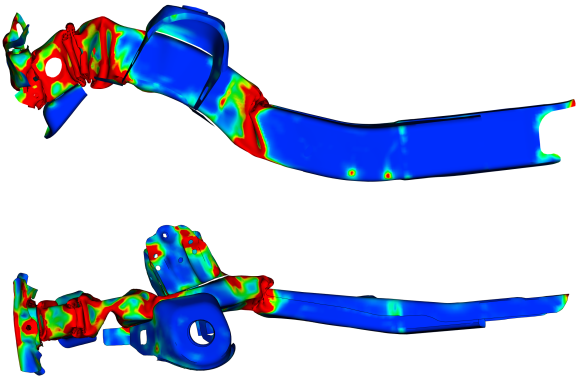

FIGURE 5.12: The embedding of the left rail can be divided into four clusters. The core cluster is *cluster2*, which contains 91% of all samples.

properly anymore, as shown by the representative of *cluster4* in figure 5.13d. The second rule suggests that if the rail thickness is above the recommended threshold, then the thickness of the lower shell of the crossbeam needs to be increased, at least above the specified value. This also makes sense, because the crossbeam holds the rail in place. If the rail is much stiffer, then the crossbeam also has to be stiffer to stabilize the rail properly.

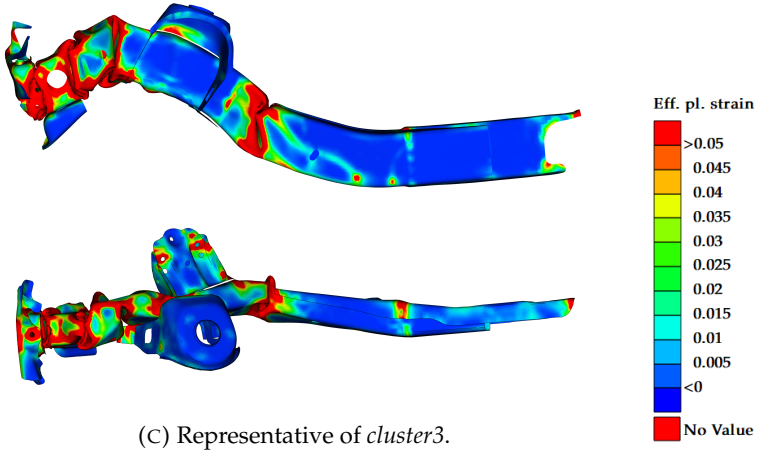
**Connection to Other Clusters** Even though the rule mining shows a reasonable dependency between the left rail and the crossbeam's thickness, no strong connection between the clusters of the crossbeam and the clusters of the left rail was found.



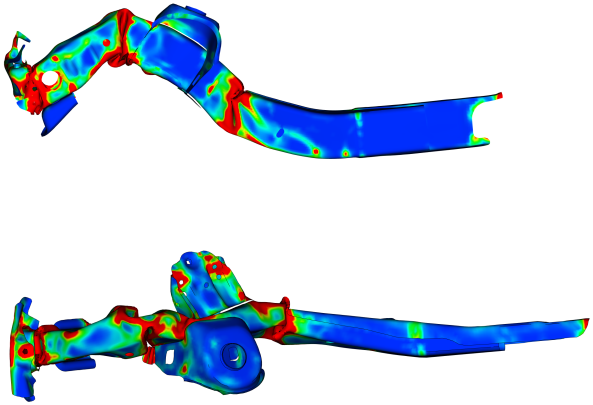

(A) Representative of *cluster1*.



(B) Representative of *cluster2*.




(C) Representative of *cluster3*.



(D) Representative of *cluster4*.

FIGURE 5.13: Representative samples for the clusters, originally shown in figure 5.12. The samples are colored according to equivalent plastic strain with a limit of 5%.

#### 5.2.2.4 Analysis of the Right Rail

The right rail is the final component to be analyzed.

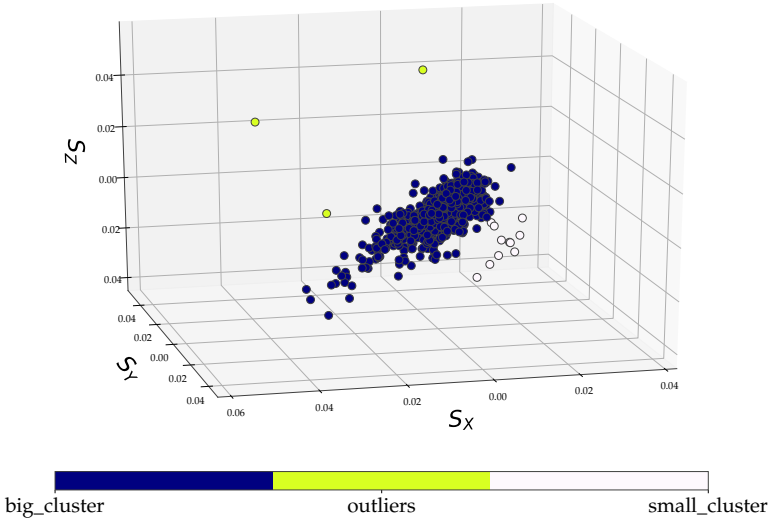
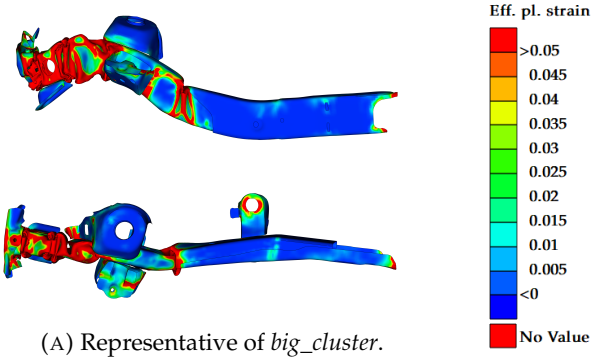


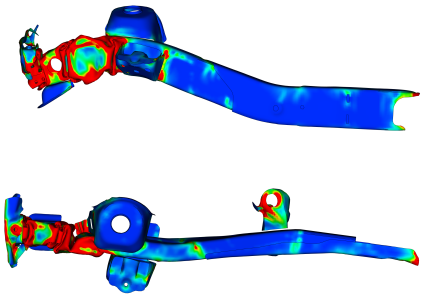

FIGURE 5.14: The embedding of the right rail is mainly a large, noisy looking cluster with a few outliers. On the right side of the cluster is also an additional, but very small cluster.

**What are the Cluster Differences?** The embedding of the right rail is shown in figure 5.14, and is simpler than the embedding of the left rail. The embedding has only one longitudinal *big_cluster* with a lot of noise surrounding it. Three *outliers* can be found beyond the noise. Next to the *big_cluster*, there is also a *small_cluster* with only a few samples.

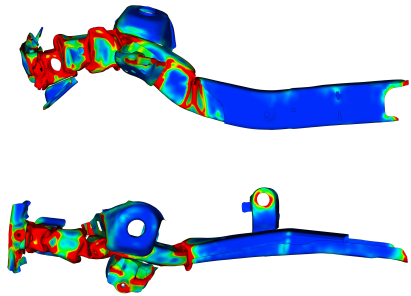
The representatives for the clusters are shown in figure 5.15. The difference between the *big_cluster* and the *small_cluster* is quite obvious. The representative of the *small_cluster* stays much straighter, in comparison to the *big_cluster*. The representative of the *outliers* shows the same kind of deformation as the *big_cluster*. Nonetheless, the frontal part does not fold perfectly, leaving an undeformed region. The visual difference between the three clusters is small, so they are




(A) Representative of *big_cluster*.



(B) Representative of *small_cluster*.



(C) Representative of *outliers*.

FIGURE 5.15: Representative samples belonging to the embedding in figure 5.14.

accepted as good. It would be interesting to understand what actually caused the outliers, but only three samples are not enough for reliable rule mining.

#### 5.2.2.5 Analysis of Responses

The frontal wall separates the engine compartment from the passenger cabin. The intrusion along this component was measured at 45 locations.

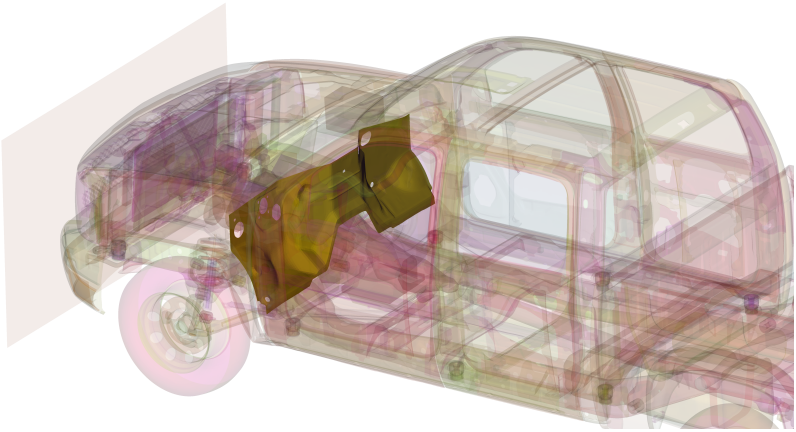


FIGURE 5.16: The intrusion was measured at several locations along the frontal wall, which is highlighted here.

**Evaluation Criterion** Whether the intrusion is acceptable or not, depends on the testing protocol. In this case, the full width frontal impact testing protocol of the European New Car Assessment Programme (Euro NCAP) [49] was chosen, which considers intrusions larger than 100 mm unacceptable. It is important to keep in mind that although this evaluation criterion is based on the Euro NCAP protocol, the virtual testing conditions itself are not, because the impact angle and velocity are also varied.

**What Intrusion Violations are Occurring?** In total, 11 of 45 points of measurement showed an intrusion exceeding 100 mm in at least one

simulation. 5 out of these 11 points have only very few samples violating the condition (below 5). In contrast, the other 6 points show much stronger violations, with at least 34, but up to 112, samples not fulfilling the criterion. As mentioned before, it must be emphasized that the impact velocity was also greatly varied. If the velocity had been equal to, or below, the specification of the testing protocol, only 2 out of 1000 samples would have violated the intrusion. In conclusion, the car seems to perform acceptably, if one does not increase the velocity, as we did.

**What is the Target?** For further analysis, a new class called *invalid_intrusion* is formed. A run is assigned to the class if at least one point of measurement violates the criterion . The target is defined to be:

$$T := \neg \textit{invalid\_intrusion}. \tag{5.2}$$

**What Rules Can be Found?** For the target of eq. 5.2, a rule search is performed with a minimum prediction confidence for the rules of $99\%$. In total, three rules were mined automatically:

$$\textit{VELOCITY} < 16037\ mm/s \rightarrow \neg \textit{invalid\_intrusion}. \tag{5.3}$$

---

$$\begin{aligned} 16037\ mm/s < \textit{VELOCITY} < 16475\ mm/s \\ 2.56\ mm < \textit{T\_FRONT0\_L} \\ 2.45\ mm < \textit{T\_FRONT0\_R} \\ 3.19\ mm < \textit{T\_MOT\_UP\_R} \end{aligned} \rightarrow \neg \textit{invalid\_intrusion}. \tag{5.4}$$

---

$$\begin{aligned} 16475\ mm/s < \textit{VELOCITY} < 16910\ mm/s \\ 2.25\ mm < \textit{T\_FRONT0\_L} \\ 2.71\ mm < \textit{T\_FRONT1\_R} \\ 3.59\ mm < \textit{T\_MOT\_LOW\_R} < 4.83\ mm \\ 2.48\ mm < \textit{T\_RAIL\_R} \\ 2.74\ mm < \textit{T\_REAR\_R} \end{aligned} \rightarrow \neg \textit{invalid\_intrusion}. \tag{5.5}$$

The first rule is eq. 5.3, which states that below the given velocity, no intrusion violation can be expected (with specified tolerance). The second rule (eq. 5.4) is already much more complex, involving three sheet thicknesses. The first two variables relate to the connection between the rail and the bumper, and the last variable is the thickness of the motor suspension. The rule contains 47 samples in total. While it makes sense that the motor has to be kept firmly in place at a higher velocity, no mechanical explanation was found for the other two variables. The last rule (eq. 5.5) is even more complex, utilizing five sheet variables. Interestingly four variables relate to the right rail. Even though the rule seems very specific, it uses 49 samples for its hypothesis. Like the previous rule, it was difficult to understand the mechanical behavior behind the hypothesis. In summary, rule 5.4 and 5.5 could not be understood properly, so that these rules should not be used blindly. In such a case, an additional investigation is recommended to check and analyze the hypothesis in detail, especially because both rules use only $5\%$ of the samples, which is a rather low. This is a very good example showing the difficult border between machine intelligence and human understanding, when recommendations become more complex.

**Relating Clusters With Clusters** An attempt was also made to find strong rules inbetween clusters of different parts, for example, the left and the right rail. However, no strong connection between the clusters could be found in this example case. Occasionally two clusters partially 'correlated', but no strong rule was ever found.

# Chapter 6

# Summary and Conclusion

The purpose of this thesis was to create a process for the analysis of large amounts of crash simulation data. The process flow consists of two key elements: dimensionality reduction and rule mining.

## 6.1 Discussion of Dimensionality Reduction

The dimensionality reduction not only derives a lightweight and mesh-free representation of the crash simulation results, but also enables the computation of a normalized similarity between them. In combination with low-dimensional embedding and clustering, groups of simulations which deform in a very similar way can be found. Finally, the engineer has to decide which deformation clusters are acceptable in terms of simulation behavior, and which are not. Solutions for an unacceptable deformation behavior can be investigated later on by using rule mining. What makes this dimensionality reduction algorithm especially attractive and easy to work with, is its white-box property for providing an understandable, intermediate representation. The resulting embeddings also usually had a good quality, even on a more detailed scale.

### 6.1.1 Geometric Simplification

The geometric simplification approximates a selected group of components to either a line or plane. Since the algorithm is based on regression, it demands that the selected components be physically connected. How much of a downside this is for crash simulation is not very clear, because from a technical point of view, it seems reasonable to select physically connected components. Analyzing subgroups

yielded better embeddings, because one does not mix up too many effects throughout the car. It is also important to keep in mind that approximating a very curvy structure with regression needs a high polynomial degree, which indirectly makes the equation system more likely to diverge, and thus cause the regression to fail. Therefore, a lot of effort had to be put into the regression algorithm. Since regression relies on averaging, it is susceptible to outlying, distant points, such as parts around the rail, which do not define its longitudinal dimension. In consequence, polynomial regression is not recommended for production stage usage and a more capable successor, which has to be less error-prone, is needed. A generalization of the geometric simplification could be a transformation of the simulation model of a car, into a mainly wireframe model.

### 6.1.2 Projection and Smoothing

The projection and smoothing of a result field onto the simplified geometry can compensate quite well for mesh and topology differences, in comparison to other methods. The method is not affected by differences, such as holes or beads, but large morphings of the structure no longer yield valid comparisons, since the simplified geometry would no longer compare the same geometrical locations. This could be taken care of by aligning the simplified geometry, which would also allow the comparison of the deformation behavior of two rails, from two different cars.

## 6.2 Discussion of Rule Mining

The rule mining algorithm, proposed by this thesis, allows the investigation of how to achieve or avoid effects, such as a certain deformation behavior or a high intrusion. One simply needs to split the simulations into two groups: one showing the desired effect and one not. The algorithm then searches the input parameters of these simulations and tries to work out why this effect occurs. The algorithm focuses on simplified regions in the variable space, in which it can safely isolate large fractions of the good simulations, and thus can predict the effect shown by these simulations. It is important to mention that the solution spaces are simplified, so that humans can understand them more

easily. An advantage is not only that the recommendations are understandable by humans, it is also fairly easy to combine multiple target conditions and use discrete and continuous variables at the same time.

### 6.2.1 Insights of Decision Tree Learning

The backbone of the rule mining algorithm is a decision tree classifier, which indirectly brings all its advantages and disadvantages with it. The most notable properties are that DTs are a white-box model, they deal well with low sample counts and also perform automatic variable selection. Especially the first property gives an engineer the opportunity of investigating the thoughts of the algorithm, in contrast to other classifiers, such as artificial neural networks. As has been stated, the DT uses only the most important variables. If two input variables are redundant, then one of them will be omitted. This could be a problem if a recommendation with the second variable was easier to realize. Decision tree learning relies heavily on heuristics, thus the quality of the heuristic learning process has a direct influence on the quality of the recommendations given to the user, by the rule mining engine. At the present time, it is difficult to state the full potential of DT-based rule mining for simulation data, because a lot of the process components can be optimized.

### 6.2.2 Limits, Flaws and Unknown Potential

In this thesis, the largest example in terms of variable count, still had about ten times more simulation samples than variables. The true limits of rule mining in this respect must be investigated and also developed further. In the field of crash simulation, it is quite easy to create a simulation model with one hundred thousand input variables. At the same time, the industry wishes to use as few samples as possible. In addition, this rule mining algorithm suffers from the curse of dimensionality, thus low sample counts in combination with a high variable count.

The reliability optimization of the rules can increase their quality a lot, but one has to keep in mind that it is based on the estimation of a probability density function. The quality of the estimation will worsen with an increasing dimensionality of the rule; thus, it is not suitable for rules with more than four variables.

It would also be interesting to compare this rule mining algorithm with others. While existing algorithms were not deemed to perform

very well, with several modifications they might be able to do so. In consequence, a lot of hidden potential can be expected in this research field, especially because rule mining for simulation data is a very young topic.

# Appendix A

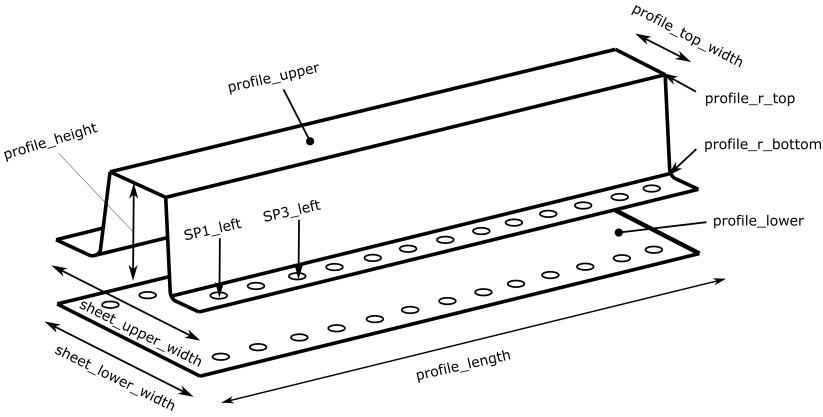
# Description of the Longitudinal Rail

The model consists of a longitudinal rail with a total length of $400mm$. The schematics are shown in figure A.1. The model resembles a drop-tower test, where a mass is dropped onto a specimen to analyze it's deformation behavior. This is frequently done in the domain of automotive engineering to investigate for example different crash boxes. The data for the model originates partially from a real, physical test.

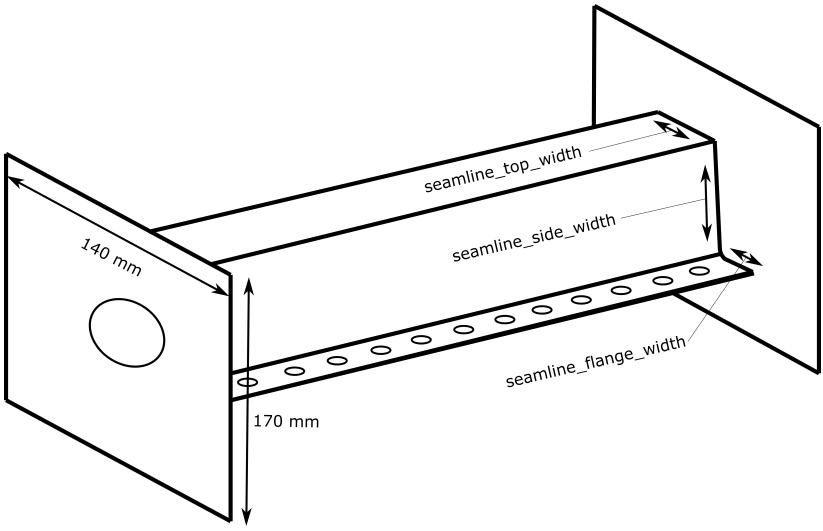
The rail consists of two sheets which are connected with 13 spot-welds along each flange. In the Design of Experiments investigation in this thesis, the design variables of the simulation model were varied with a normal distribution. In the following, the variables shall be enlisted with their distribution parameters.

| variable name | mean value [mm] | std. deviation |
|---|---|---|
| profile_top_width | 64.2 | 2. |
| profile_bot_width | 73.3 | 2. |
| profile_upper_width | 116.9 | 1. |
| profile_lower_width | 116.9 | 1. |
| profile_height | 90.3 | 1.5 |
| profile_length | 400. | 2. |
| profile_t | 2.5 | 0.1 |
| profile_r_bottom | 4. | 0.5 |
| profile_rTop | 6. | 0.5 |

TABLE A.1: Geometric variables of the simulation mode in figure A.1.




(A) Schematics of the two profile sheets.




(B) Schematics of the component with covering plates, which are connected by seamlines.

FIGURE A.1: Schematics of the longitudinal rail.

The material of the rail sheets was an AA6016-T6 aluminium. Its mechanical properties are defined in by the LS-Dyna keyword *MAT_-PIECEWISE_LINEAR_PLASTICITY also known as MAT24. The elastoplasticity is defined by a yield curve, thus an experimental curve of effective stress over effective plastic strain. If the minimum effective stress (yield criterion) is reached, elastoplastic deformations occur. The plastic behavior uses isotropic hardening. The yield criterion can in general be understood as an offset value for the entire elastoplastic curve. This offset value was used as a design variable for the lower, planar sheet in order achieve a different stiffness relation and thus trigger possibly different deformation modes. The U-Profile is created by using cold forming. There are four forming edges (two lower and to upper in figure A.1). These forming edges are represented geometrically by radii and have an own material property to account for the effects of forming, therefore the offset value is higher than the base value of the rest of the material.

| variable name | mean value [GPa] | std. deviation |
|---|---|---|
| profile_lower_matOffset | 80 | 10 |
| profile_radius_matOffset | 90 | 8 |

TABLE A.2: The material variables are the yield criterion for the upper U-profile.

The impactor is a circular disk, which drops onto the specimen with a predefined velocity. The angle of impact was varied in such a way, that the inclination is either more to the front or the rear. In consequence the plate hits the respective part of the model first. In addition, also the mass was varied along with the impact velocity. The uncertainty of the velocity was computed from the uncertainty of the height from which the plate is dropped. The variation of the mass was originally intended to consider slightly different impactors, though usually for one investigation one impactor is used for all specimens. Since the goal was to trigger different deformation modes, this was not seen as a severe issue for the reason that the variation in mass simply varies the kinetic energy a little bit more.

The lower and upper profile sheets are covered by two plates at both ends by using welding. These seam lines are also taken into account as a source of scatter, especially since such specimens are usually crafted by hand and thus have a high uncertainty in terms of their exact length.

| variable name | mean value | std. deviation |
|---|---|---|
| crushPlate_angle | 0° | 3 |
| crushPlate_mass | 165kg | 1 |
| crushPlate_initialVy | -8056mm/s | 0.06 |

TABLE A.3: For the impactor, the impact angle, the mass and the impact velocity were varied.

| variable name | mean value [mm] | std. deviation |
|---|---|---|
| seamline_top_width | 40 | 4 |
| crushPlate_mass | 50 | 4 |
| crushPlate_initialVy | 40 | 4 |

TABLE A.4: Variational parameters of the welding seams, which connect the two ending plates with the profile.

A majority of the design variables is caused by the weld spots connecting the two profile sheets, since every weld spot was modeled independently from all others. The naming has the following scheme: SP[+ and - to distinguish left and right][number]_[variable name], e.g. SP+1_radius. The variation includes not only the position, but also the stiffness, the size and the failure stresses of shear (rs) and tension (rn). Below shall only parameters be enlisted, which are different between the spotwelds (such as position).

| variable name | mean value | std. deviation |
|---|---|---|
| SP*_radius | 8 | 0.08 |
| SP*_stiff | 40 | 0.4 |
| SP*_rn | 7.46 | 0.746 |
| SP*_rs | 20. | 2. |
| SP+1_x | 48.45 | 1. |
| SP+1_y | 20. | 1. |
| SP-1_x | -48.45 | 1. |
| SP+2_y | 50. | 1. |
| SP+3_y | 80. | 1. |
| SP+4_y | 110. | 1. |
| SP+5_y | 140. | 1. |
| SP+6_y | 170. | 1. |
| SP+7_y | 200. | 1. |
| SP+8_y | 230. | 1. |
| SP+9_y | 260. | 1. |
| SP+10_y | 290. | 1. |
| SP+11_y | 320. | 1. |
| SP+12_y | 350. | 1. |
| SP+13_y | 380. | 1. |

TABLE A.5: Parameters of the spot welds connecting the two profile sheets. Only the differing variables are enlisted here (* is a placeholder adressing any variable with that pattern in the name). In total there are 6 parameters per spotweld, which are districuted along 2 flanges with 13 spotwelds. This results in 156 parameters for the spotwelds only.

# Bibliography


[1] S. Ackermann et al. *Principal component analysis for detection of globally important input parameters in nonlinear finite element analysis*. Institute of Applied and Experimental Mechanics Stuttgart. 2008.

[2] R. Agrawal et al. *Fast discovery of association rules*. Menlo Park, California: AAAI Press, 1996, pp. 307–328.

[3] J. A. Aslam, R. A. Popa, and R. L. Rivest. "On Estimating the Size and Confidence of a Statistical Audit". In: *Proceedings of the Electronic Voting Technology Workshop (EVT 07)*. 2007.

[4] G. H. Bakir, J. Weston, and B. Schölkopf. "Learning to find pre-images". In: *NIPS'03 Proceedings of the 16th International Conference on Neural Information Processing Systems*. 2003, pp. 449–456.

[5] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, 1995.

[6] B. Bohn et al. *Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods*. International Conference on Computational Science ICCS. 2013.

[7] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Application*. Springer Series in Statistics, 1997.

[8] B. Boser, I. M. Guyon, and V. N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the fifth annual workshop on Computational learning theory*. Proceedings of the fifth annual workshop on Computational learning theory - COLT (1992), p. 144. DOI: 10.1145/130385.130401.

[9] L. Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (1996), pp. 123–140.

[10] L. Breiman et al. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[11] W. W. Cohen. "Fast Effective Rule Induction". In: *Proceedings of the Twelfth International Conference on Machine Learning*. 1995.

[12] A. Criminisi, J. Shotton, and E. Konukoglu. "Decision Forests: A Unified Framework for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning". In: *Foundations and Trends in Computer Graphics and Vision* 7 (2011), pp. 81–227.

[13] C. Diez, L. Harzheim, and A. Schumacher. *Effiziente Wissensgenerierung zur Robustheitsuntersuchung von Fahrzeugstrukturen mittels Modellreduktion und Ähnlichkeitsanalyse*. Vol. 2279. VDI Reports. VDI-Platz 1, 40468 Düsseldorf: VDI Verlag GmbH, 2016.

[14] C. Diez et al. "Automated Generation of Robustness Knowledge for selected Crash Structures". In: *14th LS-DYNA Forum in Bamberg, Germany*. 2016.

[15] C. Diez et al. "Big-Data based rule-finding for analysis of crash simulations". In: *WCSMO12*. 2017.

[16] E. W. Dijkstra. "A Note on Two Problems in Connexion with Graphs". In: *Numerische Mathematlk* 1 (1959), pp. 269–271.

[17] A. Efrat, Q. Fan, and S. Venkatasubramanian. "Curve Matching, Time Warping, and Light Fields: New Algorithms for Computing Similarity between Curves". In: *Journal of Mathematical Imaging and Vision* 27.3 (2007), pp. 203–216.

[18] H. A. Eschenauer, Kobelev V. V., and A. Schumacher. "Bubble method for topology and shape optimization of structures". In: *Journal of Structural Optimization* 8 (1994), pp. 42–51.

[19] ESI US R&D. *Mineset*. http://cloud.esi-group.com/analytics. 2017.

[20] C. Feuersänger and M. Griebel. "Principal manifold learning by sparse grids". In: *Computing* 85.4 (2009), pp. 267–299.

[21] T. A. Foley and G. M. Nielson. "Knot Selection for Parametric Spline Interpolation". In: *Mathematical Methods in Computer Aided Geometric Design* (1989), pp. 261–271.

[22] Y. Freund and R. E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of Computer and System Sciences* 55.1 (1997), pp. 119–139. DOI: 10.1006/jcss.1997.1504. URL: https://doi.org/10.1006/jcss.1997.1504.

[23] J. Fürnkranz, D. Gamberger, and N. Lavraĉ. *Foundations of Rule Learning*. Springer, 2012.

[24] Dynardo GmbH. *SoS – Statistics on Structures*. https://www.dynardo.de/software/statistics-on-structures.html. 2017.

[25] Sidact GmbH. *DIFFCRASH*. http://www.sidact.com/. 2017.

[26] C. Gouette et al. "On Clustering fMRI Time Series". In: *Neuroimage* 9.3 (1999), 298–310. DOI: 10.1006/nimg.1998.0391.

[27] L. Gräning and B. Sendhoff. "Shape mining: A holistic data mining approach for engineering design". In: *Advanced Engineering Informatics* (2014).

[28] J. O. Hallquist. *LS-DYNA Theory Manual*. Livermore Sofrware Technology Corporation. 2006.

[29] J. O. Hallquist. *LS-DYNA Theory Manual*. Livermore Software Technology Corporation. 7374 Las Positas Road, Livermore, CA 94551, 2006.

[30] W. K. Hastings. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". In: *Biometrika* 57.1 (1970), pp. 97–109.

[31] H. Hotelling. "Analysis of a complex of statistical variables into principal components". In: *Journal of Educational Psychology* 24 (1933), pp. 417–441.

[32] R. L. Hyafil L. Rivest. "Constructing Optimal Binary Decision Trees is NP-complete". In: *Information Processing Letters* 5.1 (1976), pp. 15–17. DOI: 10.1016/0020-0190(76)90095-8.

[33] R Iza-Teran and J. Garcke. *Operator Based Multi-Scale Analysis of Simulation Bundles*. INS Preprint 1524. Fraunhofer SCAI, 2015.

[34] P. Jaccard. "Etude de la distribution florale dans une portion des Alpes et du Jura". In: *Bulletin de la Societe Vaudoise des Sciences Naturelles* 37.142 (1901), pp. 547–579.

[35] S. C. Johnson. "Hierarchical clustering schemes". In: *Psychometrika* 32 (1967), pp. 241–254.

[36] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2017. URL: http://www.scipy.org/.

[37] M. Kanevski, V. Timonin, and A. Pozdnukhov. *Machine Learning for Spatial Environmental Data: Theory, Applications, and Software*. New York EPFL Press, 2009.

[38] I. Katsov. *Introduction to Algorithmic Marketing: Artificial Intelligence for Marketing Operations*. Ilya Katsov, 2017.

[39] D. G. Krige. "A statistical approach to some basic mine valuation problems on the Witwatersrand". In: *Journal of the Chemical, Metal and Mining Society of South Africa* 52.6 (1951), pp. 119–139.

[40] S. Y. Kung. *Kernel Methods and Machine Learning*. Cambridge University Press, 2014.

[41] P. Kunze. "Extraktion relevanter Regeln als Verbesserungsvorschläge für Ingenieurinnen und Ingenieure mithilfe von Entscheidungsbäumen zur Robustheitsanalyse von Crashsimulationen". Bachelor Thesis. Rüsselsheim am Main, Germany: RheinMain University of Applied Sciences, Apr. 2017.

[42] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9 (2008), pp. 2579–2605.

[43] J. B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations". In: *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*. 1967.

[44] D. Marzougui et al. "Extended Validation of the Finite Element Model for the 2007 Chevrolet Silverado Pick-Up Truck (MASH 2270kg Vehicle)". In: *Transportation Research Board 92nd Annual Meeting*. 2013.

[45] S. E. Maxwell, H. D. Delaney, and K. Kelley. *Designing Experiments and Analyzing Data: A Model Comparison Perspective, Second Edition*. Routledge, 2003.

[46] W. S. McCulloch and W. Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[47] L. Mei and C.-A. Thole. "Data analysis for parallel car-crash simulation results and model optimization". In: *Sim. Modelling Practice and Theory* 16.3 (2008), pp. 329–337.

[48] N. J. Mitra. "Algorithms for Comparing and Analyzing 3D Geometry". PhD thesis. Stanford University, 2006.

[49] Euro NCAP. *Full width frontal impact testing protocol*. Dec. 2016. URL: https://www.euroncap.com/.

[50] G. M. Nielson and T. A. Foley. "A Survey of Applications of an Affine Invariant Norm". In: *Mathematical Methods in Computer Aided Geometric Design* (1989), pp. 445–467.

[51] T. A. Pastva. “Bézier Curve Fitting”. Master Thesis. Naval Postgraduate School Monterey, California, 1998.

[52] K. Pearson. “Notes on regression and inheritance in the case of two parents”. In: *Proceedings of the Royal Society of London* 58 (1895), pp. 240–242.

[53] K. Pearson. “On lines and planes of closest fit to systems of points in space”. In: *Philosophical Magazine* 2 (1901), pp. 559–572.

[54] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[55] G. Piatetsky-Shapiro. “Discovery, Analysis, and Presentation of Strong Rules”. In: *Knowledge Discovery in Databases - KDD* (1991), pp. 229–248.

[56] M. L. de Prado. *Advances in Financial Machine Learning*. John Wiley & Sons Inc, 2018.

[57] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[58] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: MIT Press, 2006. URL: http://www.gaussianprocess.org/gpml/chapters/RW.pdf.

[59] M. Reuter, F. Wolter, and N. Peinecke. “Laplace-Beltrami spectra as Shape-DNA of surfaces and solids”. In: *Computer-Aided Design* 38.4 (2006), pp. 342–366.

[60] M. Reuter et al. “Discrete Laplace-Beltrami operators for shape analysis and segmentation”. In: *Computers & Graphics* 33.3 (2009), pp. 381–390.

[61] D. W. Scott. “On optimal and data-based histograms”. In: *Biometrika* 66.3 (1979), pp. 605–610.

[62] C. E. Shannon. “A Mathematical Theory of Communication”. In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[63] R. R. Sokal and C. D. Michener. “A statistical method for evaluating systematic relationships”. In: *University of Kansas Science Bulletin* 38 (1958), pp. 1409–1438.

[64] H. Steinhaus. “Sur la division des corps matériels en parties”. In: *Bulletin de l’academié polonaise des sciences* 12 (1956), pp. 801–804.

[65] B. Tang. “Orthogonal Array-Based Latin Hypercubes”. In: *Journal of the American Statistical Association* 88.242 (1993), pp. 1392–1397. DOI: 10.2307/2291282.

[66] C.-A. Thole and H. Mierendorff. “Verfahren zur Ermittlung gemeinsamer Ursachen von Streuungen von Simulations- und / oder Messergebnissen”. Deutsch. Patent WO2011069580 A2. WO Patent App. PCT/EP2010/006,510. June 2011. URL: https://google.com/patents/WO2011069580A2?cl=de&hl=de.

[67] C.-A. Thole et al. *Advanced Mode Analysis for Crash Simulation Results*. Proceedings 9th LS-DYNA Forum Bamberg. 2010.

[68] S. Wessing. “Two-stage Methods for Multimodal Optimization”. PhD Thesis. University of Dortmund, July 2015.

[69] S. Wolff. “Simulation of random fields in structural design”. In: *11th International Probabilistic Workshop in Brno, Czech Republic*. 2013.

[70] P. Zhang, H. Qiao, and B. Zhang. “An improved local tangent space alignment method for manifold learning”. In: *Pattern Recognition Letters* 32.2 (2011), pp. 181–189.

[71] J. Zhu et al. “Multi-class AdaBoost”. In: *Statistics and Its Interface* 2 (2009), pp. 349–360.