Grunert and Fehr *Adv. Model. and Simul. in Eng. Sci.(2016)3:20*
DOI 10.1186/s40323-016-0072-x

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Identification of nonlinear behavior with clustering techniques in car crash simulations for better model reduction

Dennis Grunert* and Jörg Fehr

*Correspondence: dennis.grunert@itm.uni-stuttgart.de
Institute of Engineering and Computational Mechanics, University of Stuttgart, Pfaffenwaldring 9, 70569 Stuttgart, Germany

Abstract

Background: Car crash simulations need a lot of computation time. Model reduction can be applied in order to gain time-savings. Due to the highly nonlinear nature of a crash, an automatic separation in parts behaving linearly and nonlinearly is valuable for the subsequent model reduction.
Methods: We analyze existing preprocessing and clustering methods like k-means and spectral clustering for their suitability in identifying nonlinear behavior. Based on these results, we improve existing and develop new algorithms which are especially suited for crash simulations. They are objectively rated with measures and compared with engineering experience. In future work, this analysis can be used to choose appropriate model reduction techniques for specific parts of a car. A crossmember of a 2001 Ford Taurus finite element model serves as an industrial-sized example.
Results: Since a non-intrusive black box approach is assumed, only heuristic approaches are possible. We show that our methods are superior in terms of simplicity, quality and speed. They also free the user from arbitrarily setting parameters in clustering algorithms.
Conclusion: Though we improved existing methods by an order of magnitude, preparing them for the use with a full car model, they still remain heuristic approaches that need to supervised by experienced engineers.

Keywords: Crash simulation, Nonlinear behavior, Black box identification, Spectral clustering, K-means, Model reduction

Background

In the modern design process of cars, crash tests are simulated with highly detailed finite element (FE) models on high-performance computing clusters. Spethmann et al. summarize in [1] that in 1998 a simulation was much more cost and time efficient than building a prototype, i.e., 5000 USD and weeks vs. 300.000 USD and half a year. There is reason to believe that this great difference still exists today but with an increased quality of the simulations. On the other hand, prototypes change rapidly nowadays. A major German car manufacture has a database with around 7000 different prototypes for only one car type, which again increases the total computation time. Each of them needs to be checked for crash safety in an early development stage since major changes are impossible later

Springer Open

on. On top of that, the design engineers need fast feedback about whether the current prototype is safe before they can continue with further modifications.

This raises the need for model (order) reduction (MOR) in the simulation process. Model reduction simplifies the underlying mathematical model in such a way that the dimension, and thus hopefully the simulation time, is reduced while introducing only an acceptable error. There exist a vast collection of model reduction methods, which can be separated into two groups: linear and nonlinear methods. They differ on whether the underlying differential equation is required to be linear or whether nonlinear terms are allowed. One would expect that only nonlinear reduction techniques should be applied to a car crash due to its highly nonlinear nature. But we will propose ideas to automatically separate the car in one part with presumably linear and another part with presumably nonlinear behavior. It is now possible to apply linear MOR to the first and nonlinear MOR to the second part respectively, cf. [2]. This way, linear methods, which usually have a better ratio between computation time-saving and introduced error, can be used for most parts of the vehicle even though the crash in its entirety has nonlinear behavior. In addition, linear MOR techniques are maturer in comparison to nonlinear MOR. Radermacher and Reese successfully used a similar approach by only reducing parts with elastic material behavior [3].

In this work, we analyze modern clustering techniques for their suitability in model reduction. The clustering is not used as a way of model reduction but as a preprocessing step to identify nonlinear behavior. As described above, the knowledge of this behavior will then be used to combine linear and nonlinear model reduction methods in future work that is not part of this article.

It can be seen that certain shortcomings exist with current methods to identify nonlinear behavior in car crashes. Therefore, the major contribution of this work is an improved clustering algorithm which addresses and solves these shortcomings. This is achieved by either improving existing methods or developing new ones. Considering that the majority of the simulations in automotive development concerns crash safety [1], we will focus on car crash tests in this article.

In the next two sections, we introduce the industrial setting and provide an overview of existing methods to analyze crash behavior. The clustering methods needed in the following are discussed in detail in the “Clustering” section. After this preparation, one modern technique to separate linear from nonlinear behavior is presented in the “Preprocessing” section together with measures which are used to assess the quality of the method in the “Analysis of the preprocessing” section. The problems found in this analysis are solved in the “Improvements” section. Finally, we conclude with a summary and outlook on further research.

Setting

For the upcoming analysis, we are using the proprietary simulation software LS-DYNA [4] by the Livermore Software Technology Corporation since it is widely used by the industry for crash analysis. Additionally, it would be too costly to develop an alternative software in a research facility. As described in [5], LS-DYNA uses the following simulation procedure: After the model is read from an ASCII file, the equilibrium equations in Lagrangian formulation resulting from continuum mechanics are transformed in their corresponding weak form. The spatial dimensions are discretized via finite elements resulting in the nonlinear equation of motions

$$\boldsymbol{M} \cdot \ddot{\boldsymbol{q}}(t) + \boldsymbol{D} \cdot \dot{\boldsymbol{q}}(t) + \boldsymbol{f}_{\text{int}}(\boldsymbol{q}(t), \ldots) = \boldsymbol{f}_{\text{ext}}(t) \tag{1}$$

with the symmetric mass matrix $\boldsymbol{M}$, damping matrix $\boldsymbol{D}$, internal forces $\boldsymbol{f}_{\text{int}}$ depending, e.g., nonlinearly on the generalized coordinates $\boldsymbol{q}$ and the external forces $\boldsymbol{f}_{\text{ext}}$. If the internal forces $\boldsymbol{f}_{\text{int}}(\boldsymbol{q}(t), \ldots)$ can be written as $\boldsymbol{K} \cdot \boldsymbol{q}(t)$ with a stiffness matrix $\boldsymbol{K}$, then the system is linear. The discretization of the time is accomplished with explicit central differences or other discretization schemes, which are then solved by the built-in LS-DYNA solver. It needs to be mentioned that our work does not rely on any specific features of LS-DYNA. Thus, any other simulation tool can be used instead.

The finite element model of a 2001 Ford Taurus from the National Crash Analysis Center [6] serves as an example. The model depicted in Fig. 1 consists of almost one million (mostly shell) elements and was validated against actual hardware crash tests [7]. For simplicity, we will focus our analysis of nonlinear behavior on the crossmember shown in Fig. 2. It consists only of 3789 nodes and experiences large deformations as part of the crumple zone which makes it a suitable candidate.

Overview of crash analysis

Model reduction in mechanics would usually substitute the equation of motion (1) by a differential equation of smaller dimension, the so-called reduced system. Unfortunately, this is not possible with the closed-source software LS-DYNA. The same is true for ana-

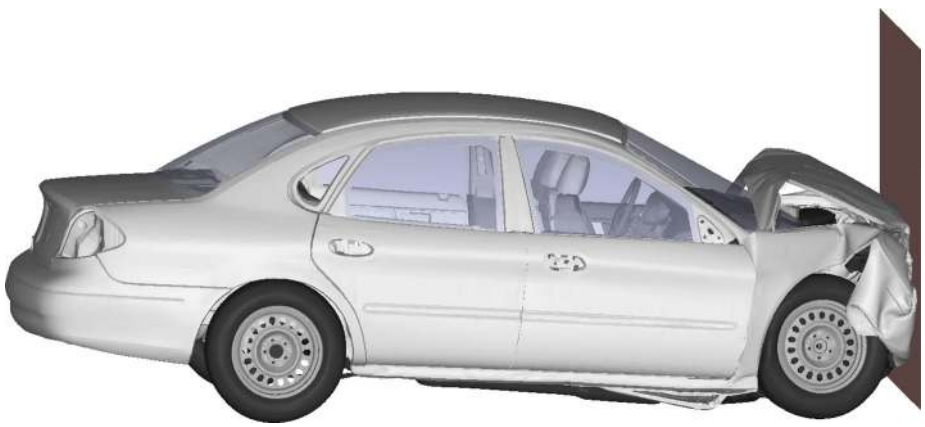

Fig. 1 Frontal crash of a 2001 Ford Taurus. The car has an initial velocity of 35 mph perpendicular to a rigid wall

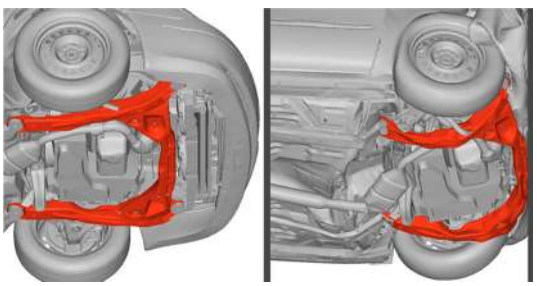

Fig. 2 The crossmember is depicted in *red* before (*left*) and after the crash (*right*)

lyzing nonlinear behavior since there is no way to access the function $\boldsymbol{f}_{\text{int}}$. Instead we can only use the simulation data resulting from $\boldsymbol{q}(t)$ for different simulation runs, which is basically a black box approach. The import routine for the binary output of LS-DYNA to MATLAB was written at the institute.

The overall goal consists of finding parts of the vehicle that most likely behave linearly or mostly nonlinearly during the crash. This leads to a separation of the generalized coordinates

$$\boldsymbol{q} = \begin{pmatrix} \boldsymbol{q}^{\text{l}} \\ \boldsymbol{q}^{\text{n}} \end{pmatrix}$$

into parts $\boldsymbol{q}^{\text{l}}$ and $\boldsymbol{q}^{\text{n}}$ corresponding to the linear and nonlinear behavior of the vehicle, respectively. The car model can now be cut along the boundaries between the parts. Several so-called substructuring methods or component mode synthesis (CMS) can be found in [8]. This allows an independent reduction of each component, which is useful for fast interchanging of specific components in the design phase of a car without the need to reapply the model reduction to the complete car. Proprietary crash simulation codes like LS-DYNA usually provide an interface to replace single components by their linearly reduced counterparts, the so-called superelements. The nonlinear reduction of one component inside LS-DYNA is still an open question but may be achieved with co-simulations via user-defined functions. For the sake of this paper, it can be assumed that the linearly behaving part will be reduced with linear MOR and the nonlinearly behaving part not reduced at all. In [2], several methods of separation, reduction techniques and the need for an additional interface reduction are described exemplarily on a model of a go-kart. The classification of linear behavior in the go-kart was performed manually by an engineer. The intention of this paper is to automate this decision as much as possible. A subsequent model reduction as described in [2] is not part of this article but the subject to current and future work.

Several ways to analyze the crash behavior were published for different needs: Running the same simulation twice can result in different outputs due to round-off errors in parallel computing. In 2005 and 2008, Mei and Thole published an algorithm to detect areas with this scatter [9,10], which was implemented in the software Diff-Crash. Since the complete algorithm is not accessible to the authors, it will not be considered in this article. The Simdata-NL project [11,12], on the other hand, used similar techniques to detect bifurcations and to group similarly behaving nodes. None of them were directly developed to separate a model for subsequent model reduction but they can be accommodated to our needs. Only nodal positions were used in the aforementioned publications. Thus, we will also restrict ourselves to this assumption in order to allow a fair comparison to our improvements.

Clustering

All aforementioned publications use either self-developed or known clustering methods such as k-means or spectral clustering. Therefore, it is vital to explain what clustering is. Clustering is the process of grouping a large data set by similarity into so-called clusters. It is a subclass of unsupervised learning since there do not exist any predefined categories. The area of application is huge: genome analysis, image segmentation, social networks and consumer analysis, to name only a few.

Despite its broad use, clustering is no Swiss army knife for data classification. In fact, [13] describes many pitfalls: First, there is no exact definition of what a cluster should look like. This lies in the eye of the beholder. Imagine for example all the books in a library as data set. One could group them by genre, by age, by the second letter of the authors family name, by their position in the shelf, by their physical dimensions, etc. Another example is the famous ambiguous optical illusion "My Wife and My Mother-in-Law" by William Ely Hill: Some see a young woman, and others see an old lady. The same is true for clustering algorithms that try to identify structure in data. Each algorithm has "its own view" on the data which does not necessarily lead to the same result as the user of the algorithm expects. Additionally, most of the real world data does not constitute of natural clusters, which need to be identified by a clustering algorithm. Instead there are many possible ways to categorize data and all of them have their own right to exist. Thus, clusters are not identified but created by the algorithms. Even in randomly distributed data, clustering algorithms will return a grouping since they enforce structure on the data instead of recognizing natural clusters or deciding if there are any clusters at all. This can also be a pitfall for experienced users. There are not only a large quantity of algorithms to choose from but they usually have several parameters which need to be defined and only have a heuristic meaning. Even the number of clusters needs to be specified most of the time in advance. In fact, there cannot be one algorithm satisfying three simple properties as shown in [14].

For consistent notation in the further description of the algorithms, we assume that n d-dimensional data points $\mathcal{X} = \{\boldsymbol{x}_i\}_{i=1}^n \subseteq \mathbb{R}^d$ are given. A clustering algorithm is supposed to group these points into K disjoint clusters $\mathcal{C} = \{C_k\}_{k=1}^K$. $\mathcal{C}$ is—mathematically speaking—a partition of $\mathcal{X}$, i.e., $C_k \neq \emptyset$ for all k, $\bigcup_k C_k = \mathcal{X}$ (particularly $C_k \subseteq \mathcal{X}$ for all k) and $C_k \cap C_{k'} = \emptyset$ for all $1 \leq k < k' \leq K$.

K-means clustering

K-means is indisputably one of the most used clustering algorithms dating back to 1955. It iteratively defines the clusters as areas around the cluster centers

$$\boldsymbol{z}_k = \frac{1}{|C_k|} \sum_{\boldsymbol{x} \in C_k} \boldsymbol{x} \tag{2}$$

of the last iteration, c.f. Algorithm 1 taken from [15]. This way it converges to a local minimum of the overall squared error, which is defined as

$$\min_{\mathcal{C}} \sum_{k=1}^{K} \sum_{\boldsymbol{x} \in C_k} \|\boldsymbol{x} - \boldsymbol{z}_k\|^2, \tag{3}$$

i.e., the clusters tend to be as dense as possible. But due to the assignment in line 4 of Algorithm 1, the clusters are always lying in convex subsets of $\mathbb{R}^d$, which restricts the ability of k-means to identify only these types of clusters. Not only does the user have to set the number K of desired clusters in advance but the random initial assignment in line 1 also makes the algorithm indeterministic. In order to implement the algorithm, one has to specify the stop criterion in line 6 and repeat it with different initial assignments to hopefully find a global and not only local minimum of (3).

Strictly speaking *k-means* refers to the optimization problem (3), which is NP-hard [16, 17]. Therefore, the above described Llyod's algorithm 1 is used to approximate the exact solution and is meant when referring to *k-means* in the remainder of this article. The computational complexity of Algorithm 1 is $\mathcal{O}(n \cdot K \cdot d \cdot \omega)$ with ω the number of iterations until satisfactory convergence is achieved in line 6. Even though ω can grow exponentially in n [18], it is in average (via smoothed analysis) polynomial in n [19]. For real data, it often can be observed that ω does not grow that fast and is considered proportional to n.

Algorithm 1 K-Means Clustering

Input: n data points $\boldsymbol{x}_i$, number of clusters K

1: Randomly assign each $\boldsymbol{x}_i$ to one of the K clusters C_k
2: Compute cluster centers $\boldsymbol{z}_k$ according to (2)
3: **repeat**
4: Reassign each $\boldsymbol{x}_i$ to the cluster C_k corresponding to the nearest center $\boldsymbol{z}_k$
5: Recompute cluster centers $\boldsymbol{z}_k$ according to (2)
6: **until** (3) converges

Output: clusters C_k

Spectral clustering

A more advanced clustering algorithm is spectral clustering, which dates back to 1973. There are several variants like [20] or [21] but we will only describe the so-called unnormalized form as described in [22].

Spectral clustering combines graph and spectral theory. Each data point $\boldsymbol{x}_i$ is viewed as a vertex of an undirected graph $G = (V, E)$. The existence and weight of the edges are calculated from the pairwise defined, symmetric similarities $s_{ij} \geq 0$ of all pairs of nodes $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The greater s_{ij}, the higher is the similarity between nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Since we define similarity in $\mathbb{R}^d$ by proximity, a good choice is the Gaussian similarity function

$$s_{ij} = e^{-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}}$$

for a parameter $\sigma > 0$.

The weight $w_{ij} \geq 0$ of each edge $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ can now be defined in several ways with $w_{ij} = 0$ meaning that $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are not connected. The resulting, weighted graph G is then called *similarity graph*. Some types of similarity graphs are listed below:

- The ϵ-neighborhood graph weights edges only if the corresponding nodes $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ have a distance below $\epsilon > 0$. Since all remaining edges have a similar distance (below ϵ), they can be weighted with 1 instead of s_{ij}. This results in the weights

$$w_{ij} = \begin{cases} 1 & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_j\| < \epsilon, \\ 0 & \text{if } \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \geq \epsilon. \end{cases}$$

- In the l-nearest neighbor graph ($l \in \mathbb{N}$), edges are weighted only if one of the nodes is among the l-nearest neighbors of the other node, i.e.,

$$w_{ij} = \begin{cases} s_{ij} & \text{if } \quad |\{1 \leq \hat{i} \leq n : \|\boldsymbol{x}_{\hat{i}} - \boldsymbol{x}_j\| < \|\boldsymbol{x}_i - \boldsymbol{x}_j\|\}| < l \\ & \text{or } \quad |\{1 \leq \hat{j} \leq n : \|\boldsymbol{x}_i - \boldsymbol{x}_{\hat{j}}\| < \|\boldsymbol{x}_i - \boldsymbol{x}_j\|\}| < l, \\ 0 & \text{otherwise.} \end{cases}$$

- The mutual l-nearest neighbor graph is a variant of the (non-mutual) l-nearest neighbor graph. The only difference is that *both* nodes need to be one of the l-nearest neighbors of the other node, i.e.,

$$w_{ij} = \begin{cases} s_{ij} & \text{if} \quad |\{1 \leq \hat{i} \leq n : \|\boldsymbol{x}_{\hat{i}} - \boldsymbol{x}_j\| < \|\boldsymbol{x}_i - \boldsymbol{x}_j\|\}| < l \\ & \text{and} \quad |\{1 \leq \hat{j} \leq n : \|\boldsymbol{x}_i - \boldsymbol{x}_{\hat{j}}\| < \|\boldsymbol{x}_i - \boldsymbol{x}_j\|\}| < l, \\ 0 & \text{otherwise.} \end{cases}$$

Either way, the result is a weighted graph $(G, \boldsymbol{W})$ with the symmetric, weighted adjacency matrix $\boldsymbol{W} := \{w_{ij}\}_{i,j=1}^n$. This matrix contains not only information about all weights but also whether there is an edge between two nodes ($w_{ij} \neq 0$) or not ($w_{ij} = 0$), therefore, replacing the set of edges E.

The similarity graph can be considered as preprocessing for the actual spectral clustering. As with all preprocessing, the right choice can make the difference. There can be no best choice for the parameters σ, ϵ and l for every use case, as discussed in the beginning of the "Clustering" section, but [22] gives some hints ("rules of thumb") that try to ensure the connectivity of the resulting graph meaning that there is a sequence of edges between every two arbitrary nodes:

- l in the l-nearest neighbor graph chosen around $\log(n)$ satisfies connectivity in the limit $n \to \infty$ for random data points $\{\boldsymbol{x}_i\}_{i=1}^n$.
- For ϵ in the ϵ-neighborhood graph, the length of the longest path in a minimal spanning tree of the (fully connected) graph is always a valid choice by definition. Recall that a minimal spanning tree connects all vertices with the least amount of edges. Since Prim's algorithm [23] for calculating such a tree has complexity $\mathcal{O}(n^2)$, this step can take a significant amount of time.

It is also considered good practice to choose σ as the length of the longest path in a minimal spanning tree as it was described for choosing ϵ. As emphasized in [22], there does not exist any rule of thumb for choosing the parameters that is based on a firm theoretical ground.

After this preparation, the algorithm focuses on the (graph) Laplacian matrix

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$$

with the degree matrix

$$\boldsymbol{D} = \{d_{ij}\}_{i,j=1}^n$$
$$d_{ij} = \begin{cases} \sum_{r=1}^n w_{ir} & \text{if } i = j, \\ 0 & \text{otherwise.} \end{cases}$$

A new matrix $\boldsymbol{U}$ is defined with the help of $\boldsymbol{L}$ by taking K linearly independent vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_K$ of the eigenspaces corresponding to the K smallest eigenvalues (counted with multiplicity) and taking them as the columns of $\boldsymbol{U} \in \mathbb{R}^{n \times K}$. The last step consists of clustering the n rows $\boldsymbol{y}_i \in \mathbb{R}^K$ of $\boldsymbol{U}$ with the k-means algorithm into K clusters. Since each row $\boldsymbol{y}_i$ corresponds to a data point $\boldsymbol{x}_i$, this induces a clustering of $\mathcal{X}$.

The transformation of the data points $\boldsymbol{x}_i$ to $\boldsymbol{y}_i$ allows the identification of non-convex clusters like rings. At first glimpse, it is not clear how this approach should work at all. Luxburg motivates it as the solution of a relaxed graph cut problem and finds analogies to a random walk as well as perturbation theory [22]. Though we can only rigorously prove some theorems like the relation of zero-eigenvalues of $\boldsymbol{L}$ to the number of connected components in G, spectral clustering seems to be a valid clustering technique.

The eigenvalue decomposition is the main factor regarding the computation time and renders this approach useless for very large data sets. A thin eigenvalue decomposition of a large matrix is usually computed with an iterative method like the Lanczos algorithm [24]. Its computational complexity depends on the number of iterations needed during the iterative procedure which again depends on the gap between the eigenvalues of the matrix [25]. Though no computational complexity can be given for spectral clustering in general, it takes at least as long as k-means discussed in the "K-means clustering" section—since k-means is the last step in the algorithm—but should be much higher in practice due to the eigenvalue decomposition. Parallelization of the algorithm [26] and out-of-sample treatment via the Nyström method [27] on sparse grids [28] are newer approaches to reduce the computational complexity. The memory consumption can be another drawback since $\boldsymbol{W}$ is only a sparse $n \times n$ matrix if the parameters ϵ and l are chosen small enough in the creation of the similarity graph.

Methods to predict the number K of clusters exist—both for general clustering algorithms and spectral clustering in particular. They are summarized in [22]. As discussed in the introduction of the "Clustering" section, the structure of the data which should be identified by a clustering algorithm depends on the expectation of the user. Thus, there cannot exist any general rule for how to choose K.

Preprocessing

In [11,12], a method was developed to cluster (finite element) nodes with different moving patterns and intensity across several simulations with small variations in the thickness of the sheet metal. In the end, the clusters were used to analyze the presence of bifurcations. We will present their method in a slightly new setting, define some quality criteria, judge the current approach and improve it.

Let there be R simulation runs with small variations in the parameters of a model with N nodes. The position of node n at time t in simulation r is depicted as $\boldsymbol{p}_n^{(r)}(t) \in \mathbb{R}^3$. After choosing an appropriate time frame $[t_0, t_1]$, the displacement of node n in simulation r is defined as

$$d_n^{(r)} := \|\boldsymbol{p}_n^{(r)}(t_1) - \boldsymbol{p}_n^{(r)}(t_0)\| \tag{4}$$

and collected in the vector

$$\boldsymbol{x}_n := \left(d_n^{(1)}, \ldots, d_n^{(R)}\right) \tag{5}$$

for each node n. Therefore, a vector $\boldsymbol{x}_n \in \mathbb{R}^R$ is assigned to each node n. The nodes can now be grouped by clustering the corresponding set $\mathcal{X} = \{\boldsymbol{x}_n\}_{n=1}^N$ with an algorithm of choice like k-means or spectral clustering, which are both described in the "Clustering" section.

This method is applied to the frontal crash described in the "Setting" section. For a better understanding, we first consider a simplification shown in Fig. 3: A beam consisting of four nodes impacts a rigid wall, which is rotated by an angle of α compared to an orthogonal contact. This simulation is repeated two times ($R = 2$) with angles $\alpha^{(1)} = 20°$ and $\alpha^{(2)} = -15°$ and the displacements $d_n^{(r)}$ plotted for each simulation in Fig. 3c. We will call such a plot *displacement plot* in the following. The red node has a higher displacement in the first simulation due the larger angle. Hence, it is located below the diagonal $\mathbb{R}\boldsymbol{e}$ with $\boldsymbol{e} = (1, 1)$ in the displacement plot.

Quality of clusters and nonlinear behavior

There needs to be a criterion to judge the quality of the clustering after the preprocessing described above. In [11], model reduction was applied to each cluster. Since no new simulations were done with a reduced system, this approach can be seen as data compression. Thus, the reconstruction error resulting from the projection on the subspace spanned by each cluster was a valid criterion to judge the quality in that case. This is, however, not true in our case. In this article, the clustering should be used for the model reduction of subsequent simulation runs with other parameters that were not part of the training data. Therefore, the real error resulting from the model reduction can only be judged after simulating the reduced model. Otherwise, there would not be any new data but only training data to assess the quality of the clustering and reduction. Error estimators or error bounds can also be useful, though they are not available for every reduction method.

So-called objective measures only rely on the data itself to rate the quality of a cluster, see [29,30] for examples. Usually structural data like the cluster density or the separation between clusters is taken into account. While objective measures can be useful for assessing the quality of clusters in general, they are likely not equivalent to the expectations of the user since these expectations can vary a lot as discussed in the "Clustering" section. The other group of cluster indicators are subjective measures that take a specific user expectation into account. In the next section, we define several new subjective measures to assess the nonlinear behavior of a crash.

Before defining the measures, it is of importance to know the sources of nonlinear behavior. According to [31], the reasons for nonlinearities in mechanics lie in the

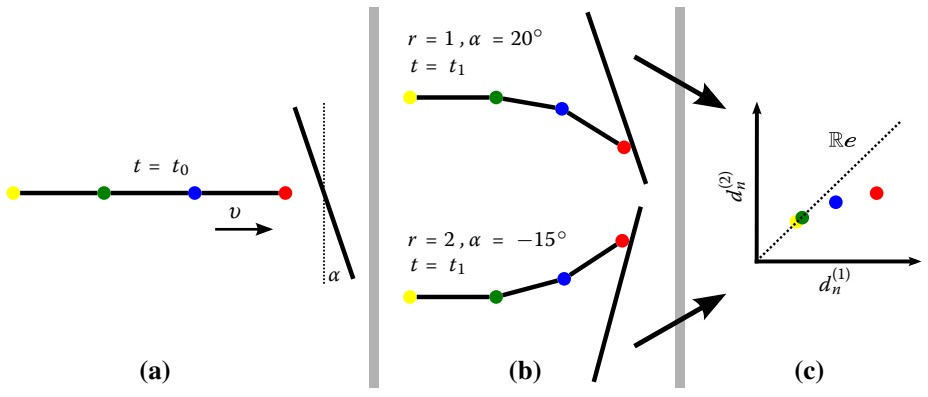

Fig. 3 **a** A simplified crash scenario of a beam against a wall rotated by an angle of α. **b** Two simulation runs for different angles. **c** Resulting displacement plot for the two simulations

- Geometry, e.g., large deformations, bifurcation or snap-through;
- Physics (material laws), e.g., plasticity;
- Boundary conditions, e.g., contact.

We will only focus on the first source of nonlinear behavior since it is most accessible for our data-driven approach.

Measures

In order to quantify the nonlinear behavior, we define measures based on the deformation and scatter of the nodes. It has been seen that the diagonal of the displacement plot represented by the vector $\boldsymbol{e} := (1, \dots, 1) \in \mathbb{R}^R$ plays a fundamental role in the analysis.

With simple geometry, the *deformation*

$$D(\boldsymbol{x}) := \frac{\langle \boldsymbol{x}, \boldsymbol{e} \rangle}{\|\boldsymbol{e}\|}$$

of a vector $\boldsymbol{x} \in \mathcal{X}$ can be defined as the length of the orthogonal projection onto the diagonal $\mathbb{R}\boldsymbol{e}$. This definition of deformation is motivated by the mechanical analog but is not equivalent, as can be seen later. Throughout this article, it is assumed that only the data $\boldsymbol{p}_n^{(r)}(t_0)$ and $\boldsymbol{p}_n^{(r)}(t_1)$ is available for every node n and simulation run r.

The distance from $\boldsymbol{x}$ to its orthogonal projection is called *(absolute) scatter* since it reflects the difference across the simulation runs. It can be calculated by

$$S^{\text{abs}}(\boldsymbol{x}) := \sqrt{\|\boldsymbol{x}\|^2 - D(\boldsymbol{x})^2}.$$

The higher $S^{\text{abs}}(\boldsymbol{x}_n)$, the more the displacement of node n differs in each of the R simulations.

The *relative scatter* S^{rel}, i.e., the absolute scatter divided by the deformation, is more important. Since all relative measures tend to be sensitive when the denominator is relatively small, we only define the relative scatter for nodes with a deformation above 2 % of the maximum deformation in the overall model:

$$D_{\max}(\mathcal{Y}) := \max_{\boldsymbol{x} \in \mathcal{Y}} D(\boldsymbol{x}) \quad \text{for } \mathcal{Y} \subseteq \mathcal{X}$$

$$S^{\text{rel}}(\boldsymbol{x}) := \begin{cases} \frac{S^{\text{abs}}(\boldsymbol{x})}{D(\boldsymbol{x})} & \text{if } D(\boldsymbol{x}) \geq \frac{2}{100} D_{\max}(\mathcal{X}), \\ 0 & \text{if } D(\boldsymbol{x}) < \frac{2}{100} D_{\max}(\mathcal{X}). \end{cases}$$

These measures can now be applied for all nodes in one cluster by averaging. The *mean deformation* of a cluster C_k is given as

$$D_{\text{mean}}(C_k) := \frac{1}{|C_k|} \sum_{\boldsymbol{x} \in C_k} D(\boldsymbol{x})$$

and the *mean relative scatter* $S^{\text{rel}}_{\text{mean}}$ as

$$S^{\text{rel}}_{\text{mean}}(C_k) := \frac{1}{|C_k|} \sum_{\boldsymbol{x} \in C_k} S^{\text{rel}}(\boldsymbol{x}).$$

A cluster with high D_{mean} represents nodes with large deformation which can be a sign of nonlinear behavior as discussed above. If a cluster has a high relative scatter $S^{\text{rel}}_{\text{mean}}$, then

the nodes in this cluster have different displacements throughout all R simulations which is an indication of chaotic behavior.

Analysis of the preprocessing

While the quality of the clustering was only judged with the experience of engineers in [11], we are now able to assess the simulation scenario described in the "Setting" section for the Taurus with the measures D_{mean} and $S^{\text{rel}}_{\text{mean}}$ defined above. We choose $[t_0, t_1] = [100, 1000]$ ms as the smallest time interval covering the complete crash. The Taurus has an initial velocity of 56 km/h (35 mph). It is approaching a rigid wall that has an angle α of 5°, 6° or 7° to the perpendicular axis in each of $R = 3$ simulations. The crash scenario is the same as shown in Fig. 3a but with the full Taurus model instead of the beam. $R > 3$ is of course possible and recommended but would prevent a visual analysis of the R-dimensional displacement plot. For the purpose of this article, it is more important to see properties of the different methods in the plots instead of choosing a higher R. High relative scatter resulting from these small parameter variations is an indicator for chaotic, therefore, nonlinear behavior. For now, we will choose k-means as clustering algorithm with $K = 3$ clusters for simplicity. This is already enough to show some major disadvantages, which can be seen in Fig. 4 in the displacement plot. The corresponding clustering of the crossmember (as a snapshot of the simulation for $t = t_1$ and $r = 1$) is shown in Fig. 5. The crossmember will always be shown for t_1, i.e., after the crash. Each color represents a cluster and the red line in the displacement plot shows the diagonal $\mathbb{R}\boldsymbol{e}$ already known from Fig. 3c. Units on the axes of the displacement plot and the view of the crossmember are always in mm.

First of all, all entries $d_n^{(r)}$ of every vector $\boldsymbol{x}_n$ are above a lower bound of 899 mm. This can be explained with the rigid body movement of the car. Even though the first time step t_0 was chosen to be the beginning of the crash, i.e., the car is in contact with the wall, the crossmember still has some distance to the wall, see Fig. 2 on the left. Therefore, the norm in Eq. (4) is dominated by the x-component, the driving direction of the car. The

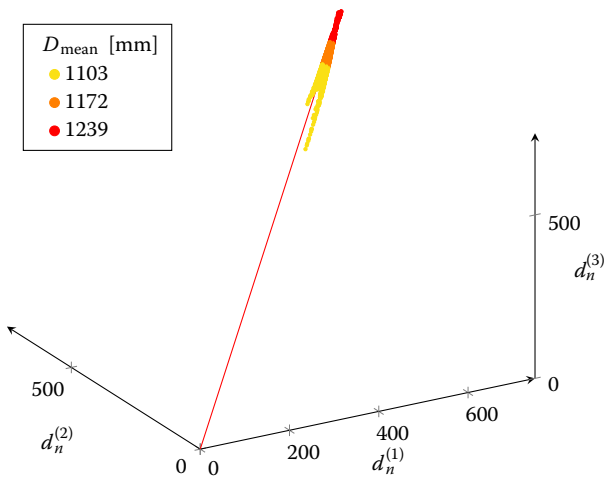

Fig. 4 Original method from [11] (displacement plot). All nodes have a high $d_n^{(r)}$ caused by rigid body movement

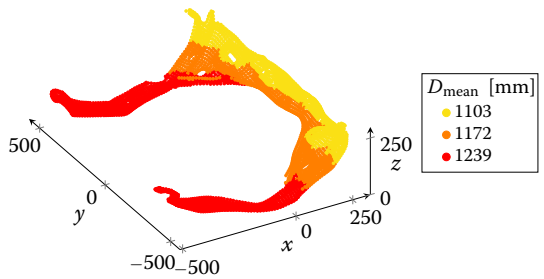

Fig. 5 Original method from [11] (crossmember)

displacements in the other two directions have no significant influence on $d_n^{(r)}$. Since this error is introduced in the beginning of the workflow, it persists in the clustering algorithm, which clusters the nodes by rigid body movement instead of (mechanical) deformation. Figure 5 also supports the hypothesis that the clustering of the nodes only depends on the x-coordinate: The nodes in the rear of the crossmember belong to the cluster with the highest deformation since their movement is decelerated the least.

Another problem lies in the distribution of the vectors $\boldsymbol{x}$. They do not form any natural clusters in Fig. 4, hence clustering algorithms uncontrollably divide the nodes in three more or less connected sets. This questions the purpose of clustering as explained in the "Clustering" section and asks for alternatives.

In this example, we have arbitrarily chosen $K = 3$. There are a few techniques to guess the number of clusters beforehand, but only if there exist any (natural) clusters in the data which is not the case here. Choosing $K = 2$ and just assuming that one cluster represents linear and the other nonlinear behavior will not suffice. Additionally, the mapping of the clusters to the (non)linear area is unclear without any of the measures that are proposed in this article.

Improvements to all these disadvantages will be given in the following.

Improvements

After preliminary work of more theoretical nature and notation, it is now possible to describe and analyze some improvements developed by the authors.

Elimination of rigid body movement

By choosing $[t_0, t_1] = [100, 1000]$ ms as the smallest time interval covering the complete crash in the "Analysis of the preprocessing" section, we have already reduced the impact of the rigid body movement. Bohn et al., instead, looked at different time steps including the full simulation, cf. [11].

In order to eliminate the rigid body movement as much as possible, a reference node is chosen in the back of the crossmember where the least deformation is expected. Let $\boldsymbol{p}_{\text{R}}(t)$ be the position of this node for every time t. The definition (4) of the displacement is now substituted by

$$d_n^{(r)} := \left\| \left(\boldsymbol{p}_n^{(r)}(t_1) - \boldsymbol{p}_n^{(r)}(t_0)\right) - \left(\boldsymbol{p}_{\text{R}}(t_1) - \boldsymbol{p}_{\text{R}}(t_0)\right) \right\| \tag{6}$$

for the remainder of this article. All following and former terms like (5) will use this new definition from now on.

Applying the new definition of displacement to the crossmember leads to much better results depicted in Fig. 6. The bending of the two arms of the crossmember in z-direction is identified correctly by assigning these nodes to the cluster with the highest mean deformation. D_{mean} is now one order of magnitude smaller, representing the actual mechanical deformation of the respective cluster much better.

One could think that shifting the nodes in Fig. 4 to the origin would lead to the same result as subtracting the rigid body movement. This is not true since the above subtraction mostly effects the x-coordinate of $\boldsymbol{p}$ as the major direction of the rigid body movement resulting in an equal weighting of all three spatial coordinates in (6). Therefore, not only the position of the nodes in the left of Fig. 6 moved to the origin, but the formation of the nodes also changed significantly. On top of that, a simple translation of the nodes would not have influenced the k-means clustering algorithm anyway since only the relative positions are important for k-means.

With this improvement, it is now worth a try to replace k-means by the presumably superior spectral clustering. Figure 7 was created with the l-nearest neighbor graph with σ chosen as the length of the longest path in a minimal spanning tree and $l = \log(N)$ respectively as described in the "Spectral clustering" section. In comparison with k-means in Fig. 6, it can be seen that spectral clustering is not restricted to convex sets. Thus, the cluster with the smallest mean deformation in the displacement plot of Fig. 7 now includes the "small arm" on the right. From a mechanical point of view, it is doubtful whether this is an improvement. In general, spectral clustering should be superior to k-means clustering but the increased number of input parameters like σ, ϵ, l leads to a greater spectrum of possible results. Simply changing σ to 1 in the above example results in an unusable clustering. This again shows that clustering algorithms like k-means and spectral clustering should only be used in the presence of natural clusters. The calculation time of spectral clustering is 2 s (additional 10 s for the minimal spanning tree) compared to only 70 ms for k-means using MATLAB R2015b on a desktop computer (Intel Xeon E31245 with 16 GB RAM). The new methods presented in the next section do not suffer from these disadvantages.

Though the subtraction of the rigid body movement with the help of a reference node already improves the clustering significantly, it can still only identify parts by their defor-

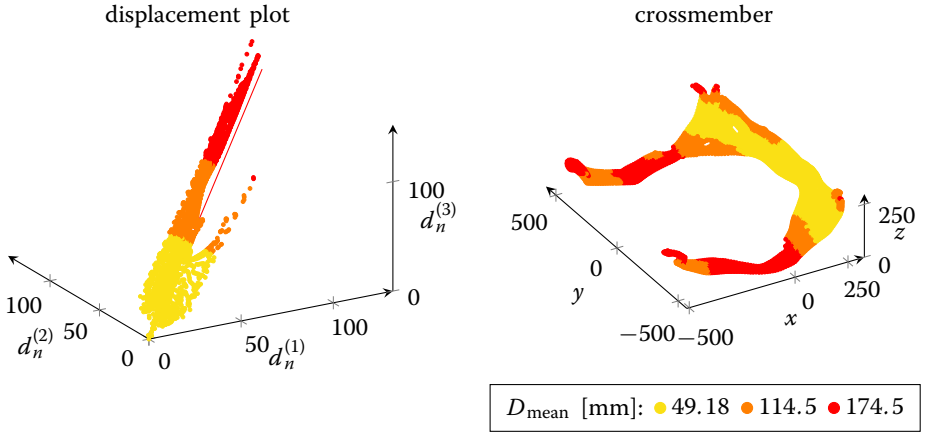

Fig. 6 Elimination of the rigid body movement with a reference node and clustered with k-means

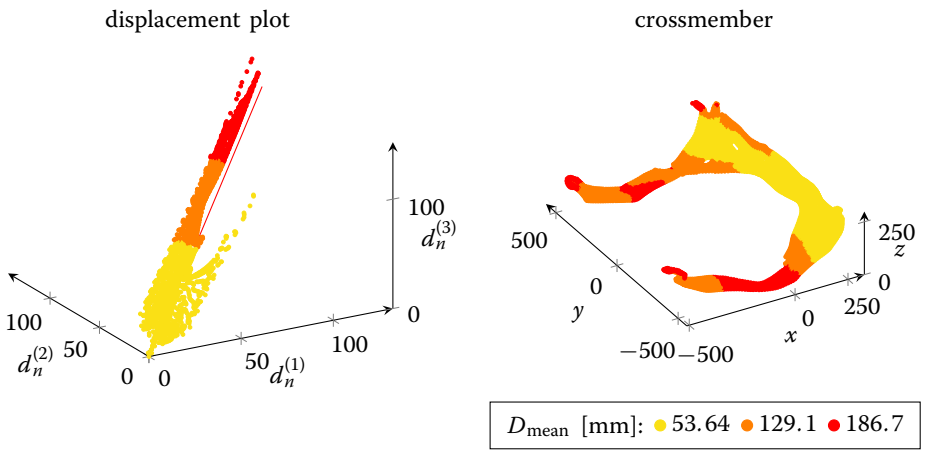

Fig. 7 Spectral clustering with *l*-nearest neighbor graph. Default parameters for σ and *l* were selected

mation behavior. But we are also interested in other sources of nonlinear behavior as described in the "Quality of clusters and nonlinear behavior" section.

Simple alternatives for clustering

With the measures defined in the "Measures" section, it is not only possible to judge existing methods but invent new approaches that use these measures for cluster optimization. This will ensure that the clustering algorithms fit to the measures that are used to assess the quality. We will present several new clustering techniques in the following.

For a general approach, let $M : \mathcal{X} \to \mathbb{R}_{\geq} 0$ be an arbitrary function on $\mathcal{X}$ representing a measure like the deformation D or relative scatter S^{rel}. We can then define the *equal clustering with respect to* M as $\mathcal{C} = \{C_k\}_{1 \leq k \leq K}$ with

$$\begin{aligned}
\alpha &:= \min\{M(\boldsymbol{x}) \colon \boldsymbol{x} \in \mathcal{X}\},\\
\beta &:= \max\{M(\boldsymbol{x}) \colon \boldsymbol{x} \in \mathcal{X}\},\\
\gamma_k &:= \alpha + k \cdot \frac{\beta - \alpha}{K}, \quad k = 0, \ldots, K,\\
C_k &:= \{\boldsymbol{x} \in \mathcal{X} \colon \gamma_{k-1} \leq M(\boldsymbol{x}) < \gamma_k\}, \qquad k = 1, \ldots, K-1,\\
C_K &:= \{\boldsymbol{x} \in \mathcal{X} \colon \gamma_{K-1} \leq M(\boldsymbol{x}) \leq \gamma_K\}.
\end{aligned}$$

This clustering method simply splits the nodes evenly with respect to M into K clusters C_k.

The results for the choice $M = D$ are presented in Fig. 8. In comparison to the clustering depicted in Fig. 6, the border between the clusters with medium and high deformation moved upwards, resulting in a larger cluster with medium deformation. This small difference seems more reasonable to the authors but the judgement what constitutes good clustering will always remain an opinion as stated in the "Clustering" section.

The calculation time of 30 ms is more than halved in comparison to k-means (70 ms) applied directly on $\mathcal{X}$ since equal clustering has linear time complexity in n. One may argue that even the 2 s computation time of spectral clustering is negligible in a preprocessing step in model reduction since the simulation time of a reduced model is higher by several orders of magnitude. As discussed in the "Spectral clustering" section, the computational complexity of spectral clustering scales nonlinearly in n. While k-means can scale linearly,

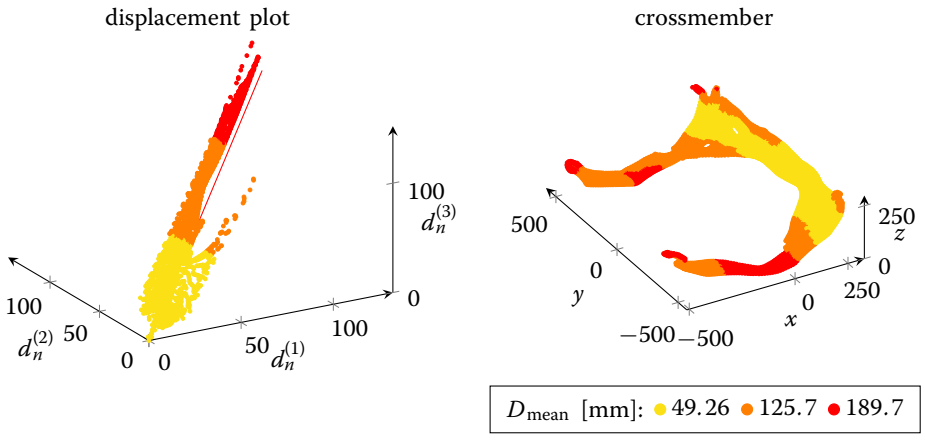

Fig. 8 Equal clustering with respect to D

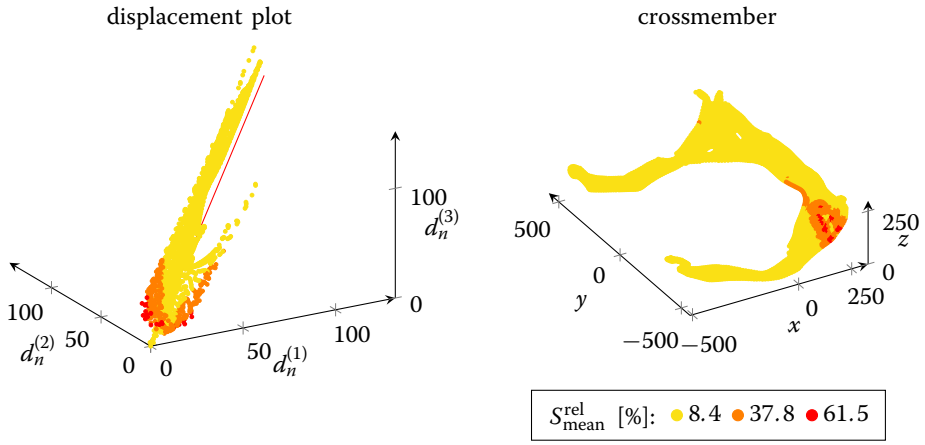

Fig. 9 Equal clustering with respect to scalar S^{rel}

it is however not guaranteed. A full model of a car has at least 1000 times more nodes than the crossmember discussed as an example in this contribution. Thus, the guaranteed linear time and memory scaling of equal clustering contributes to the overall goal of giving a design engineer fast feedback.

The case $M = S^{\text{rel}}$ depicted in Fig. 9 is more interesting since it is the first method that is able to identify areas with higher sensitivity between the simulation runs—another source of nonlinearity. This method allows us to identify the left front part of the crossmember as highly sensitive to small changes in the crash angle. The cluster with the highest sensitivity even has a relative scatter of 61.5 % as can be seen in the changed legend of Fig. 9.

While equal clustering of the scalar relative scatter gives valuable insight in the nonlinear behavior of the crossmember, it can be useful to view the scatter as a vector, i.e., the direction of the orthogonal projection of $\boldsymbol{x}$ on $\mathbb{R}\boldsymbol{e}$ relative to the deformation $D(\boldsymbol{x})$:

$$\boldsymbol{S}^{\text{abs}}(\boldsymbol{x}) := \boldsymbol{x} - \left\langle \boldsymbol{x}, \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \right\rangle \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \in \mathbb{R}^R$$

$$\boldsymbol{S}^{\mathrm{rel}}(\boldsymbol{x}) := \begin{cases} \frac{\boldsymbol{S}^{\mathrm{abs}}(\boldsymbol{x})}{D(\boldsymbol{x})} & \text{if} \quad D(\boldsymbol{x}) \geq \frac{2}{100} D_{\max}(\mathcal{X}), \\ \boldsymbol{0} & \text{if} \quad D(\boldsymbol{x}) < \frac{2}{100} D_{\max}(\mathcal{X}). \end{cases}$$

Notice the different notations: S^{rel} for the scalar and $\boldsymbol{S}^{\mathrm{rel}}$ for the vectorized relative scatter. It is now possible to cluster the vectors $\{\boldsymbol{S}^{\mathrm{rel}}(\boldsymbol{x}) : \boldsymbol{x} \in \mathcal{X}\}$ with common clustering algorithms like k-means, see Fig. 10. Two almost symmetric clusters can be observed in the displacement plot: One representing mostly nodes with $d_n^{(1)} < d_n^{(2)}$ and another cluster representing mostly nodes with $d_n^{(1)} > d_n^{(2)}$. In contrast, the scatter vector $\boldsymbol{S}^{\mathrm{abs}}(\boldsymbol{x})$, i.e., the information about the dominance of one specific simulation is lost with equal clustering with respect to S^{rel}, see Fig. 9.

Grouping

Although we have achieved a strong improvement over the original method described in the "Analysis of the preprocessing" section, it is still an open question which clusters belong to the presumably linearly and nonlinearly behaving areas and how large the number K of clusters should be chosen. We solve both problems at once by grouping the clusters.

The idea consists of choosing a high number of clusters K, selecting an appropriate measure M from the "Measures" section like $M = D_{\mathrm{mean}}$ or $M = S^{\mathrm{rel}}_{\mathrm{mean}}$ and then grouping all clusters C_k whose measure $M(C_k)$ is below or above a threshold τ, respectively. The threshold τ is calculated by

$$\begin{aligned} M_{\min} &:= \min_{1 \leq k \leq K} M(C_k), \\ M_{\max} &:= \max_{1 \leq k \leq K} M(C_k), \\ \tau &:= M_{\min} + \delta(M_{\max} - M_{\min}) \end{aligned} \tag{7}$$

after choosing a parameter $0 \leq \delta \leq 1$. The groups are then defined as

$$\begin{aligned} G_{\mathrm{lin}} &= \bigcup \{C_k : M(C_k) < \tau\}, \\ G_{\mathrm{nonlin}} &= \bigcup \{C_k : M(C_k) \geq \tau\} \end{aligned}$$

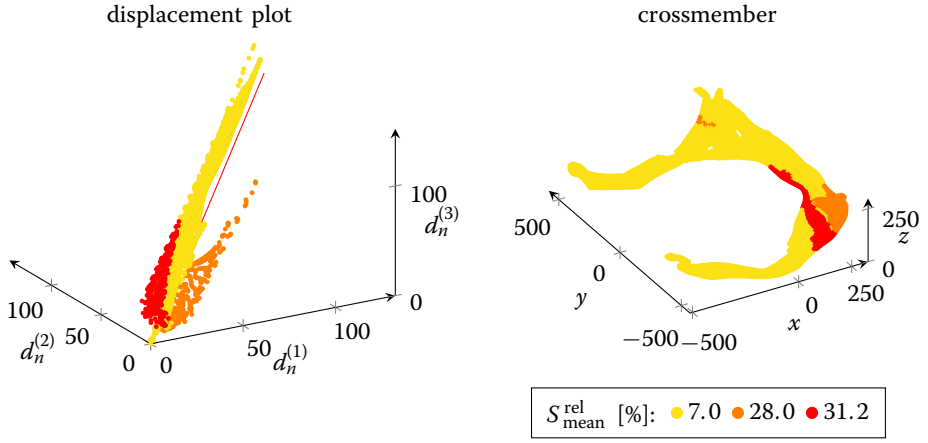

Fig. 10 Clustering of vectorized $\boldsymbol{S}^{\mathrm{rel}}$ with k-means

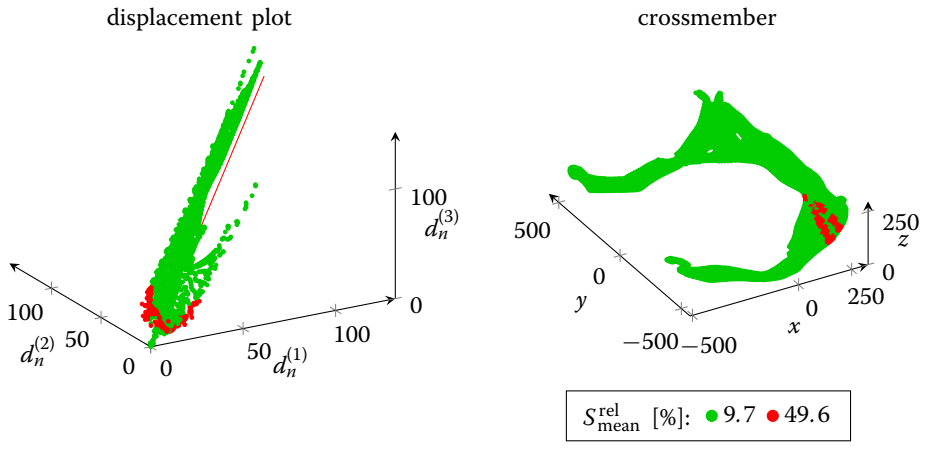

Fig. 11 Grouping of ten clusters generated by k-means with threshold $\delta = 0.7$. G_{lin} is shown in *green* and G_{nonlin} in *red*

with the two groups G_{lin} and G_{nonlin} consisting of the nodes that are behaving presumably linearly or nonlinearly, respectively. The parameter δ controls the border between these two groups whereupon $\delta = 0$ leads to $G_{\text{lin}} = \emptyset$. The higher δ, the more vectors $\boldsymbol{x}$ will be in G_{lin}.

Figure 11 shows the result of such a grouping. First, the crossmember is clustered with k-means like in Fig. 6 but this time into $K = 10$ clusters. The relative scatter of all 10 clusters is measured with $M = S_{\text{mean}}^{\text{rel}}$. $M_{\text{min}} = 3.3\,\%$ is the lowest and $M_{\text{max}} = 66.1\,\%$ is the highest relative scatter out of the 10 values. A parameter $\delta = 0.7$ leads to a threshold of $\tau = 47.3\,\%$ according to (7). Thus, the clusters with the two highest relative scatter values $S_{\text{mean}}^{\text{rel}}$ of 48.8 % and 66.06 % are grouped to G_{nonlin} since their measure is above τ, and the remaining eight clusters are merged into G_{lin}. The result can be examined in Fig. 11. It should be mentioned that the grouping method found almost the same area of high scatter as equal clustering with respect to S^{rel}, see Fig. 9, even though a different clustering method was used.

Choosing an appropriate δ and judging the resulting two groups is still a job that can only be done by an experienced engineer but the measures from the "Measures" section help a lot to automate this part of the workflow. The engineer can first look at the measures of each cluster and get an idea of their range. He can then for example decide that most of the clusters have only a low measure—which is a sign for linear behavior—and choose a high δ like 0.9. Another option would be that every cluster with a relative scatter higher than 10 % should be considered to be behaving nonlinearly. Choosing δ by solving (7) for $\tau = 10\,\%$ will lead to that clustering. Dividing the nodes into G_{lin} and G_{nonlin} would be impossible without the measures from the "Measures" section since the clusters from any clustering algorithm do not have any meaning. They are not sorted or labeled in any way. Only the measures from the "Measures" section give them a meaning like "area with high deformation".

Summary and outlook

It was observed that black box identification of nonlinear behavior can only be solved by heuristics. First, measures were defined to quantify the deformation and scatter, which correlate to nonlinear behavior. With these measures it was possible to judge improve-

ments of existing methods. The subtraction of the rigid body movement is necessary in the preprocessing. New methods like equal clustering, which clusters with respect to a measure, are more robust and faster. Additionally, they can guarantee that the clustering fits to the measures that a user wants to optimize without the need to choose any parameters like in spectral clustering. An optional grouping eliminates the need to estimate the number of clusters. Overall, they all proved to be a valuable contribution in raising the quality of the identification and matched with the experience of engineers.

The algorithms are almost ready to be applied to a full car model. The guaranteed linear time scaling of equal clustering allows the direct application on a full car model. But this can result in clusters that are scattered all over the model. For a later model reduction, it is necessary that the size of the interface between linearly and nonlinearly reduced areas is as small as possible, see [2]. Thus, the geometry of the car needs to be taken into account during the clustering, e.g., all clusters should be connected sets.

The resulting clustering can now be used to apply certain model reduction methods to the clusters in future work. Parts of the model that are considered to behave nonlinearly should be reduced with nonlinear model reduction techniques, the rest with linear methods. This approach is deemed to be advantageous over using only one reduction method. If a crash simulation is only reduced linearly, the error would be higher since nonlinear behavior cannot be reproduced. If nonlinear model reduction is used for the full car model, it would lead to higher computation times due to the usually more complex nonlinear algorithms. This needs to be proven with experiments in future work.

In this paper it was assumed that only the nodal positions are available. If the user has access to more output data like strain or stress, the aforementioned methods can be extended to this new data. It should also be beneficial to combine the results based on deformation and scatter in order to take both sources of nonlinearity into account.

Authors' contributions

DG analyzed and improved the existing methods. JF had the idea to separate linearly from nonlinearly behaving parts and implemented most of the import functions from LS-DYNA to MATLAB. Both authors read and approved the final manuscript.

Acknowledgements

The authors like to thank Anne-Kathrin Zeile for preliminary work. Additionally, they would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/1) at the University of Stuttgart. The 2001 Ford Taurus model has been developed by The National Crash Analysis Center (NCAC) of The George Washington University under a contract with the FHWA and NHTSA of the US DOT.

Competing interests

The authors declare that they have no competing interests.

Received: 19 February 2016 Accepted: 11 May 2016
Published online: 13 June 2016

References

1. Spethmann P, Thomke SH, Herstatt C. The impact of simulation on productivity and problem-solving in automative R&D. Technical report, Technische Universität Hamburg-Harburg; 2006.
2. Fehr J, Holzwarth P, Eberhard P. Interface and model reduction for efficient simulations of nonlinear vehicle crash models. Accepted by Mathematical and Computer Modelling of Dynamical Systems.
3. Radermacher A, Reese S. Model reduction in elastoplasticity: proper orthogonal decomposition combined with adaptive sub-structuring. Comput Mech. 2014;54(3):677–87. doi:10.1007/s00466-014-1020-6.
4. LS-DYNA Product Home Page. http://www.lstc.com/products/ls-dyna/. Accessed 2 Feb 2016.
5. Livermore Software Technology Corporation. LS-DYNA theory manual. Livermore: Livermore Software Technology Corporation; 2006.
6. National Crash Analysis Center. http://www.ncac.gwu.edu/ncac/. Accessed 2 Feb 2016.
7. Marzougui D, Samaha RR, Cui C, Kan C-DS. Extended validation of the finite element model for the 2001 ford taurus passenger sedan. In: Technical report NCAC 2012-W-004, The National Crash Analysis Center, The George Washington University, 45085 University Drive, Ashburn; 2012.

8. Craig R. Coupling of substructures for dynamic analyses: an overview. In: Proceedings of the AIAA dynamics specialists conference, Paper-ID 2000-1573. Atlanta; 2000.
9. Mei L, Thole CA. Clustering algorithms for parallel car-crash simulation analysis. In: Bock H, Phu H, Kostina E, Rannacher R, editors. Modeling, simulation and optimization of complex processes. Berlin: Springer; 2005. p. 331–40.
10. Mei L, Thole CA. Data analysis for parallel car-crash simulation results and model optimization. Simul Model Pract Theory. 2008;16(3):329–37.
11. Bohn B, Garcke J, Iza-Teran R, Paprotny A, Peherstorfer B, Schepsmeier U, Thole C-A. Analysis of car crash simulation data with nonlinear machine learning methods. Procedia Comp Sci. 2013;18:621–30.
12. Griebel M, Bungartz H-J, Czado C, Garcke J, Trottenberg U, Thole C-A, Bohn B, Iza-Teran R, Paprotny A, Peherstorfer B, Schepsmeier U. SIMDATA-NL – Nichtlineare Charakterisierung und Analyse von FEM-Simulationsergebnissen für Autobauteile und Crash-Tests. Final report (in German), Bundesministerium für Bildung und Forschung. 2014. http://publica.fraunhofer.de/dokumente/N-326423.html. Accessed 16 Feb 2015.
13. Jain AK. Data clustering: 50 years beyond k-means. Pattern Recog Lett. 2010;31(8):651–66.
14. Kleinberg JM. An impossibility theorem for clustering. In: Becker S, Thrun S, Obermayer K, editors. Advances in neural information processing systems, vol. 15. Cambridge: MIT Press; 2003. p. 463–70.
15. Jain AK, Dubes RC. Algorithms for clustering data. Upper Saddle River: Prentice-Hall Inc; 1988.
16. Aloise D, Deshpande A, Hansen P, Popat P. NP-hardness of euclidean sum-of-squares clustering. Mach Learn. 2009;75(2):245–8. doi:10.1007/s10994-009-5103-0.
17. Dasgupta S, Freund Y. Random projection trees for vector quantization. IEEE Transact Inform Theory. 2009;55(7):3229–42. doi:10.1109/TIT.2009.2021326.
18. Vattani A. K-means requires exponentially many iterations even in the plane. Discrete Comput Geom. 2011;45(4):596–616. doi:10.1007/s00454-011-9340-1.
19. Arthur D, Manthey B, Röglin H. Smoothed analysis of the k-means method. J ACM. 2011;58(5):1–31. doi:10.1145/2027216.2027217.
20. Ng AY, Jordan MI, Weiss Y. On spectral clustering: analysis and an algorithm. In: Advances in neural information processing systems, vol 14. MIT Press: Cambridge; 2001. p. 849–56.
21. Shi J, Malik J. Normalized cuts and image segmentation. IEEE Transact Pattern Anal Mach Intell. 2000;22(8):888–905.
22. von Luxburg U. A tutorial on spectral clustering. Stat Comput. 2007;17(4):395–416. doi:10.1007/s11222-007-9033-z.
23. Prim RC. Shortest connection networks and some generalizations. Bell Syst Tech J. 1957;36(6):1389–401. doi:10.1002/j.1538-7305.1957.tb01515.x.
24. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. J Res Natl Bur Stand. 1950;45(4):255–82.
25. Kuczyński J, Woźniakowski H. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. SIAM J Matrix Anal Appl. 1992;13(4):1094–122. doi:10.1137/0613066.
26. Zheng J, Chen W, Chen Y, Zhang Y, Zhao Y, Zheng W. Parallelization of spectral clustering algorithm on multi-core processors and GPGPU. In: Computer systems architecture conference, 2008. ACSAC 2008. 13th Asia-Pacific. 2008. p. 1–8. doi:10.1109/APCSAC.2008.4625449.
27. Bengio Y, Paiement J-F, Vincent P, Delalleau O, Roux NL, Ouimet M. Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In: Thrun S, Saul LK, Schölkopf B, editors. Advances in neural information processing systems, vol. 16. Cambridge: MIT Press; 2004. p. 177–84.
28. Peherstorfer B, Pflüger D, Bungartz H-J. A sparse-grid-based out-of-sample extension for dimensionality reduction and clustering with Laplacian eigenmaps. In: Wang D, Reynolds M, editors. Advances in artificial intelligence 2011, vol. 7106. Lecture notes in computer science. Berlin: Springer; 2011. p. 112–21.
29. Rousseeuw PJ. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math. 1987;20:53–65. doi:10.1016/0377-0427(87)90125-7.
30. Stein B, Meyer zu Eissen S, Wißbrock F. On cluster validity and the information need of users. In: Hanza MH, editor. 3rd international conference on artificial intelligence and applications (AIA 03). Anaheim, Calgary, Zurich: ACTA Press; 2003. p. 216–21
31. Wriggers P. Nonlinear finite element methods. Berlin: Springer; 2010.