Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2013) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2013

Analysis of Car Crash Simulation Data with Nonlinear Machine Learning Methods

Bastian Bohn[a], Jochen Garcke[a,b], Rodrigo Iza-Teran[b], Alexander Paprotny[c], Benjamin Peherstorfer[d,*], Ulf Schepsmeier[e], Clemens-August Thole[b]

[a]*Inst. for Numerical Simulation, University of Bonn, Wegelerstr. 6, 53115 Bonn, Germany*
[b]*Fraunhofer SCAI, Schloss Birlinghoven, 53754 Sankt Augustin, Germany*
[c]*Inst. for Mathematics, TU Berlin, Straße des 17. Juni 136, 10623 Berlin, Germany*
[d]*Department of Informatics, TU München, Boltzmannstr. 3, 85748 Garching, Germany*
[e]*Department of Mathematics, TU München, Boltzmannstr. 3, 85748 Garching, Germany*

Abstract

Nowadays, product development in the car industry heavily relies on numerical simulations. For example, it is used to explore the influence of design parameters on the weight, costs or functional properties of new car models. Car engineers spend a considerable amount of their time analyzing these influences by inspecting the arising simulations one at a time. Here, we propose using methods from machine learning to semi-automatically analyze the arising finite element data and thereby significantly assist in the overall engineering process.

We combine clustering and nonlinear dimensionality reduction to show that the method is able to automatically detect parameter dependent structure instabilities or reveal bifurcations in the time-dependent behavior of beams. In particular we study recent nonlinear and sparse grid approaches, respectively. Our examples demonstrate the strong potential of our approach for reducing the data analysis effort in the engineering process, and emphasize the need for nonlinear methods for such tasks.

Keywords: analysis of FEM data; machine learning; car crash simulation; nonlinear dimensionality reduction; sparse grids

1. Introduction

Virtual product development based on numerical simulation is nowadays an essential tool in the car industry. It is used to analyze the influence of design parameters on the weight, costs, functional properties, etc of new car models [1]. We focus on car crash tests where, e.g., the plate thickness of certain components strongly influences the intrusion of the fire wall into the passenger compartment or the acceleration behavior of a model.

We analyze data from car crash simulations with the help of methods from machine learning. Each instance is a full simulation run, which nowadays consists of more than one million finite element (FE) nodes and about hundred time steps, and is therefore very high dimensional, i.e. approx. 10^8 (time $\times$ nodes). During the development process of a new car model simulations are performed where design parameters are changed by the engineer. One simulation run takes about eight hours on a cluster with 32 cores; the number of data points consequently stays small in comparison, nowadays up to 1,000.

*Corresponding author. Tel.: +49-89-289-18630 ; fax: +49-89-289-18607 ; e-mail: pehersto@in.tum.de.
This work was supported by the German Federal Ministry of Education and Research through the project SIMDATA-NL.

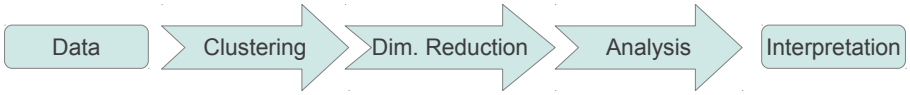


Fig. 1. Our workflow for the analysis of crash data. We cluster the finite element nodes of the car model to gain a first insight into the behavior of the crash and then use nonlinear dimensionality reduction methods to reveal even more characteristics of the data.

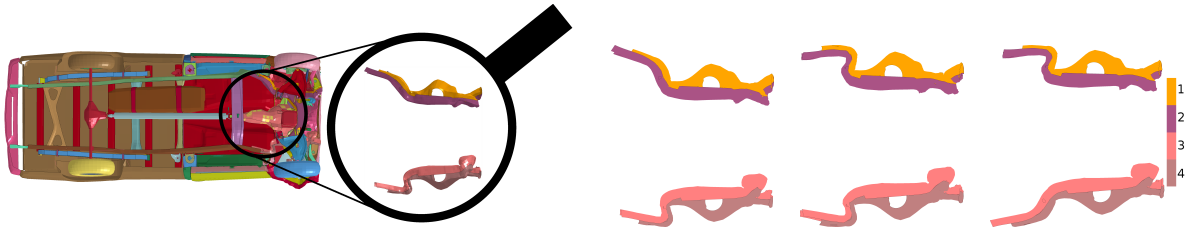
We propose a general framework for the analysis of finite element simulation data, see the workflow in Fig. 1. The use of machine learning methods allows to significantly reduce the analysis effort during the engineering process by semi-automatically revealing characteristics of the simulation runs. Currently, an engineer attempts to identify parameter dependence by sequentially analyzing the time dependent deformations using 3D visualization software. Thus, one needs to manually look at several hundred simulations individually to detect a trend in the behavior of the deformations in order to classify all simulation runs, overall a very time consuming task. Therefore the aim is to have for each simulation run only a few computed attributes left to consider, but which still describe the car crash effects in reasonable detail. One can then easily detect those simulation runs which exhibit an extreme or exceptional behavior and analyze them further. Specifically we investigate how changes in design parameters can lead to a bifurcation in the behavior of a beam of the model. We show how a low-dimensional embedding of the data of the simulation runs is able to separate them into two groups which not only divide the data by means of different behavior but also with respect to different values of one of the input parameters.

Data mining methods have been used for the analysis of simulation data in [2], where Principal Component Analysis (PCA) is used for the analysis of finite element results and in [3], where a clustering method is proposed based on local distance measures between the nodes. In [4] k-means is used on a transformed representation of the original data set. Data analysis of branching for frontal crash simulations is treated in [5].

The first step (clustering) is to obtain a fine-grained partition of the FE nodes of the simulation model where we group nodes into one cluster if they exhibit similar moving patterns and similar moving intensity. Therefore we employ clustering methods to find a partition which reflects the characteristics of the behavior of the FE nodes during the simulation. With such a clustering of the nodes, it is already possible to e.g. spot regions where the beams are crushed the most. However, in particular, the clustering can be seen as a pre-processing of the data for the dimensionality reduction in the next step of our workflow. We obtain more detailed information about the crash behavior if we apply the dimensionality reduction on each cluster separately instead on the whole data at once. We compare the results of k-means and spectral clustering with a recently introduced sub-sampling technique based on sparse grids. Both yield structurally similar results but in contrast to k-means spectral clustering can find clusters with a nonconvex shape and thus is more flexible.

In the second step (dimensionality reduction) of the workflow of Fig. 1 we employ methods to obtain a low dimensional embedding of the simulation runs which still describes most of their characteristics. Additionally, in some situations a relation of the computed attributes to physically relevant ones is possible. The standard dimensionality reduction method is PCA, which gives a low-dimensional representation of data which are assumed to lie in a subspace. But if the data resides in a nonlinear structure PCA gives an inefficient representation with too many dimensions. Therefore, and in particular in recent years, algorithms for nonlinear dimensionality reduction, or manifold learning, have been introduced, see e.g. [6]. The goal consists in obtaining a low-dimensional representation of the data which respects the intrinsic nonlinear geometry. Usually, it is advantageous to not only have the low-dimensional embedding but also to be able to reconstruct, i.e. to pick a point in the low-dimensional space and compute an approximation of the corresponding simulation run. We employ diffusion maps, a local PCA variant, and a principal manifold learning algorithm based on sparse grids which inherently allows for interpolation.

Finally, in the third step the reduced data is analyzed. By comparing the reconstruction errors of our nonlinear dimensionality reduction methods with PCA we can detect if the data contains nonlinear effects. Furthermore, the comparison of the essential dimension between the clusters allows to identify starting areas of bifurcations, since an evolving bifurcation will increase the essential dimension of the solution space. We visualize the low-dimensional embedding of the simulation runs which is expected to reflect the crash behavior. So-called “slow variables” can be obtained which are dominant with respect to the deformation of the car model. In our case the embedding clearly consists of two groups which makes it obvious that a bifurcation in the behavior of at least one beam of the simulation model exists.



(a) Truck from below.

(b) Different bending behavior of beams during crash.

Fig. 2. We consider a Chevrolet C2500 pick-up truck as model problem. In (a) we show the truck from below at time step 17 (after the crash) and the position of beams nr. 1 to 4. In (b) the four essential beams at time t_{17} for three simulation runs. The color bar shows the labels of the beams (physical parts). A different bending behavior in each simulation run can be recognized.

2. The Workflow: Clustering, Dimensionality Reduction and Analysis

In this section we describe the steps of the workflow in more detail. We apply two clustering methods and partition the FE nodes with respect to moving patterns during the crash. On these clusters, dimensionality reduction methods are applied to show the underlying characteristics of the data.

2.1. The TRUCK example

We consider a frontal crash simulation of a Chevrolet C2500 pick-up truck, a model from the National Crash Analysis Center, with around 66,000 FE nodes and 251 physical parts (called TRUCK).

With LS-DYNA and using 17 time steps we have computed 126 simulations of the whole crash where 9 of the underlying design parameters are varied, see Fig. 2. Although this model is more than one order of magnitude less than nowadays used in industry (due to the large computing resources which would be necessary to generate simulation data for current sizes), it still shows typical issues like buckling and high sensitivity to small parameter changes. In this paper we will focus our analysis on four parts, namely the longitudinal chassis beams in the front of the truck, see Fig. 2. These play a key role in the crash behavior and have been selected after consulting with car engineers. Additionally, we will restrict ourselves to a subsample of time steps, whose selection will be motivated in the presented numerical results. Overall, it takes a few minutes per time step to apply our workflow on the TRUCK example given uncompressed, preprocessed data.

Let us now consider a subset of nodes, be it the whole model, a physical part, or a cluster, and denote by M the number of nodes. With $n_1, \ldots, n_M \in \mathbb{N}$ we denote the finite element nodes, whose positions in the three dimensional space are ${}^r x_1^{t_i}, \ldots, {}^r x_M^{t_i} \in \mathbb{R}^3$, where $r \in \{1, \ldots, R = 126\}$ is the number of the simulation run and $t_i \in \{1, \ldots, T = 17\}$ is the current time step. We now define the displacement ${}^r d_l^{t_i,t_j}$ between time step t_i and t_j for one finite element node n_l and the displacement vector ${}^r \boldsymbol{d}^{t_i,t_j}$ of dimension $D = 3M$ for all nodes by

$$ {}^r d_l^{t_i,t_j} := {}^r x_l^{t_i} - {}^r x_l^{t_j} \in \mathbb{R}^3 \qquad {}^r \boldsymbol{d}^{t_i,t_j} := \left(\left({}^r d_1^{t_i,t_j} \right)^T, \ldots, \left({}^r d_M^{t_i,t_j} \right)^T \right)^T \in \mathbb{R}^D. \tag{1} $$

2.2. Clustering

In the first step of our workflow we partition the nodes of the simulation model into clusters such that nodes of a cluster behave "similar". To demonstrate why such a clustering is necessary and to describe what is meant by "similar" we show in Fig. 2 the four beams for three simulation runs at time step t_{17}. A bifurcation in the bending behavior in the rear of the beams over the different simulations can be clearly recognized. An example even as simple as this indicates that the original subdivision into parts hardly reflects similarities in terms of node behavior. The aim of the clustering is now to divide the nodes $n_1, \ldots, n_M$ into k clusters such that the displacements of the nodes in one cluster are similar. To obtain a clustering of the nodes which reflects the behavior over all simulation runs we cluster them all at once, instead of clustering each simulation run r on its own.

The inputs for our clustering algorithms are the number of clusters $k \in \mathbb{N}$ and a data matrix X which contains Euclidean distances of the displacements (1) for all simulation runs $1, 2, \ldots, R$. Thus, for each feasible pair of two time steps t_i and t_j we have

$$ X^{t_i,t_j} = \left(\| {}^r d_m^{t_i,t_j} \|_2 \right)_{m=1,\ldots,M,\ r=1,\ldots,R} \tag{2} $$

Each node $n_1, \ldots, n_M$ corresponds to one row of the data matrix X^{t_i,t_j}.

In the following, we compute a clustering of the selected four beams (see Fig. 2) with k-means in order to obtain benchmark results and with spectral clustering to show that a modern clustering method, capable of finding nonconvex clusters, is necessary. Because spectral clustering becomes computationally infeasible for many data points very quickly, we employ a recently introduced sub-sampling technique based on sparse grids [7].

2.2.1. k-means

The main idea of the k-means algorithm is to minimize the mean squared distance from each data point to its nearest centroid (center) given a set of n data points $\mathcal{Y} := \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, $\boldsymbol{e}_j \in \mathbb{R}^m$ and an integer $k \leq n$, i.e.

$$\arg\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k} \sum_{j=1}^{n} \min_{i\in\{1,\ldots,k\}} \|\boldsymbol{e}_j - \boldsymbol{\mu}_i\|_2^2,$$

where $\boldsymbol{\mu}_i$ are the cluster centers. The corresponding partition into clusters $S_1, \ldots, S_k$ is then induced by the *Voronoi cells* $V_{\boldsymbol{\mu}_i}$ of the centers, i.e. $S_i := \{\boldsymbol{e}_j \in V_{\boldsymbol{\mu}_i}\}$, where $V_{\boldsymbol{\mu}_i} := \{z \in \mathbb{R}^n \mid \|z - \boldsymbol{\mu}_i\| \leq \|z - \boldsymbol{\mu}_j\| \; \forall j \in \{1, \ldots, k\}\}$. One of the most popular heuristics for solving this problem is based on a simple iterative scheme for finding a locally minimal solution. A very efficient approach, which is based on Lloyd's algorithm [8], is the filtering method described by [9]. Even though the algorithm is proven to converge to a solution, it is in most cases just a local minimum. Therefore, the algorithm is repeated several times. In our setting the repetition rate is 10. The convergence is quite fast for a suitably chosen tolerance rate. Since most changes occur in the first few steps, typically an upper bound for the iteration steps is used which does not effect the solution decisively. If the number of clusters is significantly smaller than the number of points k-means is efficient and fast.

2.2.2. Sparse-Grid-Based Sub-Sampling for Spectral Clustering

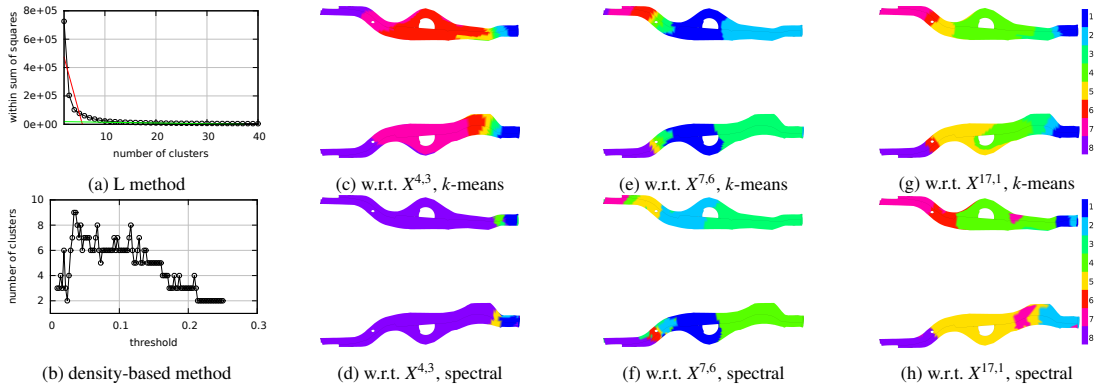
Spectral clustering has emerged as a promising technique for nonconvex clustering, e.g. it is able to find clusters with a nonconvex shape [10]. Unfortunately, because of the eigenproblem, the computational costs are in $O(M^3)$, where M is the number of data points. Here, we approximate the eigenfunctions of the spectral clustering operator on a sparse grid, recently introduced in [7]. Thus, the costs of the eigenproblem are in $O(G^3)$ rather than $O(M^3)$ where G is the number of sparse grid points which is usually distinctly smaller than the number of data points M. Furthermore, we obtain an explicit mapping between ambient and latent space which means we can treat out-of-sample points in a natural way. The cluster assignment of an out-of-sample point can be obtained by simply evaluating the approximated eigenfunctions. Depending on the number of clusters k, we approximate the eigenfunctions corresponding to the second smallest up to the k-th smallest eigenvalue.

In common out-of sample extensions, see, e.g., [11], a kernel is associated to each data point. These are then combined in the Nyström formula to approximate the eigenfunctions. Here we employ ansatz functions associated to sparse grid points rather than to data points, i.e. the eigenfunctions are approximated by functions

$$f = \sum_{i=1}^{G} \alpha_i \phi_i \in V := \operatorname{span} \Phi, \qquad \Phi := \{\phi_i \mid i = 1, \ldots, G\}, \tag{3}$$

where Φ denotes a tensor-product basis – commonly consisting of tensorized B-spline functions – settled on a grid. Unfortunately, a straightforward conventional discretization with N grid points in each direction suffers from the curse of dimensionality: The number of grid points is of the order $O(N^P)$, depending exponentially on the dimension P. For sufficiently smooth functions, sparse grids allow a reduction of the number of grid points by orders of magnitude to only $O(N \log(N)^{P-1})$ while achieving a similar accuracy as in the full grid case [12]. Note that sparse grids have also been successfully applied for classification and regression tasks in data mining, see e.g. [13, 14]. Since the cost of a sparse grid function evaluation is independent of the number of data points a cluster assignment for an out-of-sample point can be obtained very fast, see [7] for details.

For the sparse-grid-based spectral clustering, we first reduced the data to 7 dimensions with PCA (cf. Sec. 2.4) and then computed the (Euclidean) distance matrices. We employed a sparse grid with only 799 grid points and, thus, reduced the size of the eigenproblem from about 7000 to only 799. An exemplary comparison with the ordinary spectral clustering has shown that this reduction by one order of magnitude does not distinctly influence the clustering result.



(a) L method (b) density-based method (c) w.r.t. $X^{4,3}$, k-means (d) w.r.t. $X^{4,3}$, spectral (e) w.r.t. $X^{7,6}$, k-means (f) w.r.t. $X^{7,6}$, spectral (g) w.r.t. $X^{17,1}$, k-means (h) w.r.t. $X^{17,1}$, spectral

Fig. 3. Both, the L-method (a) and the density-based heuristic (b), estimate between 6 to 8 clusters. In (c)-(h) clustering of beams with k-means (top) and spectral clustering (bottom) for displacements (left to right) from t_3 to t_4, from t_6 to t_7 and from t_1 to t_{17}. The illustrations were performed using the FE model at time state t_1.

2.3. Clustering Results

We now present some results for the four beams of the TRUCK example. We employ two very different heuristics to estimate the number of clusters k: The statistical-based L-method [15], and a new heuristic based on density estimation described in [16]. In Fig. 3(a) we plot an evaluation graph which shows the within sum of squares curve. Corresponding to the L-method the x-value at the knee (intersection of the two regression lines which fit the curve most closely) is an estimate of the number of clusters [15]. The density-based heuristic assumes that the number of peaks of an estimated density function of the data is an indicator for the number of clusters. Thus, we have to look for flat regions in the curve shown in Fig. 3(b), see [16] for more details. Both methods predict between 6 to 8 clusters. Setting the number of clusters to $k = 8$, we are on the safe side, since for the dimensionality reduction in the next step of our work flow it is very important that all nodes in one cluster have a similar moving pattern and that clusters are not polluted with nodes which behave differently, cf. Sec. 2.5.2.

In Fig. 3(c)-(h) we compare the clustering obtained by k-means (top) and by spectral clustering with the sparse-grid-based sub-sampling (bottom). We see the clustering assignment of the nodes $n_1, \ldots, n_M$ with respect to $X^{4,3}$ (displacements from t_3 to t_4), with respect to $X^{7,6}$ and with respect to $X^{17,1}$. Let us first consider the clustering obtained by spectral clustering for displacements from t_3 to t_4, see Fig. 3(d). There is one big cluster in the rear and many rather small clusters in the front. This suggests that from time step t_3 to t_4 only the front parts are crushed. Indeed, we could verify this by visualizing the simulation runs. We also see, that k-means yields a distinctly different cluster assignment, cf. Fig. 3(c). With the k-means method we cannot find the small clusters in the front of the beam. This clearly shows that spectral clustering is advantageous here. For the displacements from time step t_6 to t_7, the clustering obtained by k-means and by spectral clustering looks very similar but we will see in the following that there are differences with respect to the reconstruction error in the dimensionality reduction step of our workflow. Notwithstanding the above, both methods yield one big cluster in the front and several smaller ones in the rear, cf. Fig. 3(e) and Fig. 3(f). It can again be verified that this reflects the bending behavior of the beams, i.e. mostly the rear of the beam bends. Now, if we consider the displacements from time step t_1 to t_{17} we capture the complete behavior of the crash. This is reflected by the clustering of both methods: We find multiple clusters in the rear but also in the front parts of the beams. Overall, we might conclude, that the clustering seems reasonable if compared to the actual behavior of the car during the crash.

2.4. Dimensionality Reduction

After a suitable clustering has been performed on the given data set, the resulting partitioning of data is taken as input for common dimensionality reduction and manifold learning tools. To achieve both, a compact and cost efficient description of a given simulation data set, the high-dimensional displacement vectors ${}^r\boldsymbol{d}^{t_i,t_j}$ in the Euclidean space of dimension $D = 3M$ have to be represented in a lower-dimensional space. To this end, the intrinsic dimension s of the R simulation vectors has to be found.

The probably most common approach is principal component analysis (PCA), see e.g. [6]. But as this method is linear, a nonlinear effect can not be identified. As in our application nonlinear effects emerge in form of higher-order parameter interactions, a purely linear approach does not suffice. Various nonlinear dimension reduction methods have been introduced in the recent years, see [6] for an overview. In our project we employ a variant of local tangent space alignment [17] (a simple, localized and intuitive approach), the diffusion map algorithm [18] (an often successfully used spectral approach), and a principal manifold learning algorithm with sparse grids [12, 19] (to obtain an inherent out-of-sample evaluation).

For the reconstruction experiments we consider the data set ${}^r\boldsymbol{d}^{7,6}, r = 1, \ldots, 126$. To allow for an evaluation of the reconstruction results we split the data into a training and a testing set. For the latter, we pick a set $I_{test} \subset \{1, \ldots, 126\}$ of 10 indices at random. The training set is then given by $\mathcal{Y} := \{y_r \mid r \in I_{train}\}$, $I_{train} := \{1, \ldots, 126\} \setminus I_{test}$. In a pre-processing step, we center the points (i.e. training and test data) around the mean of the training data and transform the result by a lossless PCA on the training data. Note here again that we run our algorithms for each cluster separately. With the locally linear approximation and the principal manifold learning algorithms described below, we construct approximations $\overline{y}_r$ to $y_r, r \in I_{test}$.

2.4.1. Locally Linear Approximation (LLA) Based on Tangent Space Estimation

Tangent space estimation plays an important role in various manifold learning methods, see for example diffusion maps [18] or local tangent space alignment [17]. Apart from that, estimated tangent spaces may serve as local approximations to the underlying manifold. The basic idea is as follows: the tangent space at each point in the sample is estimated by means of a weighted, intrinsically local PCA centered at the point, where, roughly speaking, the weights are chosen in such a manner that the larger the distance of a neighbor, the less its contribution to the error measure. Specifically, given the intrinsic dimension s and a point y from the sample, we obtain an estimate of the tangent space at y through the principal space of dimension s of the $k \geq s$ nearest neighbors centered at y. The nearest neighbors are with respect to geodesic distances, which we approximate by means of graph distances in a neighborhood graph. To construct the graph, we invoke the adaptive approach proposed in [20], which, as opposed to (say) the ϵ-rule or k-nearest neighbors [6, pp. 269-272], is capable of handling non-uniformly distributed data. As a refinement, we deploy a weighted PCA for the local tangent space estimation. Specifically, it is intuitively clear that the closer a neighbor is to a considered point, the more it should figure in the estimation of the tangent space thereat.

Given estimates $\tilde{T}_y$ of the tangent spaces at each sample point $y \in \mathcal{Y}$, an approximation to the manifold is given by $\tilde{M} := \bigcup_{y \in \mathcal{Y}} \rho_y(V_y)$, where $V_y := \{z \in \mathbb{R}^n \mid \|z - y\| \leq \|z - \tilde{y}\| \ \forall \tilde{y} \in \mathcal{Y}\}$ denotes the *Voronoi cell* of y with respect to $\mathcal{Y}$ and

$$\rho_y : \mathbb{R}^n \to y + \tilde{T}_y, \ z \mapsto \operatorname*{argmin}_{q \in y + \tilde{T}_y} \|q - z\|.$$

The distance of an out-of-sample point z to $\tilde{M}$ may be estimated as $\min_{y \in \tilde{M}} \|y - z\| \leq \|\rho_{\operatorname{argmin}_{y \in \mathcal{Y}} \|z-y\|}(z) - z\|$. Distances of test points y_r to $\tilde{M}$ are estimated in the same way, i.e. we set $\overline{y}_r := \rho_{\operatorname{argmin}_{y \in \mathcal{Y}} \|y - y_r\|}(y_r)$.

2.4.2. Principal Manifold Learning with Sparse Grids

We use the general idea of principal manifold learning (PML) [21], i.e. the minimization of a regularized error functional on a suitable function space, but in contrast to the original approach discretize this function space by sparse grids. The main advantage of PML in contrast to most other dimensionality reduction methods is the automatic construction of a discretized representation of the manifold. Thus, a *generative model* in the sense of [6], i.e. a function $f : [0, 1]^s \to \mathbb{R}^n$ is constructed to model the observed variables from $\mathbb{R}^n$. This allows us to pick a point in the latent space $[0, 1]^s$ and get the corresponding high dimensional point by an application of f.

In our case we look for $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|I_{train}|} \in [0, 1]^s$ and $f : [0, 1]^s \to \mathbb{R}^{|I_{train}|}$ from a discrete function space V_h which minimize

$$\frac{1}{|I_{train}|} \sum_{r \in I_{train}} \|y_r - f(\boldsymbol{s}_r)\|^2 + \lambda \sum_{j=1}^{|I_{train}|} |f^{(j)}|_{H^1_{\text{mix}}}. \tag{4}$$

The regularization parameter $\lambda > 0$ and the latent space dimension $s \in \mathbb{N}$ have to be chosen suitably. $f^{(j)}$ denotes the j-th component of f and $|f^{(j)}|_{H^1_{\text{mix}}}$ is the H^1_{mix} semi-norm of the real-valued, s-variate function $f^{(j)}$, see [22].

Table 1. Error (see (5)) of PCA, PML and LLA times 10^2 for each beam achieved after k-means.

Beam	1	2	3	4	Mean
PCA, $s = 1$	2.97	2.46	1.64	2.58	2.41
PML, $s = 1$	1.14	1.76	1.20	1.85	1.49
LLA, $s = 1$	0.50	0.86	0.49	0.82	0.67
PCA, $s = 2$	0.79	1.19	0.89	1.57	1.11
PML, $s = 2$	0.44	0.82	0.43	0.73	0.61
LLA, $s = 2$	0.43	0.73	0.44	0.70	0.58
PCA, $s = 3$	0.45	0.74	0.47	0.79	0.61
PML, $s = 3$	0.33	0.59	0.47	0.68	0.51
LLA, $s = 3$	0.36	0.65	0.38	0.58	0.49

Table 2. Error (see (5)) of PCA, PML and LLA times 10^2 for each beam achieved after spectral clustering.

Beam	1	2	3	4	Mean
PCA, $s = 1$	1.82	2.75	1.73	2.62	2.23
PML, $s = 1$	1.34	2.04	1.24	1.80	1.60
LLA, $s = 1$	0.47	0.78	0.46	0.72	0.61
PCA, $s = 2$	1.09	1.73	0.74	1.17	1.18
PML, $s = 2$	0.37	0.60	0.43	0.66	0.52
LLA, $s = 2$	0.42	0.70	0.40	0.59	0.52
PCA, $s = 3$	0.41	0.68	0.42	0.60	0.53
PML, $s = 3$	0.29	0.52	0.27	0.39	0.37
LLA, $s = 3$	0.34	0.61	0.37	0.52	0.46

The minimization is done by an alternating minimization scheme which minimizes (4) for fixed f first and for fixed $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|I_{train}|}$ subsequently. This process is iterated as in an expectation-maximization scheme. For V_h we choose a vector-valued sparse grid ansatz space with domain $[0,1]^s$, i.e. each component is discretized by a sparse grid, cf. (3). Thus, the number of coefficients in our sparse grid ansatz space scales like $O(N \cdot \log(N)^{s-1})$, cf. Sec. 2.2.2. Note, that the exponential dependence only appears with respect to the latent space dimension s and the ambient space dimension affects the computational complexity only linearly. A thorough description of PML with sparse grids and the minimization steps involved can be found in [19]. For the numerical experiments we adaptively refine the discretization spaces according to the *dimension adaptive* procedure described in [22].

After f is determined, we project the test data runs onto the manifold to determine the corresponding pre-image points $\boldsymbol{s}_r \in \mathbb{R}^s$ for $r \in I_{test}$ and to calculate the approximation $\overline{y_r} := f(\boldsymbol{s}_r)$, for details see [19].

2.4.3. Diffusion Maps

As a third nonlinear approach we consider diffusion maps [18]. Here, first a weighted graph $G = (\Omega, W)$ is constructed, where the vertices of the graph are the R observed data points ${}^r\boldsymbol{d}^{t_i,t_j} \in \mathbb{R}^D$. To be precise, we consider the Euclidean norm of the displacement vectors ${}^r\boldsymbol{d}^{1,t_j}$ at a fixed time step t_j for the simulation runs $r = 1, \ldots, R$. The weight which is assigned to the edge between two data points $u \in \Omega$ and $v \in \Omega$ is given by $w_e(u,v) = \mathrm{e}^{-\Delta^2(u,v)/\epsilon}$, where $\Delta(u,v)$ is in general an application-specific, locally defined distance measure (the kernel). A common choice is the Gaussian kernel, which we also apply in our experiments. The value of ϵ controls the neighborhood size. One defines a Markov process on the graph where the transition probability for going from u to v is given by $p(u,v) = w_e(u,v)/\sum_{z\in\Omega} w_e(u,z)$. Hence, if the points are similar then we have a high transition probability, and if they are not then the transition probability is low. The Markov chain can be iterated in q time steps and a so called "diffusion distance" D_q can be defined in a natural way by $D_q^2(u,v) = \sum_{z\in\Omega}(p_q(u,z) - p_q(v,z))^2/\phi_0(z)$, where $\phi_0(z)$ is the stationary distribution of the chain.

Diffusion maps now embeds high-dimensional data into the low-dimensional space such that the diffusion distances D_q are (approx.) preserved. It has been shown in [18] that this is accomplished by the map $\Psi_q : v \to [\lambda_1^q\psi_1(v), \lambda_2^q\psi_2(v), ..., \lambda_s^q\psi_s(v)]$, where λ_j and ψ_j are the eigenvalues and right eigenvectors of the similarity matrix $P = p(u,v)_{u,v}$. Furthermore, one can approximate the distance between transition probabilities using the Euclidean distance in the reduced space. How many terms are used for the embedding, i.e. the dimension s of the low-dimensional space, depends on the kernel and the value of ϵ. In case of a spectral gap, i.e. only the first eigenvalues are significant and the rest are small, one can get a good approximation of the diffusion distance with only a few terms. For details about the theoretical background of the approach we refer to [18].

Note that we also investigated a pre-image approximation to a test point in diffusion maps space based on nearest neighbors as described in [23, 24], but the results were not competitive with the above nonlinear methods and therefore omitted in this paper.

2.5. Analysis of Reduced Data

Due to space constraints we can only give an exemplary selection of a our experimental results here. Note here, that we empirically confirmed, that overall reconstruction on four computed clusters leads to better results than based on the four physical parts. Therefore, it is very reasonable to consider the application of a clustering algorithm before running the dimensionality reduction algorithms. Increasing the number of clusters to eight, as in the following results, increases the performance further.

2.5.1. Reconstruction Error per Beam

To be able to compare LLA and PML with respect to different clustering results, $\bar{y}_r$ is transformed into a D-dimensional vector $\overline{{}^r\boldsymbol{d}^{7,6}}$ by means of the lossless PCA matrix determined in the pre-processing step for each $r \in I_{test}$ and for each cluster. We then construct one vector $\overline{{}^r\boldsymbol{D}^{7,6}}$ by merging the vectors for all clusters. Thus, the resulting vector can be evaluated in every node of the TRUCK model. We analogously construct the vector ${}^r\boldsymbol{D}^{7,6}$ consisting of the vectors ${}^r\boldsymbol{d}^{7,6}$ for each cluster. Finally, we measure the error of beam i by evaluating

$$\frac{1}{|I_{test}|} \sum_{r \in I_{test}} \frac{\| {}^r\boldsymbol{D}_i^{7,6} - \overline{{}^r\boldsymbol{D}_i^{7,6}} \|^2}{\| {}^r\boldsymbol{D}_i^{7,6} \|^2}. \tag{5}$$

A sensible evaluation criterion consists in a comparison of our nonlinear approaches with a global PCA, where we compute reconstructions as $\bar{y}_r := \operatorname{argmin}_{y \in y_c + \mathcal{V}_s} \|y_r - y\|$, with y_c denoting the mean of the training set $\mathcal{Y}$, and $\mathcal{V}_s$ denoting the principal subspace of $\mathcal{Y} - y_c$ of dimension s. The results for the latent space dimension $s = 1, 2, 3$ and the PML-regularization parameter $\lambda = 10^{-1-s}$ can be seen in Tab. 1 and Tab. 2. As the summands in the regularization term increase exponentially in s an exponential decay in the regularization parameter ($\lambda = 10^{-1-s}$) turned out to be a reasonable choice and the results proved to be stable with respect to it.

Both nonlinear methods perform better than the standard PCA, although the difference between the errors shrinks for increasing s. For $s = 1$ we observe a clear advantage of the LLA method over the PML approach and the PCA. For increasing latent space dimension s however, the advantages of the more complex PML method begin to prevail. The partition of the spectral clustering seems to allow for a slightly lower reconstruction error than the one of the k-means algorithm in this setting.

2.5.2. Reconstruction Error per Node

Now, we want to compare the reconstruction error per node of a linear PCA approach with the LLA and PML algorithm. The parameters are the same as in the previous section. In Fig. 4, for every node $n_1, \ldots, n_M$, we plot

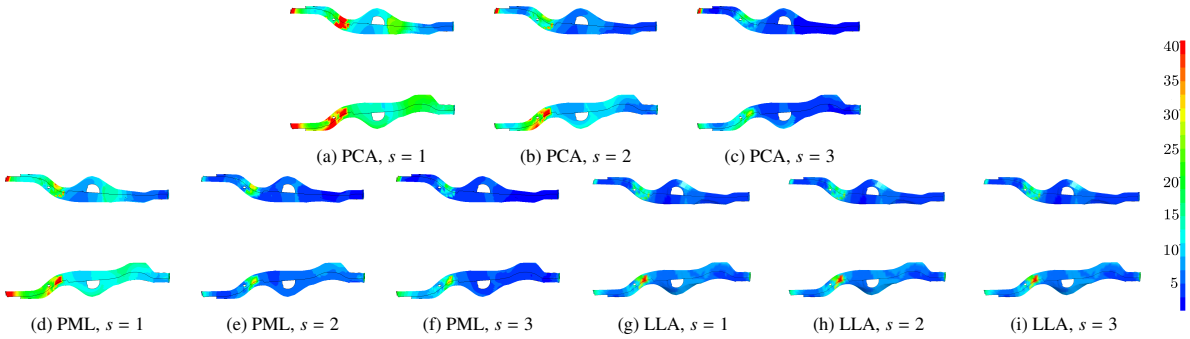
$$\frac{1}{|I_{test}|} \sum_{r \in I_{test}} \left\| \overline{{}^r d_l^{7,6}} - {}^r d_l^{7,6} \right\|^2$$

for each of the eight clusters for k-means. Note, that the results for the spectral clustering are not shown here as they behave similar to the ones in Fig. 4.

We observe, that the reconstruction error in the front of the four beams is always small since there is not much variation over all simulations for the chosen time step. The performance of the LLA and PML methods in the rear parts of the beams, where most of the bending for this time step takes place, is significantly better than the performance of the PCA. As the PML algorithm builds an actual reconstruction of the underlying manifold, it is able to find the intrinsic dimension of the data. Comparing $s = 2$ to $s = 3$ here, and in Tab. 1 and 2, the PML performance does not increase much. Therefore, it seems that two-dimensional nonlinear effects appear in these locations. Single nodes still show relatively large errors for $s = 3$. These nodes belong to small cluster regions which are surrounded by nodes that belong to other clusters.

2.5.3. Latent Space Variable Separation

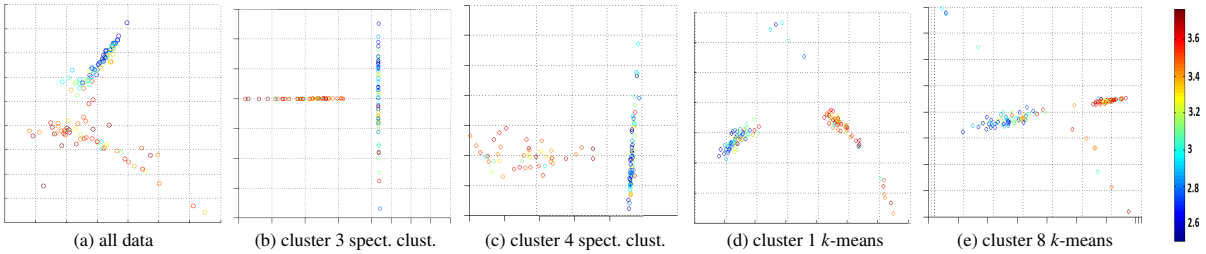
In the following, we compare the embedding of the displacements obtained by applying diffusion maps for $t_j = 6$ on the four given parts at once (cf. Sec. 2.1) and by applying it on the clusters determined by k-means and spectral clustering in Sec. 2.2. Based on the spectral gap, the data has been embedded into an only two-dimensional



(a) PCA, $s = 1$ (b) PCA, $s = 2$ (c) PCA, $s = 3$

(d) PML, $s = 1$ (e) PML, $s = 2$ (f) PML, $s = 3$ (g) LLA, $s = 1$ (h) LLA, $s = 2$ (i) LLA, $s = 3$

Fig. 4. Mean of the squared l_2 errors of the reconstructed displacements over all test runs for the k-means clustering at the time steps $t_i = 7$, $t_j = 6$ for PCA, PML and LLA. The latent space dimension s increases from left to right. Values greater than 40 are colored red.



(a) all data (b) cluster 3 spect. clust. (c) cluster 4 spect. clust. (d) cluster 1 k-means (e) cluster 8 k-means

Fig. 5. Embedding in two dimensions w.r.t. thickness in physical part 1 for all data and clusters obtained by spectral clustering and k-means.

space. We visually investigate the embedding together with the color coded value of the design parameters which were changed in the simulation runs $1, \ldots, 126$. Having the information in this form allows us to identify so-called slow variables, i.e. variables which are dominant with respect to the behavior of the deformations. In Fig. 5 we show how diffusion maps can detect the slow variable describing the thickness of the physical part 1. Overall, it can be seen that by applying diffusion maps on each cluster separately, the embedding reflects the separation of the simulation runs into two groups clearer than when diffusion maps is applied on the whole data set at once. We also found out that the bending behavior of the beams of simulations within each group is similar but it differs significantly when comparing simulation runs from different groups. Therefore, we are able to detect the bifurcation process in the bending behavior and relate it to the thickness of a beam.

3. Conclusion

By applying the presented workflow for the analysis of car crash data on a real-world example we have shown that we can indeed reveal characteristics of the crash. In the first step we obtained a cluster assignment of the nodes where we grouped nodes with similar moving intensity and moving patterns into one cluster. Based on that, we could verify that the clustering indeed captures which nodes are crushed the most at the current time step. Additionally, the results have also shown the need for a clustering method able to find nonconvex clusters because only the sparse-grid-based spectral clustering was able to find a reasonable clustering in all situations.

In the second step we employed nonlinear dimensionality reduction on the clusters obtained in the first step. Since the clustering already roughly reflects the behavior of the crash, the dimensionality reduction methods are applied on data which is pre-processed to provide a better input than just taking all nodes at once. Our example shows that e.g. the embedding obtained by diffusion maps on each cluster separately better reflects the bifurcation in the behavior of a beam than the embedding obtained by applying the method on all data. With a locally linear approximation based on tangent space estimation and the principal manifold learning with sparse grids we have also shown that we can not only obtain a low-dimensional embedding of the crash data but can also quickly reconstruct simulation runs in order to explore simulations with different parameter configurations. The analysis of the reduced data has shown that we get a better reconstruction of the simulation runs with our nonlinear methods

than with PCA, in particular we need less dimensions for a good reconstruction. Just as for the clustering, we see again that nonlinear methods are advantageous.

Overall, by using modern machine learning tools we can provide assistance to car engineers in the design process of a new car model. Using the nonlinear procedures the number of relevant parameters to be investigated can be reduced, whereas the combination of clustering and nonlinear dimensionality reduction leads to a better identification of slow variables. For example, the detection of the bifurcation process with diffusion maps, i.e. the separation of simulations into groups with different bending behavior and different input parameter values (thicknesses), allows the engineer to easily identify a group which consists of runs with a desirable behavior in terms of passenger safety and the corresponding parameter range. We envision that good reconstruction algorithms can avoid the costly computation of a full numerical simulation for suitable parameter configurations, in particular if indicators like e.g. HIC-index (Head Injury Criterion Index) or fire wall intrusion are of interest. Such a semi-automatic data analysis workflow processing bundles of simulation runs can thereby significantly reduce the time spent by the engineer analyzing the influences of design parameters on the weight, costs or functional properties of new car models.

Finally, let us note that the introduced workflow is not limited to car crash data, but can be applied to any collection of numerical simulations. It is worthwhile to investigate these approaches for other numerical data, e.g. from computations for weather forecasts, oil reservoir simulations or aerodynamic simulation of aircrafts.

References

[1] D.-B. P. Haug E. T., Scharnhorst, FEM-Crash Berechnung eines Fahrzeugfrontalaufpralls, in: VDI Berichte 613, 1981.

[2] S. Ackermann, L. Gaul, M. Hanss, T. Hambrecht, Principal component analysis for detection of globally important input parameters in nonlinear finite element analysis, IAM Institute of Applied and Experimental Mechanics, Stuttgart.

[3] L. Mei, C.-A. Thole, Data analysis for parallel car-crash simulation results and model optimization, Sim. Modelling Practice and Theory 16 (3) (2008) 329–337. doi:10.1016/j.simpat.2007.11.018.

[4] C. A. Thole, L. Nikitina, I. Nikitin, T. Clees, Advanced mode analysis for crash simulation results, in: Proc. 9th LS-DYNA Forum, 2010.

[5] R. Iza-Teran, Enabling the analysis of finite element simulation-bundles, Journal of Uncertainty QuantificationAccepted.

[6] J. A. Lee, M. Verleysen, Nonlinear dimensionality reduction, Springer, 2007.

[7] B. Peherstorfer, D. Pflüger, H.-J. Bungartz, A sparse-grid-based out-of-sample extension for dimensionality reduction and clustering with laplacian eigenmaps, in: D. Wang, M. Reynolds (Eds.), AI 2011: Adv. Artificial Intell., Vol. 7106 of LNCS, Springer, 2011, pp. 112–121.

[8] S. Lloyd, Least squares quantization in pcm, IEEE Trans. Information Theory 28 (2) (1982) 129–137. doi:10.1109/TIT.1982.1056489.

[9] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, A. Y. Wu, An efficient k-means clustering algorithm: analysis and implementation, IEEE Trans. Pattern Analysis and Machine Intelligence 24 (7) (2002) 881–892. doi:10.1109/TPAMI.2002.1017616.

[10] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (2007) 395–416.

[11] Y. Bengio, J. Paiement, P. Vincent, Out-of-Sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering, in: NIPS 16, 2003, pp. 177–184.

[12] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numerica 13 (2004) 1–123.

[13] J. Garcke, Regression with the optimised combination technique, in: W. Cohen, A. Moore (Eds.), Proceedings of the 23rd ICML '06, ACM Press, 2006, pp. 321–328. doi:doi:10.1145/1143844.1143885.

[14] D. Pflüger, Spatially Adaptive Sparse Grids for High-Dimensional Problems, Verlag Dr. Hut, 2010.

[15] S. Salvador, P. Chan, Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms, in: 16th IEEE Internat. Conf. on Tools with Artificial Intelligence (ICTAI)., 2004, pp. 576 – 584. doi:10.1109/ICTAI.2004.50.

[16] B. Peherstorfer, D. Pflüger, H.-J. Bungartz, Clustering based on density estimation with sparse grids, in: B. Glimm, A. Krüger (Eds.), KI 2012: Adv. Artificial Intell., Vol. 7526 of LNCS, Springer, 2012, pp. 131–142.

[17] Z. Zhang, H. Zha, Principal manifolds and nonlinear dimension reduction via local tangent space alignment, SIAM SISC 26 (1) (2004) 313–338.

[18] R. Coifman, F. Lafon, Diffusion maps, Applied and computational harmonic analysis 21 (2006) 5–30.

[19] C. Feuersänger, M. Griebel, Principal manifold learning by sparse grids, Computing 85 (4) (2009) 267–299.

[20] J. Giesen, U. Wagner, Shape dimension and intrinsic metric from samples of manifolds with high co - dimension, in: Proc. 19th Annual ACM Symposium on Computational Geometry (SoCG), 2003, pp. 329–337.

[21] A. J. Smola, S. Mika, B. Schölkopf, R. C. Williamson, Regularized principal manifolds, J. of Mach. Learn. Research 1 (2001) 179–209.

[22] B. Bohn, M. Griebel, An adaptive sparse grid approach for time series predictions, in: J. Garcke, M. Griebel (Eds.), Sparse grids and applications, Vol. 88 of Lecture Notes in Computational Science and Engineering, Springer, 2012, pp. 1–30.

[23] S. M. Buchman, A. B. Lee, C. M. Schafer, High-Dimensional Density Estimation via SCA: An Example in the Modelling of Hurricane Tracks, Statistical Methodology 8 (1) (2011) 18–30. arXiv:0907.0199.

[24] D. Kushnir, A. Haddad, R. Coifman, Anisotropic diffusion on sub-manifolds with application to earth structure classification, Applied and Computational Harmonic Analysis 32 (2) (2012) 280–294. doi:10.1016/j.acha.2011.06.002.