# A sparse grid based method for generative dimensionality reduction of high-dimensional data

Bastian Bohn[a], Jochen Garcke[a,b], and Michael Griebel[a,b]

[a] *Inst. for Numerical Simulation, University of Bonn, Wegelerstr. 6, 53115 Bonn, Germany*
[b] *Fraunhofer SCAI, Schloss Birlinghoven, 53754 Sankt Augustin, Germany*

**Abstract**

Generative dimensionality reduction methods play an important role in machine learning applications because they construct an explicit mapping from a low-dimensional space to the high-dimensional data space. We discuss a general framework to describe generative dimensionality reduction methods, where the main focus lies on a regularized principal manifold learning variant. Since most generative dimensionality reduction algorithms exploit the representer theorem for reproducing kernel Hilbert spaces, their computational costs grow at least quadratically in the number $n$ of data. Instead, we introduce a grid-based discretization approach which automatically scales just linearly in $n$. To circumvent the curse of dimensionality of full tensor product grids, we use the concept of sparse grids.

Furthermore, in real-world applications, some embedding directions are usually more important than others and it is reasonable to refine the underlying discretization space only in these directions. To this end, we employ a dimension-adaptive algorithm which is based on the ANOVA (analysis of variance) decomposition of a function. In particular, the reconstruction error is used to measure the quality of an embedding. As an application, the study of large simulation data from an engineering application in the automotive industry (car crash simulation) is performed.

*Keywords:* Generative Models, Machine Learning, Sparse Grids, Dimensionality Reduction, Numerical Simulation Data, Car-Crash Analysis
*2010 MSC:* 62J02, 65K10, 65D15, 68W25

## 1. Introduction

In real-world applications, nominally high-dimensional data often resides on a lower-dimensional manifold. Due to this observation, one of the main topics in mathematical learning theory, see e.g. [1, 2, 3], and machine learning algorithms, see e.g. [4, 5], is the dimensionality reduction of a high-dimensional data set. To this end, it is assumed that there exists an $m$-dimensional manifold $\mathcal{M}$ embedded in $\mathbb{R}^d$ on which the data resides. The task of dimensionality reduction is to build an approximate representation of the low-dimensional manifold $\mathcal{M}$

from the available data in $\mathbb{R}^d$ and by that to obtain a description of the data in the new coordinate system $T \subset \mathbb{R}^m$ governed by the manifold.

While there is a vast amount of algorithms to achieve this, the class of *generative* algorithms in this context is quite small, see [5, 6]. Most algorithms result in a mapping $P : \mathbb{R}^d \to T$ which allows to describe the available data in terms of the manifold coordinates. A generative approach however provides two maps $P : \mathbb{R}^d \to T$ and $f : T \to \mathbb{R}^d$. This way, given a generative algorithm, we can project new data points from $\mathcal{M}$ into $T$ by means of $P$, we can interpolate between two data points in the low-dimensional representation of the manifold and thus generate new, meaningful data on $\mathcal{M}$ by means of $f$.

While the principal component analysis (PCA) is the method of choice for linear manifolds $\mathcal{M}$, the nonlinear case is more relevant for practical applications but also more involved. Here, the explicit determination of general manifolds $\mathcal{M}$ is quite complicated. To this end, generative methods usually assume that there exists a bijective map $\zeta : [0,1]^m \to \mathcal{M}$, i.e. the manifold can be described by one chart. The generative algorithm then takes $T = [0,1]^m$ and approximates $\zeta$ by $f$. A generative model is of special interest when the high-dimensional data space $\mathbb{R}^d$ represents simulation results, for instance the coefficient vector of a finite element solution of a partial differential equation. Here, the comparison of original data and reconstructed surrogates of the $d$-dimensional simulations is inevitable for the quality control of the resulting mapping $f$.

Two well-known examples for nonlinear generative methods are the generative topographic mapping (GTM), see [7], and the regularized principal manifold learning (PML) algorithm, see [8]. Here, the question arises how the mappings $f$ and $P$ are computed. In the original GTM [7] a discretization of $f$ is performed by a full tensor product grid approach. This naturally suffers from the *curse of dimensionality*, see [9], i.e. the case $m > 3$ cannot be treated computationally. In the PML algorithm [8] a kernel-based ansatz centered in the data points is chosen for $f$. This relies on the representer theorem for reproducing kernel Hilbert spaces, see e.g. [4]. While this approach is well-suited if the number $n$ of data is moderate (up to several hundred data points), a grid-based approach is more favorable for large $n$. However, as for the GTM, a full tensor product grid does not work in the case $m > 3$.

To circumvent the curse of dimensionality at least to some extent, sparse grid approaches have been introduced for both the GTM, see [10, 11], and the PML, see [12]. With a sparse grid discretization it is possible to exploit higher regularity of $\zeta$, i.e. if $\zeta$ has Sobolev regularity of bounded mixed smoothness we obtain almost the same approximation quality as in the full tensor product grid approach but with significantly lower computational complexity, see [13].

In this paper we will introduce an adaptive sparse grid PML approach which is an enhancement of [12]. To this end, we will present an alternating minimization scheme to solve the PML optimization problem over a generalized sparse grid to obtain $f : [0,1]^m \to \mathbb{R}^d$. We suggest error indicators which rely on the so-called hierarchical surplus of a sparse grid basis function. These error indicators are then used to refine the underlying sparse grid space in spatial directions in which the current approximation of $f$ varies the most. The alter-

nating minimization and the refinement of the underlying sparse grid space are then iterated until a given computational complexity threshold of a maximum refinement level is reached.

We will apply our method to a high-dimensional data set of finite element simulations. This specific scenario is based on the virtual product development arising in typical engineering applications, in particular we consider the automotive industry. Here, as part of the development cycle for a new car model, the influence of design parameters on the crash behavior of a car is analyzed by an engineer with the help of numerical crash simulations [14]. Each simulation consists of approximately one million finite element nodes and up to several hundred snapshots in time, and is therefore very high-dimensional, i.e. approximately $10^8$ (time $\times$ nodes). The resulting huge data bases of finite element simulations can be used for sophisticated data-driven analysis steps, see [15, 16]. For example, the detection of the number of different effects such as tearing or bending behavior in a set of crash simulation data is of special interest. As an example, we will study the deformation of several beams in the front part of the car with the help of our dimension-adaptive PML method.

The outline of this paper is as follows: In section 2 we will introduce the idea of generative dimensionality reduction in the most general setting followed by a more detailed view of two common generative approaches - the PCA and the PML methods - and how they fit into our general setting. In section 3 we will shortly review the concept of sparse grids and introduce an adaptive version of the sparse grid PML method. Section 4 deals with a state-of-the-art big data manifold learning problem which stems from current demands in the automotive industry. We apply our adaptive sparse grid PML algorithm to this problem and compare the results to the linear PCA method. Section 5 contains some concluding remarks.

## 2. Generative dimensionality reduction

Let us assume that we are given $n$ data points $\{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$ which have been drawn i.i.d. according to an unknown probability measure $\rho$ on $\mathbb{R}^d$ which is absolutely continuous with respect to the Lebesgue measure. Furthermore, let $\rho$ be supported on an unknown compact manifold $\mathcal{M} \subset \mathbb{R}^d$ of dimension $m$. Since the treatment of general manifolds is very involved we impose an additional condition on the structure of $\mathcal{M}$ and assume that there exists a bijective map $\zeta : [0,1]^m \to \mathcal{M}$, i.e. the compact manifold $\mathcal{M}$ can be described by a single chart. The problem of determining $\mathcal{M}$ now becomes the problem of approximating $\zeta$. Note here that $\rho$ contains more information about $\mathcal{M}$ than $\zeta$ does because it also gives a weight to each region on $\mathcal{M}$. The fact that this information is lost when we are just given $\zeta$ is resembled by the invariance of $\mathcal{M}$ under reparametrization, i.e. any bijective map $\psi : [0,1]^m \to [0,1]^m$ can be employed and $\zeta \circ \psi : [0,1]^m \to \mathcal{M}$ is still bijective. Thus there are infinitely many possibilities to describe the manifold. It is reasonable that the general generative approach also takes the weighting of different regions in $\mathcal{M}$ into account.

*2.1. General generative approach*

A general approach reads

$$\min_{g \in G} \mathcal{A}_\rho(g)$$

where $\mathcal{A}_\rho : G \to \mathbb{R}$ is a (possibly nonlinear) functional over a set of functions $G \subset \{g : [0,1]^m \to \mathbb{R}^d\}$. $\mathcal{A}_\rho(g)$ describes the error that is made when using $g$ to describe the manifold $\mathcal{M}$ given implicitly by $\rho$. Usually $\mathcal{A}_\rho$ has the form

$$\mathcal{A}_\rho(g) = \int_{\mathbb{R}^d} c(g, \mathbf{x}) \mathrm{d}\rho(\mathbf{x}), \tag{1}$$

where $c : G \times \mathbb{R}^d \to [0, \infty)$ is called cost function. Since $\rho$ is unknown, it is substituted with the empirical measure

$$\frac{1}{n} \sum_{i=1}^{n} \delta_{\mathbf{x}_i}. \tag{2}$$

Here $\delta_{\mathbf{x}_i}$ denotes the Dirac distribution centered in $\mathbf{x}_i$. This leads to the empirical minimization functional

$$\mathcal{A}_{\mathrm{emp}}(g) = \frac{1}{n} \sum_{i=1}^{n} c(g, \mathbf{x}_i) \tag{3}$$

instead of $\mathcal{A}_\rho$. Depending on the choice of $G$ the minimization of $\mathcal{A}_{\mathrm{emp}}$ can be an ill-posed problem and therefore a restriction of the search set can be necessary for numerical minimization. To this end, typically an additional regularization term $\lambda S(g)$ is added to $\mathcal{A}_{\mathrm{emp}}$ with regularization parameter $\lambda > 0$. Then, the solution $f$ of a general generative approach is defined by

$$f := \arg\min_{g \in G} \mathcal{A}_{\mathrm{emp}}(g) + \lambda S(g). \tag{4}$$

Here, $S$ usually penalizes large norms of $g$, i.e. $S(g) = \|g\|_H^2$ where the space $H$ reflects the smoothness requirements on $g$. Two examples are the Sobolev-Bochner space

$$\begin{aligned} H &= H^k([0,1]^m; \mathbb{R}^d) \\ &= \left\{ g : [0,1]^m \to \mathbb{R}^d \mid \|g\|_{H^k}^2 := \sum_{\|\mathbf{l}\|_1 \le k} \int_{[0,1]^m} \|\mathrm{D}^{\mathbf{l}} g(\mathbf{t})\|_{\ell_2}^2 \mathrm{d}\mathbf{t} < \infty \right\} \end{aligned}$$

of order $k \in \mathbb{N}$, see e.g. [8], and the Sobolev-Bochner space

$$\begin{aligned} H &= H^k_{\mathrm{mix}}([0,1]^m; \mathbb{R}^d) \\ &= \left\{ g : [0,1]^m \to \mathbb{R}^d \mid \|g\|_{H^k_{\mathrm{mix}}}^2 := \sum_{\|\mathbf{l}\|_\infty \le k} \int_{[0,1]^m} \|\mathrm{D}^{\mathbf{l}} g(\mathbf{t})\|_{\ell_2}^2 \mathrm{d}\mathbf{t} < \infty \right\} \end{aligned} \tag{5}$$

of dominating mixed smoothness of order $k \in \mathbb{N}$, see e.g. [12]. Here, we denote by $\mathbf{l} = (l_1, \ldots, l_m) \in \mathbb{N}^m$ a multivariate index and use the multivariate derivative operator $\mathrm{D}^{\mathbf{l}} := \frac{\partial^{(l_1,\ldots,l_m)}}{\partial_{\mathbf{t}_1}^{l_1} \ldots \partial_{\mathbf{t}_m}^{l_m}}$.

As depicted in [5] in a detailed manner, dimensionality reduction methods can be categorized into distance preserving and topology preserving methods. The first ones aim at maintaining the pairwise distances between the original data points also for the embedded points in the low-dimensional space. The latter ones target at preserving proximities in the data in a qualitative way, i.e. points which are close to (or far from, respectively) each other in the original data should be mapped to low-dimensional points which are also close to (or far from, respectively) each other. A generative dimensionality reduction approach with smoothness regularization can be interpreted as a topology preserving method. To this end, assume that $g \in G$ is Lipschitz-continuous with constant $C > 0$ and let $\mathbf{t}_1, \mathbf{t}_2 \in [0,1]^m$ be points in the low-dimensional space. Then, we obtain

$$\|g(\mathbf{t}_1) - g(\mathbf{t}_2)\|_{\ell_2} \leq C \|\mathbf{t}_1 - \mathbf{t}_2\|_{\ell_2}.$$

Therefore, if two points on the manifold are far apart from each other, i.e. $\|g(\mathbf{t}_1) - g(\mathbf{t}_2)\|_{\ell_2}$ is large, then the corresponding low-dimensional points are also far apart from each other, i.e. $\|\mathbf{t}_1 - \mathbf{t}_2\|_{\ell_2}$ is large, or the function $g$ has a large Lipschitz constant, i.e. $C$ is large. Since the optimal Lipschitz constant is directly related to the first derivative of $g$, a dimensionality reduction algorithm with an $H^1$ regularization can be interpreted as a topology preserving method in the this sense. However, since $H^1([0,1]^m; \mathbb{R}^d)$ is no reproducing kernel Hilbert space (RKHS), see e.g. [17], for $m \geq 2$, a regularization with the $H^1$ norm can still lead to an ill-posed problem. Therefore, higher order Sobolev spaces or spaces of dominating mixed smoothness are a suitable choice as they lead to both, a well-posed minimization problem, and a topology preserving method.

We will now shortly review how the linear principal component analysis (PCA) and the nonlinear principal manifold learning (PML) algorithm fit into this setting. Note that also the generative topographic mapping (GTM) is compatible with our general framework. Since we focus on PCA and PML we provide a corresponding description for the GTM only in the appendix for the sake of completeness.

### *2.2. Principal Component Analysis*

There exist many different interpretations of the principal component analysis [18] in terms of a minimization problem, see e.g. [19]. We will stick to a geometrically motivated variant here. To simplify our notation we assume that $\mathbb{E}_\rho[\mathbf{x}] = \int_{\mathbb{R}^d} \mathbf{x} \, \mathrm{d}\rho(\mathbf{x}) = \mathbf{0}$. Then, $g \in G = \{\mathbf{W} \mid \mathbf{W} \in \mathbb{R}^{d \times m}, \mathbf{W}^T\mathbf{W} = \mathbf{I}\}$ is a linear orthogonal map, i.e. $g$ is a matrix, and

$$\mathcal{A}_\rho(g) = \int_{\mathbb{R}^d} \min_{\mathbf{t} \in [0,1]^m} \|\mathbf{x} - g\mathbf{t}\|_2^2 \mathrm{d}\rho(\mathbf{x}),$$

i.e. $c(g, \mathbf{x}) = \min_{\mathbf{t}\in[0,1]^m} \|\mathbf{x} - g\mathbf{t}\|_2^2$. In the case of finite data and unknown $\rho$ this becomes

$$\mathcal{A}_{\text{emp}}(g) = \frac{1}{n}\sum_{i=1}^{n} \min_{\mathbf{t}_i\in[0,1]^m} \|\mathbf{x}_i - g\mathbf{t}_i\|_2^2.$$

Note that the assumption of centered data has to be made for this representation, i.e.

$$0 = \mathbb{E}_{\frac{1}{n}\sum_{i=1}^{n}\delta_{\mathbf{x}_i}}[\mathbf{x}] = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i.$$

If this is not the case the data has to be centered first. Without centering, the representation is more involved, see e.g. [19].

It can easily be shown that the PCA minimization problem is well-posed. Therefore, no regularization is needed. To compute the optimal $f$, an eigendecomposition of the covariance matrix of the data $\mathbf{x}_i, i = 1, \ldots, n$, has to be done. Then, the columns of $f \in \mathbb{R}^{d\times m}$ are the $m$ eigenvectors corresponding to the $m$ largest eigenvalues of the data covariance matrix. For a $g \in G$, the preimage points $\mathbf{t}_i$ of $\mathbf{x}_i$ are the images

$$P(\mathbf{x}_i) = \operatorname*{arg\,min}_{\mathbf{t}_i\in[0,1]^m} \|\mathbf{x}_i - g\mathbf{t}_i\|_{\ell_2}^2 \tag{6}$$

under the projection $P : \mathbb{R}^d \to T$. For the optimal matrix $f$, they can be determined by a basis transform of $\mathbf{x}_i$ into the eigenbasis of the covariance matrix and a subsequent truncation of the result.

The PCA is clearly a linear method. It works well in many situations (provided that $m$ is sufficiently large) but can deteriorate or even fail in finding a low-dimensional representation for nonlinear manifold data. Then a nonlinear approach like the PML is superior.

### *2.3. Principal Manifold Learning*

For principal manifold learning we let $g \in G$ with

$$G = L_2([0,1]^m; \mathbb{R}^d) := \left\{ g : [0,1]^m \to \mathbb{R}^d \mid \|g\|_{L_2}^2 := \int_{\mathbb{R}^d} \|g(\mathbf{t})\|_{\ell_2}^2 \mathrm{d}\mathbf{t} < \infty \right\}.$$

Analogously to [4], the goal is to find a function $g \in G$ such that the $\rho$-dependent error

$$\mathcal{A}_\rho(g) := \int_{\mathbb{R}^d} \inf_{\mathbf{t}\in[0,1]^m} \operatorname{dist}(\mathbf{x}, g(\mathbf{t}))\, \mathrm{d}\rho(\mathbf{x})$$

for a fixed distance function $\operatorname{dist} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$ is minimized. A common choice for dist is the squared Euclidean norm of the difference of the arguments. Thus, we get

$$\mathcal{A}_\rho(g) := \int_{\mathbb{R}^d} \inf_{\mathbf{t}\in[0,1]^m} \|\mathbf{x} - g(\mathbf{t})\|_{\ell_2}^2\, \mathrm{d}\rho(\mathbf{x})$$

and $c(g, \mathbf{x}) = \inf_{\mathbf{t}\in[0,1]^m} \|\mathbf{x} - g(\mathbf{t})\|_{\ell_2}^2$. The evaluation of $c$ now involves solving a global nonlinear optimization problem in $m$ dimensions which can be very

hard to handle numerically. However, as we will explain in the next section, our discretized variant of the PML will deal with a search set $G$ which consists of piecewise linear functions. This can be exploited for the efficient evaluation of $c$. After substituting $\rho$ by the empirical measure (2) we obtain

$$\mathcal{A}_{\mathrm{emp}}(g) := \frac{1}{n} \sum_{i=1}^{n} \inf_{\mathbf{t}_i \in [0,1]^m} \|\mathbf{x}_i - g(\mathbf{t}_i)\|_{\ell_2}^2.$$

Due to the structure of the cost function $c$ the preimage $\mathbf{t}_i = P(\mathbf{x}_i)$ of a data point $\mathbf{x}_i$ is individually determined by solving the $m$-dimensional minimization problem

$$P(\mathbf{x}_i) = \arg\min_{\mathbf{t}_i \in [0,1]^m} \|\mathbf{x}_i - g(\mathbf{t}_i)\|_{\ell_2}^2. \tag{7}$$

Note that even if $P(\mathbf{x}_i)$ exists (e.g. for continuous $g \in G$) it is not necessarily unique. Analogously to the evaluation of $c$, the computation of $P(\mathbf{x}_i)$ also involves the solution of a global optimization problem which is only feasible for specific choices of $g$.

As we directly see, the functional $\mathcal{A}_\rho$ and the projection $P$ are exactly the same for PCA and PML. The two methods only differ in the search set $G$ which is much larger for PML than for PCA. Thus, for principal manifold learning, the minimization of $\mathcal{A}_{\mathrm{emp}}$ over $G$ is an ill-posed problem and a regularization term $S(g) = \|f\|_H^2$ has to be added. Therefore, the overall minimization problem for PML becomes

$$f = \arg\min_{g \in G} \mathcal{A}_{\mathrm{emp}}(g) + \lambda \|g\|_H^2 \tag{8}$$

with a fixed regularization parameter $\lambda$, see also (4). In [8] different choices of reproducing kernel Hilbert spaces were introduced for $H$. In [12] the regularization is done in the mixed Sobolev-Bochner space $H = H^1_{\mathrm{mix}}([0,1]^m; \mathbb{R}^d)$ which is also an RKHS.

The numerical minimization is done analogously to the expectation-maximization (EM) scheme, where the minimization with respect to $g$ and the minimization with respect to $\mathbf{t}_i, i = 1, \ldots, n$, is split into separate minimizations for each argument. This scheme is then iterated until convergence is reached. In the first step, $\mathcal{A}_{\mathrm{emp}}(g) + \lambda \|g\|_H^2$ is minimized for fixed $\mathbf{t}_i, i = 1, \ldots, n$, i.e. the $\inf_{\mathbf{t}_i \in [0,1]^m}$ in front of $\|\mathbf{x}_i - g(\mathbf{t}_i)\|_{\ell_2}^2$ disappears. In [8], the famous representer theorem for reproducing kernel Hilbert spaces is used in this step to obtain a finite kernel expansion

$$g(\mathbf{t}) = \sum_{j=1}^{n} \boldsymbol{K}(\mathbf{t}_j, \mathbf{t}) \boldsymbol{\alpha}_j \tag{9}$$

for the principal manifold $g$, where $\boldsymbol{\alpha}_j \in \mathbb{R}^d, j = 1, \ldots, n$. Here, $\boldsymbol{K} : H \times H \to \mathbb{R}^{d \times d}$ denotes the matrix-valued reproducing kernel of $H$. We refer to [4, 17] for details on the representer theorem and on matrix-valued reproducing kernels. With (9) we see that only a finite-dimensional optimization problem has to be solved, even though the minimization takes place in the possibly infinite-dimensional RKHS $H$. In the second step of the alternating scheme $g$ is kept

fixed and (7) is evaluated to obtain updated values for $P(\mathbf{x}_i) = \mathbf{t}_i$ for every $i = 1, \ldots, n$. Note that also in the case of reproducing kernel Hilbert spaces $P(\mathbf{x}_i)$ still has to be computed by solving an $m$-dimensional global nonlinear optimization problem. The two steps of the alternating scheme are then iterated until convergence.

## 3. Discretized PML with adaptive sparse grids

As we showed in (9), the application of the representer theorem leads to a finite expansion of $g$ in terms of the point evaluated kernel functions $\boldsymbol{K}(\mathbf{t}_j, \mathbf{t}), j = 1, \ldots, n$. Then, it is easy to see that for fixed $\mathbf{t}_1, \ldots, \mathbf{t}_n$ a minimizer of (8) solves the system of linear equations

$$(\bar{\mathbf{K}} + \lambda \mathbf{I})\vec{\boldsymbol{\alpha}} = \vec{\mathbf{x}}$$

with an $n \times n$ kernel block matrix given by $\bar{\mathbf{K}}_{i,j} = \boldsymbol{K}(\mathbf{t}_i, \mathbf{t}_j) \in \mathbb{R}^{d \times d}$. Here, $\mathbf{I}$ denotes the $nd$-dimensional identity matrix, $\vec{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_n)^T$ is the coefficient vector and $\vec{\mathbf{x}} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)^T$ represents the data vector. Since $\bar{\mathbf{K}}$ is usually a dense matrix, the computational costs of solving this equation system scale like $(nd)^3$ for a naive direct approach and at least like $(nd)^2$ for more involved algorithms. Therefore, this is not feasible for problems with large data set size $n$. Alternatively, it is possible to work with localized kernel functions such that most $\bar{\mathbf{K}}_{i,j}$ are negligible, see [6]. In this case however, the problem of choosing an appropriate scale parameter for the kernel has to be dealt with.

To overcome these problems, we again use a finite basis representation as in (9), but this time choose a grid based ansatz instead of the kernel basis. To simplify our notation we consider the components $g^{(i)}, i = 1, \ldots, d$, of $g$ independently, i.e. for every $g^{(i)}$ we have

$$g^{(i)}(\mathbf{t}) = \sum_{k=1}^{N_i} \beta_k^{(i)} \gamma_k^{(i)}(\mathbf{t}), \tag{10}$$

where $N_i$ is the number of basis functions spent to represent $g^{(i)}$, $\gamma_k^{(i)} : [0,1]^m \to \mathbb{R}$ is the $k$-th such basis function and $\beta_k^{(i)} \in \mathbb{R}$ is the corresponding coefficient. Using the notation (10) in (8) we obtain the following alternating minimization approach:

**Algorithm 1 - Grid-based PML**

1. Initialize $\mathbf{t}_i$ by a PCA of the data $\mathbf{x}_i, i = 1, \ldots, n$.
2. Solve the minimization problem

$$\min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - g(\mathbf{t}_i)\|_{\ell_2}^2 + \lambda \|g\|_H^2.$$

If $H$ is a Hilbert space and $g \in G$ is defined by (10), the minimization problem can be translated into $d$ systems of linear equations ($i = 1, \ldots, d$):

$$\left((\boldsymbol{B}^{(i)})^T\boldsymbol{B}^{(i)} + n\lambda\boldsymbol{C}^{(i)}\right)\vec{\boldsymbol{\beta}}^{(i)} = (\boldsymbol{B}^{(i)})^T\vec{\mathbf{x}} \tag{11}$$

with matrices defined by $\boldsymbol{B}^{(i)}_{j,k} = \gamma^{(i)}_k(\mathbf{t}_j)$, $\boldsymbol{C}^{(i)}_{k,l} = \langle\gamma^{(i)}_k, \gamma^{(i)}_l\rangle_H$ and the coefficient vector $\vec{\boldsymbol{\beta}}^{(i)} = \left(\beta^{(i)}_1, \ldots, \beta^{(i)}_{N_i}\right)^T$.

3. Solve the $m$-dimensional nonlinear minimization problem (7) to determine the updated $\mathbf{t}_i$ for each $i = 1, \ldots, n$.

The steps 2 and 3 are then iterated until convergence, see [12] for details.

For better readability, we will restrict ourselves to the scalar-valued case in the following explanations and omit the upper index $(i)$. A vector-valued function is then built by the $d$ scalar-valued component functions (10).

### *3.1. Isotropic sparse grids*

A common choice for a multivariate basis function $\gamma_k : [0,1]^m \to \mathbb{R}$ is a tensor product of piecewise linear hat functions. To this end, let $\phi : \mathbb{R} \to [0,1]$ be defined by

$$\phi(t) := \begin{cases} 1 - |t|, & \text{if } t \in [-1,1] \\ 0 & \text{else} \end{cases}$$

and let

$$\phi_{l,i}(t) := \phi(2^l \cdot t - i)|_{[0,1]}$$

for any $l, i \in \mathbb{N}$ be a dilated and rescaled version of $\phi$ restricted to the interval $[0,1]$. The construction of an $m$-variate hat function is now straightforward via the tensor product

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{t}) := \prod_{j=1}^{m} \phi_{l_j,i_j}(\mathbf{t}_j),$$

where $\mathbf{l} = (l_1, \ldots, l_m) \in \mathbb{N}^m$ is the multivariate level index and $\mathbf{i} = (i_1, \ldots, i_d) \in \mathbb{N}^m$ denotes the multivariate position index. After proper re-indexing, the $\gamma_k$ can now be identified with the $\phi_{\mathbf{l},\mathbf{i}}$.

Next, we will define full grid spaces and sparse grid spaces. To this end, let

$$\mathbf{I}_\mathbf{l} := \left\{ \mathbf{i} \in \mathbb{N}^m \;\middle|\; \begin{array}{ll} 0 \le i_j \le 1, & \text{if } l_j = 0 \\ 1 \le i_j \le 2^{l_j} - 1,\ i_j \text{ odd} & \text{if } l_j > 0 \end{array} \quad \text{for all } 1 \le j \le m \right\} . \tag{12}$$

Then, we define the hierarchical increment space $W_\mathbf{l} := \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \mathbf{I}_\mathbf{l}\}$ and the space of piecewise $m$-linear functions on the regular (isotropic) full grid of level $l \in \mathbb{N}$ by

$$V_l := \bigoplus_{|\mathbf{k}|_\infty \le l} W_\mathbf{k} ,$$

represented in the so-called hierarchical basis

$$\{\phi_{\mathbf{k},\mathbf{i}} \mid \mathbf{i} \in \mathbf{I}_\mathbf{k}, |\mathbf{k}|_\infty \le l\} .$$

Here, $|\mathbf{k}|_\infty := \max_{i=1,\dots,m} k_i$ denotes the $\ell_\infty$ norm. If

$$h_l = \sum_{|\mathbf{k}|_\infty \leq l} \sum_{\mathbf{i} \in \mathbf{I}_\mathbf{k}} \beta_{\mathbf{k},\mathbf{i}} \phi_{\mathbf{k},\mathbf{i}}$$

is the $L_2$-best approximation of $h \in H^2([0,1]^m)$ in $V_l$ it holds that

$$||h - h_l||_{L_2([0,1]^m)} = \mathcal{O}\left(2^{-2l}\right). \tag{13}$$

However, since the dimension $|V_l| = \left(2^l + 1\right)^m$ of $V_l$ grows exponentially in $m$, this approach is not feasible for $m > 3$ already for moderate level $l$. Therefore, we introduce the sparse grid space with similar approximation properties but significantly smaller basis size. We define the regular sparse grid space of level $l$ by

$$V_l^s := \bigoplus_{\substack{\mathbf{k} \in \mathbb{N}^m \\ \theta_m(\mathbf{k}) \leq l}} W_\mathbf{k}\,, \tag{14}$$

where $\theta_m(\mathbf{0}) := 0$ and

$$\theta_m(\mathbf{k}) := |\mathbf{k}|_1 - m + |\{j \mid k_j = 0\}| + 1$$

for every other $\mathbf{k} \in \mathbb{N}^m$. Here, $|\mathbf{k}|_1 := \sum_{j=1}^m |k_j|$ denotes the $\ell^1$ norm. This specific definition of $\theta_m$ guarantees that the resolution of grids on the boundary is the same as the resolution of grids in the interior of the domain. If

$$h_l^s(\mathbf{t}) = \sum_{\substack{\mathbf{k} \in \mathbb{N}^m \\ \theta_m(\mathbf{k}) \leq l}} \sum_{\mathbf{i} \in \mathbf{I}_\mathbf{k}} \beta_{\mathbf{k},\mathbf{i}} \phi_{\mathbf{k},\mathbf{i}}(\mathbf{t}) \in V_l^s$$

is the $L_2$-best approximation of $h \in H^2_{\text{mix}}([0,1]^m)$ in $V_l^s$, it holds that

$$||h - h_l^s||_{L_2([0,1]^m)} = \mathcal{O}\left(2^{-2l} l^{m-1}\right).$$

Thus, compared to (13), the accuracy is only slightly worse by a factor $l^{m-1}$. However, the number of basis functions in the sparse grid ansatz space is just $N = \mathcal{O}\left(2^l \cdot l^{m-1}\right)$ and the exponential dependence of $m$ now only affects the level $l$ instead of $2^l$. For a thorough treatment of sparse grids, approximation results and complexity issues we refer to [13] and the references cited therein.

Let us now consider the computational costs of algorithm 1 with a sparse grid basis and an $H^1_{\text{mix}}$ regularization: In step 1 of the algorithm a PCA of the data has to be calculated. To this end, the $d \times d$ covariance matrix is built and an eigen-decomposition of it is computed. Therefore, the runtime of this step is $\mathcal{O}\left(nd^2 + d^3\right)$. When dealing with a few data points in a high-dimensional space, i.e. $d > n$, one can also do an eigen-decomposition of the Gramian matrix instead of the covariance of the data, see e.g. [5]. Then, the computational costs of step 1 become $\mathcal{O}\left(dn^2 + n^3\right)$. Usually, since we only have to compute the eigenvalues of the covariance matrix once, the runtime of the initial PCA is

still much smaller than the computational effort a kernel PML which solves an $nd \times nd$ equation system in every iteration of the alternating minimization scheme. However, if the initial PCA is still infeasible one can also initialize the $\mathbf{t}_i \in [0,1]^m$ randomly. In this case, algorithm 1 might end up in a local minimum of $\mathcal{A}_{\text{emp}}(\cdot) + \lambda \| \cdot \|_H^2$ which does not characterize the true manifold as good as the PCA-initialized variant. Another, more sophisticated initialization method, would be to use the projection of the input data points onto the hyperplane which results from a PCA of a small random subset of the input data.

To solve the systems of linear equations in step 2 of algorithm 1 we employ a diagonally-preconditioned CG-solver. To this end, we do not have to assemble the matrices $B^{(i)}$ and $C^{(i)}$ explicitly for $i = 1, \ldots, d$ but compute their application to a vector instead. Here, the structure of the sparse grid and the locality of the basis functions allow for a runtime of $\mathcal{O}\left(d \cdot \left(2^{m+l} \cdot l^{m-1} + n l^{m-1}\right)\right)$ for a single step of the CG-algorithm with a sparse grid of level $l$ in all $d$ components, see [20] for details on exploiting the specific sparse grid structure to obtain a fast iterative solver. Although the number of CG iterations is not constant in general, a suitable choice of the regularization parameter $\lambda$ leads to a small number of iterations until the CG algorithm reaches a prescribed error bound. Therefore, we neglect the iterations count in the complexity analysis.

In the third step of algorithm 1, we have to solve the optimization problem (7) for a given function $g \in V_l^s$. As we mentioned above, the application of a nonlinear optimization algorithm on the whole domain $[0,1]^m$ can not be afforded in general. Therefore, we will instead use a heuristic approach which exploits the sparse grid structure and restricts the search domain to a small hyperrectangle on which $g$ is linear. To this end, we first compute the best sparse grid point $\mathbf{p}$, i.e.

$$\mathbf{p}_i := \operatorname*{arg\,min}_{\mathbf{t}_i \in \text{grid}(V_l^s)} \|\mathbf{x}_i - g(\mathbf{t}_i)\|_{\ell_2}^2$$

for each $i = 1, \ldots, n$ which costs $\mathcal{O}(dnN)$. Here, $N$ is the number of basis functions $|V_l^s|$ and

$$\text{grid}(V_l^s) := \{\mathbf{t} \mid \phi_{\mathbf{l},\mathbf{i}}(\mathbf{t}) = 1 \text{ for some } \phi_{\mathbf{l},\mathbf{i}} \in V_l^s\}$$

is the set of points in the sparse grid of level $l$. Then, we employ a Newton-type minimization on a restricted domain $D(\mathbf{p}_i)$ to obtain the minimum of (7) for each $i = 1, \ldots, n$. This search domain $D(\mathbf{p}_i)$ is the hyperrectangle $R$ with corner $\mathbf{p}_i$ on which $g$ is multilinear and for which $\left(\nabla \|g(\mathbf{p}_i) - \mathbf{x}_i\|_{\ell_2}^2\right)^T \omega(R)$ is smallest, where $\omega(R) \in \{-1, 1\}^m$ is the unique vector such that $\mathbf{p}_i + \omega(R)$ points into the direction of $R$. This can be interpreted as one steepest descent step to determine the optimal hyperrectangle $R$. Within the domain $D(\mathbf{p}_i)$ the Newton-type algorithm only needs $\mathcal{O}(1)$ iterations to converge. The determination of $D(\mathbf{p}_i)$ can be done in $\mathcal{O}(2^m dN)$. In each Newton step the gradient and the Hessian of the target function in (7) have to be computed and an $m \times m$ system of linear equations has to be solved for each $i = 1, \ldots, d$. Therefore, the overall runtime of step 3 can be bounded from above by $\mathcal{O}\left(m^3 dNn\right)$.

Finally, assuming that a constant number of iterations of the alternating minimization scheme is used, the computational costs of algorithm 1 with a sparse grid discretization can be bounded by

$$\begin{aligned} & \mathcal{O}\left(nd^2 + d^3 + d \cdot \left(2^{m+l} \cdot l^{m-1} + nl^{m-1}\right) + dnN + 2^m dN + m^3 dnN\right) \\ = \quad & \mathcal{O}\left(d\left(nd + d^2 + n2^{m+l}l^{m-1}\right)\right), \end{aligned}$$

where we used $N = \mathcal{O}\left(2^l l^{m-1}\right)$ and $m^3 = \mathcal{O}\left(2^m\right)$. As we directly see, the runtime is linear with respect to the number of samples $n$ in contrast to a kernel-based method. As we mentioned above, another initialization could be used in step 1 to get rid of the higher order dependence on the dimension $d$.

### *3.2. Dimension-adaptive sparse grids*

Our sparse grid construction has been purely isotropic so far. Thus, every spatial direction is equally resolved by the discretization. However, in most practical applications some directions are more important than others and it is reasonable to resolve these directions in more detail. Since the importance of different directions is not known a-priori, an adaptive approach is the method of choice to achieve an appropriate discretization. The dimension-adaptive algorithm we employ here is based on [21].

First, we define the altered index sets

$$\tilde{\mathbf{I}}_{\mathbf{l}} := \left\{ \mathbf{i} \in \mathbb{N}^m \;\middle|\; \begin{array}{ll} i_j = 0, & \text{if } l_j = -1 \\ i_j = 1, & \text{if } l_j = 0 \qquad \text{for all } 1 \le j \le m \\ 1 \le i_j \le 2^{l_j} - 1,\ i_j \text{ odd} & \text{if } l_j > 0 \end{array} \right\} \tag{15}$$

and also allow the negative level $-1$. Furthermore, we define the univariate basis function $\phi_{-1,0} := \chi_{[0,1]}$ to be the indicator function of the interval $[0, 1]$. With this and the definition

$$\tilde{W}_{\mathbf{l}} := \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} \mid \mathbf{i} \in \tilde{\mathbf{I}}_{\mathbf{l}}\}$$

we see that $W_{\mathbf{l}}$ and $\tilde{W}_{\mathbf{l}}$ are the same for a multilevel index $\mathbf{l}$ with $l_j \ge 1$ for all $j = 1, \ldots, m$. This way, we just have split the space of linear functions on $[0, 1]$, which was previously spanned by the two linear basis functions associated to the two boundary points, further into the sum of one constant (level $-1$) and one linear function (level 0). If we define the $\ell_1$ norm of a level index with possibly negative coordinates as

$$|\mathbf{l}|_1 := |(\max(l_1, 0), \ldots, \max(l_d, 0))|$$

we can maintain our previous definition of sparse grid spaces (14) using

$$\tilde{\theta}_m(\mathbf{k}) := \begin{cases} 0 & \text{if } k_j \le 0 \text{ for all } 1 \le j \le m \\ |\mathbf{k}|_1 - m + |\{j \mid \mathbf{k}_j \le 0\}| + 1 & \text{else} \end{cases}$$

instead of $\theta_m(\mathbf{k})$. This slightly different choice of the hierarchical increment spaces $\tilde{W}_\mathbf{l}$ results in a direct analogy of the altered discretization to the so-called analysis-of-variance (ANOVA) decomposition, see [22] for a detailed explanation.

We now return to the vector-valued notation to describe the dimension-adaptive procedure. The main component for the algorithm is the error indicator which decides if the ansatz space is refined in a certain direction. To this end, let $g = (g^{(1)}, \dots, g^{(d)})^T$ be a vector valued function with components $g^{(i)}, i = 1, \dots, d$, given in the form

$$g^{(i)} = \sum_{\mathbf{k} \in \mathbb{K}^{(i)}} \sum_{\mathbf{j} \in \tilde{\mathbf{I}}_\mathbf{k}} \beta^{(i)}_{\mathbf{k},\mathbf{j}} \phi_{\mathbf{k},\mathbf{j}},$$

analogously to (10). Here $\mathbb{K}^{(i)} \subset (\mathbb{N} \cup \{-1\})^m$ is the set of level indices of the grid in direction $i \in \{1, \dots, d\}$. Then, the error indicator for component $i$ is defined by

$$\epsilon^{(i)}_\mathbf{k} := \max_{\mathbf{j} \in \tilde{\mathbf{I}}_\mathbf{k}} \left|\left| \beta^{(i)}_{\mathbf{k},\mathbf{j}} \phi_{\mathbf{k},\mathbf{j}} \right|\right|_{L_2([0,1]^m)}.$$

For more elaborate techniques and details on how to choose a reliable *and* efficient indicator for the case of specific norms of the error, we refer to [23].

Given a fixed threshold $\epsilon$, the adaptive algorithm can now be described by the following steps.

**Algorithm 2 - Dimension-adaptive sparse grid PML**

1. Set $g = (g^{(1)}, \dots, g^{(d)})^T$ to the result of the alternating minimization algorithm (Algorithm 1, steps 1-3) with regular sparse grid ansatz spaces $\mathbb{V}^{(i)} := V^s_{l_\text{start}}$ of small level $l_\text{start}$ for each component $i = 1, \dots, d$.
2. Initial Compression: For each $i = 1, \dots, d$: For each $\tilde{W}_\mathbf{l} \subset \mathbb{V}^{(i)}$ check if $\epsilon^{(i)}_\mathbf{k} \leq \epsilon \|g^{(i)}\|_{L_2([0,1]^m)}$ for all $\mathbf{k}$ with $\mathbf{k} \geq \mathbf{l}$ and $\tilde{W}_\mathbf{k} \subset \mathbb{V}^{(i)}$. If this is the case, remove all these $\tilde{W}_\mathbf{k}$ from $\mathbb{V}^{(i)}$.
3. Set $g = (g^{(1)}, \dots, g^{(d)})^T$ to the result of the alternating minimization algorithm (Algorithm 1, steps 1-3) with ansatz spaces $\mathbb{V}^{(i)}$ for each component $i = 1, \dots, d$.
4. Adaption: For each $i = 1, \dots, d$: For each $\tilde{W}_\mathbf{l} \subset \mathbb{V}^{(i)}$ check if $\epsilon_\mathbf{l} \geq \epsilon \|g^{(i)}\|_{L_2([0,1]^m)}$. If this is the case add the spaces $\tilde{W}_\mathbf{k}$ to $\mathbb{V}^{(i)}$ for all $\mathbf{k} \leq \mathbf{l} + \mathbf{e}_j$ with all $j \in \{1, \dots, m\}$ with $l_j \neq -1$. Here, $\mathbf{e}_j$ denotes the $j$-th unit vector.

The steps 3 and 4 are then iterated until a maximum refinement level $l_\text{end}$ is reached in at least one direction. Then the algorithm ends after executing step 3 one last time. The solution $f$ is defined as the resulting function $g$. In step 4 of algorithm 2 the choice of directions $j$ with $l_j \neq -1$ resembles the fact that the algorithm only refines in directions in which the function $g^{(i)}$ is non-constant, see [22] for details. This has the effect that the components of the ANOVA decomposition of $g^{(i)}$ which have no relevance at all, i.e. $l_j = -1$ after the compression in step 2, are not refined again.

The dimension-adaptive procedure builds ansatz spaces $\mathbb{V}^{(i)}$ for each $i = 1, \ldots, d$, which are refined according to the relevance of a direction in terms of the size of the coefficients in the sparse grid basis expansion. Note that we now have to take the union of the different adaptive sparse grids in each component function to find the best grid point for the projection onto the manifold (step 3 of algorithm 1) before running a Newton minimizer on the corresponding restricted domain again. For a more elaborate description of the dimension-adaptive algorithm, its benefits and the relation to the ANOVA decomposition we refer to [21, 22].

Finally, note that the regular sparse grid PML and even the dimension-adaptive sparse grid PML become computationally infeasible if the reduced dimension $m$ is still quite large, e.g. $m > 10$. The reason for this is that if $l_{\text{start}} \geq 0$ the initial space $V^s_{l_{\text{start}}}$ in algorithm 2 contains at least $2^m$ functions. To overcome this problem, the dimension-adaptive procedure could be changed in such a way that we start with a grid which only consists of 1 point, i.e. $l_{\text{start}} = -1$, and refine it once into each direction $j = 1, \ldots, m$. In order to allow for appropriate refinement in the further adaption steps we then have to add all spaces $\tilde{W}_{\mathbf{k}}$ with $\mathbf{k} \leq \mathbf{l} + \mathbf{e}_j$ for each $j \in \{1, \ldots, m\}$ in step 4 of algorithm 2. Thus, in contrast to the original algorithm we would refine into all directions instead of choosing only the ones where $g^{(i)}$ is non-constant. Although these changes lead to a valid refinement heuristic which performs well in many settings, see e.g. [24], they are not directly compatible with the theory of the ANOVA decomposition since we would use information of lower-order ANOVA terms to determine if we add higher-order terms into the representation of $g^{(i)}$, see [21] for details. Another problem for large dimension $m$ is the fast application of the matrices $\mathbf{C}^{(i)}$ in the CG-algorithm in step 2 of algorithm 1 which also introduces a factor of $2^m$ to the computational runtime. To avoid this, one could consider direct matrix assemblation of $\mathbf{C}^{(i)}$ in the case of a moderate number $N$ of basis functions or alternatively change the regularization term such that the resulting matrices allow for a faster matrix-vector multiplication, see e.g. [24]. However, we do not need to consider the above mentioned alterations to algorithm 1 and algorithm 2 as our application at hand is manifold learning where the dimension $m$ is usually small (up to 7) for most real-world problems.

## 4. Application of the dimension-adaptive PML

In this section we present results for the application of the PML method from section 3. We use the $H^1_{\text{mix}}$-semi norm, i.e. we omit the $\mathbf{l} = \mathbf{0}$ term in the sum in (5), to regularize the dimension-adaptive sparse grid PML approach. We start with a toy problem: a two-dimensional S-shaped manifold embedded in three dimensions where we will discuss the performance of the adaptivity. Subsequently, we consider a state-of-the-art engineering problem where we aim to build a generative model for numerical simulation data, which allows the detection of intrinsic effects, and whose quality can be quantified by measuring the reconstruction error. To this end, we compare the results of the dimension-adaptive sparse grid PML to a linear PCA.

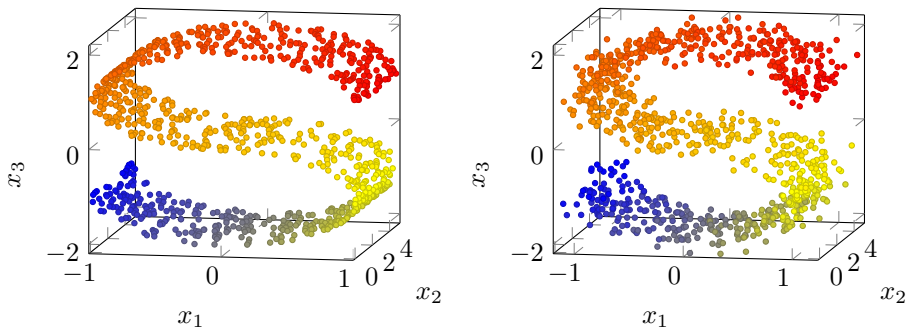



Figure 1: 1000 samples of the S-shaped surface, without (left) and with (right) Gaussian noise.

*4.1. A toy example: S-shaped manifold*

We consider the commonly used two-dimensional S-shaped surface which is given by

$$S(t_1, t_2) := (x_1, x_2, x_3)^T = (\sin(t_1), t_2, \text{sign}(t_1)(\cos(t_1) - 1))^T \tag{16}$$

for $t_1 \in \left[-\frac{3}{2}\pi, \frac{3}{2}\pi\right], t_2 \in [0, 5]$. We create input data $\mathbf{x}_1, \ldots, \mathbf{x}_{1000}$ by drawing 1000 independent, uniformly distributed samples $\mathbf{t}_1, \ldots, \mathbf{t}_{1000}$ from $\left[-\frac{3}{2}\pi, \frac{3}{2}\pi\right] \times [0, 5]$, applying $S$ and adding three-dimensional $\mathcal{N}(\mathbf{0}, 0.01 \cdot \mathbf{I})$ distributed Gaussian noise to each sample. The resulting data can be seen in Figure 1.

We now run the dimension-adaptive sparse grid PML algorithm with $l_{\text{start}} = 2, l_{\text{end}} = 7$. To determine the optimal $\epsilon \in \{0.05, 0.04, 0.03, 0.02, 0.01\}$ and $\lambda \in \{0.1 \cdot 2^{-i}, i = 0, \ldots, 10\}$ we split the input data into a training set of size 900 and a test set of size 100 randomly and run the dimension-adaptive algorithm on the training set for all possible parameter combinations. The respective model $f_{\epsilon,\lambda}$ is then applied to measure the mean squared error on the test data, i.e.

$$\frac{1}{100} \sum_{i=1}^{100} \|\tilde{\mathbf{x}}_i - f_{\epsilon,\lambda}(\tilde{\mathbf{t}}_i)\|_2^2,$$

where $(\tilde{\mathbf{t}}_i, \tilde{\mathbf{x}}_i)$ for $i = 1, \ldots, 100$ are the test data points. To reduce the influence of the specific splitting into training and test data, we repeated this process for five different randomly picked test sets and measured the average of the mean squared errors for each parameter pair. The smallest average error of 0.0202 is obtained for $(\epsilon, \lambda) = (0.01, 0.1 \cdot 2^{-8})$. The results we achieved with the adaptive PML algorithm for these parameters on the whole data set of size 1000 are displayed in Figure 2. As we see, the structure of the initial manifold is perfectly recovered by the algorithm. The overall number of degrees of freedom in the sparse grid representation are 93. To resolve the S-shaped manifold with a similar accuracy by a regular sparse grid PML we have to employ a sparse grid of level four at least. This corresponds to an overall number of basis functions of 339.

Let us assume that a further reduction of the degrees of freedom is desirable. To this end, instead of the parameter pair which achieves the best average MSE on the test data, we consider the pair which leads to the smallest average basis size and for which the average MSE is still smaller than 0.1. We now run the dimension-adaptive sparse grid PML algorithm for the resulting parameters $\lambda = 0.1 \cdot 2^{-4}$ and $\epsilon = 0.02$. The results are shown in Figure 3. Again, the algorithm recovers the structure of the original manifold nicely. This time the reconstruction does not exhaust the whole S-shape and a few points cluster at the boundary of the shape. However, the general order of the data points is kept also in this projection. For this parameter set, our algorithm is able to detect almost perfectly that each component of the S-shaped function depends on only one variable as we see in Figure (4) which shows the three component sparse grids of the solution. We observe that grid points are located almost solely on the $t_1$ or the $t_2$-axes. This means that the corresponding basis functions employ level $-1$ in the other direction, i.e. they are constant along that direction. The effort needed to resolve the first component function is slightly overestimated as it actually only depends on $t_1$. The second component of (16) is linear in $t_2$. To this end, two grid points would suffice to describe it perfectly. Our algorithm only spends one extra grid point here. The third component is resolved perfectly as all employed basis functions only vary with respect to $t_1$. The overall number of basis functions in all three components is 35. Thus, the dimension-adaptive algorithm is able to represent the manifold with only 10% of the amount of the degrees of freedom that the regular sparse grid algorithm needs.

### *4.2. FEM simulations in automotive engineering*

Let us first describe the considered data, which stem from numerical simulations in the automotive industry. Here, finite element simulations of a car crash have become inevitable in order to allow a sophisticated virtual product development process for new car models with respect to passenger safety and to avoid the huge financial expenses of real crash tests as much as possible. In the design process, parameters like plate thickness or material properties are changed by the engineer to inspect the crash behavior of the respective model. This results in a number of different, but related numerical simulations, one for each specific choice of values for the set of parameters. Each simulation is represented by a point in the high-dimensional data space. Since the different simulations follow the same physical laws and observe the same mesh configuration and constraints, it is clear that the variation of the model parameters result in simulation data which, in general, form a nonlinear, low-dimensional structure in the high-dimensional simulation data space. Usually one simulation run takes about half a day on the compute cluster resources available to the engineer. The number of data points consequently stays small and is typically in the range of only a few hundred.

As an example, we consider a frontal crash simulation of a Ford Taurus using

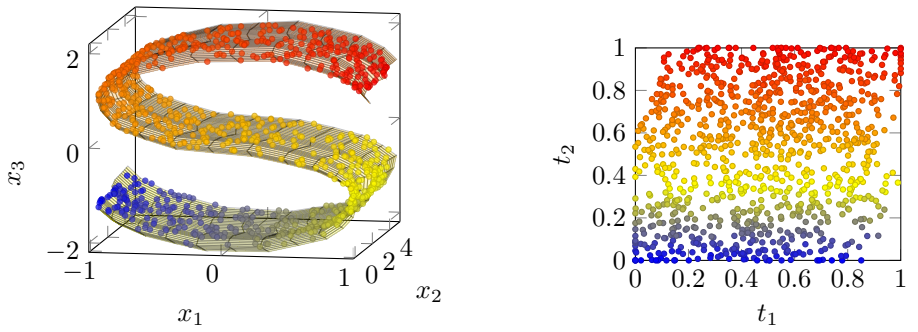



Figure 2: Results of the dimension-adaptive PML for $\epsilon = 0.01, \lambda = 0.1 \cdot 2^{-8}$. Left: the projection of the input data onto the principal manifold computed by the dimension-adaptive PML; Right: the projected input data in reduced (two-dimensional) coordinates.

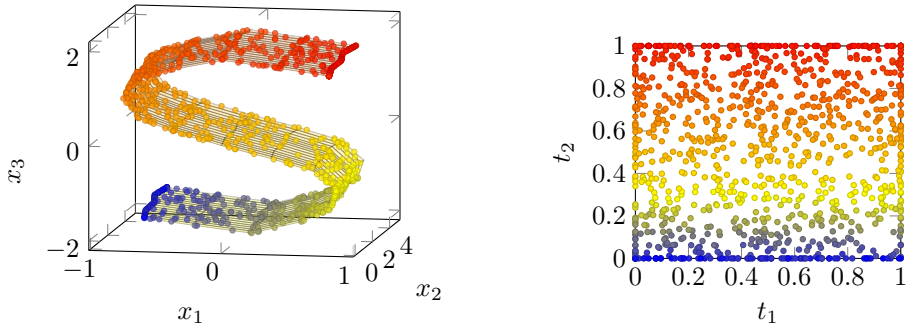



Figure 3: Results of the dimension-adaptive PML for $\epsilon = 0.02, \lambda = 0.1 \cdot 2^{-4}$. Left: the projection of the input data onto the principal manifold computed by the dimension-adaptive PML; Right: the projected input data in reduced (two-dimensional) coordinates.

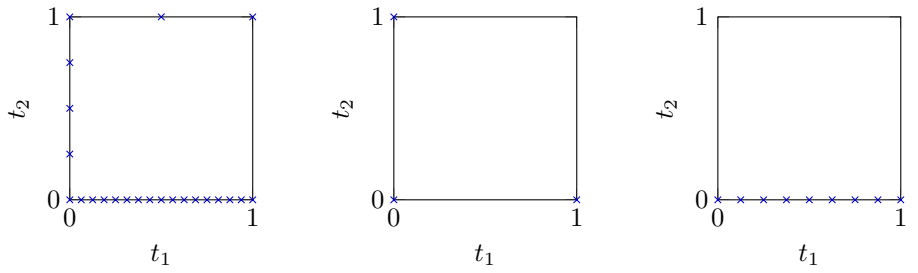



Figure 4: The dimension-adaptive sparse grids for the three component functions of the reconstructed S-shaped surface with 23 grid points for the $x_1$ component (left), 3 grid points for the $x_2$ component (mid) and 9 grid points for the $x_3$ component (right).

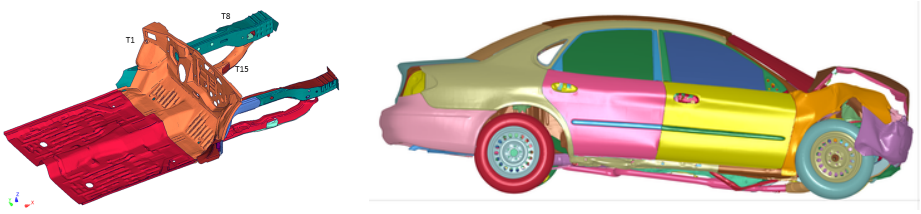



Figure 5: For the experiments the plate thickness of 19 parts (shown on the left) in the front of the car were changed by up to 5% to generate a realistic setup of numerical simulations. A deformed car is shown on the right.

a model from the National Crash Analysis Center[1], which is on-par with current industry discretization resolutions. It involves around 900.000 finite element nodes. With LS-DYNA[2], $n = 274$ crash simulations with 300 time steps were computed at the Fraunhofer Institute SCAI in cooperation with partners from the automotive industry in the project "SIMDATA-NL". Here, 19 underlying parameters, namely the plate thicknesses of the 19 parts (= 15 beams and 4 attached further parts) shown in Figure 5, were varied by up to 5% each. This led to $n$ different crash simulations with $n$ different outcomes. Note that a similar approach for a smaller-sized toy problem can be found in [15].

Ultimately, a car engineer is interested in the variability of the simulations with respect to the design parameter changes and also in the number of different effects (such as bending in different directions) in safety-critical parts of the car model. To this end, we look for the minimal embedding dimension of the high-dimensional simulation data such that the resulting embedding describes the data sufficiently well.

*4.2.1. Input data for the generative algorithms*

In the following we will consider the displacements of the finite element nodes between the time step 150 (when most of the crash impact took place and the car is not yet bouncing back from the obstacle) and the initial time step 0 (where the simulation starts with specified speed for the car movement fixed for all $n$ simulations). For each simulation such a displacement vector now has a dimension of approximately $3 \cdot 900.000$ where the factor 3 appears due to the $x$-,$y$- and $z$-coordinates of the finite element nodes in physical space.

The number of different effects that appear between time step 0 and 150 in the simulations can be understood as the number $m$ of dimensions of the intrinsic representation of the manifold on which the $n$ displacement vectors reside, see [15, 16] for details. A dimensionality reduction of the $3D$-dimensional vectors would result in a manifold which describes the global behavior of the whole car model. Since we are only interested in certain critical components of the

[1] http://www.ncac.gwu.edu/ This model has been developed by the NCAC of The George Washington University under a contract with the FHWA and NHTSA of the US DOT.

[2] http://www.lstc.com/products/ls-dyna

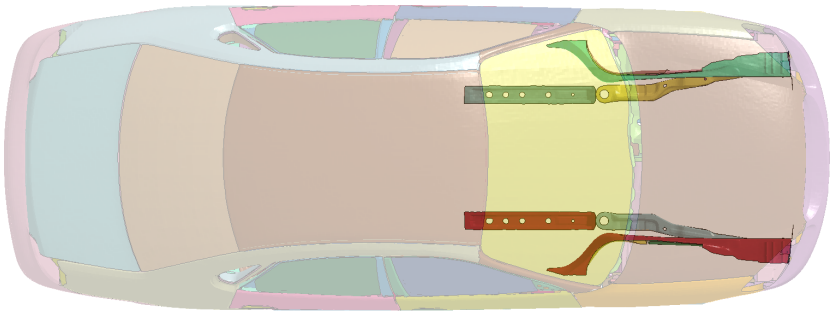


Figure 6: Top view of the Taurus. The 15 relevant beams and their positions in the car are highlighted.

car, we will restrict our analysis to 15 parts of the car, the so-called beams, see Figure 6. These parts absorb most of the impact during the frontal crash. They are designed to have essential influence on the deformations resulting from the crash and therefore on the safety of the passengers. We will denote the set of these 15 parts by $B$. The number of finite element nodes in each part lies between 934 and 4.675.

Let us now be more precise with the definition of our input data. We consider the set $p_{b,1}, \ldots, p_{b,k_b} \in \mathbb{N}$ of $k_b$ nodes in the FE-model belonging to beam $b$ where $b \in B$. We denote their positions in the three dimensional space by $z^i_{b,1}, \ldots, z^i_{b,k_b} \in \mathbb{R}^3$, where $i \in \{1, \ldots, n\}$ is the number of a simulation run. We next define the displacement $\delta^i_{b,j}$ between time step 150 and 0 for one finite element node $p_{b,j}$, $j \in \{1, \ldots, k_b\}$, and the displacement vector $\mathbf{x}_{b,i}$ of dimension $d_b = 3k_b$ for all nodes by

$$\delta^i_{b,j} := z^i_{b,j} - \bar{z}_{b,j} \in \mathbb{R}^3, \quad \mathbf{x}_{b,i} := \left( \left(\delta^i_{b,1}\right)^T, \ldots, \left(\delta^i_{b,k_b}\right)^T \right)^T \in \mathbb{R}^{d_b}, \tag{17}$$

where $\bar{z}_{b,j} \in \mathbb{R}^3$ is the position of node $p_{b,j}$ at time 0. An $\mathbf{x}_{b,i}$ is therefore a point in the high-dimensional space $\mathbb{R}^{d_b}$ and describes the displacement of beam $b \in B$ for the $i$-th numerical simulation, where $i = 1, \ldots, n$. Thus, for each beam $b \in B$ we have translated the car-crash simulation results into $n$ input vectors of dimension $d_b$ for the generative dimensionality reduction method.

*4.2.2. Comparison of generative methods and data pre-processing*

To achieve both, a compact and cost efficient description of the given simulation data set, the high-dimensional displacement vectors $\mathbf{x}_{b,i}$, $i \in \{1, \ldots, n\}$, in Euclidean space of dimension $d_b = 3k_b$ now have to be represented in a lower-dimensional space. To this end, the intrinsic dimension $m_b$ of the $n$ simulation vectors has to be found for each $b \in B$. As mentioned earlier, $m_b$ corresponds to the number of different intrinsic effects in beam $b$ that influence the crash behavior. Now the task arises to find the smallest possible $m_b$ such that the reconstruction error made by the representation of the $d_b$-dimensional simulations as vectors in $[0,1]^{m_b}$ is still sufficiently small.

Note that - in contrast to [15] - the size of our simulation data set at hand is comparable to the size in real-world industry applications. Nevertheless, for each beam $b$ the number of data points ($n = 274$) in the corresponding dimensionality reduction problem is still moderate and therefore a kernel method could also be considered despite its quadratic or even cubic runtime in $n$. However, the ultimate goal in the design-process in automotive industries is to consider several simulation data sets with different car geometries and parameter changes at once. Then, the amount of data points that serve as input for a dimensionality reduction method is in the range of several thousands and therefore untreatable by any data-based discretization such as kernel expansions. In this paper, we focus on the case of a simulation data set with non-varying car model geometry where only plate thicknesses change, i.e. scalar values for which a parametrized design of experiments is easily computable. Note here that the process of creating a parametrized framework for different realistic geometries is very involved, requires sophisticated engineering knowledge and goes beyond the scope of this paper.

To measure the reconstruction error we follow the usual machine learning setup of training and testing data. All of the following steps are performed for each $b \in B$ individually. We split a given data set $\{\mathbf{x}_{b,i} \mid i = 1, \dots, n\} \subset \mathbb{R}^d$ into a training and a testing set. To this end, we pick a set $I_{\text{test}} \subset \{1, \dots, n\}$ of 10 indices at random. The training set is then given by $\mathcal{Y}_b := \{\mathbf{x}_{b,i} |\ i \in I_{\text{train}}\}$, $I_{\text{train}} := \{1, \dots, n\} \setminus I_{\text{test}}$. To reduce the impact of the randomly chosen test data set on the results, we repeated this process five times to result with five different $I^s_{\text{train}}$, $I^s_{\text{test}}$, $\mathcal{Y}^s_b$ for $s = 1, \dots, 5$ and averaged all of our results with respect to these splittings as we will describe in the next section.

Moreover we perform three pre-processing steps of the data, which are done for each $s = 1, \dots, 5$ individually. As an initial step we first center the data (i.e. training and testing data) around the mean of $\mathcal{Y}^s_b$ and then perform a lossless principal component analysis on $\mathcal{Y}^s_b$. The training and testing data is then transformed into a representation with respect to the corresponding eigenvectors. Finally, the transformed training data is used to compute the maps[3] $f_{\text{PCA}}$, $P_{\text{PCA}}$ by the PCA and $f_{\text{PML}}$, $P_{\text{PML}}$ by the dimension-adaptive sparse grid PML method (Algorithm 2) for fixed dimension $m$ and beam $b$. In the following $*$ will stand for PCA or PML, respectively. For reasons of simplicity we will assume that $f_*$ and $P_*$ map directly between $[0, 1]^m$ and $\mathbb{R}^d$, i.e. in our notation we neglect the initial lossless PCA step which first maps $\mathbf{x}_{b,i} \in \mathbb{R}^d$ to $\tilde{\mathbf{x}}_{b,i} \in \mathbb{R}^{\tilde{d}}$ with $m < \tilde{d} = |I^s_{\text{train}}| = 264 < d$. Therefore, we can write

[3] Up to this point we called the resulting functions of any generative method $f$ and $P$. As we are now dealing with both PCA and PML, we denote the functions corresponding to the PCA by $f_{\text{PCA}}$ and $P_{\text{PCA}}$ and the functions corresponding to the PML by $f_{\text{PML}}$ and $P_{\text{PML}}$ respectively. Note that these functions also depend on the dimension $m$, the beam $b$, the splitting index $s$ and - in the case of PML - also on the sparse grid levels $l_{\text{start}}$, $l_{\text{end}}$ and the threshold $\epsilon$. We omit these parameters for ease of notation.

the residual as

$$\mathrm{Res}_*(b,i) := (\mathrm{Id} - f_* \circ P_*)\,(\mathbf{x}_{b,i}) = \mathbf{x}_{b,i} - f_* \circ P_*(\mathbf{x}_{b,i}) \tag{18}$$

for each $i = 1, \ldots, n$ and $b \in B$. In other words, we first map a vector $\mathbf{x}_{b,i}$ from the high-dimensional space $\mathbb{R}^d$ by $P_*$ into the low-dimensional space and then map the result by $f_*$ back to the high-dimensional space. To estimate the quality of both $P_*$ and $f_*$ we consider a suitable norm of (18) which will be explained in the next section.

*4.2.3. Results*

We will now introduce two different error measures for the residual (18).

*Reconstruction error per beam.* First we measure the average relative reconstruction error on beam $b \in B$ on the testing data by

$$\mathrm{BeamError}(b) = \frac{1}{5}\sum_{s=1}^{5} \frac{1}{10} \sum_{i \in I^s_{\mathrm{test}}} \frac{\|\mathbf{x}_{b,i} - f_* \circ P_*(\mathbf{x}_{b,i})\|^2_{\ell_2}}{\|\mathbf{x}_{b,i}\|^2_{\ell_2}}. \tag{19}$$

To simplify the presentation of the quantitative results we give only the average error over all 15 beams, which is be computed by

$$\mathrm{AverageBeamError}(B) = \frac{1}{15}\sum_{b \in B} \mathrm{BeamError}(b). \tag{20}$$

We now compare the results of our sparse grid PML approach with the results of the PCA. We use manifold dimensions $m_b = m \in \{1, 2, 3\}$ for $b \in B$ and the PML parameters[4] $\lambda = 10^{-3}, l_{\mathrm{start}} = 2, l_{\mathrm{end}} = 7, \epsilon = 10^{-2}$. The results can be found in Table 1.

Clearly, the dimension-adaptive sparse grid PML outperforms the standard PCA for each $m$ with respect to reconstruction accuracy. Only on beam 7 for $m = 3$ the PML error is slightly larger than the PCA error. However, both errors are already very small and the regularization of the PML method was probably too strong for this specific beam. Note at this point that the computational costs for a PCA are of course smaller than that for the PML. However, given the fact that we are interested in the number of effects, we are more interested in the smallest possible embedding dimension $m$ for a given error threshold than just a cost efficient representation of the data. Furthermore, it is important to allow for interactive exploration and visualization of the low-dimensional embedded data to aid the engineer in the car design process [25, 26]. To this end, it is necessary to obtain a suitable representation of the data which is as low-dimensional as possible. Let us consider the value $\nu := 10^{-6}$ as an example

[4]Note that we also computed the errors for different parameters $\lambda, l_{\mathrm{start}}, l_{\mathrm{end}}$ and $\epsilon$ but the qualitative results proved to be stable within a certain range and we thus keep the parameters fixed for our presentation here.

Table 1: Beam error (see (19)) and average beam error (see (20)) times $10^{-5}$ of PCA and PML. $m_b = m$ is the same for every beam $b \in B$

| $m$ | 1 | 2 | 3 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| | PCA | | | PML | | |
| Beam 1 | 1.76 | 0.96 | 0.43 | 0.92 | 0.43 | 0.29 |
| Beam 2 | 4.79 | 3.34 | 2.55 | 3.41 | 2.00 | 1.56 |
| Beam 3 | 1.24 | 0.63 | 0.25 | 0.58 | 0.23 | 0.15 |
| Beam 4 | 2.77 | 2.18 | 1.60 | 2.23 | 1.31 | 1.04 |
| Beam 5 | 3.62 | 2.47 | 1.68 | 3.15 | 1.52 | 1.24 |
| Beam 6 | 3.58 | 2.48 | 1.77 | 1.94 | 1.28 | 1.04 |
| Beam 7 | 0.92 | 0.41 | 0.09 | 0.42 | 0.17 | 0.11 |
| Beam 8 | 1.84 | 1.45 | 1.22 | 1.38 | 0.88 | 0.74 |
| Beam 9 | 2.47 | 1.57 | 0.96 | 1.75 | 0.97 | 0.81 |
| Beam 10 | 0.68 | 0.39 | 0.25 | 0.31 | 0.16 | 0.11 |
| Beam 11 | 0.61 | 0.31 | 0.11 | 0.28 | 0.14 | 0.10 |
| Beam 12 | 2.95 | 1.77 | 0.97 | 1.79 | 0.78 | 0.58 |
| Beam 13 | 2.34 | 1.17 | 0.63 | 1.17 | 0.62 | 0.41 |
| Beam 14 | 2.13 | 0.98 | 0.55 | 1.08 | 0.46 | 0.34 |
| Beam 15 | 2.29 | 1.12 | 0.63 | 1.34 | 0.48 | 0.36 |
| **Average** | **2.27** | **1.41** | **0.91** | **1.45** | **0.76** | **0.59** |

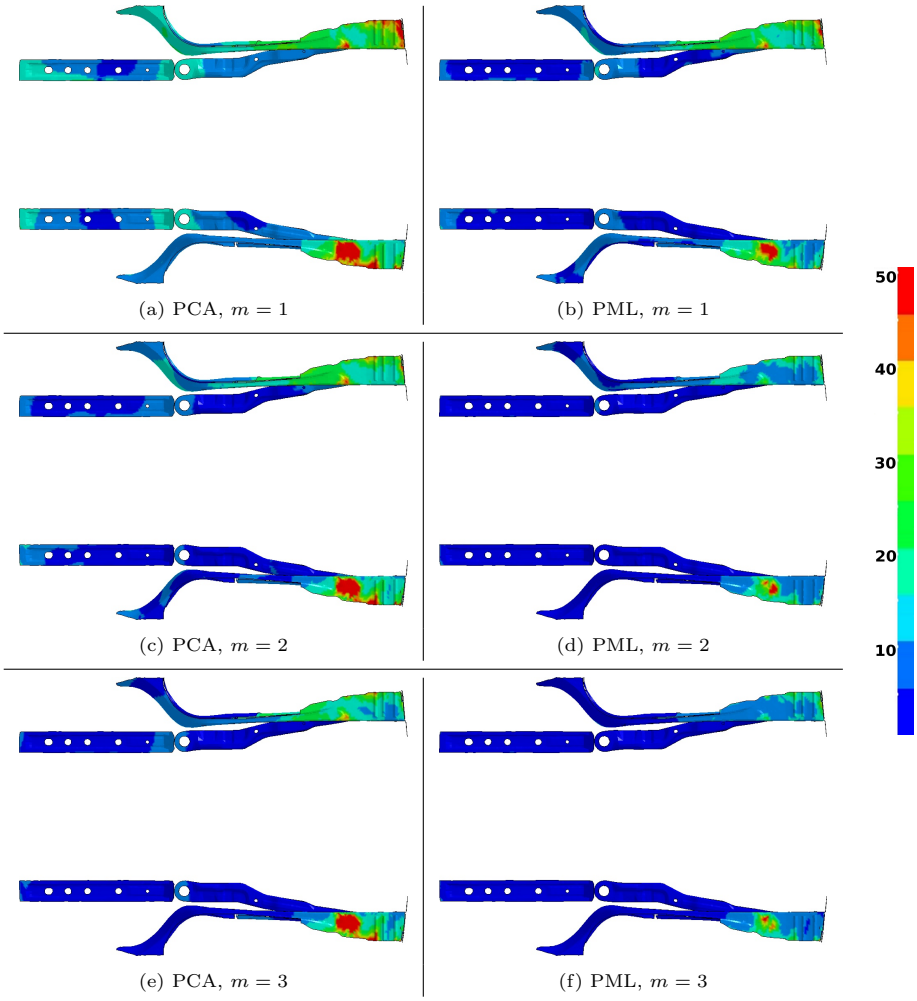
for the overall error threshold. Then, as we can see in Table 1, for $m = 1$ the PCA achieves an error below this threshold for 3 beams and the PML for 5 beams. For $m = 2$ the result is even clearer as the PCA achieves a BeamError which is smaller than $\nu$ on 6 beams while the PML does this on 11 beams. For $m = 3$ the results of PCA (10 BeamErrors below $\nu$) and PML (11 BeamErrors below $\nu$) are almost equal and the benefit of using the nonlinear PML is less evident than for $m = 1, 2$. Since the results for PML do not improve for $m = 3$ compared to $m = 2$ we see that two dimensions are sufficient to describe the main effects in the simulations when using the PML. The linear PCA however needs at least three dimensions to reduce the error to $\nu$ for as many beams as the PML does.

*Reconstruction error per node.* Now we want to compare the reconstruction error per node of a linear PCA approach and that of the PML algorithm. The parameters are the same as in the previous section. In Figure 7, we plot

$$\text{NodeError}(b, j) := \frac{1}{5} \sum_{s=1}^{5} \frac{1}{10} \sum_{i \in I^s_{\text{test}}} \left|\left| (\mathbf{x}_{b,i})_j - (f_* \circ P_*(\mathbf{x}_{b,i}))_j \right|\right|^2_{\ell_2} \tag{21}$$

for every node $p_{b,j}$, $j = 1, \ldots, k_b$, and every beam $b \in B$. Here the index $j$ extracts the 3 spatial coordinates of node $j$. The displacements have been measured in millimeters and therefore (21) is given in $\text{mm}^2$.

Also for the error (21) we observe that the results of the dimension-adaptive sparse grid PML method are significantly better than the results of the PCA in




(a) PCA, $m = 1$ (b) PML, $m = 1$

(c) PCA, $m = 2$ (d) PML, $m = 2$

(e) PCA, $m = 3$ (f) PML, $m = 3$

Figure 7: The pointwise errors (21). The latent space dimensions $m_b = m$ are the same for each $b \in B$ and increase from top to bottom. Values greater than 50 are colored red. The scale runs from $0\text{mm}^2$ (blue) to $50\text{mm}^2$ and larger (red).

the sense that a smaller dimension $m$ suffices to achieve the same reconstruction error as with the PCA. In almost every node the reconstruction error of the PML method with fixed dimension $m$ is at least as good as the reconstruction error of the PCA with dimension $m+1$. Furthermore, in the front part of the beams on the right side, there is a region where the reconstruction error of the PCA method is larger than $50\text{mm}^2$ for each $m = 1, 2, 3$. Here, the PML method is able to resolve this region more accurately already for $m = 1$.

The fact that there is not much difference between $m = 2$ and $m = 3$ for the PML reinforces the claim that two dimensions are sufficient to describe the most important effects with the PML. To confirm the statement that three di-

mensional effects do not seem to be of much importance in the data, we can examine the case $m = 3$ and look at the number of sparse grid points used to learn the manifold. We observe that the average number of basis functions (per component) which are constant in two of the three directions is 379, i.e. these basis functions only vary along one coordinate. The average number of basis functions which is constant in one of the three directions is 926, i.e. these basis functions vary along two of the three coordinates. Finally, the average number of basis functions which is non-constant in every direction is only 235. Therefore, the dimension-adaptive algorithm detected that the contribution of basis functions which are effectively two-dimensional is far more important than the contribution of functions which vary along all of the three coordinates.

Note that there are beams on which the PML with $m = 1$ already suffices to reduce the error to $5\text{mm}^2$ and below in almost every node while there also exist certain local parts on large beams on which the error for the PML with $m = 3$ is still of the size $20\text{mm}^2$ or larger. To overcome this effect to some extent, a more fine-grained analysis of the beams could be considered. To this end, for example an a-priori clustering of the car geometry can help to obtain a segmentation of the car model which reflects the local crash behavior in a better way than the fixed partitioning into design parts, see e.g. [15, 27].

## 5. Conclusion

In this paper, we discussed the idea of generative dimensionality reduction for two examples: Principal Component Analysis (PCA), and Principal Manifold Learning (PML). We explained how the PML method can be discretized in terms of sparse grids and introduced a dimension-adaptive generalization of the algorithm in [12]. Furthermore, we pointed out the relation to the ANOVA decomposition of multivariate functions. As a typical field of application for a generative dimensionality reduction method, we considered a state-of-the-art machine learning problem from the field of automotive engineering. Here, we were able to show that the application of our nonlinear PML algorithm leads to superior results in comparison to the linear PCA. In particular we needed less dimensions for a good reconstruction. Thus, using the PML, the number of relevant effects governing the crash behavior is smaller than for the PCA. This is due to the presence of nonlinear behavior and nonlinear effects in the car crash simulation data set. Ultimately, an engineer would like to optimize design parameters with respect to given safety-relevant target values. To this end, it is crucial to detect the smallest possible embedding dimension of the high-dimensional simulation data and to construct the corresponding embedding as a first step towards such an optimization procedure. Here, already a difference of one in the dimensionality of the low-dimensional space can lead to a significant reduction in the amount of work that needs to be done for an analysis of the car crash behavior during product development. Note finally that the difference in the necessary dimensions for the PCA and the PML is more profound for more complex nonlinear load cases.

**Acknowledgements**

The authors were partially supported by the *Hausdorff Center for Mathematics* in Bonn, by the Sonderforschungsbereich 1060 *The Mathematics of Emergent Effects* funded by the Deutsche Forschungsgemeinschaft and by the project *SIMDATA-NL* funded by the German Federal Ministry of Education and Research. We cordially thank Rodrigo Iza-Teran for generating the car crash simulation data.

## Appendix

Although we mainly deal with principal manifolds in this paper, for reasons of completeness and comparison, we will shortly discuss how the generative topographic mapping [7] fits into our generic generative approach from section 2. For the GTM the density corresponding to $\rho$ is approximated by

$$q_{g,\beta}(\mathbf{x}) := \left(\frac{\beta}{2\pi}\right)^{\frac{d}{2}} \int_{[0,1]^m} \exp\left(-\frac{\beta}{2}\|g(\mathbf{t}) - \mathbf{x}\|_{\ell_2}^2\right) \mathrm{d}\mathbf{t},$$

with a parameter $\beta \in (0, \infty)$ which models the Gaussian noise in the measurements. $q_{g,\beta}$ is a perturbation of a transformed $m$-dimensional uniform distribution. Here, $g \in G = \{\mathbf{W}(\xi_1(\cdot), \ldots, \xi_N(\cdot))^T \mid \mathbf{W} \in \mathbb{R}^{d\times N}\}$ is a vector-valued linear combination of $N$ Gaussian kernel functions $\xi_i$ with fixed centers $\mathbf{c}_i$ for $i = 1, \ldots, N$, i.e. the $j$-th component function $g^{(j)}$ of $g$ can be written as

$$g^{(j)}(\mathbf{t}) = \sum_{i=1}^{N} w_{ji}\xi_i(\mathbf{t}) = \sum_{i=1}^{N} w_{ji} \exp\left(-\frac{\|\mathbf{t} - \mathbf{c}_i\|^2}{\sigma^2}\right) \tag{22}$$

with a fixed variance $\sigma^2$. Usually, the centers $\mathbf{c}_i$ are chosen to form an isotropic grid in $[0, 1]^m$. The correct coupling between the number of basis functions $N$ and the variance parameter $\sigma$ is crucial for a successful dimensionality reduction, see [7] for details. Since the centers $\mathbf{c}_i$ and the variance $\sigma^2$ are fixed, the only remaining degrees of freedom are the entries of $\mathbf{W}$.

It can be shown that the original formulation of the GTM is equivalent to minimizing the so-called cross-entropy between $q_{g,\beta}$ and $\rho$, i.e.

$$\mathcal{A}_\rho(g) = -\inf_{\beta\in(0,\infty)} \int_{\mathbb{R}^d} \log(q_{g,\beta}(\mathbf{x}))\mathrm{d}\rho(\mathbf{x})$$

and therefore $c(g, \mathbf{x}) = -\log(q_{g,\beta}(\mathbf{x}))$, see [10] for details. Note that there is an additional outer minimization over $\beta$. After the substitution of $\rho$ by the empirical measure (2) the functional reads

$$\mathcal{A}_{\mathrm{emp}}(g) = -\inf_{\beta\in(0,\infty)} \frac{1}{n}\sum_{i=1}^{n} \log(q_{g,\beta}(\mathbf{x}_i)).$$

Depending on the a-priori choice of the Gaussian kernels $\xi_i$ the GTM can be run with or without regularization, see [11] for an $H^1_{\text{mix}}$-regularized version of the GTM and [7, 10] for unregularized versions.

To solve the GTM-minimization problem an additional density function $\psi : [0,1]^m \times \mathbb{R}^d \to (0,\infty)$ has to be introduced. Then it can be shown that the minimization of $\mathcal{A}_\rho$ is equivalent to the minimization of the free energy form

$$\min_{\psi,g,\beta} \int_{\mathbb{R}^d} \int_{[0,1]^m} \psi(\mathbf{t},\mathbf{x}) \left( \log(\psi(\mathbf{t},\mathbf{x})) + \frac{\beta}{2} \|g(\mathbf{t}) - \mathbf{x}\|^2_{\ell_2} \right) \mathrm{d}\mathbf{t}\mathrm{d}\rho(\mathbf{x}) - \frac{d}{2} \log\left(\frac{\beta}{2\pi}\right),$$

see [10]. Note that the minimization with respect to $g$ directly translates into a minimization with respect to the matrix $W$ from (22). Similar as for the PML, this problem can be treated analogously to the expectation-maximization algorithm, where the minimization with respect to $\psi$, $g$ and $\beta$ is split into separate minimizations for each argument. This scheme is then iterated until convergence is reached. For the GTM, the projection $P(\mathbf{x}_i)$ is usually defined as the expected value of the so called responsibilities, i.e.

$$P(\mathbf{x}_i) = \sum_{\mathbf{t}_j \in Q} \mathbf{t}_j \frac{\exp\left(-\frac{\beta}{2}\|g(\mathbf{t}_j) - \mathbf{x}_i\|^2_{\ell_2}\right)}{\sum_{\mathbf{t}_k \in Q} \exp\left(-\frac{\beta}{2}\|g(\mathbf{t}_k) - \mathbf{x}_i\|^2_{\ell_2}\right)}.$$

Here the points $\mathbf{t}_j$ form a fixed (usually isotropic) grid $Q$ in $[0,1]^m$ which has a significantly higher resolution than the grid of the Gaussian basis function centers $\mathbf{c}_i$ from (22). In [7], the scaling $|Q| \approx 10^m \cdot N$ is suggested to obtain meaningful results.

## References


[1] F. Cucker, D. Zhou, Learning Theory, Cambridge Monographs on Applied and Computational Mathematics, 2007.

[2] L. Györfi, M. Kohler, A. Krzyżak, H. Walk, A Distribution-Free Theory of Nonparametric Regression, Springer, 2002.

[3] V. Vapnik, Statistical Learning Theory, John Wiley & Sons, 1998.

[4] B. Schölkopf, A. Smola, Learning with Kernels – Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press – Cambridge, Massachusetts, 2002.

[5] J. Lee, M. Verleysen, Nonlinear Dimensionality Reduction, Springer Science, 2007.

[6] S. Gerber, T. Tasdizen, R. Whitaker, Dimensionality reduction and principal surfaces via kernel map manifolds, in: 2009 IEEE 12th International Conference on Computer Vision, 2009, pp. 529–536.

[7] C. Bishop, M. Svensen, C. Williams, GTM: The generative topographic mapping, Neural Computation 10 (1998) 215–234.

[8] A. Smola, S. Mika, B. Schölkopf, R. Williamson, Regularized principal manifolds, Journal of Machine Learning Research 1 (2001) 179–209.

[9] R. Bellman, Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.

[10] M. Griebel, A. Hullmann, Dimensionality reduction of high-dimensional data with a nonlinear principal component aligned generative topographic mapping, SIAM Journal on Scientific Computing 36 (2014) A1027–A1047.

[11] M. Griebel, A. Hullmann, A sparse grid based generative topographic mapping for the dimensionality reduction of high-dimensional data, in: H. Bock, X. Hoang, R. Rannacher, J. Schlöder (Eds.), Modeling, Simulation and Optimization of Complex Processes - HPSC 2012, Springer International Publishing, 2014, pp. 51–62.

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

[13] H.-J. Bungartz, M. Griebel, Sparse grids, Acta Numerica 13 (2004) 147–269.

[14] M. Meywerk, CAE-Methoden in der Fahrzeugtechnik, Springer, 2007.

[15] B. Bohn, J. Garcke, R. Iza-Teran, A. Paprotny, B. Peherstorfer, U. Schepsmeier, C.-A. Thole, Analysis of car crash simulation data with nonlinear machine learning methods, in: Procedia Computer Science, Proceedings of the ICCS 2013, Barcelona, volume 18, 2013, pp. 621–630. Supplementary material `http://garcke.ins.uni-bonn.de/research/pub/simdata_supplementary.pdf`.

[16] R. Iza Teran, Enabling the analysis of finite element simulation bundles, Internat. Jour. for Uncertainty Quantification 4 (2014) 95–110.

[17] C. Micchelli, M. Pontil, On learning vector-valued functions, Neural Computation 17 (2005) 177–204.

[18] K. Pearson, On lines and planes of closest fit to systems of points in space, Philosophical Magazine 2 (1901) 559–572.

[19] A. Ilin, T. Raiko, Practical approaches to principal component analysis in the presence of missing values, Journal of Machine Learning Research 11 (2010) 1957–2000.

[20] H. Bungartz, Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung, Ph.D. thesis, Fakultät für Informatik, Technische Universität München, 1992.

[21] C. Feuersänger, Sparse Grid Methods for Higher Dimensional Approximation, Ph.D. thesis, Institute for Numerical Simulation, University of Bonn, 2010.

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

[23] M. Griebel, Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences, Computing 61 (1998) 151–179.

[24] D. Pflüger, B. Peherstorfer, H.-J. Bungartz, Spatially adaptive sparse grids for high-dimensional data-driven problems, Journal of Complexity 26 (2010) 508–522.

[25] C. Schöne, R. Iza-Teran, J. Garcke, A framework for simulation process management and data mining, in: 1st International Simulation Data and Process Management Conference, Salzburg, Jun 9-12, 2013.

[26] J. Garcke, R. Iza-Teran, Machine learning approaches for repositories of numerical simulation results, in: 10th European LS-DYNA Conference 2015, 2015.

[27] B. Peherstorfer, D. Pflüger, H.-J. Bungartz, Clustering based on density estimation with sparse grids, in: B. Glimm, A. Krüger (Eds.), KI 2012: Advances in Artificial Intelligence, volume 7526 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2012, pp. 131–142.