B. Bohn, J. Garcke, M. Griebel

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

INS Preprint No. 1514

July 2015
A sparse grid based method for generative dimensionality reduction of high-dimensional data

Bastian Bohn\textsuperscript{a}, Jochen Garcke\textsuperscript{a,b}, and Michael Griebel\textsuperscript{a,b}

\textsuperscript{a}Inst. for Numerical Simulation, University of Bonn, Wegelerstr. 6, 53115 Bonn, Germany
\textsuperscript{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 meaningful 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
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]. 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 $f : T \to \mathbb{R}^d$. Therefore, 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 [6], and the regularized principal manifold learning (PML) algorithm, see [7]. Here, the question arises how the mappings $f$ and $P$ are computed. In the original GTM [6] a discretization of $f$ is performed by a full tensor product grid approach. This naturally suffers from the curse of dimensionality, see [8], i.e. the case $m > 3$ is impossible for any meaningful computation. In the PML algorithm [7] 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 [9, 10], and the PML, see [11]. 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 [12].

In this paper we will introduce an adaptive sparse grid PML approach which is an enhancement of [11]. 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 alternating 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 [13]. 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. approx. $10^8$ (time $\times$ nodes). The resulting huge databases of finite element simulations can be used for sophisticated data-driven analysis steps, see [14, 15]. For example, the detection of the number of different effects such as tear or bending behaviors 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 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 three of the most common generative approaches - the PCA, the GTM 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 $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ 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 $M \subset \mathbb{R}^d$ of dimension $m$. Since the treatment of general manifolds is very involved we impose an additional condition on the structure of $M$ and assume that there exists a bijective map $\zeta : [0,1]^m \to M$, i.e. the compact manifold $M$ can be described by a single chart. The problem of determining $M$ now becomes the problem of approximating $\zeta$. Note here that $\rho$ contains more information about $M$ than $\zeta$ does because it also gives a weight to each region on $M$. The fact that this information is lost when we are just given $\zeta$ is resembled by the invariance of $M$ under reparametrization, i.e. any bijective map $\psi : [0,1]^m \to [0,1]^m$ can be employed and $\zeta \circ \psi : [0,1]^m \to M$ 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 $M$ into account.

2.1. General generative approach

A general approach reads

$$\min_{g \in G} A_\rho(g)$$

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

$$A_\rho(g) = \int_{\mathbb{R}^d} c(g, x) d\rho(x). \quad (1)$$

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

$$\frac{1}{n} \sum_{i=1}^{n} \delta_{x_i}. \quad (2)$$

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

$$A_{\text{emp}}(g) = \frac{1}{n} \sum_{i=1}^{n} c(g, x_i) \quad (3)$$

instead of $A_\rho$. Depending on the choice of $G$ the minimization of $A_{\text{emp}}$ 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 $A_{\text{emp}}$ with a regularization parameter $\lambda > 0$. Then, the solution $f$ of a general generative approach is defined by

$$f := \arg \min_{g \in G} A_{\text{emp}}(g) + \lambda S(g). \quad (4)$$

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

$$H = H^k([0,1]^m; \mathbb{R}^d)$$

$$= \left\{ g : [0,1]^m \to \mathbb{R}^d \mid \|g\|_{H^k}^2 := \sum_{|l|_1 \leq k} \int_{[0,1]^m} \|D^l g(t)\|_{L_2}^2 dt < \infty \right\}$$

of order $k \in \mathbb{N}$, see e.g. [2], and the Sobolev-Bochner space

$$H = H_{\text{mix}}^k([0,1]^m; \mathbb{R}^d)$$

$$= \left\{ g : [0,1]^m \to \mathbb{R}^d \mid \|g\|_{H_{\text{mix}}^k}^2 := \sum_{|l|_\infty \leq k} \int_{[0,1]^m} \|D^l g(t)\|_{L_2}^2 dt < \infty \right\} \quad (5)$$
of dominating mixed smoothness of order \( k \in \mathbb{N} \), see e.g. [11]. Here, we denote by \( \mathbf{l} = (l_1, \ldots, l_m) \in \mathbb{N}^m \) a multivariate index and use the multivariate derivative operator \( D^\mathbf{l} := \frac{\partial^{l_1 \ldots l_m}}{\partial x_1^{l_1} \ldots \partial x_m^{l_m}} \).

We will now shortly review how the linear principal component analysis and the nonlinear generative topographic mapping as well as the nonlinear principal manifold learning algorithm fit into this setting.

2.2. Principal Component Analysis

There exist many different interpretations of the principal component analysis [16] in terms of a minimization problem, see e.g. [17]. We will stick to a geometrically motivated variant here. To simplify our notation we assume that \( \mathbb{E}_\rho [\mathbf{x}] = \int_{\mathbb{R}^d} \mathbf{x} \, d\rho(\mathbf{x}) = \mathbf{0} \). Then, \( g \in G = \{ \mathbf{W} | \mathbf{W} \in \mathbb{R}^{d \times m}, \mathbf{W}^T \mathbf{W} = \mathbf{I} \} \) is a linear orthogonal map, i.e. \( g \) is a matrix, and

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

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

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

Note that the assumption of centered data has to be made for this representation as well, 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. [17].

It can easily be shown that the PCA minimization problem is well-posed. Therefore, no regularization is needed. To compute the optimal \( f \), an eigen-decomposition 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. The preimage points \( t_i \), i.e. the images \( P(\mathbf{x}_i) \) of \( \mathbf{x}_i \) under the projection \( P : \mathbb{R}^d \to T \), which are mapped onto \( \mathbf{x}_i, i = 1, \ldots, n \), by \( f \) 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 GTM or the PML is superior.

2.3. Generative Topographic Mapping

Although we will mainly deal with principal manifolds in this paper, we will shortly discuss how the generative topographic mapping [6] fits into our
generic generative approach. For the GTM the density corresponding to $\rho$ is approximated by

$$q_{g,\beta}(x) := \left(\frac{\beta}{2\pi}\right)^{\frac{d}{2}} \int_{[0,1]^m} \exp \left(-\frac{\beta}{2} \|g(t) - x\|_2^2\right) dt,$$

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, $g \in G = \{W(\xi_1(\cdot), \ldots, \xi_N(\cdot))^T \mid W \in \mathbb{R}^{d \times N}\}$ is a vector-valued linear combination of $N$ Gaussian kernel functions $\xi_1, \ldots, \xi_N$ with fixed centers.

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.

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

and therefore $c(g,x) = -\log(q_{g,\beta}(x))$, see [9] 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

$$A_{\text{emp}}(g) = -\inf_{\beta \in (0, \infty)} \frac{1}{n} \sum_{i=1}^n \log(q_{g,\beta}(x_i)).$$

Depending on the a-priori choice of the Gaussian kernels $\xi_i$ the GTM can be run with or without regularization, see [10] for an $H_\text{mix}^1$-regularized version of the GTM and [6, 9] 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 $A_\rho$ is equivalent to the minimization of the free energy form

$$\min_{\psi,g,\beta} \int_{\mathbb{R}^d} \int_{[0,1]^m} \psi(t,x) \left(\log(\psi(t,x)) + \frac{\beta}{2} \|g(t) - x\|_2^2\right) dt d\rho(x) - \frac{d}{2} \log \left(\frac{\beta}{2\pi}\right),$$

see [9]. This problem can be treated by a so-called expectation-maximization (EM) 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 $P(x_i)$ is usually defined as the expected value of the so-called responsibilities, i.e.

$$P(x_i) = \sum_{t_j \in Q} \frac{\exp \left(-\frac{\beta}{2} \|g(t_j) - x_i\|_2^2\right)}{\sum_{t_k \in Q} \exp \left(-\frac{\beta}{2} \|g(t_k) - x_i\|_2^2\right)}.$$

Here the points $t_j$ form a fixed (usually isotropic) grid $Q$ in $[0,1]^m$. 

6
2.4. Principal Manifold Learning

For principal manifold learning we let \( g \in G \) with
\[
G = L_2([0, 1]^m; \mathbb{R}^d) := \left\{ g : [0, 1]^m \to \mathbb{R}^d \mid \|g\|_{L_2}^2 := \int_{\mathbb{R}^d} \|g(t)\|_2^2 \, dt < \infty \right\}.
\]

Analogously to \([3]\), the goal is to find a function \( g \in G \) such that the \( \rho \)-dependent error
\[
A(g) := \int_{\mathbb{R}^d} \inf_{t \in [0, 1]^m} \text{dist}(x, g(t)) \, d\rho(x)
\]
for a fixed distance function \( \text{dist} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+ \) is minimized. A common choice for \( \text{dist} \) is the squared Euclidean norm of the difference of the arguments. Thus, we get
\[
A(g) := \int_{\mathbb{R}^d} \inf_{t \in [0, 1]^m} \|x - g(t)\|_2^2 \, d\rho(x)
\]
and \( c(g, x) = \inf_{t \in [0, 1]^m} \|x - g(t)\|_2^2 \). Note that this is the same minimization functional as for the PCA but the search set \( G \) is much larger here. After substituting \( \rho \) by the empirical measure \((2)\) we obtain
\[
A_{\text{emp}}(g) := \frac{1}{n} \sum_{i=1}^{n} \inf_{t_i \in [0, 1]^m} \|x_i - g(t_i)\|_2^2.
\]

Note that, for the GTM, the preimage \( g^{-1}(A) \) of a subset \( A \subset \mathbb{R}^d \) where \( \rho(A) \) is large, has a large Lebesgue measure in \([0, 1]^m\) due to the structure of \( q_{g,p} \). Thus, the preimages of the data points are all at once implicitly determined by \( q_{g,p} \). However, in the PML setting there is no implicit assumption on \( g \) and the approach is thus more general than for the GTM case. Due to the structure of the cost function \( c \) the preimage \( t_i = P(x_i) \) of a data point \( x_i \) is individually determined by solving the \( m \)-dimensional minimization problem
\[
\inf_{t_i \in [0, 1]^m} \|x_i - g(t_i)\|_2^2.
\]

Since the minimization of \( A_{\text{emp}} \) over \( G \) is an ill-posed problem, a regularization term \( S(g) = \|f\|_{H}^2 \) is added and the overall minimization problem becomes
\[
f = \arg \min_{g \in G} A_{\text{emp}}(g) + \lambda \|g\|_{H}^2
\]
for a fixed regularization parameter \( \lambda \), see also \([4]\). In \([7]\) different choices of reproducing kernel Hilbert spaces (RKHS), see e.g. \([18]\), were introduced for \( H \). In \([11]\) the regularization is done in the mixed Sobolev-Bochner space \( H = H_1^{\text{mix}}([0, 1]^m; \mathbb{R}^d) \) which is also an RKHS.

As for the GTM, the numerical minimization is again performed by an alternating expectation-maximization scheme. In the first step, \( A_{\text{emp}}(g) + \lambda \|g\|_{H}^2 \) is minimized for fixed \( t_i, i = 1, \ldots, n \), i.e. the \( \inf_{t_i \in [0, 1]^m} \) in front of \( \|x_i - \)
$g(t_i) \|_{L_2}^2$ disappears. In [7], the famous representer theorem for reproducing kernel Hilbert spaces is used in this step to obtain a finite kernel expansion

$$g(t) = \sum_{j=1}^{n} K(t_j, t) \alpha_j$$

for $\alpha_j \in \mathbb{R}^d$, $j = 1, \ldots, n$, where $K : H \times H \to \mathbb{R}^{d \times d}$ denotes the matrix-valued reproducing kernel of $H$. We refer to [8, 13] for details on the representer theorem and on matrix-valued reproducing kernels. With (8) 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 (6) is evaluated to obtain updated values for $P(x_i) = t_i$ for every $i = 1, \ldots, n$. These two steps are then iterated until convergence.

3. Discretized PML with adaptive sparse grids

As we showed in [8], the application of the representer theorem leads to a finite expansion of $g$ in terms of the point evaluated kernel functions $K(t_j, t), j = 1, \ldots, n$. Then, it is easy to see that for fixed $t_1, \ldots, t_n$ a minimizer of (7) solves the system of linear equations

$$(\tilde{K} + \lambda I) \bar{\alpha} = \bar{x}$$

with an $n \times n$ kernel block matrix given by $\tilde{K}_{i,j} = K(t_i, t_j) \in \mathbb{R}^{d \times d}$. Here, $I$ denotes the $nd$-dimensional identity matrix, $\bar{\alpha} = (\alpha_1, \ldots, \alpha_n)^T$ is the coefficient vector and $\bar{x} = (x_1, \ldots, x_n)^T$ represents the data vector. Since $K$ 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 $n$.

To overcome this problem, we again use a finite basis representation as in [8], 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

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

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 [9] in [7] we obtain the following alternating minimization approach:

Algorithm 1 - Grid-based PML

1. Initialize $t_i$ by a PCA of the data $x_i, i = 1, \ldots, n$. 

8
2. Solve the minimization problem

\[
\min_{g \in G} \frac{1}{n} \sum_{i=1}^{n} \|x_i - g(t_i)\|_{\ell_2}^2 + \lambda\|g\|_H^2.
\]

If \( H \) is a Hilbert space and \( g \in G \) is defined by (9), the minimization problem can be translated into \( d \) systems of linear equations (\( i = 1, \ldots, d \)):

\[
\left( (B^{(i)})^T B^{(i)} + n\lambda C^{(i)} \right) \vec{\beta}^{(i)} = (B^{(i)})^T \vec{x}
\]

with matrices defined by \( B^{(i)}_{j,k} = \gamma_k^{(i)}(t_j) \), \( C^{(i)}_{k,l} = \langle \gamma_k^{(i)}, \gamma_l^{(i)} \rangle_H \) and the coefficient vector \( \vec{\beta}^{(i)} = (\beta_1^{(i)}, \ldots, \beta_N^{(i)})^T \).

3. Solve the \( m \)-dimensional nonlinear minimization problem (6) to determine the updated \( t_i \) for each \( i = 1, \ldots, n \).

The steps 2 and 3 are then iterated until convergence, see [11] 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 (9).

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

\[
\phi_{l,i}(t) := \prod_{j=1}^{m} \phi_{l_j,i_j}(t_j),
\]

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

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

\[
I_l := \left\{ i \in \mathbb{N}^m \left| \begin{array}{c}
0 \leq i_j \leq 1, \\
1 \leq i_j \leq 2^{l_j} - 1, \quad i_j \text{ odd} \\
0 \quad \text{if } l_j = 0 \\
1 \quad \text{if } l_j > 0
\end{array} \right. \right\}.
\]
Then, we define the hierarchical increment space \( W_l := \text{span} \{ \phi_{l,i} \mid i \in I_l \} \) and the space of piecewise \( m \)-linear functions on the regular (isotropic) full grid of level \( l \in \mathbb{N} \) by

\[
V_l := \bigoplus_{|k|_\infty \leq l} W_k ,
\]

represented in the so-called hierarchical basis

\[
\{ \phi_{k,i} \mid i \in I_k, |k|_\infty \leq l \} .
\]

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

\[
h_l = \sum_{|k|_\infty \leq l} \sum_{i \in I_k} \beta_{k,i} \phi_{k,i}
\]

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

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

(12)

However, since the dimension \(|V_l| = (2^l + 1)^m \) 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

\[
V^s_l := \bigoplus_{k \in \mathbb{N}^m, \theta_m(k) \leq l} W_k ,
\]

(13)

where \( \theta_m(0) := 0 \) and

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

for every other \( k \in \mathbb{N}^m \). Here, \(|k|_1 := \sum_{j=1}^m |k_j| \) 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

\[
h^s_l(t) = \sum_{k \in \mathbb{N}^m, \theta_m(k) \leq l} \sum_{i \in I_k} \beta_{k,i} \phi_{k,i}(t) \in V^s_l
\]

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

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

Thus, compared to (12), 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 \( N = O \left( 2^l \cdot l^{m-1} \right) \) 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 [12] and the references cited therein. Assuming that a constant number of iterations is used, the computational costs of the alternating minimization algorithm with sparse grids is now bounded by \( O \left( N + nl^{m-1} \right) \), see [11] for details.
3.2. Dimension-adaptive sparse grids

Our sparse grid construction has been purely isotropic so far. Thus, every spatial direction is resolved equally 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 [19].

First, we define the altered index sets

\[
\bar{I}_l := \left\{ i \in \mathbb{N}^m \mid \begin{array}{ll}
i_j &= 0, & \text{if } l_j = -1 \\
i_j &= 1, & \text{if } l_j = 0 \\
1 \leq i_j \leq 2^{l_j} - 1, & \text{if } l_j > 0 \\
i_j &\text{ odd if } l_j > 0
\end{array} \right\}
\]

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

\[
\tilde{W}_l := \text{span}\{\phi_{l,i} \mid i \in \bar{I}_l\}
\]

we see that \(W_l\) and \(\tilde{W}_l\) are the same for a multilevel index \(l\) with \(l_j \geq 1\) 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

\[
|l|_1 := |(\max(l_1,0), \ldots, \max(l_d,0))|
\]

we can maintain our previous definition of sparse grid spaces [13] using

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

instead of \(\theta_m(k)\). However, this slightly different choice of the hierarchical increment spaces \(\tilde{W}_l\) results in a direct analogy of the altered discretization to the so-called analysis-of-variance (ANOVA) decomposition, see [20] 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 \(g = (g^{(1)}, \ldots, g^{(d)})^T\) be a vector valued function with components \(g^{(i)}, i = 1, \ldots, d\), given in the form

\[
g^{(i)} = \sum_{k \in K^{(i)}} \sum_{j \in \bar{I}_k} \beta^{(i)}_{k,j} \phi_{k,j},
\]
analogously to (9). Here $\mathbb{R}^{(i)} \subset (\mathbb{N} \cup \{-1\})^m$ is the set of level indices of the grid in direction $i \in \{1, \ldots, d\}$. Then, the error indicator for component $i$ is defined by

$$
\epsilon_k^{(i)} := \max_{j \in \tilde{I}_k} \left\| \beta_k^{(i)} g_k \phi_j \right\|_{L_2([0,1]^m)}.
$$

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 [21]. 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 $g = (g^{(1)}, \ldots, g^{(d)})^T$ to the result of the alternating minimization algorithm (Algorithm 1, steps 1-3) with regular sparse grid ansatz spaces $V^{(i)} := V_{\text{start}}^{(i)}$ of small level $l_{\text{start}}$ for each component $i = 1, \ldots, d$.

2. Initial Compression: For each $i = 1, \ldots, d$: For each $\tilde{W}_l \subset V^{(i)}$ check if $\epsilon_k^{(i)} \leq \epsilon \|g^{(i)}\|_{L_2([0,1]^m)}$ for all $k$ with $k \geq l$ and $\tilde{W}_k \subset V^{(i)}$. If this is the case, remove all these $\tilde{W}_k$ from $V^{(i)}$.

3. Set $g = (g^{(1)}, \ldots, g^{(d)})^T$ to the result of the alternating minimization algorithm (Algorithm 1, steps 1-3) with ansatz spaces $V^{(i)}$ for each component $i = 1, \ldots, d$.

4. Adaption: For each $i = 1, \ldots, d$: For each $\tilde{W}_l \subset V^{(i)}$ check if $\epsilon_k^{(i)} \geq \epsilon \|g^{(i)}\|_{L_2([0,1]^m)}$. If this is the case add the spaces $\tilde{W}_k$ to $V^{(i)}$ for all $k \leq l + e_j$ with all $j \in \{1, \ldots, m\}$ with $l_j \neq -1$. Here, $e_j$ 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 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 [20] 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 $V^{(i)}$ 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. For a more elaborate description of the dimension-adaptive algorithm, its benefits and the relation to the ANOVA decomposition we refer to [19] [20].

**4. Application of the dimension-adaptive PML**

Now, let us apply our dimension-adaptive sparse grid PML method from section 3 to a state-of-the-art engineering problem and compare the results to a linear PCA. 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.
4.1. FEM simulations in automotive engineering

Let us first describe the considered data, which stem from numerical simulations for a problem from 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 results in 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 a model from the National Crash Analysis Center\footnote{http://www.ncac.gwu.edu/} which is on-par with current industry discretization resolutions. It involves around 900,000 finite element nodes. With LS-DYNA\footnote{http://www.lstc.com/products/ls-dyna}, \( n = 274 \) 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” which was funded by the German Federal Ministry of Education and Research (BMBF). Here, 19 underlying parameters, namely the plate thicknesses of the 19 parts (= 15 beams and 4 attached further parts) shown in Figure\footnote{} 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\footnote{[14]}.

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. 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...
Figure 1: 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.

Figure 2: Top view of the Taurus. The 15 relevant beams and their positions in the car are highlighted.

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 [14, 15] 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 car, we will restrict our analysis to 15 parts of the car, the so-called beams, see Figure 2. 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 4675.

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 \( b \in B \). We denote their positions in the three dimensional space by \( z_{b,i}^1, \ldots, z_{b,i}^3 \in \mathbb{R}^3 \), where \( i \in \{1, \ldots, n\} \) is the number of a simulation run. We next define the displacement \( \delta_{b,j} \) between time step 150 and 0 for one finite
element node $p_{b,j}$, $j \in \{1, \ldots, k_b\}$, and the displacement vector $x_{b,i}$ of dimension $d_b = 3k_b$ for all nodes by

$$
\delta_{b,j}^i := z_{b,j}^i - \bar{z}_{b,j} \in \mathbb{R}^3, \quad x_{b,i} := \left( (\delta_{b,1}^i)^T, \ldots, (\delta_{b,k_b}^i)^T \right)^T \in \mathbb{R}^{d_b},
$$

where $\bar{z}_{b,j} \in \mathbb{R}^3$ is the position of node $p_{b,j}$ at time 0. An $x_{b,i}$ 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.

#### 4.3. 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 $x_{b,i}$, $i \in \{1, \ldots, n\}$, 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.

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 $\{x_{b,i} \mid i = 1, \ldots, n\} \subset \mathbb{R}^d$ into a training and a testing set. To this end, we pick a set $I_{\text{test}} \subset \{1, \ldots, n\}$ of 10 indices at random. The training set is then given by $Y_b := \{x_{b,i} \mid i \in I_{\text{train}}\}$, $I_{\text{train}} := \{1, \ldots, n\} \setminus I_{\text{test}}$.

Moreover, we perform three pre-processing steps of the data. As an initial step we first center the data (i.e. training and testing data) around the mean of $Y_b$ and then perform a principal component analysis on $Y_b$ where we keep 99% of the variance. 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 $f_{\text{PCA}}$, $P_{\text{PCA}}$ 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 $\ast$ will stand for PCA or PML respectively. For reasons of simplicity we will assume that $f_{\ast}$ and $P_{\ast}$ map directly between $[0, 1]^m$ and $\mathbb{R}^d$, i.e. in our notation we neglect the initial PCA step which first maps $x_{b,i} \in \mathbb{R}^{d_b}$ to $\tilde{x}_{b,i} \in \mathbb{R}^{\tilde{d}_b}$ with $m < \tilde{d}_b < d$.

---

3Up 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 $b$ 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 the ease of notation.
Table 1: Average beam error (see (18)) of PCA and PML. \( m_b = m \) is the same for every beam \( b \in B \).

<table>
<thead>
<tr>
<th>( m )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCA</td>
<td>( 2.49 \times 10^{-5} )</td>
<td>( 1.51 \times 10^{-5} )</td>
<td>( 8.99 \times 10^{-6} )</td>
<td>( 5.29 \times 10^{-6} )</td>
</tr>
<tr>
<td>PML</td>
<td>( 1.44 \times 10^{-5} )</td>
<td>( 7.22 \times 10^{-6} )</td>
<td>( 5.91 \times 10^{-6} )</td>
<td>( 5.15 \times 10^{-6} )</td>
</tr>
</tbody>
</table>

Therefore, we can write the residual as

\[
\text{Res}^*(b, i) := (\text{Id} - f_* \circ P_*) (x_{b,i}) = x_{b,i} - f_* \circ P_* (x_{b,i})
\]

for each \( i = 1, \ldots, n \) and \( b \in B \). In other words, we first map a vector \( x_{b,i} \) 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 (16) which will be explained in the next section.

4.4. Results

We will now introduce two different error measures for the residual (16).

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

\[
\text{BeamError}(b) = \frac{1}{10} \sum_{i \in I_{test}} \frac{\| x_{b,i} - f_* \circ P_* (x_{b,i}) \|^2_{L^2}}{\| x_{b,i} \|^2_{L^2}}.
\]

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

\[
\text{AverageBeamError}(B) = \frac{1}{15} \sum_{b \in B} \text{BeamError}(b).
\]

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

Clearly, the dimension-adaptive sparse grid PML outperforms the standard PCA with respect to reconstruction accuracy although the difference between the errors shrinks for increasing \( m \). Let us consider the value \( 10^{-6} \) as an example for the overall error threshold. Then, as we can see in Table 1, two dimensions are sufficient to describe the simulations when using the PML. The linear PCA however would need 3 dimensions to describe the effects up to the

---

4Note that we also computed the errors for different parameters \( \lambda, l_{\text{start}}, l_{\text{end}} \) 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.
prescribed accuracy. 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.

**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 3, we plot

$$\text{NodeError}(b, j) := \frac{1}{10} \sum_{i \in I_{test}} \left\| (x_{b,i})_j - (f \circ P)(x_{b,i})_j \right\|_2^2$$

(19)

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 (19) is given in mm$^2$.

Also for the error (19) we observe that the results of the dimension-adaptive sparse grid PML method are significantly better than the results of the PCA in 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 50mm$^2$ for each $m = 1, 2, 3$. Here, the PML method is able to resolve this region more accurately already for $m = 1$.

5. Conclusion

In this paper, we discussed the idea of generative dimensionality reduction for three examples: Principal Component Analysis (PCA), Generative Topographic Mapping (GTM) 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 [11]. 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
Figure 3: The pointwise errors \[b\]. 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 0mm\(^2\) (blue) to 50mm\(^2\) and larger (red).
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.

References


