Visualization of High-dimensional Scalar Functions using Principal Parameterizations

Rafael Ballester-Ripoll, Member, IEEE and Renato Pajarola, Senior Member, IEEE

Abstract—Insightful visualization of multidimensional scalar fields, in particular parameter spaces, is key to many fields in computational science and engineering. We propose a principal component-based approach to visualize such fields that accurately reflects their sensitivity to input parameters. The method performs dimensionality reduction on the vast $L^2$ Hilbert space formed by all possible partial functions (i.e., those defined by fixing one or more input parameters to specific values), which are projected to low-dimensional parameterized manifolds such as 3D curves, surfaces, and ensembles thereof. Our mapping provides a direct geometrical and visual interpretation in terms of Sobol’s celebrated method for variance-based sensitivity analysis. We furthermore contribute a practical realization of the proposed method by means of tensor decomposition, which enables accurate yet interactive integration and multilinear principal component analysis of high-dimensional models.

Index Terms—Parameter space visualization, dimensionality reduction, sensitivity analysis, tensor decomposition

1 INTRODUCTION

Dimensionality reduction is a crucial data processing step to interactively visualize and explore large complex data sets in many visual data analysis methods and systems. Past efforts in the field, including those based on linear projections, are largely tailored for scattered data visualization. Such data sets comprise a finite amount of available data samples that are expected to follow an often unknown pattern; hopefully, an interesting manifold that explains well the sampled data points’ distribution within their high-dimensional domain. Much of the visualization challenge therein consists of learning and revealing this low-dimensional underlying structure.

However, many important problems are actually dense in nature. For example, parameter spaces in engineering and life sciences or econometric models (including black-box systems, simulations, and metamodels) are often high-dimensional and may accept a large or even infinite number of valid parameter combinations. General scattered data approaches are less effective for such situations. Parameter space analysis and multidimensional scalar function visualization have thus become a research field of their own, for which specialized techniques have been introduced. How to perform dimensionality reduction in this scenario, as is customary for the scattered case, is still a major challenge. The usual linear vs. non-linear distinction [34] still applies, but this dense setting brings about some new specific visualization challenges and use cases [35].

In this paper we pursue the goal of meaningfully depicting a model’s dependence with respect to its input variables as well as joint interactions of arbitrary groups of variables. Critically, we also want to understand how the model’s behavior evolves as these variables (and groups thereof) take different specific values. The dimensionality reduction we propose is inspired by the popular Sobol method for sensitivity analysis [37], which partitions a model into orthogonal projected subfunctions that depend on different subsets of variables. It is an example of analysis of variance (ANOVA) method that has long attracted great interest in uncertainty quantification and reliability engineering.

Consider a domain $\Omega = \Omega_1 \times \cdots \times \Omega_N \subset \mathbb{R}^N$ over which $N$ variables $x_1, \ldots, x_N$ move and a multidimensional function $f : \Omega \to \mathbb{R}$. Given some variables of interest $x_{i_1}, \ldots, x_{i_K}$, we introduce the principal parameterization with respect to these variables as a mapping $\pi : \Omega_{i_1} \times \cdots \times \Omega_{i_K} \to \mathbb{R}^D$ that is as similar as possible to the original $f$. We will detail the precise desired notion of similarity in the central sections of this paper. $D$ is the target reduced dimensionality; w.l.o.g. here we will always use $D = 3$ for effective visualization, namely an embedding into a 3D system of coordinates. Hence, for $K = 1$ (one variable of interest) we produce parameterized curves in $\mathbb{R}^3$, for $K = 2$ surfaces (or equivalently, ensembles of curves), for $K = 3$ volumes (or ensembles of surfaces), etc.

As opposed to strategies that focus mainly on certain critical or topologically interesting points (for example, local extrema for Morse-
Smale complexes [21]), our approach is a dimensionality reduction that takes the full model \( f \) and its exponentially large domain \( \Omega \) into consideration, not unlike e.g. the active subspace method [13]. This kind of catch-all approaches are good at conveying context within a model and are thus attractive for the so-called global-to-local user navigation paradigm [35]. However, they have been exploited to a limited degree only due to the curse of dimensionality. For instance, one often needs to compute multidimensional integrals over the entire \( \Omega \). This is an intensively researched and computationally expensive task, and is all the more challenging with the computing time limitations that usually arise in a visualization context. As a second contribution we make the proposed system computationally feasible and real-time responsive by means of a convenient compact representation, namely a tensor decomposition [31]. That way, one can efficiently manipulate dense high-dimensional scalar functions and reconstruct (decompress) regions of interest or derived properties interactively. In principle, the proposed transformation could also be computed using another back-end numerical framework, for instance (quasi-)Monte Carlo or sparse grid-based approximate integration.

**Notation**

Given a function of interest \( f : \Omega \subset \mathbb{R}^N \to \mathbb{R} \), we will refer to its partial functions (or just partials) as the functions that arise by fixing some \( K \geq 1 \) of \( f \)'s variables. For example, \( f^{x_1} \) is an \((N-1)\)-variate function defined as \( f^{x_1} = f(x_1 = \alpha, \ldots) = f(x_1 = \alpha, x_2, \ldots, x_N) \). Whenever we wish to identify one of these subfunctions we will simply write \( f^{x_1} \). For \( K \geq 2 \) we have higher-order partials that fix two or more variables, for example \( f^{x_n,x_m} \).

Tensors in this context are multidimensional data arrays, including vectors, matrices, volumes and so forth. Vectors and matrices are denoted using bold italic lowercase (\( \mathbf{u} \)) and roman uppercase (\( \mathbf{U} \)) respectively, whereas general tensors use calligraphic letters (\( \mathcal{T} \)).

### 2 RELATED WORK

#### 2.1 Parameter Space Visualization

Several visualization frameworks lend themselves well to parameter space analysis. These include continuous parallel coordinates [24], dimensional stacking [42], the HyperSlice [40] and Sliceexplorer [39] tools, etc. Others are highly domain-specific, such as Tuner [38] for brain segmentation or [10] for physically-based simulations in graphics. In terms of the conceptual framework defined by the comprehensive survey by Sedlmair et al. [35], our proposed approach is geared towards global-to-local exploration and places a special emphasis on the sensitivity analysis task. We refer the reader to [35] for a more inclusive literature overview and limit this section to our particular scope and use cases.

In sensitivity-oriented visualization there is a parameter space \( f(x_1, \ldots, x_N) \) whose variables are either freely tunable by the user (usually in a controlled experimental or simulated environment) or naturally governed by a probability density function (PDF). In the latter case, the complexity that is due to the model function \( f \) adds to that of its underlying PDF, which may or may not be known in closed form. If one wishes to place a strong emphasis on the PDF, one may take a set of representative samples distributed accordingly and then simply apply their favorite scattered data visualization technique [30, 34]. Conversely, if only a set of scattered samples is known from an otherwise dense parameter space, a surrogate model may be fitted in order to estimate the true model during interactive visualization. This strategy provides more contextual information than a bare-bones collection of scattered points because a surrogate can be evaluated cheaply at previously uncharted locations within the domain. This enables, among others, derivative-based feature visualization in points’ neighborhoods (gradients, extremal/saddle structure and other local properties) using for example flow-based scatterplots [12] or multiscale lensing [36].

Here we focus on the case where the PDF is of limited interest or even uniform (and especially, when parameters may be set at will). Note that this is a common scenario in sensitivity analysis. In other words, we are concerned with understanding and visualizing the complexity ascribed to the high-dimensional model \( f \) itself, rather than to its parameters’ distribution. A popular approach is to track and visualize \( f \)'s topological properties; watershedding segmentation, Morse-Smale complexes [21] and topological spines [14] belong to this paradigm. Such methods are very sensitive to high-frequencies and irregularities in the model, and they often resort to a smoothing hyperparameter to filter out noise and reveal topological features at different scales. The active subspace method [13], which is very similar to the structure tensor idea for images and volumes [28], is perhaps one of the closest to the present work: it is also based on extracting principal directions of variance within an \( L^2 \) space and inner product. However, while active subspaces arise from uncentered covariances between the model’s gradient components \( \frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_N} \) across the domain \( \Omega \), our method uses the covariance between all partial functions, be it of single variables or variable tuples. In particular, we are not limited solely to global structure. Rather, we can look at variations that occur as one or more input parameters evolve. This is a crucial feature that facilitates effective global-to-local navigation as motivated earlier.

#### 2.2 Sobol Method for Sensitivity Analysis

Several decades after its inception, Sobol’s method [37] remains one of the most prominent for sensitivity analysis of high-dimensional scalar functions [18, 33]. Its main insight is to realize that every variable’s influence can be decomposed into two orthogonal components:

- The additive (or first-order) term \( S_i \) of a variable \( x_i \) measures how strongly the model’s average is affected when \( x_i \) moves. A purely additive variable \( x_i \) means that we can separate it from \( f \) and write
  \[
  f = g(x_n) + h(x_1, \ldots, x_{n-1}, x_{n+1}, \ldots, x_N). \tag{1}
  \]

- The high-order term of a variable \( x_i \) measures its impact on the model that is not attributable to changes of its average. Hence, a purely high-order variable \( x_i \) means that the function’s expected value \( E[f^{x_i}] \) is not affected as \( x_i \) moves.

Often, these two components show up together. Their aggregation is the so-called total effect, denoted as \( S_{i,t} \). The first-order index \( S_i \) gives a precise measure of its variable’s isolated effect, but disregards joint interactions altogether. On the other hand, the total effect \( S_{i,t} \) accounts for such interactions, although it does not identify what orders of interactions are prevalent and how partner variables are interacting. Note that the Sobol components are defined for tuples of variables as well.

Sobol’s method is excellent at robustly capturing joint relationships among interacting groups of variables. For example, while the presence of a strongly additive variable may destroy the local extrema of an otherwise topologically rich function, it will not alter the Sobol relative importances between the remaining variables. However, a drawback of the method is that for each variable (or tuple thereof), its effect over the entire domain is summarized into a scalar quantity only. Thus, it fails at reflecting changes at different specific values of these variables. For instance, a variable may play an overall important role, but only when it takes extreme values, or it may be additive in an interval and high-order in another. The need to convey this important, more granular information calls for a novel visualization-oriented methodology that interplays well with the principles of Sobol’s ANOVA framework.

This is the main motivation behind our method.

#### 2.3 Tensor Metamodelling and Visualization

The dimensionality reduction technique we propose is based on PCA projection in vector spaces of very high dimensionality. To cope with this computational challenge we will use a framework known as tensor decomposition that we briefly review here.

Tensor decompositions approximate arbitrary data tensors (multidimensional arrays) as expansions of simpler, separable tensors that can cope well with the curse of dimensionality [29]. In the context of surrogate modeling, tensors are often defined as discretizations of parameter spaces over regular grids, whereby each tensor entry corresponds
to one particular combination of parameters. For instance, given a simulation depending on $N = 8$ parameters, we may discretize each axis into 64 possible values to yield a tensor with $64^8 \approx 3 \cdot 10^{14}$ elements. It is often possible to handle such massive tensors succinctly using a suitable decomposition, so that they never need to be managed in a raw explicit form. In this paper we use the tensor train (TT) model [31], which in recent years has been used increasingly for surrogate modeling and visualization [5, 23, 41] as well as for sensitivity analysis [6, 7, 9]. Tensor model fitting is an active research field, and multiple options exist nowadays for either a given set of training samples or when new data points can be sampled on demand. Here we follow a precomputed metamodeling paradigm: the surrogate is built offline, taking as many samples as needed to ensure a sufficiently low estimated generalization error. No further samples are acquired during visualization, and in particular no steering is considered.

The techniques we present take advantage of the unique strengths of tensor decompositions and, in particular, the TT model. Classical regressors such as Gaussian processes (kriging) or radial basis functions are popular for surrogate modeling in certain cases, e.g. when the available ground-truth samples are fixed and very limited in number. Nonetheless, they are less adequate for the kind of multidimensional integration and multilinear PCA projection required by the proposed visualization. The general idea of using compressed tensor coefficients as features for model (post-)processing and analysis is not new [25, 43], and PCA is a long-established framework that has been used in the past for low-dimensional parameterization (prominently, trajectory curves over time [11, 19]). However, the proposed sensitivity-aware dimensionality reduction for dense, high-parametric models is new. We cover it in detail over the next sections.

3 PROPOSED DIMENSIONALITY REDUCTION

Consider an $N$-dimensional parameter space represented as a function $f : \Omega \to \mathbb{R}$. In the multivalued case $f : \Omega \to \mathbb{R}^M$ one can handle each output $1, \ldots, M$ separately or, if a joint analysis for all outputs is desired, reduce the problem to the single-valued version by stacking all outputs to form an extra dimension: $f : \Omega \times \{1, \ldots, M\} \to \mathbb{R}$. For simplicity, let us also assume that all inputs are continuous and scaled to $[0, 1]$ (alternative cases work analogously).

3.1 Single Variable Case

For the sake of clarity, let us start with $K = 1$, i.e. there is only one variable of interest $x_n$. Our goal is to understand its effect on the high-dimensional function $f$, or, in other words, the relationship between the $(N-1)$-dimensional partial functions $f^{x_n} = f(\ldots, x_n = \alpha, \ldots)$ as $\alpha$ thus $x_n$ moves between 0 and 1. Of course, each partial may have a structure (almost) as complex as the original $f$, so their joint behavior is just as potentially intricate and challenging.

We propose to consider the $L^2$ space $\mathcal{F}$ of all functions that map $[0,1]^{N-1}$ to $\mathbb{R}$. Clearly, $f^{x_n} \in \mathcal{F}$ for every $0 \leq x_n \leq 1$. Let us summarize this collection of partial functions as a parameterized curve, i.e. map each to a point in $\mathbb{R}^3$. By starting after averaging each partial over its free variables to get a single scalar, i.e. computing the global average of $f^{x_n}$ for any fixed $x_n$. To this end, we consider the projection function $f_n$:

$$f_n(x_1, \ldots, x_N) := \mathbb{E}[f^{x_n}] = \int_{[0,1]^{N-1}} f(x_1, \ldots, x_N) \, dx_1 \cdots dx_{n-1} dx_{n+1} \cdots dx_N. \quad (2)$$

The projection $f_n$ is constant along all variables but $x_n$. It captures the aggregated additive behavior as per variable $x_n$, and hence determines the first-order Sobol index (Sec. 2.2), which is defined as $S_n := \text{Var}[f_n]/\text{Var}[f]$. Such an axis-aligned projection has already some visualization power and has been used in prior literature, but obviously still gives limited information on $f^{x_n}$’s inner workings as $x_n$ varies. The missing information is contained in the corrected function $f^{x_n} := f - f_n$, which is the main idea driving Sobol’s ANOVA method. It removes the additive component that is due to $x_n$ and gives rise to the total index: $S_n := \text{Var}[f^{x_n}]/\text{Var}[f] + S_n$.

We now extend this idea to our setting, which is concerned with partial functions $f^{x_n}$, and split each of those partials similarly as

$$f^{x_n} = f^{x_n}_n + f^{x_n}_n.$$

For any $x_n$, these two components are furthermore orthogonal with one another w.r.t. the $L^2$ inner product:

$$\int f^{x_n}_n f^{x_n}_n = \mathbb{E}[(f^{x_n}_n)^2] \cdot (f^{x_n} - \mathbb{E}[f^{x_n}]) \cdot (f^{x_n}_n - \mathbb{E}[f^{x_n}_n]) = 0 \text{ for all } 0 \leq x_n \leq 1.$$

This motivates us to represent the original space of partial functions in terms of two orthogonal subspaces and brings us to the core part of the proposed mapping. In order to facilitate visualization we will decompose each $f^{x_n}$ in terms of a coordinate triplet:

- One coefficient $\pi_x(x_n)$ that encodes the projection of each $f^{x_n}$ onto the subspace spanned by $f^{x_n}_n$’s partials along $x_n$, i.e. $\{f^{x_n}_n | 0 \leq x_n \leq 1\}$. As argued earlier, each of those partials is a constant function. Therefore, they are all multiple of one another, and thus they form a subspace of dimension 1 within $\mathcal{F}$. Hence, one coordinate is enough to encode the projection exactly, and it is precisely the expected value $\mathbb{E}[f^{x_n}_n]$ as per Eq. 2.

- Two coefficients $\pi_y(x_n)$ and $\pi_z(x_n)$ that encode the projection of each $f^{x_n}$ onto the subspace spanned by $f^{x_n}_n$’s partials, $\{f^{x_n}_n | 0 \leq x_n \leq 1\}$. Since this subspace’s dimension is in general infinite, we resort to a truncated basis expansion. We choose an optimal basis in the $L^2$ sense, namely the two leading eigenfunctions of the Karhunen-Loeve expansion (KLE) for the pairwise covariance function $\text{Cov}(\alpha, \beta) := \mathbb{E}[(f^{x_n}_n + \alpha)(f^{x_n}_n + \beta)]$ for all $0 \leq \alpha, \beta \leq 1$.

In summary, each individual $f^{x_n}$ is reduced to a point in 3D $(\pi_x(x_n), \pi_y(x_n), \pi_z(x_n))$ and thus the set of all $x_n \in [0,1]$ is mapped to a parameterized curve in $\mathbb{R}^3$. This is a subspace-constrained KLE: we force a specific vector to appear in the basis, and want to find others that best summarize the remaining subspace that is orthogonal to that vector. The fixed vector gathers absolute information, since coordinate $\pi_x$ equals the partial function’s mean. On the other hand, the two other vectors to be sought encode relative information, as absolute positions $(\pi_y, \pi_z)$ on the $yz$-plane are not directly interpretable, but the distances between points are. Although the fixed vector is not generally one of the KLE’s leading eigenfunctions, it is still a reasonable basis choice in $L^2$ terms and it often captures a significant amount of the model’s variance.

3.2 Multivariate Case

We have just mapped a single variable’s corresponding collection of partial functions onto a 3D curve ($K = 1$-dimensional manifold). The higher-dimensional case ($K \geq 2$) follows naturally from that. The main difference is that we have now collections that are indexed by two or more variables, and we handle higher-order partial functions, e.g. $f^{x_n,x_m}$ for $K = 2$, that arise from fixing several variables and are $(N-K)$-dimensional. As a result we no longer obtain parameterized curves but higher-order manifolds (surfaces, volumes, etc.) that are parameterized by triplets of multidimensional functions $\pi_x, \pi_y, \pi_z : [0, 1]^K \to \mathbb{R}$.

4 PRACTICAL ALGORITHM

4.1 Algorithm Outline

For the ease of exposition, we assume that the $K$ variables of interest are the first $1, \ldots, K$. The dimensionality reduction that we motivated in Sec. 3 boils down to a three-step processing pipeline:
Stage A: The within-mean of each partial function of \( f \) is computed (to be used as \( x \)-coordinate during visualization) and subtracted from the original. This way we derive a new function \( f_{1...K} \) whose partials are zero-centered.

Stage B: The cross-mean of \( f_{1...K} \) (i.e., its average over target variables \( 1, ..., K \)) is subtracted from each of its elements to yield a new collection \( f_{1...K} \). By doing so we are shifting the collection’s origin of coordinates to its barycenter as is often done in PCA to achieve a more meaningful projection.

Stage C: We compute the two leading eigenfunctions of \( [0, 1]^K \times [0, 1] \rightarrow \mathbb{R} \) covariance function that maps every possible pair of elements of \( f_{1...K} \) to their inner product. For each partial, its coefficients in terms of this basis define its embedding on the \( yz \)-plane.

See Fig. 2 for an example of the within- and cross-means for dimension \( N = 3 \) and target variable \( x_1 \). Note that Stages A and B are orthogonal to each other: the within-mean (Stage A) is computed as an average over the non-target variables, whereas the cross-mean (Stage B) is an average over the remaining target variables.

![Fig. 2: Isosurface renderings for an example 3D function](image)

Fig. 3 illustrates the full three-stage pipeline for a toy example of a 4D scattered data set displayed in parallel coordinates.

### 4.2 Discretization

Given an infinite collection of functions, each of which is an element of an infinite-dimensional vector space, how can we find a good truncated basis for it? As a first step, let us work on a discretely sampled version of the problem, whereby we quantize the collection into a number of representative bins. This makes the procedure numerically tractable, namely via an eigensolver, so that we can approximate the original function space’s KLE. Essentially, given a parameter space with \( N \) inputs, w.l.o.g. we discretize the input function \( f \) along each axis using \( I \) bins to yield an \( N \)-dimensional data tensor \( T \) of size \( I^N \). This way, partial functions become hypersurfaces of tensors. Instead of a continuous 3D parameterization \((x_1, y_1, z_1)\), we then seek three corresponding discrete (coordinate) tensors \((X, Y, Z)\), each of size \( I^K \)

For example, for \( I = 64 \) and \( K = 3 \) we will obtain a triplet of \( 64^3 \) coordinate volumes that can be visualized as e.g. an ensemble of 64 3D surfaces, each represented as a quadmesh of \( 64^2 \) vertices. See Alg. 1 for a practical, discretized version of the proposed algorithm.

Alg. 1 relies on a range of powerful tensor operations: computing means along several axes, element-wise subtracting tensors, computing a large covariance matrix \( C \) among many tensor slices, and eigen-decomposition of that matrix. In particular, the entries of \( C \) require very large-scale dot products that are non-trivial to compute. A classical method to estimate such products is Monte Carlo (MC) integration, which is simple but costly as it converges slowly [26]. In addition, for higher values \( K \), \( C \) may grow to become a massive dense matrix with billions of entries, so its eigen-decomposition poses a challenge on its own.

Given an infinite collection of functions, each of which is an element of an infinite-dimensional vector space, how can we find a good truncated basis for it? As a first step, let us work on a discretely sampled version of the problem, whereby we quantize the collection into a number of representative bins. This makes the procedure numerically tractable, namely via an eigensolver, so that we can approximate the original function space’s KLE. Essentially, given a parameter space with \( N \) inputs, w.l.o.g. we discretize the input function \( f \) along each axis using \( I \) bins to yield an \( N \)-dimensional data tensor \( T \) of size \( I^N \). This way, partial functions become hypersurfaces of tensors. Instead of a continuous 3D parameterization \((x_1, y_1, z_1)\), we then seek three corresponding discrete (coordinate) tensors \((X, Y, Z)\), each of size \( I^K \)

For example, for \( I = 64 \) and \( K = 3 \) we will obtain a triplet of \( 64^3 \) coordinate volumes that can be visualized as e.g. an ensemble of 64 3D surfaces, each represented as a quadmesh of \( 64^2 \) vertices. See Alg. 1 for a practical, discretized version of the proposed algorithm.

### 4.3 Tensor Decomposition Algorithm

We propose to use tensor decomposition, and in particular the tensor train (TT) model, to represent and work with our discretized parameter space. It is an extremely convenient format for the problem at hand because (a) it can compress a full parameter space very compactly, circumventing the curse of dimensionality; (b) it allows for very fast multidimensional integration; and (c) it can encode the covariance matrix needed in a TT-compressed form of its own, from which principal components are then easy to extract. The TT format approximates each entry \( 1 \leq i_1, ..., i_N \leq I \) of our discretized tensor \( T \) as a product of matrices:

\[
T[i_1, \ldots, i_N] \approx T^{(1)}[i_1] \cdots T^{(N)}[i_N]
\]

where every \( T^{(n)} \) is a 3D tensor known as core, namely an array of \( I \) matrices indexed by \( i_n \); \( T^{(1)} \) and \( T^{(N)} \) contain row and column vectors, respectively. In other words, the model’s behavior for any dimension \( n \) and any value of \( i_n \) is completely governed by the elements in its corresponding matrix \( T^{(n)}[i_n] \). The TT representation allows us to perform all required operations efficiently, provided that the input parameter space itself is given discretized and in the TT format. For instance, to compute a function’s average along axes \( 1, ..., K \) one only needs to compute averages of the corresponding
Dimensionality reduction

The truncated PCA yields the projection \( \pi \) that, by means of a reduced orthonormal basis, best preserves a given collection of vectors in the \( L^2 \) sense:

\[
\arg \min_{\pi} \sum_{u} \| u - \pi^{-1}(\pi(u)) \|^2,
\]

where \( \pi^{-1}(\cdot) \) is the expansion back into the original high-dimensional space using the same basis. This means that distances between vectors are also preserved well:

\[
\| v - u \| \approx \| \pi(v) - \pi(u) \|
\]

and likewise relative distance changes:

\[
\| w - v \| - \| v - u \| \approx \| \pi(w) - \pi(v) \| - \| \pi(v) - \pi(u) \|
\]

and similarly for any level of repeated subtraction. In other words, notions like speed of change or acceleration tend to be reflected well in the projected space. This has important and desirable consequences from a visualization point of view. So, for example, if a collection of partial functions \( f^x_n \) for \( 0 \leq x_n \leq 1 \) follows a straight line in \( \mathcal{F} \), then its projection \( \pi : [0,1] \rightarrow \mathbb{R}^3 \) will necessarily evolve in a linear fashion:

\[
\pi(x_n) = \pi(0) + x_n \cdot (\pi(x_n) - \pi(0)), \quad 0 \leq x_n \leq 1
\]

which is a straight line connecting 3D points \( \pi(0) \) and \( \pi(1) \). Furthermore, a curved line hints at a sequence of vectors that changes non-linearly. Sudden changes in the curve mirror sudden changes also in the original high-dimensional \( \mathcal{F} \) (as intuitively expected), periodic behavior is mapped to rings, etc. Also, note that the global mean of the model \( \mathbb{E}[f] \) coincides with the barycenter of any principal parameterization in 3D, which has the form \( (\mathbb{E}[f], 0, 0) \). The cosine similarity \( \frac{\pi \cdot v}{\| \pi \| \| v \|} \) is approximately preserved as well, and is rendered in 3D as angles between vectors. To help gain intuition and demonstrate the expressive power of the proposed parameterizations, we show a number of examples in Figs. 4 and 5. All were taken from the models listed later in Sec. 6.1.

Certainly, since we are projecting vectors of huge dimensionality using three basis elements only, much of the detail along the unimportant variables is likely to be smoothed out. On the other hand, because the manifold has as many dimensions as there are variables of interest, the trajectories of these variables are captured well and can be tracked visually in full detail. For example, any sharp corner or feature in a curve traces back unambiguously to one specific value of its variable \( x_n \). We argue this is a strength of the proposed method: it is able to abstract complex spaces over many dimensions while still retaining full resolution along a few selected target variables.

5.2 Global Sensitivity Analysis

As outlined in the introduction, there are several interesting connections relating our projection \( \pi \) with the Sobol indices [37] (Sec. 2).

Recall that for the Sobol index of the \( n \)-th variable we have \( S_n \propto \text{Var}[f_n] \), whereas the total Sobol index \( S_n^T \) accounts for both the additive and high-order effects: \( S_n^T \propto \text{Var}[f_n] + \text{Var}[f_{-n}] \). Furthermore, we have that \( \text{Var}[f_n] = \| f_n \|^2 \| \pi_x \|^2 \) (exact projection on the \( x \)-axis) and \( \text{Var}[f_{-n}] = \| f_{-n} \|^2 \| (\pi_y, \pi_z) \|^2 \) (approximately; it is the projection on the \( yz \)-plane using a truncated expansion). Therefore,

- The curve’s evolution along the \( x \)-axis mirrors the corresponding partial’s mean value \( \mathbb{E}[f^x_n] \) as \( x_n \) moves, and \( S_n \) is proportional to the curve’s variance along that axis. In other words, by tracking the curve’s \( \pi_x \) coordinate we can infer the overall additive behavior of our \( N \)-dimensional model as that variable progresses. In particular, the correlation between \( x_n \) and the model output equals that between \( x_n \) and the curve’s \( \pi_x \):

\[
\rho(x_n, f) = \rho(x_n, \mathbb{E}[f^x_n]) = \rho(x_n, \pi_x(x_n)),
\]

Thus, curves that generally move towards the right of the \( x \)-axis indicate a positive correlation, and vice versa. Any purely additive variable \( x_n \) (i.e. \( S_n = S_n^A \)) will not use the \( yz \)-plane, i.e. is mapped to a line segment that is perfectly aligned to the \( x \)-axis. See also examples in Fig. 4a,b,c,d,g.

- The higher-order component measures exclusively the influence due to the interplay between the variable of interest and the rest of variables. This interaction is reflected as variations along the \( yz \)-plane. The manifold’s second-order moment on that plane, i.e. its summed squared distance to the \( x \)-axis, is proportional to \( S_n^A - S_n \). Any purely high-order variable \( x_n \) (i.e. \( S_n = 0 \)) does not move along the \( x \)-axis, but only on the \( yz \)-plane, e.g. as in Fig. 4h.

- The total index \( S_n^T \) measures both effects and is approximately proportional to the total second spatial moment (that is, including all \( X, Y, \) and \( Z \) axes) of the principal parameterization. In other words, the more spread out the parameterization is, the more global influence its variable has on the model. Conversely, a tuple of irrelevant variables will be collapsed into a point.

In a nutshell, additive effects make the curve move along the \( x \)-axis, while high-order interactions pull it away in various ways.

5.3 Local Sensitivity Analysis

The differential structure around a point of the manifold tells us about the local behavior of the variables of interest when they take some specific value. Consider the derivatives \( \frac{dx}{dx_n} \) and \( \frac{dx}{dx_m} \) of a principal surface at a certain point \( x_n, x_m \). The angle between them is a proxy of the similarity between the local effects caused by small increments
Fig. 4: The proposed 1D and 2D principal parameterizations capture a wide range of single- and multiple-effect patterns as well as global/local properties. For example, (f) shows a non-differentiable point due to a min function, whereas (l) demonstrates a variable that is not influential when another variable is small, but is quite influential otherwise. The x-axis is depicted in each case as a black arrow and marks the model’s average over all abstracted variables for any specific values of the parameterized target variables (color arrows and surfaces). Rows and columns in the surfaces’ checkerboard patterns reflect variable isolines, i.e. the path followed when a variable moves and the other one is fixed.

Fig. 5: We can display 3D trajectory curves so as to track individual points in a parameterized surface as a third variable moves.

6 RESULTS

We used Python 3.5, Qt and OpenGL to implement the proposed algorithm. All visual results are displayed using a range of custom widgets and diagrams. Our visualization front-end uses PyQtGraph [1], a 2D/3D graphics and GUI library for scientific Python, as well as PyQt and PyOpenGL which enable Python interfacing with Qt and OpenGL, respectively. Besides NumPy, our numerical back-end exploits the auxiliary libraries ttpy [2] and ttreuces [3] for tensor train manipulation.

6.1 Models Tested

We considered five analytical models of varying complexity and dimensionality:

- The Nassau County [8]: a 6D voting system where any coalition of political agents can pass a motion if and only if their combined votes reach a simple majority. It is a naturally discrete problem; we model all 64 possible coalitions as a small tensor of size $2^6$.
- The Cell Cycle [20]: a 4D multivalued time-dependent model, which builds on an earlier model by Goldbeter [22] and is discretized as a tensor of size $32 \times 32 \times 32 \times 64 \times 5$. The second-to-last dimension is the time $t$, while the last dimension indexes the five outputs of the model.
- The Damped Oscillator [17]: an 8D physical model measuring the peak force in a spring system connecting two oscillating

\[ \frac{\partial^2 \pi}{\partial x_n \partial x_m} = 0 \]
\[ \frac{\partial^2 \pi}{\partial x_n \partial x_m} \neq 0 \]

\[ \frac{\partial \pi}{\partial x_n} \approx 0 \]
\[ \frac{\partial \pi}{\partial x_n} \approx \| \frac{\partial \pi}{\partial x_n} \| \cdot \| \frac{\partial \pi}{\partial x_m} \| \]

\[ \frac{\partial \pi}{\partial x_n, \partial x_m} \]

\[ \frac{\partial \pi}{\partial x_n} \]

\[ \frac{\partial \pi}{\partial x_n, \partial x_m} \]

\[ \frac{\partial \pi}{\partial x_n} \]

\[ \frac{\partial \pi}{\partial x_n, \partial x_m} \]
masses. This and the following models were discretized using 64 bins per dimension.

- The Robot Arm [4]: an 8D model measuring the distance attained by a 4-segment robot arm. It includes an irrelevant variable and three periodic variables (elbow joint angles moving between 0 and $2\pi$).

- The Ebola Spread [15]: an 8D model predicting Ebola virus infection rate in Liberia and Sierra Leone, based on statistics from the 2014 outbreak. Two variables can be influenced to decrease the infection rate in Liberia and Sierra Leone, based on statistics from the 2014 outbreak.

We built TT metamodels for those use cases by means of the cross-approximation adaptive sampling algorithm [32], in particular the implementation released in the tippy Python toolbox [2]. It is an incremental sampling technique that uses a growing validation set at each sample acquisition step; the process is stopped as soon as the relative error w.r.t. the current prediction is below a user-defined threshold: $\|\hat{\mathbf{x}} - \mathbf{x}\| / \|\mathbf{x}\| \leq \epsilon$. We used $\epsilon := 10^{-4}$ in all cases.

### 6.2 Curve Array

The simplest kind of diagram we implemented is a collection of $N$ static parameterized curves in 3D, one per input variable $x_1, \ldots, x_N$. This basic visualization conveys the summarized global structure of the model as its inputs move individually: among others, it reflects correlations between the model’s output and each of its variables. Note that inter-variable interactions are only given in abstracted form as curve movements on the $yz$-plane.

Let us start with the small Nassau County example in order to gain intuition on the proposed method of operation. This model considers 6 county districts with different voting weights. In order for any political motion to succeed, it must be backed by a coalition of districts that reach a vote majority. In political sciences and game theory, the Banzhaf power index [8] is often used for this kind of settings to assess the true influence of individual agents (which may be far from their nominal voting weight). For each agent, its index is defined as the fraction of all possible winning coalitions in which it is a necessary member, i.e. the coalition reaches a majority but would not do so without that agent. Since each district party can either be or not be in a coalition, we model the problem using a binary variable for each. The domain is thus $\Omega = \{0, 1\}^6$. We arrange all possible coalitions in a small tensor of size $2^6 = 64$, where each entry is 1 if the corresponding coalition would reach majority and 0 otherwise. We then compute an array of principal curves using our method. For each district, its curve has only two points, and is thus a line segment that connects them. We have found the variance of these two endpoints (i.e. the segment’s squared length) to be proportional to their corresponding district’s Banzhaf power index; see also Fig. 8.

As a second, more complex example we consider a 4D, 5-valued system of ordinary differential equations (ODE), namely the Cell Cycle. It is a time-dependent system modeling protein concentrations during cell division, hence the temporal axis $t$ is particularly important as an explanatory variable. We select $t$ as our variable of interest and gather all 5 outputs of the ODE into an extra dimension for a joint analysis. This way, we enforce their 5 principal curves to share the same 3D system of projected coordinates. We have furthermore added a slider to govern any of the 4 parameters separately and thus add one extra degree of user interaction. The slider can be adjusted in real time and prompts an immediate update on the 5 curves displayed. Parameter $K_6$ was found to have a strong effect on the speed of change of several outputs; see Fig. 9 for some example renderings.

Note that several quantities (Fig. 9a) exhibit an oscillatory pattern as is explained by their interaction with each other in the ODE. They differ in their asymptotic behavior: they oscillate around different concentration levels, and the attractor point is in each case differently affected by $K_6$. Besides compactly showing the rate of evolution of each quantity as time progresses, the proposed visualization also reflects periods of similarity between two or more curves’ behaviors.
6.3 Plot Matrix

The SPLOM (scatterplot matrix) and the HyperSlice [40] are diagrams that arrange pair-wise relations in a square matrix fashion and line up unary items along its diagonal. Drawing on this idea we combine all 1st and 2nd-order parameterizations in an $N \times N$ plot matrix where every entry $(n,m)$ contains the principal surface for variables $x_n$ and $x_m$. The special case $n = m$ yields a curve. This diagram generalizes and is more expressive than the curve array; see Fig. 10 and the teaser Fig. 1 for some examples using the Damped Oscillator and Robot Arm models. For example, the periodic variable $\phi_2$ (corresponding to the angle in a robot elbow) stands out as the purple central curve in Fig. 10. Its interaction with non-periodic variables (lengths $L_1$ and $L_2$ of that elbow’s arm segments) is encoded as the purple-orange and purple-magenta umbrella- and cylinder-like parameterized surfaces.

Fig. 10: Plot matrix diagram (zoomed-in version showing 3 variables only) giving a compact depiction of single and pair-wise effects in the Robot Arm high-dimensional model. The user can navigate freely within the 3D scene in order to magnify and observe details from any desired angle. See the paper teaser for zoomed-out examples.

6.4 Variable Selection

The plot matrix layout outlined above is highly compact, but it becomes impractical for more than 2 variables of interest as the number of possible combinations increases exponentially. For this reason, we support various forms of variable selection. There is a tight association between our diagrams and various Sobol indices as discussed in Sec. 5.2. This gives us a forthright criterion to select interesting variables interactively: high Sobol indices reveal strong effects and interactions, and vice versa. Furthermore, such indices can be extracted efficiently from any TT surrogate [6, 16]. We have implemented a contextual minimap as shown in Fig. 11 that extends the fanovaGraph sensitivity analysis chart [18]. It is a graph in a circular layout that displays all first- and second-order Sobol indices of either the whole function or any arbitrary partial function. We furthermore use two palettes, one for darker and one for lighter colors. The area of the darker inner circle within each variable’s node is proportional to its additive effect, while the lighter outer circle encodes its higher-order effect. The same applies to arcs for order-2 indices using the width instead of area.

Fig. 11: Our contextual radial graph (Ebola Spread model) displays all 1st- and 2nd-order Sobol indices for any given values of the current active tuple of variables. Additive and high-order effects are shown as darker and lighter colors, respectively.

Navigation is governed by an active tuple of variables that is empty at the beginning. By clicking a node or an arc on the Sobol minimap, the user can select a variable or pair of variables for further analysis. For instance, if we select an arc connecting an additive and a high-order variable, we expect their joint surface to be mostly rectangularly tilted. The surface will be more or less bent depending on whether there are further high-order interactions with even more variables, as is signaled by the lighter part of their arc.

Then, the corresponding curve or surface visualization is launched in the right 3D widget. The user can also click on the curve array to set a further $n$-th variable to a specific value $x_n := \alpha$, and update this value interactively by sliding a sphere back and forth. Moving the sphere also updates the minimap, which then shows the Sobol indices of the selected 1-variable partial function, as well as whatever curve or surface that is visible in the right widget. This system goes one step further beyond the plot matrix (Sec. 6.3) as it can show up to third-order interactions smoothly.

6.5 Widget-based Tool

We have combined the features previously discussed into an integrated widget-based visualization application (see Figs. 12 and 13 for example snapshots). Global, single-variable structure is shown via a curve array (Sec. 6.2) within a 3D widget (top left). There is also a 2D minimap widget (bottom left) that starts showing Sobol indices for the overall model, and a 3D widget on the right to show individual parameterized curves and surfaces in detail.

As a last example we use the proposed widget-based tool to explore the Ebola Spread model. The most important goal in this problem is ascertaining how resources can be best allocated to reduce the infection rate $R$. With this in mind, the easiest variables to alter in practice are the hospitalization rate $\psi$ and the proper burial rate $\omega$ [15]. Fig. 13 shows two snapshots of the tool for this model. Note that there are four variables that generally increase the rate $R$ and four that reduce it. An insight that stands out immediately from the curve array in Fig. 13 is...
that ψ has a much stronger effect than ω at reducing the rate R. This is precisely one of the main conclusions reached in the study by Diaz et al. [15]. Furthermore, we can use the proposed widgets to understand under what circumstances this disparity is more or less acute. The model is also interesting for its accelerations and decelerations. For example, variations in the hospitalization rate ψ make much more of a difference for low ψ, whereas increasing an already high ψ yields a vanishing improvement only. In addition, the influence of other variables changes drastically when ψ varies. We can examine this scenario in depth by setting ψ to a high value in our application’s curve array widget to find out what other parameters would then become useful at further reducing the infection rate.

![Ebola Spread](image)

Fig. 13: Ebola Spread: we vary transmission rate β₁ (cyan curve; position indicated by the spherical marker) and study its impact on the effectiveness of hospitalization rate ψ (large gray curves) at reducing Ebola infection rate R (black axes). Note that ψ is extremely high at high values of β₁ (b), where it can halve the overall infection rate. Correspondingly, the gray node in the radial widget in (b) becomes the largest one.

### 7 Conclusions

We have contributed a principal component-based dimensionality reduction for global-to-local visualization and sensitivity analysis of dense parameter spaces. To this end we consider the set of all possible partial functions of a model f and project them onto two orthogonal components in the spirit of Sobol’s decomposition method for high-dimensional ANOVA. We summarize those components using a few spatial coordinates to form various parameterized manifolds including curves, surfaces, and ensembles thereof.

In its simplest form, our algorithm boils down to higher-order tensor PCA, on top of which we contributed three conceptual and computational developments:

- The abstraction of taking arbitrary partial functions as the set of vectors to project, including those that are defined with respect to groups of variables and thus give rise to multidimensional manifolds;
- We split the original $L^2$ space into additive and high-order subspaces so as to separately capture the different kinds of influences that variables (or groups thereof) can have. This gives the proposed mapping a direct interpretation in terms of the ANOVA decomposition and the Sobol indices.
- We are aware that computing the principal components of large collections of high-dimensional discretized vectors (with e.g. billions of entries) is a challenging task. We exploited a numerical framework, the tensor train decomposition, that is key to ensure responsiveness and interactivity within the proposed visualization system. The parameter space is cast as a tensor grid and approximated as a low-rank expansion; we extract its principal subspaces directly from the compressed domain.

The visualization diagrams made possible by those ingredients are able to readily communicate interactions between up to three input variables. They also provide the user with discriminative information that allows him or her to select interesting combinations of variables as well as specific values for those variables. We identified how several interesting high-dimensional global and local properties are mapped to specific patterns in 3D curves and shapes, and how individual versus joint-variable effects stand out from our visualization. To the best of our knowledge, this is the first visualization system that is able to communicate such structure in a global-to-local, time-effective manner.

### Future Work

As outlined in the introduction, in this paper we have focused on dense parameter spaces. In particular, no scattered data set (i.e. given collection of samples at fixed locations) was considered as a ground-truth input. Although the domain of application we pursued is attractive, we believe the discrete case remains an equally important target. We believe the proposed method is adaptable to this end: instead of abstracting partial functions over the entire domain, we can show parameterizations that summarize regions or neighborhoods only, for example around feature points or samples from given scattered data. This way we can combine the strengths of global/contextual information (as is only made possible via surrogate modeling) with local structural information as arising from possibly complex sample distributions. We would also like to let users define and move anchor points similarly to interactive PCA [27], in order to provide extra layers of informed interaction.

### References

[1] PyQtGraph: Scientific Graphics and GUI Library for Python. http://www.pyqtgraph.org/

[2] ttpy: a tensor train toolbox implemented in Python. http://github.com/oseledets/ttppy

[3] ttrecipes: a high-level Python library of tensor train numerical utilities. https://github.com/rballester/ttrecipes

[4] J. An and A. B. Owen. Quasi-regression. *Journal of Complexity*, 17(4):588–607, 2001.

[5] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. A surrogate visualization model using the tensor train format. In SIGGRAPH ASIA 2016 Symposium on Visualization, pp. 13:1–13:8, 2016.

[6] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. Sobol tensor trains for global sensitivity analysis. ArXiv e-print 1712.00237, 2017.

[7] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. Tensor algorithms for advanced sensitivity metrics. *SIAM/ASA Journal on Uncertainty Quantification*, 6(3):1172–1197, 2018.

[8] J. F. Banzhaf. Weighted voting doesn’t work: A mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[9] D. Bigoni, A. Engsig-Karup, and Y. Marzouk. Spectral tensor-train decomposition. *SIAM Journal on Scientific Computing*, 38(4):A2405–A2439, 2016.

[10] S. Bruckner and T. Moeller. Result-driven exploration of simulation parameter spaces for visual effects design. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1467–1475, 2010.

[11] J. C. Castura, A. K. Baker, and C. F. Ross. Using contrails and animated sequences to visualize uncertainty in dynamic sensory profiles obtained from temporal check-all-that-apply (TCATA) data. *Food Quality and Preference*, 54:90–100, 2016.

[12] Y.-H. Chan, C. D. Correa, and K.-L. Ma. Flow-based scatterplots for sensitivity analysis. In *IEEE Symposium on Visual Analytics Science and Technology*, pp. 43 – 50, 2010.

[13] P. G. Constantine. Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2015.
