The Visual Computer manuscript No.
(will be inserted by the editor)

High-dimensional Scalar Function Visualization Using Principal Parameterizations

Rafael Ballester-Ripoll · Gaudenz Halter · Renato Pajarola

Abstract Insightful visualization of multidimensional scalar fields, in particular parameter spaces, is key to many computational science and engineering disciplines. We propose a principal component-based approach to visualize such fields that accurately reflects their sensitivity to their input parameters. The method performs dimensionality reduction on the 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.

Keywords Scientific visualization · Sensitivity analysis · Dimensionality reduction · Tensor decompositions

1 Introduction

Dimensionality reduction is a crucial data processing step to interactively visualize and explore large complex data sets. Visualizing parameter spaces is particularly challenging: multidimensional functions often appear in engineering, finance or the life sciences, they arise from different sources (such as black-box experiments, simulations, or surrogate models), and they may accept a large or even infinite number of valid parameter combinations [34].

In this context, *sensitivity analysis* [32] is one of the most important tools for the understanding of parameter spaces. In particular, the method of Sobol [36] is a powerful framework for global sensitivity analysis and interpretation of general multidimensional functions, specifically models subject to hyperparameters or uncertain variables as they move across their entire possible range. However, the method only yields global indices out of the variables of interest on their entire domain, without revealing local details as they move within that domain.

In this paper we aim to reconcile global and local sensitivity analysis by depicting how the model's behavior *evolves* as these variables (and subsets thereof) take different specific values. At the same time, the dimensionality reduction we propose is inspired by Sobol's method in the sense that we partition a model into orthogonal projected subfunctions that only depend on different subsets of variables. Consider a domain $\Omega = \Omega_1 \times \cdots \times \Omega_N \subset \mathbb{R}^N$ over which N variables, $x_1, \dots, x_N$, vary. Let $f : \Omega \to \mathbb{R}$ be a function on these variables. Given a subset of variables of interest, $x_{i_1}, \dots, x_{i_K}$, we introduce a *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. K is the number of variables of interest: for $K = 1$ we produce parameterized curves in $\mathbb{R}^3$, for $K = 2$ we produce surfaces (or equivalently, ensembles

R. Ballester-Ripoll (corresponding author;
e-mail: rafael.ballester@ie.edu)
IE University, Madrid, Spain

G. Halter (e-mail: halter@ifi.uzh.ch)
University of Zurich, Zurich, Switzerland

R. Pajarola (e-mail: pajarola@ifi.uzh.ch)
University of Zurich, Zurich, Switzerland

of curves), for $K = 3$ we produce 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 [20]), our approach is a dimensionality reduction that takes the *full* model f and its exponentially large domain Ω 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 [34]. However, they have only been exploited to a limited degree due to the *curse of dimensionality*. For example, one often needs to compute multidimensional integrals over the entire Ω. This task is typically too computationally expensive to be supported in an interactive visualization system.

To make this feasible and as a second contribution, we design a system that is responsive in real time by means of a convenient compact representation, namely a *low-rank tensor decomposition* [30]. That way, one can efficiently manipulate and integrate dense multidimensional scalar functions and reconstruct (decompress) regions of interest or derived properties interactively. In principle, the proposed transformation could also be computed using another backend numerical framework, for instance (quasi-) Monte Carlo or sparse grid-based approximate integration. Nonetheless, low-rank decompositions are a much more natural match for the proposed dimensionality reduction, which is based in linear projections (Sec. 3).

We provide a global-to-local navigation method that first presents an overall summary and allows the user to refine the view by incrementally adding variables to a *tuple of interest* and, optionally, fixing other variables; see Sec. 6 for further details. Fig. 1 summarizes the modeling and visualization components of our system that implements the proposed approach.

Notation

The hat notation $\hat{x}_n$ refers to a list of variables where x_n is missing, as in $x_1, \dots, x_{n-1}, x_{n+1}, \dots, x_N$.

Given a function of interest $f : \Omega \subset \mathbb{R}^N \to \mathbb{R}$, we will refer to its *slices* as the functions that arise by fixing some $K \geq 1$ of f's variables. For example, $f^{x_1}(x_2, \dots, x_N) = f^{x_1}(\hat{x}_1)$ is a function that maps $\mathbb{R}^{N-1}$ to $\mathbb{R}$ defined for each α as $f^{x_1=\alpha}(\hat{x}_1) := f(x_1{=}\alpha, x_2, \dots, x_N)$. For $K \geq 2$ we have slices 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 ($\boldsymbol{u}$) 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 [43] and Sliceplorer [38] tools, etc. Others are highly domain-specific, such as Tuner [37] for brain segmentation or [10] for physically-based simulations. In terms of the conceptual framework defined by the comprehensive survey by Sedlmair et al. [34], 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 [34] 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 $f(x_1, \dots, x_N)$ 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 [29, 33]. 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 [35].

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 multidimensional model f itself, rather than to its parameters' distribution. A popular approach is to track and visualize f's topological

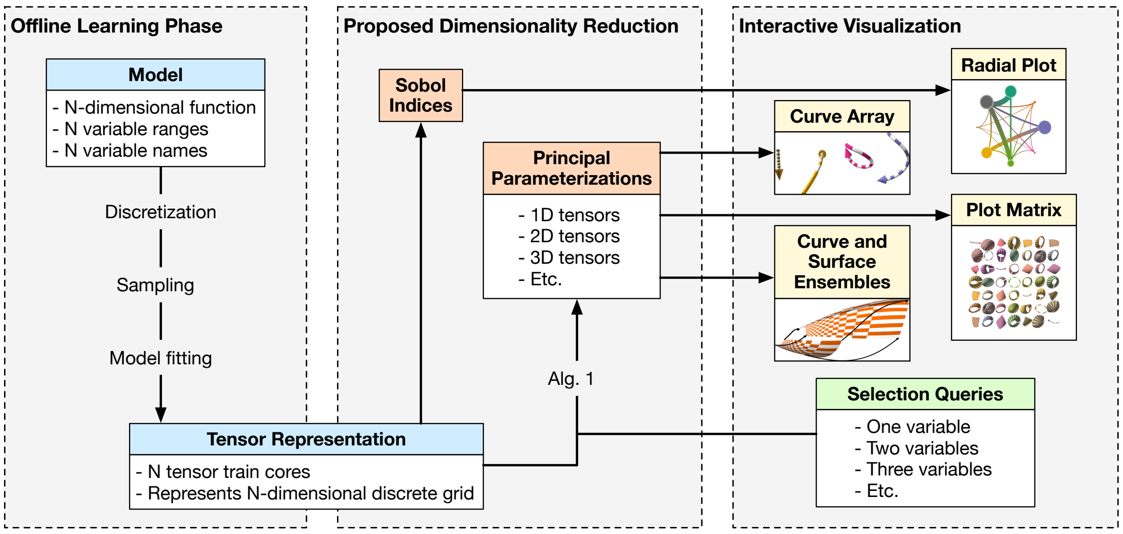


Fig. 1: Our system combines a stage for building the tensor representation of the underlying model of interest (left), a numerical backend that performs the proposed dimensionality reduction via Alg. 1 (center), and an interactive 3D visualization frontend featuring multiple widgets (right).

properties; watershedding segmentation, Morse-Smale complexes [20] 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. Contribution to the sample mean (CSM) plots [9] give a very detailed account of the model's average as a variable moves, but disregard the high-order interactions of the variable with other variables. In addition, CSM plots cannot focus on variable tuples whereas our method can (via the proposed PP-surfaces).

The active subspace method [13], which is very similar to the structure tensor idea for images and volumes [27], is perhaps one of the closest to our 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 $\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_N}$ across the domain Ω, our method uses the covariance between all function slices, 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 and novel 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 [36] remains one of the most prominent for sensitivity analysis of multidimensional scalar functions [17,32]. Its main insight is to realize that every variable's influence can be decomposed into two orthogonal components:

- The *first-order* term S_n of a variable x_n measures how strongly the model's average is affected when x_n changes. A purely first-order variable x_n means that we can separate it from f and write

$$f = g(x_n) + h(x_1, \dots, x_{n-1}, x_{n+1}, \dots, x_N). \quad (1)$$

- The *high-order* term of a variable x_n measures its impact on the model that is not attributable to changes of its average. Hence, a purely high-order variable x_n means that the function's mean $\mathbb{E}[f^{x_n}]$ is not affected by changing x_n.

Often, these two components show up together. Their aggregation is the so-called *total effect*, denoted as S_n^T. The first-order index S_n gives a precise measure of its variable's isolated effect, but disregards any joint interactions. On the other hand, the total effect S_n^T accounts for these, 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 excells at robustly capturing joint relationships among interacting groups of variables. For example, while the presence of a strongly first-order 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 single scalar quantity. 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 first-order in an interval and high-order in another. The need to convey this important, more fine grained information calls for a novel visualization-oriented methodology that works well with the principles of Sobol's ANOVA framework. This is the main motivation behind our method.

2.3 Tensor Metamodeling 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 [28]. 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 [30], which in recent years has been used increasingly for surrogate modeling and visualization [4, 22, 41] as well as for sensitivity analysis [5, 6, 8]. 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 nor needed.

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, 44]. PCA is a long-established framework that has been extensively used for dimensionality reduction, including low-dimensional representation of image-typed data [39, 40], trajectory curves over time [11, 18], and more. However, the proposed sensitivity-aware projection and visualization 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 $1, \dots, M$ 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: $f : \Omega \times \{1, \dots, M\} \to \mathbb{R}$. 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 multidimensional function f or, in other words, the relationship between the $(N-1)$-dimensional function slices $f^{x_n}(\hat{x}_n) = f(\dots, x_n, \dots)$ as x_n changes between 0 and 1. Of course, each slice 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, every slice f^{x_n} $0 \le x_n \le 1$ belongs to $\mathcal{F}$. Let us summarize this collection of (infinite many) slices as a parameterized curve, i.e. map each to a point in $\mathbb{R}^3$. We start by averaging each slice over its remaining free variables to get a single scalar, i.e. computing the global average of f^{x_n} for a fixed x_n. We call this function f_n,

$$f_n(x_n) := \mathbb{E}[f^{x_n}] = \\ = \int_{[0,1]^{N-1}} f(x_1, \dots, x_N)\, \not{d}x_n, \tag{2}$$

where $\not{d}x_n$ indicates that all variables except x_n are integrated.

Function f_n captures the aggregated first-order behavior as per variable x_n and hence determines the first-order Sobol index (Sec. 2.2), which is defined by the variances of f_n and f as $S_n := \mathrm{Var}[f_n]/\mathrm{Var}[f]$. Obviously, such an averaging still gives limited information on f^{x_n}'s inner workings as x_n varies. The missing information is contained in the *residual function* $f_{-n} := f - f_n$. The slices of this residual are the zero-centered slices of f and contain the *higher-order* information of f along variable x_n. In Sobol's method the residual is important, since its variance gives rise to the total index: $S_n^T := \mathrm{Var}[f_{-n}]/\mathrm{Var}[f] + S_n$.

The core part of our proposed mapping is to decompose each original slice as the sum of two orthogonal components, namely its average and the residual:

$$f^{x_n}(\hat{x}_n) = f_n(x_n) + f_{-n}^{x_n}(\hat{x}_n). \tag{3}$$

Then, we represent those two parts as follows:

- One coefficient $\pi_x(x_n)$ for the average at x_n, i.e. $\pi_x(x_n) := f_n(x_n)$.
- Two coefficients $\pi_y(x_n)$ and $\pi_z(x_n)$ for the residual slice. Since that slice still depends on $N-1$ continuous variables, 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-Loève expansion (KLE) for the pairwise covariance function between all residuals along x_n: $\mathrm{Cov}(\alpha, \beta) := \mathbb{E}[f_{-n}^{x_n=\alpha} \cdot f_{-n}^{x_n=\beta}]$ for all $\alpha, \beta \in [0,1]$. This yields two scalar functions $\pi_y(x_n)$ and $\pi_z(x_n)$. The discrete equivalent of this dimensionality reduction, i.e. when one wants to project a collection of vectors onto 2D, is to keep the two leading eigenvectors of the vectors' covariance matrix.

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 $\mathbb{R}^3$ (see Fig. 2 for a simple example). Let us call this 1D-manifold a *principal-parametrization* or *PP*-curve.

Our dimensionality reduction 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. Our choice of the fixed vector gathers *absolute* information, since coordinate π_x equals the slice's mean. On the other hand, the two other vectors to be sought encode *relative* information as absolute positions (π_y, π_z) on the yz-plane. While these absolute positions are not directly interpretable, 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 function slices f^{x_n} onto a PP-curve, a $K=1$-dimensional manifold. The higher-dimensional case ($K \geq 2$) extends naturally from that. The main difference is that we now have collections of function slices $f^{x_n,\cdots}$ that are indexed by two or more variables, and we get functions that arise from fixing several variables and are $(N-K)$-dimensional, e.g. f^{x_n,x_m} for $K=2$. As a result we no longer obtain parameterized curves but higher-order PP-manifolds (surfaces, volumes, etc.) that are again parameterized by triplets of multidimensional functions, e.g. $(\pi_x(x_n,x_m), \pi_y(x_n,x_m), \pi_z(x_n,x_m)) : [0,1]^2 \to \mathbb{R}$.

In Fig. 3 we show an example parametrized surface from the *Damped oscillator* model where we select the two variables k_p and z_s of interest and get a PP-surface in $\mathbb{R}^3$ given by all points $(\pi_x(k_p,z_s), \pi_y(k_p,z_s), \pi_z(k_p,z_s))$. An example considering three variables of interest is shown in Fig. 5.

4 Geometric Interpretation

4.1 Approximate Isometries

The truncated KLE indicated above yields the projection π that, by means of a reduced orthonormal basis, best preserves a given collection of vectors in the L^2 sense:

$$\arg\min_{\pi} \sum_{\boldsymbol{u}} \|\boldsymbol{u} - \pi^{-1}(\pi(\boldsymbol{u}))\|^2$$

In the above equation, $\pi^{-1}(\cdot)$ is the expansion back into the original multidimensional space using the same basis. This means that distances between vectors are also preserved well,

$$\|\boldsymbol{v} - \boldsymbol{u}\| \approx \|\pi(\boldsymbol{v}) - \pi(\boldsymbol{u})\|,$$

and likewise relative distance changes:

$$\|\boldsymbol{w} - \boldsymbol{v}\| - \|\boldsymbol{v} - \boldsymbol{u}\| \approx \|\pi(\boldsymbol{w}) - \pi(\boldsymbol{v})\| - \|\pi(\boldsymbol{v}) - \pi(\boldsymbol{u})\|,$$

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 as well. This has important and desirable consequences from a visualization point of view. For example, if a set of function slices f^{x_n} for $0 \leq x_n \leq 1$ are all multiples of each other, $\mathcal{F}$, then its projection $\pi : [0,1] \to \mathbb{R}^3$ will consequently evolve in a linear fashion too:

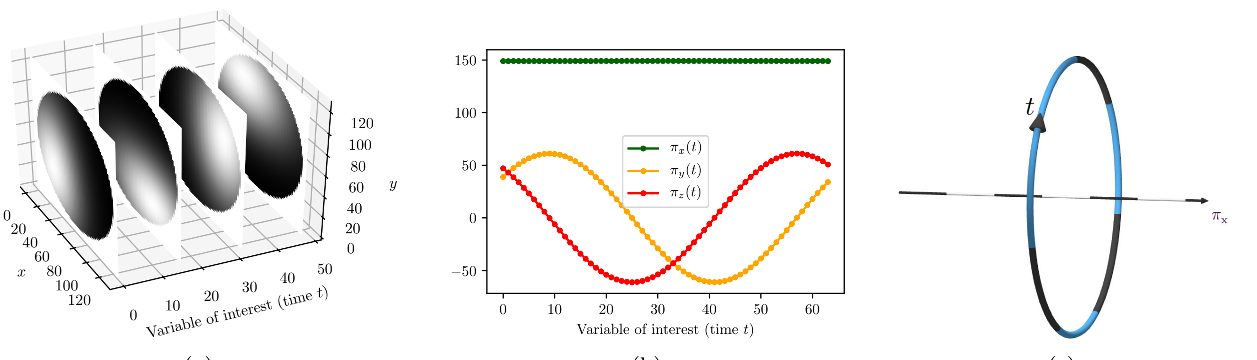


Fig. 2: (a) A simple 3D model, B/W video of a static ball with a light source rotating around it, with coordinates (x, y, t), shown as slices along a variable of interest (time t). (b) Principal parameterization for t, $(\pi_x(t), \pi_y(t), \pi_z(t))$. Since all video frames have the same average intensity, $\pi_x(t)$ is constant. The rotating pattern is captured as two wavelike functions: $(\pi_y(t), \pi_z(t)) = (A\cos(\omega t), A\sin(\omega t))$. Each dot represents one discrete value of $t = 0, 1, \ldots, 63$. (c) Resulting 3D visualization of the resulting curve without explicitly showing the axes π_y, π_z. Since $\pi_x(t)$ is constant the curve projects onto a single point on the x-axis, and since $\pi_y(t)$ and $\pi_z(t)$ are periodic the curve describes a circle in the yz-plane.

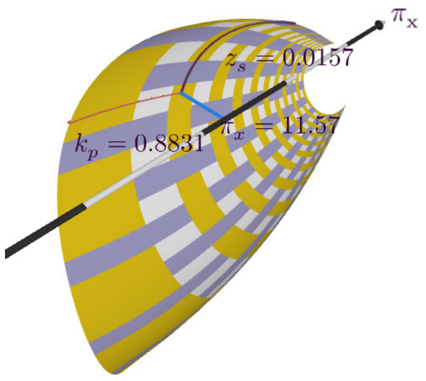


Fig. 3: We use interwoven isolines to visually track constant parameter values over our principal-parametrization surface. By hovering over a point, the user is shown its corresponding parameter values (here k_p and z_s from the *Damped oscillator* model, Sec. 6) along with its projected coordinate $\pi_x(k_p, z_s)$.

$$\pi(x_n) = \pi(0) + x_n \cdot (\pi(x_n) - \pi(0)), \ 0 \le x_n \le 1,$$

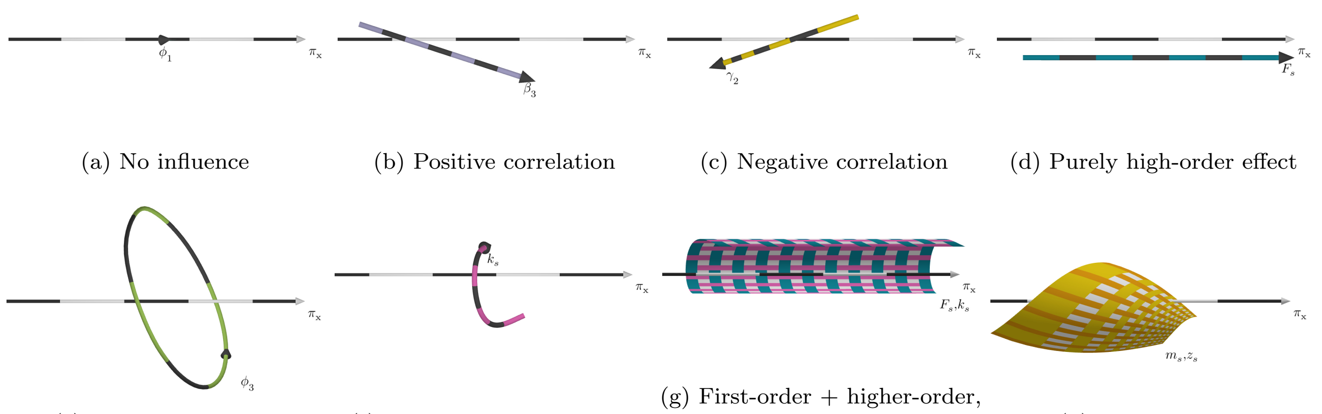
which is a straight line connecting the 3D points $\pi(0)$ and $\pi(1)$.

Conversely, a curved projection π hints at a sequence of vectors that changes non-linearly. Sudden changes in the curve mirror sudden changes also in the original higher-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 $(\mathbb{E}[f], 0, 0)$. Furthermore, the cosine similarity $\frac{\boldsymbol{u}\cdot\boldsymbol{v}}{\|\boldsymbol{u}\|\cdot\|\boldsymbol{v}\|}$ is approximately preserved as well, and is displayed in 3D as angles between vectors. To help gain an 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.

Certainly, since we are projecting vectors of high dimensionality onto three basis elements only, much of the detail along the unimportant variables is likely to be smoothened out. On the other hand, because the resulting visualized 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 PP-curve $\pi(x_n)$ traces back unambiguously to one specific value of its variable x_n. We argue this is a key 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.

4.2 Global Sensitivity Analysis

As outlined in the introduction, there are several interesting connections relating our projection π with the Sobol indices [36] (see Sec. 2). Recall that for the Sobol index of the n-th variable we have $S_n \propto \mathrm{Var}[f_n]$, whereas the total Sobol index S_n^T accounts for both the first- and high-order effects: $S_n^T \propto \mathrm{Var}[f_n] + \mathrm{Var}[f_{-n}]$. Furthermore, we have that $\mathrm{Var}[f_n] = \|f_n\|^2 \propto \|\pi_x\|^2$ (exact projection on the x-axis) and $\mathrm{Var}[f_{-n}] = \|f_{-n}\|^2 \propto \|(\pi_y, \pi_z)\|^2$ (approximately; it is the projection on the yz-plane us-



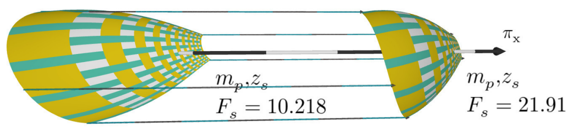
(a) No influence (b) Positive correlation (c) Negative correlation (d) Purely high-order effect

(e) Periodic behavior (f) Purely high-order effect (g) First-order + higher-order, little interaction (h) Mixed effects

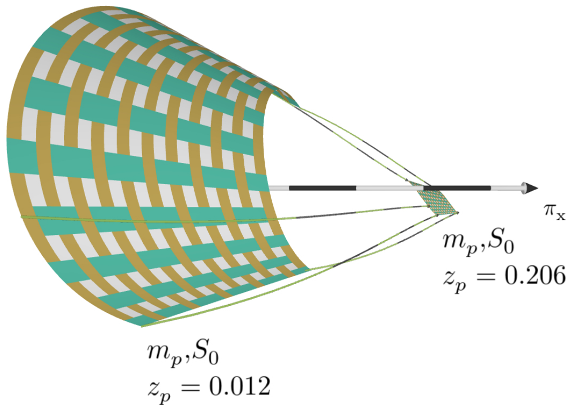
Fig. 4: The proposed PP-curves and PP-surfaces capture a wide range of single- and multiple-effect patterns. No correlation, as well as positive or negative correlations between the selected variable and the model's (mean) output, is shown in (a,b,c,d) by the relatively simple curves. Periodic and high-order effects of a single variable are illustrated in (e) and (f). PP-surfaces are shown in (g) and (h). In (g), one variable has a small global influence as revealed by the surface's small spread around the x-axis (dark cyan isolines), while the other variable independently exhibits a first-order contribution (light purple isolines). In all graphs, the x-axis is depicted as a black-and-white arrow. The x coordinate indicates the model's average over all abstracted variables for any specific values of the parameterized target variables (color curves and surfaces).

ing a truncated expansion). Therefore, we can make the following observations:

- The curve's evolution along the x-axis mirrors the corresponding slice's mean value $\mathbb{E}[f^{x_n}]$ as x_n changes, and S_n is proportional to the PP-curve's variance along that axis. In other words, by tracking the curve's π_x coordinate while changing x_n we can infer the overall first-order behavior of our N-dimensional model as that variable varies. In particular, the correlation ρ between x_n and the model output equals that between x_n and the PP-curve's π_x coordinate: $\rho(x_n, f) = \rho(x_n, \mathbb{E}[f^{x_n}]) = \rho(x_n, \pi_x(x_n))$. Thus curves that point towards the right of the x-axis indicate a positive correlation, and vice versa. Any purely first-order variable x_n (i.e. $S_n = S_n^T$) will not cause any variation in the yz-plane, i.e. the curve is mapped to a line segment that is perfectly aligned to the x-axis. See also examples in Fig. 4(a,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 in 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 the difference between the total and plain Sobol indices $S_n^T - S_n$. For instance, a curve orbiting far from the x-axis means its corresponding variable has strong interactions with other variables. Any purely high-order variable x_n (i.e. that *does not* influence the model's average, with $S_n = 0$) does not vary along the x-axis at all but only in the yz-plane, e.g. as in Fig. 4(f).
- A periodic behavior of a variable on the model's output is mapped to closed curves, e.g. as in Figs. 2(c) and 4(e).
- For two variables x_n and x_m (that is, $K = 2$) the resulting PP-surface shows their joint effect on the model's mean value and their individual effects as highlighted by surface isolines (Figs. 4(g,h) and 5). For example, Fig. 4(g) shows a pair of variables (F_s and k_s) having a separable influence on the first- and high-order effects of the model's average respectively, while Fig. 4(h) shows another pair (m_s and z_s) exhibiting a mixed interacting effect.
- The total index S_n^T measures the sum of average and higher-order effects and is 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 in all directions, the more global influence its variable has on the model. Conversely, a tuple of irrelevant variables will be collapsed into a point (see also Fig. 5).



(a) Linear effect



(b) Diminishing effect

Fig. 5: An ensemble of surfaces can show the effect of three variables in one visualization. In (a) a linear effect is shown with respect to changing the third variable F_s. In (b) the contraction of the principal parametrization surface with respect to increasing the third variable z_p means that the variables m_p and S_0 become irrelevant for certain values of z_p.

In a nutshell, first-order effects make the projected PP-manifolds move *along* the x-axis, while high-order interactions *pull* them away in various ways.

5 Practical Algorithm

5.1 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 $\mathcal{T}$ of size I^N. This way, function slices become tensor slices. Instead of a continuous 3D parameterization (π_x, π_y, π_z), we then seek three corresponding discrete (coordinate) tensors $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$, 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.

5.2 Algorithm Outline

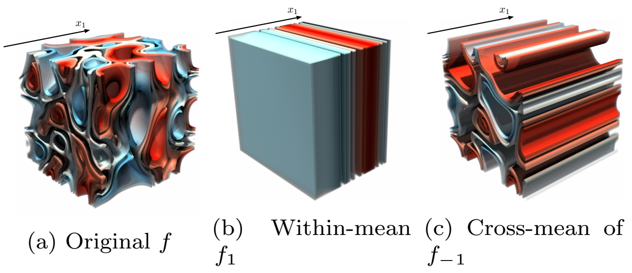
For the ease of exposition, we assume that the K variables of interest are the first $1, \dots, K$. The dimensionality reduction that we motivated in Sec. 3 boils down to a three-step processing pipeline:

Stage A: The *within-mean* of each slice of f is computed (to be used as x-coordinate during visualization) and subtracted from the original. This way we derive a new function $f_{-1 \dots K}$ whose slices are zero-centered.

Stage B: The *cross-mean* of $f_{-1 \dots K}$ (i.e., its average over target variables $1, \dots, K$) is subtracted from each of its elements to yield a new collection $f_{\overline{-1 \dots K}}$. 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 the $[0,1]^K \times [0,1]^K \to \mathbb{R}$ covariance function that maps every possible pair of elements of $f_{\overline{-1 \dots K}}$ to their inner product. For each slice, its coefficients in terms of this basis define its embedding on the yz-plane.

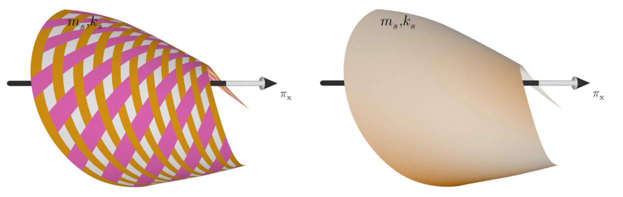
See Fig. 6 for an example of the within- and cross-means for dimension $N = 3$ and target variable x_1. Note that Stages A and B are orthogonal to each other: the within-mean (Stage A) is computed as an average over the non-target variables, whereas the cross-mean (Stage B) is an average over the remaining target variables. See App. B for an illustrated flow chart of the proposed algorithm.



(a) Original f (b) Within-mean f_1 (c) Cross-mean of f_{-1}

Fig. 6: Isosurface renderings for an example 3D function $f(x_1, x_2, x_3)$ (a) with target variable x_1. The within-mean (b) is an average over axes x_2, x_3 that only varies along x_1, whereas the cross-mean (c) is an average over axis x_1 that only varies along x_2, x_3.

The PCA truncation we just described also allows us to compute the projection error for each point in a principal parameterization: it is the distance between the corresponding slice and its approximation $\epsilon(x_n, x_m) := \|f^{x_n,x_m} - \pi^{-1}(\pi(f^{x_n,x_m}))\|$, and is given by the sum of squared truncated eigenvalues. The user can toggle the color texture of principal surfaces back and forth between isolines and projection error; an example in Fig. 7.



(a) Isolines (b) $\epsilon(m_s, k_s)$

Fig. 7: We provide two surface texture options: (a) interwoven parameter isolines; (b) PCA projection error ϵ at each point. The model shown is the *Damped Oscillator* with variables of interest m_s and k_s.

See Alg. 1 for a practical, discretized version of the proposed algorithm. It relies on a range of expensive tensor operations: computing means along several axes, element-wise subtracting tensors, computing a large covariance matrix $\mathbf{C}$ among many tensor slices, and eigendecomposition of that matrix. In particular, the entries of $\mathbf{C}$ 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, $\mathbf{C}$ may grow to become a massive dense matrix with billions of entries, so its eigendecomposition poses a challenge on its own. For example, for $K = 3$ and a moderate discretization size of 64 bins per dimension, the method must compute the leading eigenvectors of a matrix of size $64^3 \times 64^3$. While MC estimation may be sufficient in some cases for offline dimensionality reduction and visualization, it is hardly practical for interactive navigation, which is the more desirable goal. A suitable algorithm, therefore, is required as presented below.

5.3 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:

Algorithm 1 Practical numerical procedure operating on discrete tensors. Input: an N-dimensional function f discretized as a tensor $\mathcal{T}$ of shape I^N and $1 \leq K < N$ variables of interest (for simplicity, here assumed to be the first $1, \ldots, K$). Output: 3 tensors $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ that describe their discretized principal parameterization, namely a K-dimensional manifold in $\mathbb{R}^3$. Note that all tensors are manipulated in the tensor train format; for more details on how to perform such operations (subtraction, averaging, dot products, etc.) we refer the reader to the seminal paper [30].

```
                         {Stage A}
 1: // Within-mean of each slice: average over non-target variables
 2: $\mathcal{T}_{1\ldots K} := \mathrm{mean}(\mathcal{T}; K+1, \ldots, N)$
 3: // Separate and subtract the within-mean
 4: $\mathcal{T}_{-1\ldots K} := \mathcal{T} - \mathcal{T}_{1\ldots K}$
                         {Stage B}
 5: // Cross-mean among all slices: average over target variables
 6: $\mathcal{T}_{\overline{-1\ldots K}} := \mathrm{mean}(\mathcal{T}_{-1\ldots K}; 1, \ldots, K)$
 7: // Separate and subtract the cross-mean
 8: $\mathcal{M} := \mathcal{T}_{-1\ldots K} - \mathcal{T}_{\overline{-1\ldots K}}$
                         {Stage C}
 9: // Compute covariances among all pairs of tensor slices
10: $\mathcal{C} := \mathrm{zeros}(I, \ldots, I)$   // Tensor of size $I^{2K}$
11: for $i_1, \ldots, i_K = 1, \ldots, I$ do
12:     for $j_1, \ldots, j_K = 1, \ldots, I$ do
13:         $\mathcal{C}(i_1, \ldots, i_K, j_1, \ldots, j_K) = \langle \mathcal{M}^{i_1 \cdots i_K}, \mathcal{M}^{j_1 \cdots j_K} \rangle$
14:     end for
15: end for
16: $\mathbf{C} := \mathrm{reshape}(\mathcal{C}, I^K \times I^K)$   // Covariance matrix
17: // Compute the two leading eigenpairs of $\mathbf{C}$
18: $\boldsymbol{\Lambda}, \mathbf{U} := \mathrm{EIG}(\mathbf{C}; 2)$
19: // Gather and return 3D parameterization tensors
20: $\mathcal{X} := \mathcal{T}_{1\ldots K}$   // The $x$ coordinate is exactly the within-mean
21: $\mathcal{Y} := \mathrm{firstColumn}(\boldsymbol{\Lambda} \cdot \mathbf{U})$   // Vector with $I^K$ elements
22: $\mathcal{Y} := \mathrm{reshape}(\mathcal{Y}, I \times \cdots \times I)$
23: $\mathcal{Z} := \mathrm{secondColumn}(\boldsymbol{\Lambda} \cdot \mathbf{U})$   // Vector with $I^K$ elements
24: $\mathcal{Z} := \mathrm{reshape}(\mathcal{Z}, I \times \cdots \times I)$
25: return $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$
```

(a) it can compress a full parameter space very compactly, circumventing the curse of dimensionality; (b) allows for very fast multidimensional integration [31]; and (c) 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 $1 \leq i_1, \ldots, i_N \leq I$ of our discretized tensor $\mathcal{T}$ as a product of matrices,

$$\mathcal{T}[i_1, \ldots, i_N] \approx \mathcal{T}^{(1)}[i_1] \cdot \ldots \cdot \mathcal{T}^{(N)}[i_N], \tag{4}$$

where every $\mathcal{T}^{(n)}$ is a 3D tensor known as *core*, namely an array of I matrices $\mathcal{T}^{(n)}[i_n]$ indexed by i_n; $\mathcal{T}^{(1)}$ and

$\mathcal{T}^{(N)}$ 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 $\mathcal{T}^{(n)}[i_n]$.

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 cores $\mathcal{T}^{(1)}, \dots, \mathcal{T}^{(K)}$. Furthermore, we do not actually require explicit expensive loops (corresponding to lines 11 to 15 in Alg. 1) to populate all elements of the $I^K \times I^K$ covariance matrix $\mathbf{C}$: we hold all entries of this matrix in the tensor train format and extract its leading eigenpairs efficiently in the compressed domain (see App. C).

Note that this framework also allows us to quantify the relative error introduced by the PCA truncation, which is a relation between the eigenvalues we keep and the original norm of the covariance matrix: $1 - \frac{\sqrt{\lambda_1^2+\lambda_2^2}}{\|\mathbf{C}\|}$, where λ_1, λ_2 are the two leading eigenvalues of $\mathbf{C}$ and $\|\mathbf{C}\|$ its Frobenius norm, i.e. square root of the sum of its squared elements. This error was below 0.25 in all parameterization extracted in this paper.

6 Results

We tested our methods and implementation on four analytical models of varying complexity and dimensionality. For each of these models, we used an adaptive sampling algorithm known as *cross-approximation* [31] to built the corresponding TT metamodel. This step corresponds to the *Offline Learning Phase* in Fig. 1. See the App. A for more details on cross-approximation.

6.1 Software

We used Python 3.6, Flask, and Three.js to implement the proposed algorithm in a webapp framework. All visual results are displayed using a range of custom widgets and diagrams, as demonstrated in Fig. 12 for the *Damped Oscillator* example described in Sec. 6.5. Our visualization frontend is implemented in JavaScript: we use Three.js for 3D rendering and interaction with the principal parameterizations, and Bootstrap with JQuery for the user interface. The Flask server hosts our numerical backend, which exploits the NumPy, ttpy [1], and ttrecipes [2] libraries for tensor train manipulation. The models used in this paper are all available in the ttrecipes package.

6.2 Nassau County

Our first model is the voting system studied by Banzhaf [7], which considers the six voting agents in the 1964 Nassau County Board of Supervisors. In order for any political motion to succeed, it must be backed by a coalition of districts that reach a vote majority (58 votes or more). In political sciences and game theory, the *Banzhaf power index* [7] is often used to assess the true influence of individual agents in a voting body. 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. Banzhaf argued that the three weakest districts of Nassau County were actually powerless even though they collectively held 22% of the total votes.

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 tensor of size $2^6 = 64$, where each entry is 1 if the corresponding coalition would reach majority and 0 otherwise. We then compute the PP-curve corresponding to each district and visualize them together in a single system of 3D coordinates; we call this widget a *curve array* (Fig. 8).

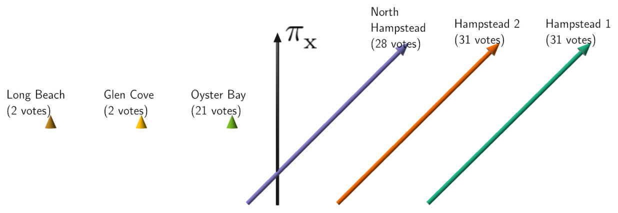


Fig. 8: When applied to a majority voting system, our parameterization yields the Banzhaf power index [7]. The powerless districts are the three leftmost curves, which are simply zero-length segments.

Since each district's variable can only take values 0 or 1, its PP-curve is just a line connecting two endpoints. We have found the variance of these two points (i.e. the segment's squared length) to be proportional to their corresponding district's Banzhaf power index. In particular, the three PP-curves of Fig. 8 with zero length (i.e. only their arrowhead is visible) identify the three powerless districts: their votes have no influence on the model's output. On the other hand, the three Hampstead districts are mapped to three equal straight lines, indicating that they share the effective power evenly.

6.3 Cell Cycle

As a second, more complex example we consider a 4D, multivalued system of ordinary differential equations (ODE), namely the *Cell Cycle* [19,21]. It models five protein concentrations (`cyclin`, `protease`, `cdc2k`, `cyclin inhibitor`, and `complex inhibitor`) during cell division. It is a time-dependent system, hence the temporal axis t is particularly important as an explanatory variable. The five system outputs are known to have an oscillatory behavior around five stationary concentration levels, and our goal is to understand how three uncertain input parameters (`V1p`, `K6`, and `V3p`) influence those levels.

To perform this analysis using the proposed PP-curves, we select t as our variable of interest and gather all five outputs of the ODE into an extra dimension for a joint analysis; the resulting tensor is five-dimensional. This way, we force their five principal curves to share the same 3D system of projected coordinates. Furthermore, we add a slider to govern any of the three 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 five curves displayed. Concentration `K6` was found to have a strong effect on the speed of change of several outputs; see Fig. 9 for some example renderings.

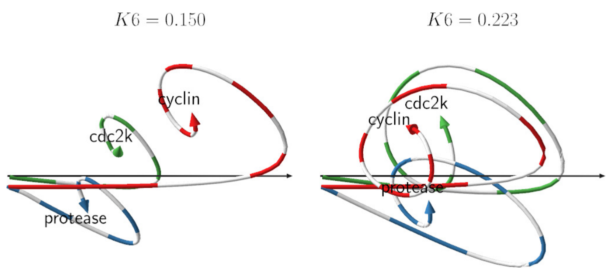


Fig. 9: A dynamic 5-valued ODE modeling the cell division cycle [19] and principal curves showing three of the five output quantities with variable of interest t (time) and two different values of the input parameter `K6`.

As can be seen in Fig. 9, the PP-curves show how the attractor point is different for each output and, further, it is differently affected by `K6`. In particular, higher values of `K6` lead to more similar stationary concentrations of the proteins of interest. 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.

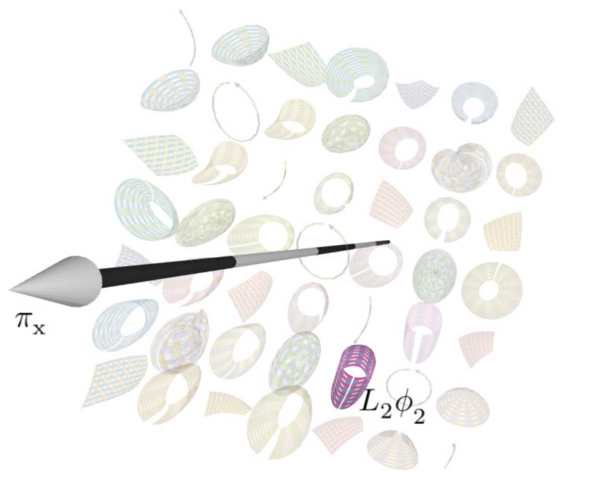
6.4 Robot Arm

Next, we use an 8D model measuring the distance from the origin attained by a 4-segment robot arm that can only move within a 2D plane [3]. Its variables are L_1, L_2, L_3, L_4, which model the length in centimeters of each segment, and $\phi_1, \phi_2, \phi_3, \phi_4$, which model the angle of each elbow joint. The model output is periodic w.r.t. those angles, which can all move between 0 and 2π. The first angle is non-influential because it rotates the full arm and does not change the distance between the arm's tip and the origin. Other than ϕ_1, we expect strong interactions between the variables of this model, given that the segments are connected to each other and their lengths and angles affect the tip position in a complex, high-order way.

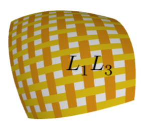
We explore this model using two different visualizations. The first is a plot matrix: we combine all 1st and 2nd-order principal parameterization 1D or 2D manifolds in an $N \times N$ table where every entry (n, m) contains the PP-surface for variables x_n and x_m. The special case $n = m$ yields its PP-curve. Similarly to the curve array, the parameterizations are evenly spread on the yz-plane, but their x axis has an absolute interpretation in all cases. This visualization is inspired by the SPLOM (scatterplot matrix) and the HyperSlice [43] diagrams, which arrange pair-wise relations in a square matrix fashion and line up unary items along its diagonal.

The plot matrix generalizes and is more expressive than the curve array. See Fig. 10a for an example using the *Robot Arm* model. Note how periodic variables (the angles ϕ_i) are mapped into closed curves along the diagonal. Each surface summarizes the pair of variables it corresponds to; see the zoomed-in examples at the bottom of Fig. 10. For example, a rectangular surface (Fig. 10c) represents the interaction between two linear variables. A cylinder (Fig. 10c) captures the joint behavior of a periodic and a linear variable, and the cylinder's girth and height convey how strong their respective influences are on the model output. The dome in Fig. 10d tells us that the influence of ϕ_4 vanishes when L_4 is small, which reflects the fact that the last elbow angle becomes irrelevant when the last robot arm segment is very short.

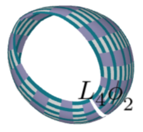
Second, we also implement a contextual minimap (Fig. 11) that extends the *fanovaGraph* sensitivity analysis chart [17]. It is a graph in a circular layout that captures all first- and second-order Sobol indices of either the whole function or any arbitrary function slice. We furthermore use two palettes, one for darker and one for lighter colors. The area of the darker inner circle within each variable's node is proportional to its first-order effect, while the lighter outer circle encodes



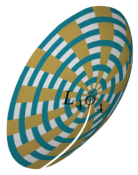
(a) Plot matrix



(b) Rectangular



(c) Cylindrical



(d) Dome

Fig. 10: Top: plot matrix diagram giving a compact depiction of single and pair-wise effects (*Robot Arm* model). The x axis is shown in black-and-white. The user can navigate freely within the 3D scene in order to magnify and observe details from any desired angle. Hovering over a curve or surface highlights it and shows its variables. Bottom: detail of three distinct surface types.

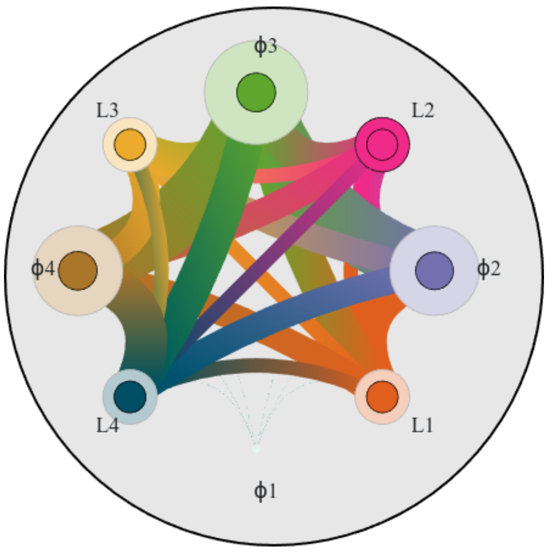


Fig. 11: Our contextual radial graph (*Robot Arm* model) displays all 1st- and 2nd-order Sobol indices for any given values of the current active tuple of variables. First- and high-order effects are shown as darker and lighter colors, respectively.

its higher-order effect. The same applies to the curved arcs connecting two nodes for order-2 indices using the width instead of area.

We can see in Fig. 11 that the most influential variables (larger graph nodes) are the elbow angles, and that the strongest interactions (wide arcs) occur between these variables. Note that, since ϕ_1 has no effect, its node is just a point (circle of radius zero).

6.5 Damped Oscillator

Third, we consider an 8D physical model known as the *Damped Oscillator* [16]. It models a primary mass m_p connected to a wall by a spring and to a secondary mass m_s by another spring. The system is subject to a white-noise impulse of strength S_0 and the model output is the peak force experienced by the secondary spring (the higher this force, the more likely the system is to fail). The other variables are the spring stiffnesses k_p and k_s, the damping rates z_p and z_s, and the force capacity of the secondary spring F_s.

To explore this model we have combined all features previously discussed into an integrated web-based visualization application emphasizing the global-to-local paradigm (Fig. 12). When the user selects a model to be visualized, a plot matrix (as in Sec. 6.4) is launched in the main view alongside with with an array of PP-curves (Sec. 6.2). These widgets are furthermore linked to the Sobol minimap (Fig. 11), which is initialized to the overall model.

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 thereof for further analysis. For instance, if we select an arc connecting a first-order variable with a high-order one, we expect their joint surface to be mostly rectangularly tiled (as we showed in Fig. 4(g)). 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. Alternatively, the user can select a tuple by clicking on the parameterized surface directly in the plot matrix.

As soon as such a selection is made, the plot matrix in the main view is minimized, and the corresponding curve or surface visualization is maximized instead. Using a dropdown menu, the user can further select yet another variable x_l to a specific value $x_l := \alpha$, and update this value interactively via a slider. Moving the slider also updates the minimap, which then shows the Sobol indices of the selected 1-variable function slice,

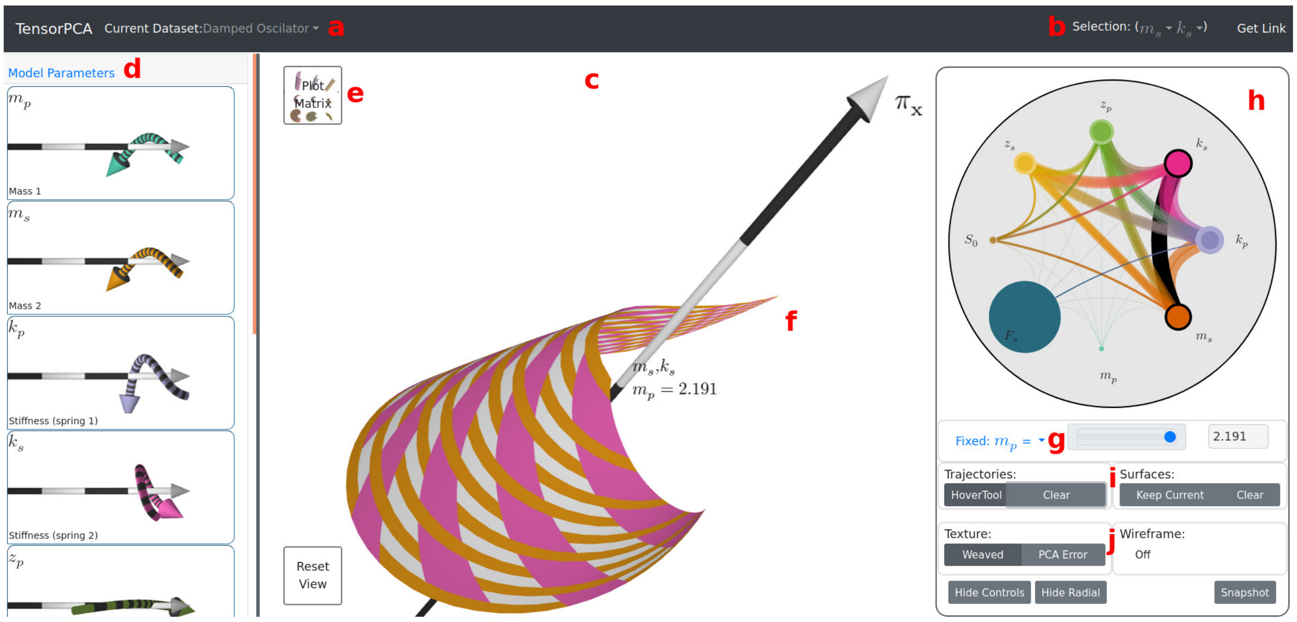


Fig. 12: Snapshot of the entire application (*Damped Oscillator* model) and its most important features: (a) model selector; (b) parameter selector; (c) spotlight; (d) curve array; (e) plot matrix toggle; (f) hoverable 3D parameterizations; (g) selector and slider for an optional third parameter; (h) radial minimap; (i) selector for optional ensembles of curves or surfaces; (j) surface texture menu.

as well as whatever curve or surface that is visible in the right widget. A hover tool allows the user to investigate the current principal parameterization. Hovering over a surface makes a trajectory appear that displays the 3D evolution of $\pi(x_n, x_m, x_l)$ as x_l takes different values. This system can show up to third-order interactions smoothly, and thus goes one step beyond the plot matrix of Sec. 6.4. The system responds interactively: computing a principal parameterization takes well under a second, whereas extracting Sobol indices from a tensor train is a matter of milliseconds.

After looking at the curve array in the left-hand side of Fig. 12, we notice that the arrows for m_p and m_s flow in the against the x axis (depicted as a black-and-white straight arrow). This leads to the insight that the higher these masses the smaller the peak force becomes and, therefore, the more robust the oscillator will be. On the other hand, we see that the spring stiffnesses k_p, k_s have a small impact as their arrows move largely perpendicularly to the x axis. The radial minimap (Fig. 12h) shows that, although F_s is the most influential variable, it barely interacts with other variables – meaning that the model can be simplified as the sum of a term depending on F_s plus a term depending on the rest. Last, the central widget shows the interaction between m_s and k_s as we move m_p (currently set to 2.191 in the slider g). The highly obtuse angles between the magenta and orange isolines tell us that variations in m_s and k_s have effects that are largely opposing to each other.

6.6 Ebola Spread

We consider an 8-variable model of the Ebola virus infection rate R_0 in Liberia and Sierra Leone, based on statistics from the 2014 outbreak [15]. The most important goal in this scenario is ascertaining how resources can be best allocated to reduce R_0. According to the authors of the study, the two variables that can be influenced most easily to decrease the number of infections are the hospitalization rate ϕ and the proper burial rate ω. The rest mostly depend on environmental factors, including $\beta_1, \beta_2, \beta_3$ (infection parameters), ρ_1 (mortality rate for unhospitalized patients) and γ_1, γ_2 (avg. disease duration without and with hospitalization, respectively).

Fig. 14 shows two snapshots of the tool for this model. As the model's curve array shows (Fig. 13), there are four variables that generally increase the rate R_0 and four that reduce it. It stands out that ϕ has a much stronger effect than ω at reducing the rate R_0. This is precisely one of the main conclusions reached analytically 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 ϕ make much more of a difference for low ϕ, whereas increasing an already high ϕ yields a vanishing improvement only. In addition, the influence of other variables changes drastically when ϕ varies. We can examine this scenario in depth by setting ϕ to a high value in the slider to find out what other parameters would then become useful at further reducing the infection rate.

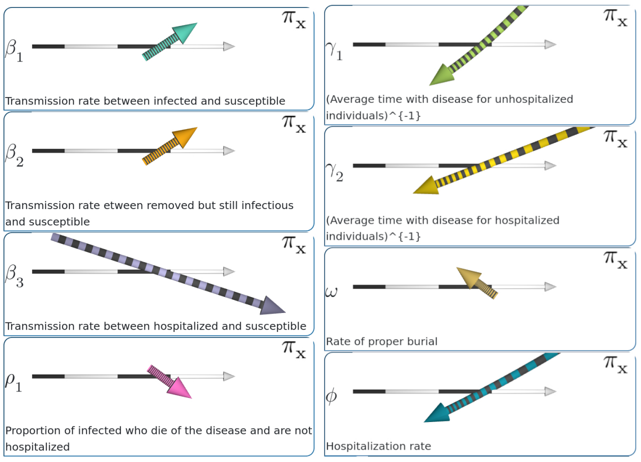


Fig. 13: The curve array summarizes the influence exerted by each parameter. Depicted are all 8 parameters for the *Ebola Spread* model. Parameters $\beta_1, \beta_2, \beta_3$ and ρ_1 are roughly aligned with the x axis (black-and-white arrow), which tells us that they correlate positively with the model's output (viral infection rate).

6.7 Comparison

Last, we comparatively analize the *Robot Arm* using the CSM [9] and Sliceplorer [38] techniques, and we note that the proposed PP-curve can better account for periodic variables as shown in Fig. 15. Not only can our method reflect local changes of selected parameters, but it does so while succinctly encoding the remaining parameters' contribution into (y, z) coordinates (instead of fixing, randomly sampling, or averaging these contributions).

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 slices of a model f and project them onto two orthogonal components in the spirit of Sobol's decomposition method for ANOVA. We summarize those components using a few spatial coordinates to form various parameterized manifolds including curves, surfaces, and ensembles thereof.

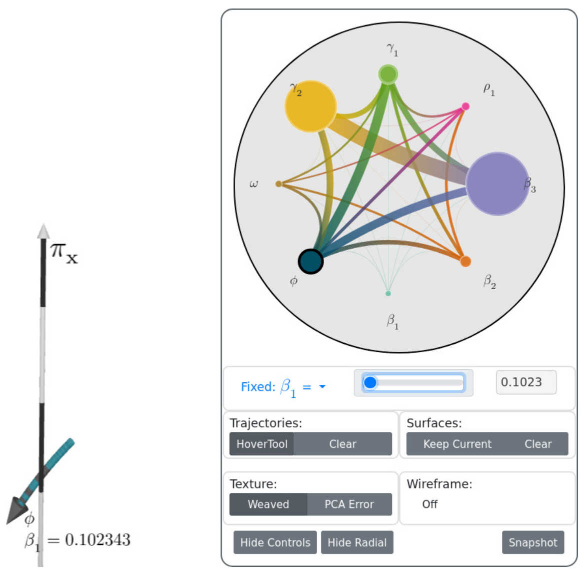
In its simplest form, our algorithm boils down to higher-order tensor PCA, on top of which we contributed three conceptual and computational developments:

- The abstraction of taking function slices 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 first- 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 multidimensional 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 types of inter-variable relationships are mapped to specific patterns in 3D curves and shapes, and how individual versus joint-variable effects stand out from our visualization. Although our examples only considered uniformly distributed input variables, use cases with non-uniform distributions are well contemplated in the general framework of variance-based sensitivity analysis and are supported in our implementation. 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

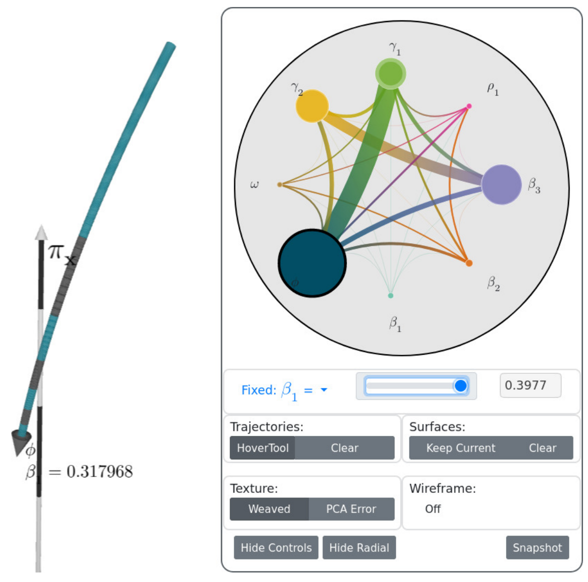
As outlined in the introduction, in this paper we have focused on dense parameter spaces. In particular, no scattered data set (i.e. given collection of samples at fixed locations) was considered as a ground-truth input. Although we pursued an important domain of application,



Fixed: β_1 = 0.1023

Wireframe: Off

(a) Low transmission rate β_1

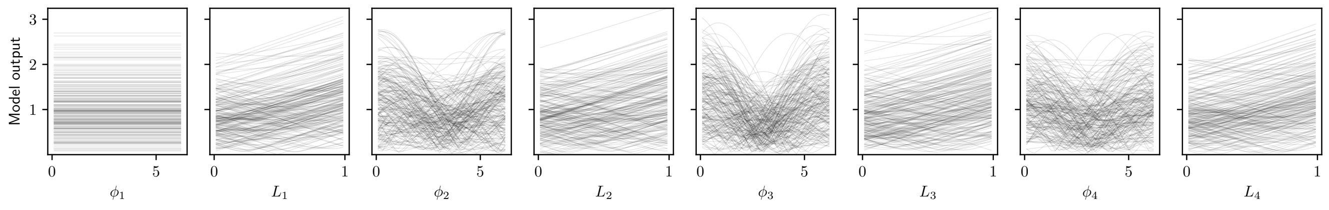


Fixed: β_1 = 0.3977

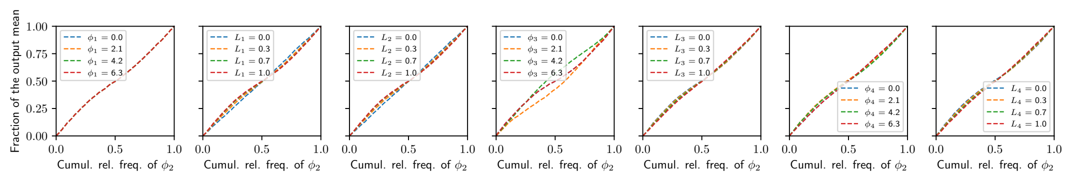
Wireframe: Off

(b) High transmission rate β_1

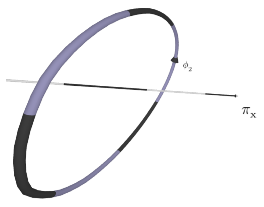
Fig. 14: *Ebola Spread*: using the blue slider on the right sidebar, we vary the viral transmission rate β_1 and study its impact on the effectiveness of hospitalization rate ϕ (large dark blue curve arrows) at reducing Ebola infection rate R_0 (black axis). We see that ϕ is extremely important at high values of β_1 as shown by the large blue curve in (b), where it can halve the overall infection rate. Correspondingly, the dark blue node in the radial widget on the right figure becomes the largest one in (b).



(a) Sliceplorer plots [38] for all eight variables.



(b) CSM plots [9] for ϕ_2 while varying the other seven parameters one at a time.



(c) PP-curve for ϕ_2.

Fig. 15: *Robot Arm* model analyzed with (a) Sliceplorer and (b) CSM, two prior techniques for parameter space visualization. Although these methods are able to capture non-influential variables, such as ϕ_1, and reflect how strong the interactions between variables are, they do not support the visual analysis of variable tuples and do not succinctly express properties such as periodicity, which can better be visualized by our PP-curve in (c).

the discrete case remains an equally attractive 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.

The current visualization system supports a limited number of dimensions only; for example, a plot matrix would become cluttered for a model with more than a score of dimensions. Efforts to support higher dimensionality will have to focus more on this aspect, which is more of a limitation than the numerical backend (the tensor train decomposition has been used for hundreds dimensions in the literature).

Appendix

A Details on Cross-approximation

Cross-approximation is a surrogate modeling technique that incrementally builds a compressed TT tensor approximating a target black-box function by evaluating the function at a batch of samples per iteration. These samples are chosen in a clever adaptive manner so as to minimize the model's relative error using the smallest possible number of evaluations. The error is defined as $\|\tilde{\mathbf{X}}-\mathbf{X}\|/\|\mathbf{X}\|$, where $\mathbf{X}$ are the groundtruth evaluations and $\tilde{\mathbf{X}}$ is the model's prediction. These samples are also used as a validation set, and the process is stopped as soon as the error is below a user-defined threshold ϵ. This is a standard way to approximate the generalization error when building compressed tensors, and the estimation is known [31] to be reliable in practice.

We used $\epsilon := 10^{-4}$ in all cases, which required between 10^5 and 10^7 samples and resulted in a few seconds' time to build each TT metamodel. Although not needed for our work, note that there are alternatives to cross-approximation for the case when function evaluation is expensive [23].

In this paper we have used the implementation of cross-approximation that is released in the *ttpy* Python toolbox [1].

B Step-by-step Illustration of the Algorithm

In Fig. 16 we show a flowchart of the proposed algorithm using an intermediate visualization along all its steps. To this end, we take a simplistic 3D use case function that is simple to understand and can directly be displayed both as a 3D rendering as well as a sequence of 2D slices.

C Operations in the TT Compressed Domain

The TT format allows for efficient computation of several operations without having to explicitly decompress the tensor it represents. This appendix covers the three main operations that are leveraged in the paper.

Multidimensional Integrals

Suppose our original function is expressed as a TT with cores $\mathcal{T}^{(1)}, \dots, \mathcal{T}^{(N)}$. To marginalize the n-th variable away (i.e. compute the expected value along x_n), one simply needs to replace the core $\mathcal{T}^{(n)}$ by $\widehat{\mathcal{T}}^{(n)}$ where

$$\widehat{\mathcal{T}}^{(n)} := \frac{1}{I_n} \sum_{i=1}^{i=I_n} \mathcal{T}^{(n)}[i],$$

where I_n is the shape of the grid along dimension n. In other words, we simply need to average the n-th core along its second dimension.

Element-wise Arithmetics

If two tensors are in the TT format with cores $\mathcal{T}_1^{(1)}, \dots, \mathcal{T}_1^{(N)}$ and $\mathcal{T}_2^{(1)}, \dots, \mathcal{T}_2^{(N)}$, then their element-wise sum $\mathcal{T}_3 = \mathcal{T}_1 + \mathcal{T}_2$ is given by the following cores:

$$\mathcal{T}_3^{(n)}[i_n] := \begin{cases} \left(\mathcal{T}_1^{(1)}[i_1];\ \mathcal{T}_2^{(1)}[i_1] \right) & \text{if } n = 1; \\ \begin{pmatrix} \mathcal{T}_1^{(n)}[i_n]; & 0 \\ 0; & \mathcal{T}_2^{(n)}[i_n] \end{pmatrix} & \text{if } n = 2, \dots, N-1; \\ \begin{pmatrix} \mathcal{T}_1^{(N)}[i_N] \\ \mathcal{T}_2^{(N)}[i_N] \end{pmatrix} & \text{if } n = N. \end{cases}$$

To subtract two tensors, it is enough to flip the sign of the second one by flipping the sign of its first core, and then sum them as above.

The operations just described allow us to obtain up to $\mathcal{M}$ (line 8 from Alg. 1) in the TT format.

PCA Projection

In practice, once we have $\mathcal{M}$, we found it more efficient to compute its PCA projection in the compressed domain via the SVD decomposition as follows. Let $\mathcal{M}$, which represents a matrix of size $I^K \times I^{N-K}$, be given by TT cores $\mathcal{M}^{(1)}, \dots, \mathcal{M}^{(N)}$. We proceed as follows:

1. We *left-orthogonalize* [31] the TT. This is equivalent to finding the RQ decomposition of $\mathcal{M}$, with the first K cores representing the $\mathbf{R}$ part (a matrix of shape $I^K \times R$, where R is the K-th TT rank of $\mathcal{M}$) and the remaining cores the $\mathbf{Q}$ part (of shape $R \times I^{N-K}$).
2. We discard $\mathbf{Q}$ by keeping the first K cores only.
3. We perform *rank-truncation* [31] on the last rank in order to decrease it to 3. This involves a SVD decomposition on the last core and is equivalent to keeping the three leading left singular vectors of $\mathbf{R}$.

The resulting tensor has shape $I^K \times 3$, as desired, and represents a mapping from the original I^K grid to $\mathbb{R}^3$, i.e. a discretized K-dimensional manifold, that is as close as possible to $\mathcal{M}$ in the L^2 sense.

Compliance with Ethical Standards

The authors declare that they have no conflict of interest. This work involved no human participants or animals.

Data Availability Statement

The datasets generated during and analyzed during the current study are available in the *ttrecipes* repository: `https://github.com/rballester/ttrecipes/tree/master/models`. The code for the method is available in `https://github.com/ghalter/pca-webapp-paper`, and a deployed application can be found at `http://pcatestwebapp.westeurope.cloudapp.azure.com/`.

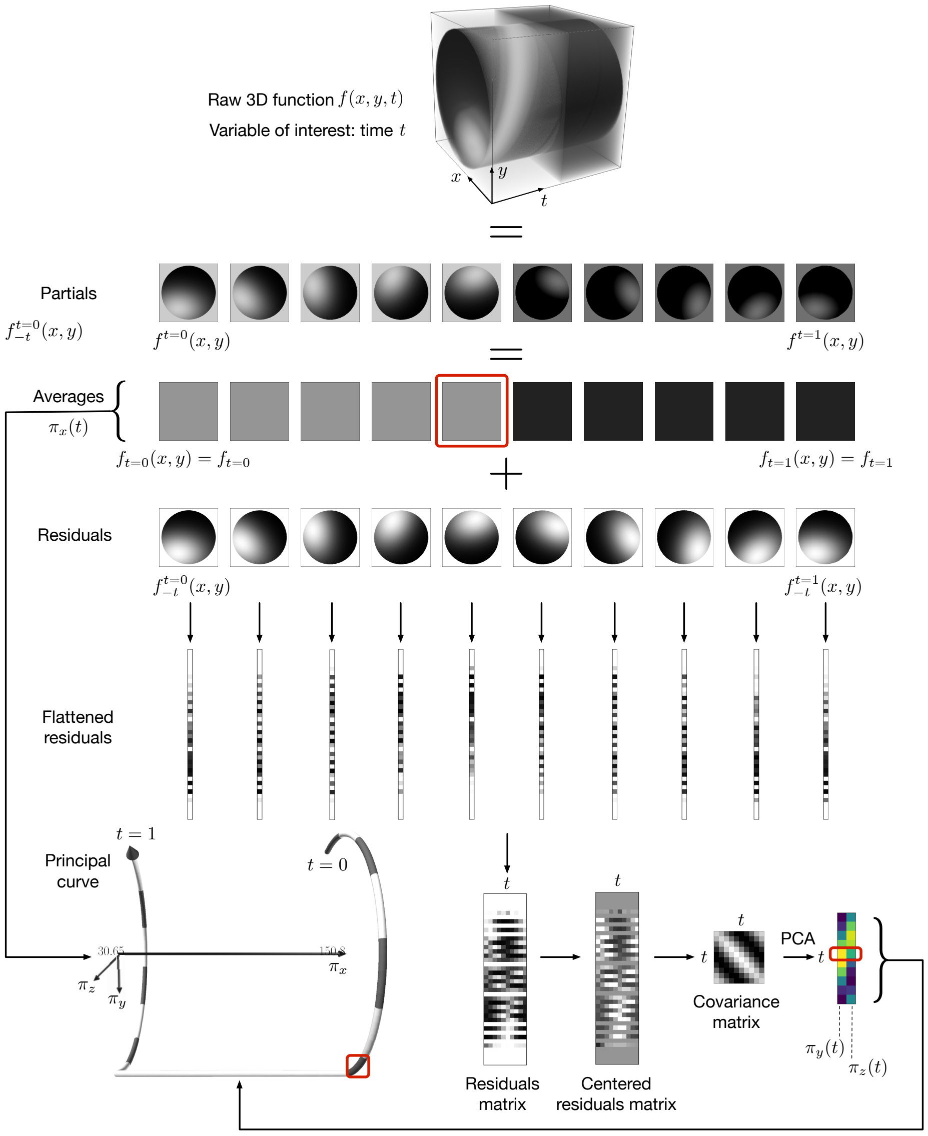


Fig. 16: Flowchart of the proposed method. For illustrative purposes, we consider a 3D function f representing a grayscale video: $f(x,y,t)$ is the intensity of pixel (x,y) at time $0 \leq t \leq 1$. The variable of interest is t. The video captures a static ball in a fixed scene and camera position where the light source moves in a circular fashion. In addition, the light is suddenly dimmed in the middle of the video, which is manifested as a jump in t's curve along the π_x axis. The three red boxes correspond to the coordinates $(\pi_x(t), \pi_y(t), \pi_z(t))$ at time $t = 0.5$, right before the light source is dimmed. The original video is a tensor of shape $256 \times 256 \times 256$; for the sake of presentation, only 10 timesteps are shown in this diagram.

References

1. ttpy: a tensor train toolbox implemented in Python. `http://github.com/oseledets/ttpy`
2. ttrecipes: a high-level Python library of tensor train numerical utilities. `https://github.com/rballester/ttrecipes`
3. An, J., Owen, A.B.: Quasi-regression. Journal of Complexity **17**(4), 588–607 (2001)
4. Ballester-Ripoll, R., Paredes, E.G., Pajarola, R.: A surrogate visualization model using the tensor train format. In: SIGGRAPH ASIA 2016 Symposium on Visualization, pp. 13:1–13:8 (2016)
5. Ballester-Ripoll, R., Paredes, E.G., Pajarola, R.: Sobol tensor trains for global sensitivity analysis. ArXiv e-print 1712.00233 (2017)
6. Ballester-Ripoll, R., Paredes, E.G., Pajarola, R.: Tensor algorithms for advanced sensitivity metrics. SIAM/ASA Journal on Uncertainty Quantification **6**(3), 1172–1197 (2018)
7. Banzhaf, J.F.: Weighted voting doesn't work: A mathematical analysis. Rutgers Law Review **19**, 317–343 (1965)

8. Bigoni, D., Engsig-Karup, A., Marzouk, Y.: Spectral tensor-train decomposition. SIAM Journal on Scientific Computing **38**(4), A2405–A2439 (2016)
9. Bolado-Lavin, R., Castaings, W., Tarantola, S.: Contribution to the sample mean plot for graphical and numerical sensitivity analysis. Reliability Engineering and System Safety **94**(6), 1041–1049 (2009). DOI https://doi.org/10.1016/j.ress.2008.11.012. URL `https://www.sciencedirect.com/science/article/pii/S0951832008002743`
10. Bruckner, S., Moeller, T.: Result-driven exploration of simulation parameter spaces for visual effects design. IEEE Transactions on Visualization and Computer Graphics **16**(6), 1467–1475 (2010)
11. Castura, J.C., Baker, A.K., Ross, C.F.: 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. Chan, Y.H., Correa, C.D., Ma, K.L.: Flow-based scatterplots for sensitivity analysis. In: IEEE Symposium on Visual Analytics Science and Technology, pp. 43 – 50 (2010)
13. Constantine, P.G.: Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2015)
14. Correa, C., Lindstrom, P., Bremer, P.T.: Topological spines: A structure-preserving visual representation of scalar fields. IEEE Transactions on Visualization and Computer Graphics **17**(12), 1842–1851 (2011)
15. Diaz, P., Constantine, P.G., Kalmbach, K., Jones, E., Pankavich, S.: A modified SEIR model for the spread of Ebola in western Africa and metrics for resource allocation. Applied Mathematics and Computation **324**, 141–155 (2018)
16. Dubourg, V., Sudret, B., Deheeger, F.: Metamodel-based importance sampling for structural reliability analysis. Probabilistic Engineering Mechanics **33**, 47–57 (2013)
17. Fruth, J., Roustant, O., Muehlenstaedt, T.: The fanovaGraph package: Visualization of interaction structures and construction of block-additive kriging models. HAL preprint 00795229 (2013). URL `https://hal.archives-ouvertes.fr/hal-00795229`
18. Gallagher, M., Downs, T.: Visualization of learning in multilayer perceptron networks using principal component analysis. IEEE Transactions on Systems, Man, and Cybernetics, Part B **33**(1), 28–34 (2003)
19. Gardner, T.S., Dolnik, M., Collins, J.J.: A theory for controlling cell cycle dynamics using a reversibly binding inhibitor. In: National Academy of Sciences of the United States of America, vol. 95(24), pp. 14,190–14,195 (1998)
20. Gerber, S., Bremer, P.T., Pascucci, V., Whitaker, R.: Visual exploration of high dimensional scalar functions. IEEE Transactions on Visualization and Computer Graphics **16**(6), 1271–1280 (2010)
21. Goldbeter, A.: A minimal cascade model for the mitotic oscillator involving cyclin and CDC2 kinase. In: National Academy of Sciences of the United States of America, vol. 88(20), pp. 9107 – 9111 (1991)
22. Gorodetsky, A.A., Jakeman, J.D.: Gradient-based optimization for regression in the functional tensor-train format. ArXiv e-print 1801.00885 (2018). URL `https://arxiv.org/abs/1801.00885`
23. Grasedyck, L., Kressner, D., Tobler, C.: A literature survey of low-rank tensor approximation techniques. GAMM-Mitteilungen **36**(1), 53–78 (2013)
24. Heinrich, J., Weiskopf, D.: State of the art of parallel coordinates. In: Proceedings Eurographics (State of the Art Reports), pp. 95–116 (2013)
25. Insuasty, E., Van den Hof, P.M.J., Weiland, S., Jansen, J.D.: Flow-based dissimilarity measures for reservoir models: a spatial-temporal tensor approach. Computational Geosciences **21**(4), 645–663 (2017)
26. Iooss, B., Lemaître, P.: A Review on Global Sensitivity Analysis Methods, pp. 101–122. Springer US, Boston, MA (2015)
27. Knutsson, H.: Representing local structure using tensors. Tech. Rep. LiTH-ISY-I, 8765-4321, Computer Vision Laboratory, Linköping University (1989)
28. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009)
29. Liu, S., Maljovec, D., Wang, B., Bremer, P.T., Pascucci, V.: Visualizing high-dimensional data: Advances in the past decade. IEEE Transactions on Visualization and Computer Graphics **23**(3), 1249–1268 (2017)
30. Oseledets, I.V.: Tensor-train decomposition. SIAM Journal on Scientific Computing **33**(5), 2295–2317 (2011)
31. Oseledets, I.V., Tyrtyshnikov, E.: TT-cross approximation for multidimensional arrays. Linear Algebra Applications **432**(1), 70–88 (2010)
32. Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global Sensitivity Analysis: The Primer. John Wiley & Sons, Ltd. (2008)
33. Sedlmair, M., Brehmer, M., Ingram, S., Munzner, T.: Dimensionality reduction in the wild: Gaps and guidance. Tech. Rep. TR-2012-03, University of British Columbia, Department of Computer Science (2012)
34. Sedlmair, M., Heinzl, C., Bruckner, S., Piringer, H., Möller, T.: Visual parameter space analysis: A conceptual framework. IEEE Transactions on Visualization and Computer Graphics **20**(12), 2161–2170 (2014)
35. Shao, L., Mahajan, A., Schreck, T., Lehmann, D.J.: Interactive regression lens for exploring scatter plots. Computer Graphics Forum **36**(3), 157–166 (2017)
36. Sobol', I.M.: Sensitivity estimates for nonlinear mathematical models (in Russian). Mathematical Models **2**, 112–118 (1990)
37. Torsney-Weir, T., Saad, A., Möller, T., Hege, H.C., Weber, B., Verbavatz, J.M.: Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. IEEE Transactions on Visualization and Computer Graphics **17**(12), 1892 – 1901 (2011)
38. Torsney-Weir, T., Sedlmair, M., Möller, T.: Sliceplorer: 1D slices for multi-dimensional continuous functions. Computer Graphics Forum **36**(3), 167 – 177 (2017)
39. Turk, M., Pentland, A.: Eigenfaces for recognition. Journal of Cognitive Neuroscience **3**(1), 71–86 (1991)
40. Vasilescu, M.A.O., Terzopoulos, D.: Multilinear analysis of image ensembles: Tensorfaces. In: European Conference on Computer Vision, pp. 447–460 (2002)
41. Vervliet, N., Debals, O., Sorber, L., Lathauwer, L.D.: Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis. IEEE Signal Processing Magazine **31**(5), 71–79 (2014)
42. Ward, M.O., LeBlanc, J.T., Tipnis, R.: N-land: a graphical tool for exploring N-dimensional data. In: Proceedings Computer Graphics International, pp. 95–116 (1994)
43. van Wijk, J.J., van Liere, R.: HyperSlice: Visualization of scalar functions of many variables. In: Proceedings IEEE VIS, pp. 119–125 (1993)

44. Wu, Q., Xia, T., Chen, C., Lin, H.Y.S., Wang, H., Yu, Y.: Hierarchical tensor approximation of multidimensional visual data. IEEE Transactions on Visualization and Computer Graphics **14**(1), 186–199 (2008)