TENSOR ALGORITHMS FOR ADVANCED SENSITIVITY METRICS*

RAFAEL BALLESTER-RIPOLL†, ENRIQUE G. PAREDES†, AND RENATO PAJAROLA†

Abstract. Following up on the success of the analysis of variance (ANOVA) decomposition and the Sobol indices (SI) for global sensitivity analysis, various related quantities of interest have been defined in the literature including the effective and mean dimensions, the dimension distribution, and the Shapley values. Such metrics combine up to exponential numbers of SI in different ways and can be of great aid in uncertainty quantification and model interpretation tasks, but are computationally challenging. We focus on surrogate based sensitivity analysis for independently distributed variables, namely via the tensor train (TT) decomposition. This format permits flexible and scalable surrogate modeling and can efficiently extract all SI at once in a compressed TT representation of their own. Based on this, we contribute a range of novel algorithms that compute more advanced sensitivity metrics by selecting and aggregating certain subsets of SI in the tensor compressed domain. Drawing on an interpretation of the TT model in terms of deterministic finite automata, we are able to construct explicit auxiliary TT tensors that encode exactly all necessary index selection masks. Having both the SI and the masks in the TT format allows efficient computation of all aforementioned metrics, as we demonstrate in a number of example models.

Key words. Variance-based sensitivity analysis, surrogate modeling, tensor train decomposition, Sobol indices

AMS subject classifications. 65C20, 15A69, 49Q12

1. Introduction.

Variance-based sensitivity analysis (SA) is a fundamental tool in many disciplines including reliability engineering, risk assessment and uncertainty quantification. It captures the behavior of simulations and systems in terms of how much of their output's variability is explained by each input (and combinations of inputs), and has received a great deal of academic and industrial interest over the last decades. These efforts have resulted in widely popular metrics such as the *Sobol indices* (SI) [52, 25] and an increasing number of more recent related quantities of interest (QoI). They help analysts assess which groups of variables have the strongest influence on the output's uncertainty and, for example, which ones may be frozen with the least possible impact [47].

A number of long-standing hurdles make such tasks challenging. First of all, directly sampling the whole domain of variables is rarely a feasible option. Usually one has either a given sparse set of fixed samples, or a simulation/experiment that can be run on demand with arbitrary parameters (for example the so-called *non-intrusive modeling*, also known as *black-box sampling*). Uncertainty estimations are thus often bound to have a margin of error. Second, the well-known *curse of dimensionality* poses a challenge for high-parametric models. Points tend to lie *far* from each other as the number of variables N grows, and a rather large number of samples may be required in practice to attain a reasonable accuracy. Furthermore, the number of possible index combinations and sensitivity metrics scales exponentially with N. Several algorithms and sampling schemes have been proposed that can partially tackle this problem; for example to estimate specific aggregated indices (e.g. the *total effects*) or as exact formulas to compute analytical values from certain classes of mathematical functions. Many methods limit themselves to computing indices that are relative to

* Submitted to the editors: 6th December 2017.

Funding: This work was partially supported by the University of Zurich's Forschungskredit "Candoc", grant number FK-16-012.

†Department of Informatics, University of Zurich (rballester@ifi.uzh.ch, egparedes@ifi.uzh.ch, pajarola@ifi.uzh.ch).

single variables only. Unfortunately, such simplifications risk overlooking sizable joint interactions. Often, an effect due to a specific combination of 2 or 3 variables might be stronger and more significant than the mere knowledge that these variables are important on their own.

A more powerful strategy for SA consists in using a limited set of samples to train a regressor that acts as a *surrogate model*, i.e. a routine that can estimate the true model's output for any combination of input values. This has become a standard choice for many SA tasks [42, 58, 29, 26], especially when many samples are needed. Even though such models are typically very fast to evaluate, sampling schemes operating on them still suffer from the curse of dimensionality, and certain QoIs or queries can be highly time-consuming to compute. In particular, some advanced metrics are defined on and combine many or even the whole set of SI. Even though interactions involving many variables tend to be very small in practice, there is an exponentially large number of them; hence their aggregated contributions should not be ignored in general.

The present work tackles surrogate modeling based SA under the paradigm of low-rank tensor decompositions, namely the *tensor train* (TT) model [33]. This model lends itself very well to variance-based SA as the SI can be extracted directly from its compressed representation without explicit sampling [44, 13, 3]. Throughout this paper we assume that a low-rank TT tensor surrogate exists that approximately predicts the model behavior at all possible input variable combinations. This assumption holds for many families of models [18], including many with high orders of variable interactions, and also in the presence of categorical variables. Many algorithms have been proposed to build such TT representations, be it via fixed sets of samples [55, 19, 20], via adaptively sampling black-box simulations and analytical functions [35, 50, 7], or from other alternative low-rank tensor decompositions [22, 27, 3]; see also [21, 2]. In this work we focus on adaptive sampling by the so-called *cross-approximation* technique [35].

We contribute a range of procedures to compute the *effective dimension* (in the *superposition*, best-possible ordering *truncation*, and *successive* senses [8, 30]), the *mean dimension* [8], the full *dimension distribution* [37], and the *Shapley values* [51, 39, 40]. Current state-of-the-art approaches for these advanced metrics are narrow in scope and/or face important limitations: [53] resorts to randomly sampling the vast space of possible variable permutations to approximate the Shapley values; [31] is able to estimate statistical moments of the dimension distribution; and [28] approximates some effective dimensions by using bounds on related surrogate metrics, a method that is less effective for higher-order interaction models [5]. In contrast, we propose to use a highly compact data structure, the Sobol tensor train [3]. To the best of our knowledge, ours is the first framework that can obtain all these metrics in an efficient manner. Our algorithms exploit the fact that a certain class of finite automata can be compactly encoded using tensor networks, in particular including the TT format (see [12] for an early application of this interpretation for eigenstate energy minimization, and [43] in the context of string weighted automata). We want to highlight that these automata-like representations are exact, i.e. compress masks of size 2^N without loss, and therefore do not introduce additional error when manipulating the entries in the Sobol TT. Thanks to the advantageous numerical properties of the TT model, the proposed approach is flexible and scales well with the model dimensionality, also when the queried metrics are defined as aggregations of up to exponential numbers of other indices. Fig. 1 summarizes the state-of-the-art on TT-based SA and the algorithms here contributed, which build on the work from [3] and thus reinforce the advantages

of using the Sobol TT representation altogether.

The rest of the paper is organized as follows. Sec. 2 reviews the main definitions and concepts we use, including the ANOVA decomposition, the Sobol indices, and all other sensitivity metrics considered. Sec. 3 outlines the mathematical tools that are fundamental for our algorithms: the tensor train decomposition (TT) as a framework for surrogate modeling and the Sobol tensor train. In Sec. 4 we contribute and construct explicitly several TT tensors with 2^N entries which behave as deterministic finite automata. In Sec. 5 we show how these automata can be combined with Sobol TTs to efficiently produce all sensitivity metrics listed in Sec. 2. Numerical results are presented in Sec. 6, and concluding remarks in Sec. 7.

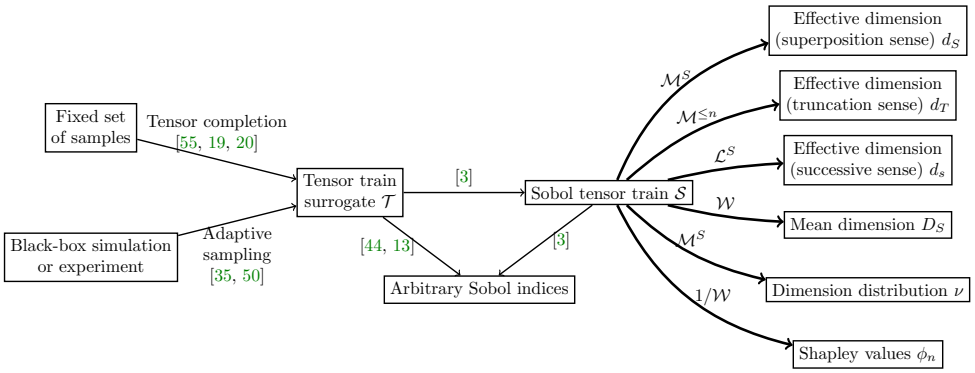

Fig. 1: Previous work has established the TT model as a valuable tool for sensitivity analysis. Based on this, in this paper we explicitly construct several tensors (bold edges) that select and aggregate indices from $\mathcal{S}$ in convenient ways and produce a variety of advanced QoIs (rightmost boxes).

2. Sensitivity Analysis: Definitions and Metrics. We write multiarrays (tensors) differently depending on their dimension: italic scalars (e.g. s), vectors with boldface italics (e.g. $\boldsymbol{u}$), matrices with boldface capitals (e.g. $\mathbf{U}$), and higher-order tensors with calligraphic capitals (e.g. $\mathcal{T}$). We point to their elements via NumPy-like notation, for example $\mathcal{T}[:, :k, :]$ are the slices $0, \dots k-1$ of a 3D tensor along its second axis. The Kronecker and element-wise products for matrices and tensors are written as $\mathbf{U} \otimes \mathbf{V}$ and $\mathcal{A} \circ \mathcal{B}$, respectively. Our accuracy metric between a tensor $\mathcal{T}$ and its approximation $\widetilde{\mathcal{T}}$ (this includes vectors, matrices, etc.) is always the *relative error*: $\|\mathcal{T} - \widetilde{\mathcal{T}}\|/\|\mathcal{T}\|$, where $\|\cdot\|$ is the Frobenius norm (defined as the Euclidean norm of the tensor's vectorization).

In this paper we often deal with tensors that have size 2^N and use them to index tuples of model variables. To access a position in a 2^N TT tensor indexed by a tuple $\boldsymbol{\alpha} \subseteq \{1, \dots, N\}$ we use subscripts: $\mathcal{T}_{\boldsymbol{\alpha}} \equiv \mathcal{T}[\alpha_1, \dots, \alpha_N]$ where $\alpha_n = 1$ iff $n \in \boldsymbol{\alpha}$, and 0 otherwise. For example, if $\mathcal{T}$ is a 3D tensor of size $2 \times 2 \times 2$, then $\mathcal{T}_{2,3}$ denotes the element $\mathcal{T}[0, 1, 1]$. We write complements of tuples as $-\boldsymbol{\alpha} := \{1, \dots, N\} \setminus \boldsymbol{\alpha}$.

2.1. ANOVA Decomposition and Sobol Indices. Let $f : \Omega \to \mathbb{R}$ be an L^2-integrable function on a rectangle $\Omega = [0, 1]^N$ and $F(\boldsymbol{x}) = F_1(x_1) \cdots F_N(x_N)$ a separable probability density function on Ω. The *ANOVA decomposition*, also known as the Sobol or Hoeffding decomposition [24, 52], splits up f into 2^N terms, each of which depends on a different subset of its input variables $\{1, \dots, N\}$:

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \subseteq \{1,\dots,N\}} f_{\boldsymbol{\alpha}}(\boldsymbol{x}) \tag{1}$$

where each $f_{\boldsymbol{\alpha}}(\boldsymbol{x})$ only depends effectively on $\boldsymbol{x}_{\boldsymbol{\alpha}}$ and is built as

$$\int_{\Omega_{-\boldsymbol{\alpha}}} \left(f(\boldsymbol{x}) - \sum_{\boldsymbol{\beta} \subsetneq \boldsymbol{\alpha}} f_{\boldsymbol{\beta}}(\boldsymbol{x}_{\boldsymbol{\beta}}) \right) dF_{-\boldsymbol{\alpha}}(\boldsymbol{x}_{-\boldsymbol{\alpha}}) \tag{2}$$

with $f_\emptyset(\boldsymbol{x}) = f_\emptyset = \mathbb{E}[f] = \int_\Omega f(\boldsymbol{x})\, dF(\boldsymbol{x})$. This defines a partition of the model's statistical variance as the sum of variances of each subfunction: $\mathrm{Var}[f] = \sum_{\boldsymbol{\alpha} \neq \emptyset} \mathrm{Var}[f_{\boldsymbol{\alpha}}]$.

The *variance components*, here denoted as $S_{\boldsymbol{\alpha}}$, are the relative variance contributions of each subfunction (except $f_\emptyset$), normalized by the total variance: $S_{\boldsymbol{\alpha}} := \mathrm{Var}[f_{\boldsymbol{\alpha}}]/\mathrm{Var}[f]$ for all $\boldsymbol{\alpha} \neq \emptyset$. They are nonnegative and sum up to 1. They are thus interpretable in terms of set cardinalities and can be used to define a set algebra. The *Sobol indices*, or SI for short, are specific aggregations and combinations of variance components. There are two main types, namely the *total Sobol indices* $\mathcal{S}^T$ (sometimes also known as *total effects* or *first-order indices*) and the *closed Sobol indices* $\mathcal{S}^C$ [38]:

$$S^T_{\boldsymbol{\alpha}} := \sum_{\boldsymbol{\beta} \cap \boldsymbol{\alpha} \neq \emptyset} S_{\boldsymbol{\beta}}, \;\; S^C_{\boldsymbol{\alpha}} := \sum_{\boldsymbol{\beta} \subseteq \boldsymbol{\alpha}} S_{\boldsymbol{\beta}} \tag{3}$$

which satisfy

$$\begin{gathered} 0 \leq S_{\boldsymbol{\alpha}} \leq S^C_{\boldsymbol{\alpha}} \leq S^T_{\boldsymbol{\alpha}} \leq 1 \\ S^T_{\boldsymbol{\alpha}} = 1 - S^C_{-\boldsymbol{\alpha}} \text{ and } S^C_{\boldsymbol{\alpha}} = 1 - S^T_{-\boldsymbol{\alpha}} \\ \sum_n S_n \leq 1 \leq \sum_n S^T_n. \end{gathered} \tag{4}$$

2.2. Effective Dimension. More advanced sensitivity metrics take into account the size of the variable tuple $\boldsymbol{\alpha}$. One example is the *effective dimension*, which has been defined in three ways at least:

- *Superposition sense* [8]:

$$d_S := \arg\min_k \left\{ k \,\middle|\, \sum_{\boldsymbol{\alpha} \,\middle|\, |\boldsymbol{\alpha}| \leq k} S_{\boldsymbol{\alpha}} \geq 1 - \epsilon \right\}$$

- *Truncation sense* [8]:

$$d_T := \arg\min_k \left\{ k \,\middle|\, S^C_{\{1\dots k\}} \geq 1 - \epsilon \right\}$$

- *Successive sense* [30]:

$$d_s := \arg\min_k \left\{ k \,\middle|\, \sum_{\boldsymbol{\alpha} \,\middle|\, \mathrm{len}(\boldsymbol{\alpha}) \leq k} S_{\boldsymbol{\alpha}} \geq 1 - \epsilon \right\}$$

In the above equation, $\text{len}(\boldsymbol{\alpha}) := \arg\max_n\{n \in \boldsymbol{\alpha}\} - \arg\min_n\{n \in \boldsymbol{\alpha}\} + 1$ and ϵ is a small tolerance for unexplained effects (say, 5%). The superposition sense d_S measures the minimal order of interactions needed to capture most of the model variability. In other words, it means that the model f is roughly a sum of d_S-dimensional subfunctions; interactions involving higher numbers of variables may be safely discarded. On the other hand, d_T is the number of *leading* variables needed to capture a $1-\epsilon$ fraction of the variance. If we allow reordering the variables, d_T is the minimal integer such that there exists *some* tuple $\boldsymbol{\alpha}$ with $|\boldsymbol{\alpha}| = d_T$ such that $S^C_{\boldsymbol{\alpha}} \geq 1-\epsilon$. Last, d_s is informative when all variables have an inherent ordering, for example in a time series; it means that the model consists of subfunctions that depend on *neighboring* variables only.

2.3. Mean Dimension. The *mean dimension* in the superposition sense [37] is the expected value of $|\boldsymbol{\alpha}|$, if one were to select $\boldsymbol{\alpha}$ with probability proportional to its variance:

$$D_S := \sum_{\boldsymbol{\alpha}} S_{\boldsymbol{\alpha}} \cdot |\boldsymbol{\alpha}|. \tag{5}$$

With 3 variables, for example, $D_S = S_1 + ... + 2 \cdot S_{1,2} + ... + 3 \cdot S_{1,2,3}$. This metric is a non-integer number, unlike the effective dimension, and measures the average complexity or dimensionality of a model. A result by Liu and Owen [31] states that the mean dimension equals the sum of all first-order total SI: $D_S = \sum_{n=1}^{N} S^T_n$.

2.4. Dimension Distribution. Denoted as ν, it was defined by Owen [37] as the probability mass function of the random variable $|\boldsymbol{\alpha}|$, if one were to select $\boldsymbol{\alpha}$ as described before. It is a discrete variable over the domain $\{1, \ldots, N\}$, and each value $1 \leq n \leq N$ has probability

$$\nu(n) = \sum_{\boldsymbol{\alpha} \,\big|\, |\boldsymbol{\alpha}|=n} S_{\boldsymbol{\alpha}}. \tag{6}$$

Its expected value is the mean dimension D_S. Also, knowing the dimension distribution allows a direct computation of the effective dimension in the superposition sense: if we write $\bar{\nu} := \text{CDF}(\nu)$, then $d_S = \lceil \bar{\nu}^{-1}(1-\epsilon) \rceil$.

2.5. Shapley Values. They originated in game theory [51] to determine the just retributions that each individual player $1, ..., N$ should receive from a set of potential coalitional tasks:

$$\phi_n := \sum_{\boldsymbol{\alpha} \subseteq -\{n\}} \frac{|\boldsymbol{\alpha}|!(N - |\boldsymbol{\alpha}| - 1)!}{N!} \cdot (C_{\boldsymbol{\alpha} \cup \{n\}} - C_{\boldsymbol{\alpha}}). \tag{7}$$

Here $C_{\boldsymbol{\alpha}}$ represents the productivity or goodness of each coalition $\boldsymbol{\alpha}$. Recently, Shapley values have been reinterpreted as variance contributions in the context of SA, whereby a connection with the Sobol indices was established [39]: If the productivity of each subset of variables $\boldsymbol{\alpha}$ is taken to be $C_{\boldsymbol{\alpha}} := S^C_{\boldsymbol{\alpha}}$, then the n-th Shapley value can be computed with the simpler formula

$$\phi_n = \sum_{\boldsymbol{\alpha}|n\in\boldsymbol{\alpha}} \frac{S_{\boldsymbol{\alpha}}}{|\boldsymbol{\alpha}|}. \tag{8}$$

For example, $\phi_1 = S_1 + S_{1,2}/2 + S_{1,3}/2 + S_{1,2,3}/3$. This interpretation has been further investigated in various settings [53, 26, 40] and found to be a good compromise between the more granular closed indices S^C and the coarser total indices S^T: it holds that $S_n = S_n^C \leq \phi_n \leq S_n^T \; \forall n = 1, \ldots, N$. It has been extended for the case of dependent input variables as well. The Shapley values ϕ also have the desirable property that their sum equals 1 [39], unlike for example the total effects.

3. Tensor Train Decomposition.

3.1. Fundamentals. The tensor train model (TT) is a tensor decomposition proposed by Oseledets [33] that is also known as *linear tensor network* in other communities [22, 10]. The format assigns each physical dimension to one 3D *core tensor* (see Fig. 2). To decompose a model f into a tensor $\mathcal{T}$ of size $I_1 \times \cdots \times I_N$, we first discretize each axis of the domain $\Omega = \Omega_1 \times \cdots \times \Omega_N$ so that $f(\boldsymbol{x}) \approx \mathcal{T}[\boldsymbol{i}]$ with $0 \leq i_n < I_n$ for $n = 1, \ldots, N$. Element-wise, each entry of $\mathcal{T}$ is a product of matrices:

$$\mathcal{T}[\boldsymbol{i}] = \mathcal{T}^{(1)}[0, i_1, :] \cdot \mathcal{T}^{(2)}[:, i_2, :] \cdot \ldots \cdot \mathcal{T}^{(N-1)}[:, i_{N-1}, :] \cdot \mathcal{T}^{(N)}[:, i_N, 0] =$$
$$\sum_{\boldsymbol{r=0}}^{\boldsymbol{R-1}} \mathcal{T}^{(1)}[0, i_1, r_1]\mathcal{T}^{(2)}[r_1, i_2, r_2] \ldots \mathcal{T}^{(N-1)}[r_{N-2}, i_{N-1}, r_{N-1}]\mathcal{T}^{(N)}[r_{N-1}, i_N, 0].$$

Every core $\mathcal{T}^{(n)}$ is thus a collection of matrices that are stacked along its second dimension (corresponding to the slices in Fig. 2). The matrix sizes $R_1, \ldots, R_{N-1}$ are known as *TT ranks* and capture the complexity of the compressed tensor. We sometimes write $\mathcal{T} = [[\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(N)}]]$ to denote a TT decomposition in terms of its cores.

Let us write $I := \max\{I_1, \ldots, I_N\}$ and $R := \max\{R_1, \ldots, R_{N-1}\}$. The storage cost of a TT tensor is $O(NIR^2)$, i.e. depends only linearly on the number of variables N. There is of course no free lunch: the upper bounds for R are high, namely $R = I^{N/2}$ and $O(I^N)$ elements are required to represent a tensor of size I^N in the TT format *exactly* in the worst case. This happens when the $(N/2)$-th *TT unfolding matrix* [33] (which has size $I^{N/2} \times I^{N/2}$ when N is even) has full rank. Fortunately, in many applications this large matrix has many small (or zero) singular values, and much lower values for R achieve sufficiently small error levels. See also the discussion below.

3.2. Black-box Sampling for TT Surrogate Modeling. The TT format is a powerful generalization of the matrix low-rank representation that can compactly encode the multidimensional structure of a wide family of functions and models. It has thus become a successful tool for high-dimensional interpolation and integration in physics and chemistry, and even in low dimensions via the so-called *tensorization* process [36] (which introduces artificial dimensions in a vector via reshaping operations).

Given a black-box routine that can be evaluated on demand, one can often build an accurate TT representation of it via an adaptive sampling scheme over a structured set of samples, for example *cross-approximation* [35]. This is the method we use in this

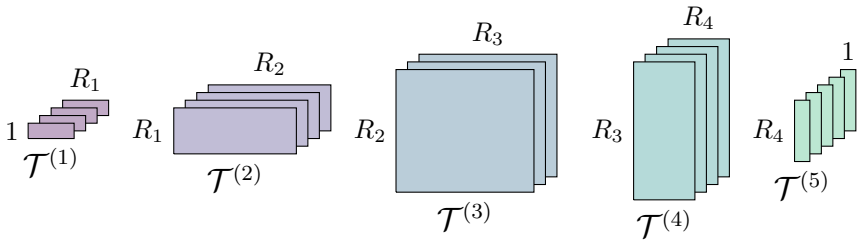

Fig. 2: A 5D tensor train approximating a tensor of spatial sizes $4 \times 4 \times 3 \times 4 \times 5$. Graphically, these sizes are the number of core slices (matrices) along the depth dimension, while the TT ranks $(1, R_1, R_2, R_3, R_4, 1)$ are distributed horizontally and vertically as matrix sizes. The ranks are usually larger around the central dimensions.

paper to build all considered models and, although an in-depth discussion would be out of scope, we give here an introduction. Cross-approximation is best understood as a generalization of the *pseudoskeleton* decomposition for matrices, which approximates a matrix in terms of a number R of its columns and rows. The main challenge here is how to select a good set of samples that is economic and yet spans the original matrix's vector space as accurately as possible. It was proven [17] that the best approximation is given by those columns and rows whose intersection has maximal determinant in modulus, and an alternating row/column selection heuristic was proposed that works well in practice [56]. For higher-order tensors, columns and rows generalize to *fibers*, i.e. samples acquired by fixing all parameters except one which is moved throughout its full range With cross-approximation these fibers are chosen as columns and rows of various *matricizations* of the original tensor, which they approximate by means of the pseudoskeleton decomposition. The more fibers are sampled, the larger TT cores one can construct and more accurate the approximation will be.

The numbers of TT-ranks $R_1, \ldots, R_{N-1}$ needed to achieve a good approximation (i.e. with low relative error) are unknown a priori. Adaptive algorithms estimate them by starting with small cores that are progressively expanded as new fibers are sampled. The process is stopped when the approximation is accurate enough, namely when the relative error $\|\hat{\boldsymbol{x}} - \boldsymbol{x}\| / \|\boldsymbol{x}\|$ between the new samples $\boldsymbol{x}$ and the current model's estimation $\hat{\boldsymbol{x}}$ is below a user-defined threshold. Thus, each sampling batch acts as a validation set plus stopping criterion. All in all, for every dimension n, $O(R_n I_n R_{n+1})$ samples are taken to fit a TT core of size $R_n \times I_n \times R_{n+1}$. The ranks often vary from core to core; for example, suppose that a model f is separable with respect to two groups of variables, $f(\boldsymbol{x}) = g(x_1, \ldots, x_k) \cdot h(x_{k+1}, \ldots, x_N)$. Then it holds that $R_k = 1$, and good sampling heuristics will realize this after casting very few fibers for that rank. Cross-approximation needs $O(NIR^2)$ samples overall to produce a compressed tensor of $O(NIR^2)$ coefficients, so the number of function evaluations is proportional to the number of degrees of freedom in the fitted model. The asymptotic complexity is $O(NIR^3)$ operations (since, among other things, linear systems of size up to $R \times R$ need to be solved), so tensors with ranks up to $R \approx 100$ are usually tractable on a regular workstation in a matter of seconds. This rank budget covers a large class of models, since low-rank structure is quite frequent in practice [18]. A number of variants have been proposed to structure the adaptive sampling plan [35, 50, 49, 7]. We make use of an *alternating minimal energy* strategy [14] to handle the rank selection,

as is readily provided in the *ttpy* toolbox [15].

On top of surrogate modeling [57, 2, 18], the TT model has been also used for sensitivity analysis as well [13, 44, 59, 7, 3]. For a more in-depth review on TT model building techniques from either fixed sets of samples or black-box settings, we also refer the reader to the surveys [21, 54].

3.3. Sobol Tensor Trains. The *Sobol tensor train* was recently introduced [3] as a compressed TT tensor that can be extracted from any N-variable TT surrogate model and approximately represents its full set of $2^N - 1$ variance components and SIs. Denoted by $\mathcal{S}$, the index for any tuple $\boldsymbol{\alpha}$ is approximated by the corresponding tensor entry $S_{\boldsymbol{\alpha}} \approx \mathcal{S}_{\boldsymbol{\alpha}} = \mathcal{S}[\alpha_1, \dots, \alpha_N]$. Since $\mathcal{S}$ is in the TT format, this entry is decompressed as a product of matrices:

$$\mathcal{S}^{(1)}[:, \alpha_1, :] \cdot \ldots \cdot \mathcal{S}^{(N)}[:, \alpha_N, :] \tag{9}$$

The Sobol TT contains a 0 at the corner: $\mathcal{S}_\emptyset = \mathcal{S}[0, ..., 0] = 0$. It has size 2^N (i.e. each α_n can only take values in $\{0, 1\}$) and therefore each core has just 2 slices, as illustrated in Fig. 3 for a 7D indexing example.

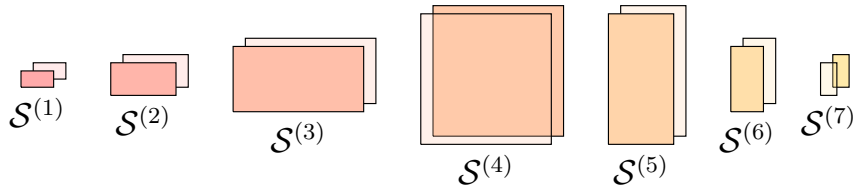

Fig. 3: A 7-variable model yields a Sobol TT $\mathcal{S}$ of size 2^7 [3]. As an example, multiplying the 7 highlighted slices yields the element $\mathcal{S}[0, 0, 0, 1, 0, 0, 1] = \mathcal{S}_{4,7} \approx S_{4,7}$.

Throughout this paper we compute a TT surrogate to approximate each model studied and extract its Sobol TT via the method described in [3]. The Sobol TT is, to our advantage, a highly compact representation for the complete set of indices. Furthermore, many aggregation and manipulation operations can be performed in the TT compressed domain including linear combinations (in $O(NIR^2)$ operations), differentiation/integration ($O(NIR^2)$ again), element-wise functions ($O(NIR^3)$ operations), etc.

4. Tensor Train Masks. The sensitivity metrics we overviewed in Sec. 2 rely on selecting and weighting various indices according to their tuple order $|\boldsymbol{\alpha}|$. We observe that these orders can be appropriately and compactly accounted for in the TT format, thanks to a connection between tensor networks and finite automata [12, 43]. Next we contribute the explicit construction of a number of weighted tensor masks and automata followed by their application for the direct computation of advanced metrics as further detailed in Sec. 5. All proposed TT masks are N-dimensional, i.e. use N cores, and have 2^N entries.

4.1. Hamming Weight Tensor. We define first the *Hamming weight tensor* $\mathcal{W}$, which stores at each position $\boldsymbol{\alpha} \in \{0, 1\}^N$ the number of '1' elements in $\boldsymbol{\alpha}$: $\mathcal{W}_{\boldsymbol{\alpha}} := |\boldsymbol{\alpha}|$. Note that $\mathcal{W}$ can be written as the sum of N separable terms. Each n-th term has size 2^N and adds 1 to each entry $\boldsymbol{\alpha}$ if and only if $\alpha_n = 1$:

$$\mathcal{W} = \sum_{n=1}^{N} \overbrace{\begin{pmatrix}1\\1\end{pmatrix} \otimes ... \otimes \begin{pmatrix}1\\1\end{pmatrix}}^{n-1 \text{ terms}} \otimes \begin{pmatrix}0\\1\end{pmatrix} \otimes \overbrace{\begin{pmatrix}1\\1\end{pmatrix} \otimes ... \otimes \begin{pmatrix}1\\1\end{pmatrix}}^{N-n \text{ terms}}. \tag{10}$$

In the TT format, $\mathcal{W}$'s rank is just 2. This is due to $\mathcal{W}_{\boldsymbol{\alpha}} = \alpha_1 + \cdots + \alpha_N$ being a purely additive function. Such functions benefit from the following identity [34]:

$$\sum_{n=1}^{N} \alpha_n = \begin{pmatrix}\alpha_1 & 1\end{pmatrix} \cdot \begin{pmatrix}1 & 0\\ \alpha_2 & 1\end{pmatrix} \cdots \begin{pmatrix}1 & 0\\ \alpha_{N-1} & 1\end{pmatrix} \cdot \begin{pmatrix}1\\ \alpha_N\end{pmatrix}. \tag{11}$$

Each n-th core consists of two slices: the first one is the n-th matrix in Eq. 11, setting $\alpha_n = 0$; the second one is analogous but with $\alpha_n = 1$. See Fig. 4 for an example illustration of this construction for a 3-variable model.

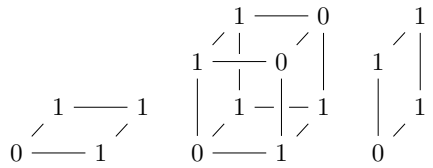

Fig. 4: The Hamming weight TT $\mathcal{W}$ for $N = 3$. At each position $\boldsymbol{\alpha} \in \{0,1\}^N$ it records the bit sum $\mathcal{W}_{\boldsymbol{\alpha}} = |\boldsymbol{\alpha}|$. It is compressed with N cores (TT rank 2) using $8N - 8$ elements in total.

4.2. Hamming Mask Tensor. The N-dimensional *Hamming mask tensor* of type $\leq n$ contains a 1 at each entry $\boldsymbol{\alpha}$ if and only if $|\boldsymbol{\alpha}| \leq n$, and a 0 otherwise. We denote it as $\mathcal{M}^{\leq n}$. It is equivalent to a deterministic finite automaton (DFA) that reads exactly N symbols from the input alphabet {'0', '1'} and has $n + 2$ possible states $\{s_0, s_1, \ldots, s_n, R\}$. There is an accepting state $s_0, \ldots, s_n$ per each value of $|\boldsymbol{\alpha}|$ for which $|\boldsymbol{\alpha}| \leq n$ and one extra rejecting state, R, for any other value $|\boldsymbol{\alpha}| > n$.

To construct a tensor equivalent to this automaton we need to mark one of the $n + 2$ possible states as the active one at each processing step. We represent the active state with a vector of $n+1$ binary elements by using *dummy encoding*, with the rejecting state R encoded as the all-zeros vector. Each core updates the current state by multiplying the state vector generated in the previous step with the slice indexed by the corresponding input symbol. Thus, each slice encodes the state transition matrix for one of the input symbols ('0' or '1'). Using these conventions we manage to translate in a comprehensible manner our *Hamming mask* automaton into a *Hamming mask tensor* (see also Fig. 5):

- The state of the automaton after processing one input symbol can only be s_0 if the processed symbol was '0' (at $[0, 0, 0]$), or s_1 if it was '1' (at $[0, 1, 1]$). Therefore, the first core is all zeros in each slice except at those state positions.
- The cores $2, \ldots, N - 1$ have as first slice (input '0') an identity matrix so as to leave the state unchanged, while the second slice (input '1') is the identity shifted to the right since it is encoding the transition from s_j to s_{j+1} by

shifting any '1' in the input state vector (bit at position j) one position to the right (to $j+1$). An intuitive explanation for these transition matrices is shown in the DFA state diagram of Fig. 5, where transitions labeled with the '0' input symbol always point to the current state and the ones labeled '1' point to the next state.

- The last core collapses the state vector so that the result is '1' if and only if the final state is one of the accepting states $\{s_0, s_1, ..., s_n\}$ after processing the last input symbol, that is, if n or fewer '1' bits were read. That means that for the last symbol being a '0' any '1' in the state vector indicates an acceptable state $\{s_0, ..., s_n\}$, but for the last symbol being a '1' the state vector cannot have a '1' in the s_n-th position. Hence the first slice is all ones for checking the current state being within $\{s_0, ..., s_n\}$, and the second slice is all ones but at the last position to indicate the transition from s_n to R.

This pattern for building a *Hamming mask tensor* can be generalized to any length N and any value n by changing the number of cores to N and their rows and columns to $n+1$ accordingly, while always keeping 2 slices.

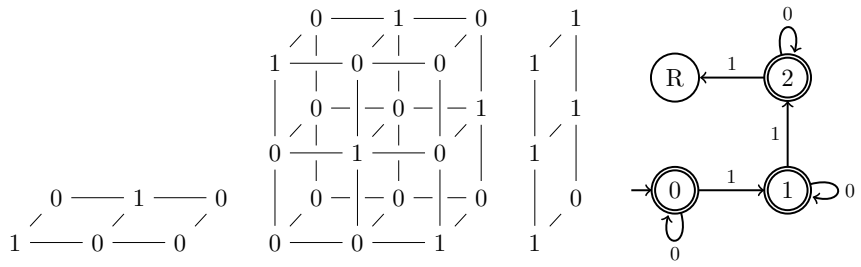

Fig. 5: The Hamming mask TT $\mathcal{M}^{\leq n}$ for $N=3$ and $n=2$ (left) and an equivalent DFA that reads 3 symbols (right, 'R' stands for the rejecting state). It contains a 1 at every position $\boldsymbol{\alpha} \in \{0,1\}^N$ such that $|\boldsymbol{\alpha}| = n$, and a 0 everywhere else. It has TT rank $n+1$ and uses $2(n+1)^2(N-2) + 4(n+1)$ elements.

4.3. Hamming State Tensor. The last core of the mask tensor just described collapses the state vector into one scalar (namely, 0 or 1) and depends on n. However, it is also useful to output the Hamming weight using an explicit one-hot encoded result vector. We define the *Hamming state tensor* $\mathcal{M}^S$, which maps every tuple $\boldsymbol{\alpha}$ to a vector of $N+1$ elements $\boldsymbol{u}$ as:

$$u_i = \begin{cases} 1 & \text{if } i = |\boldsymbol{\alpha}| \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

To build it we proceed similarly to $\mathcal{M}^{\leq N}$, and change the last core for a 3D one, namely the same as the central ones. So $\mathcal{M}^S$ has TT ranks $1, N+1, \ldots, N+1, N+1$, i.e. it is not a standard TT in the sense that its last rank is not 1. The Hamming state tensor needs $2(N+1)^2(N-1) + 2(N+1)$ elements to be stored.

4.4. Length Tensors. The effective dimension in the successive sense motivates us to build tensors that are sensitive to tuple length $|\boldsymbol{\alpha}|$, i.e. the distance between the first and the last variables present in the tuple. We define first the *length mask*

tensor $\mathcal{L}^{\leq n}$, which filters tuples of variables whose length exceeds a threshold (i.e are too spread out). Element-wise,

$$\mathcal{L}^{\leq n}_{\boldsymbol{\alpha}} := \begin{cases} 1 & \text{if } \mathrm{len}(\boldsymbol{\alpha}) \leq n \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

We need again a state vector with $n+1$ elements to encode a state set with $n+2$ states $\{s_0, s_1, \ldots, s_n, R\}$ using dummy encoding. As before, the rejecting state R is represented by an all zeros state vector. Any state transition will lead to R if a tuple length longer than n is detected. An example of this tensor's encoded automaton ($n = 2, N = 3$) is shown in Fig. 6:

- The first core initializes the state vector in the same way as above in Sec. 4.2.
- The cores $2, \ldots, N-1$ follow a regular pattern: the second slice is the identity shifted to the right and therefore encodes the s_j to s_{j+1} transition for any input state. The first slice represents the transitions for the input symbol '0', which in this case are more complex:
  1. When the current state is s_0 (a '1' in the first position of the state vector), the output state should remain the same and thus a '1' is placed at $[0, 0, 0]$ in the first row. This means that no input symbol '1' has been yet found and thus the length counter remains at 0.
  2. When the current state is $s_1, \ldots, s_{n-1}$ (a '1' in the second position of the state vector in the example), the output state should be shifted to the next one, exactly in the same way as for an input '1', thus same row as for the second slice.
  3. When the current state is s_n (a '1' in the third position of the state vector in the example), the output state also remains s_n. This encodes the case in which the distance to the first input symbol '1' is already larger than n and therefore the condition will be violated as soon as an input symbol '1' appears. However, if the remaining input symbols are all '0' it would mean that the actual distance to the last input symbol '1' was shorter or equal to n and thus the final output would be '1'.
- The last core collapses the state vector to '1' if the final state belongs to the set of accepted states in the same way as above in Sec. 4.2.

Based on this tensor and analogously to the Hamming state tensor (Sec. 4.3), we define now the *length state tensor* $\mathcal{L}^S$ which maps every $\boldsymbol{\alpha}$ to a vector of $N+1$ elements that stores the value $\mathrm{len}(\boldsymbol{\alpha})$ using one-hot encoding. It has the same storage complexity as $\mathcal{M}^S$.

Tab. 1 shows a few examples of the values taken by the main tensors we have introduced in this section.

Table 1: Several tensors for the case $N = 5$ evaluated at three example tuples.

Tuple $\boldsymbol{\alpha}$	**Binary form**	$\mathcal{W}_{\boldsymbol{\alpha}}$	$\mathcal{M}^{\leq 3}_{\boldsymbol{\alpha}}$	$\mathcal{M}^{S}_{\boldsymbol{\alpha}}$	$\mathcal{L}^{\leq 3}_{\boldsymbol{\alpha}}$	$\mathcal{L}^{S}_{\boldsymbol{\alpha}}$
$\{5\}$	$[0,0,0,0,1]$	1	1	$[1,0,0,0,0]$	1	$[1,0,0,0,0]$
$\{1,5\}$	$[1,0,0,0,1]$	2	1	$[0,1,0,0,0]$	0	$[0,0,0,0,1]$
$-\{3\} = \{1,2,4,5\}$	$[1,1,0,1,1]$	4	0	$[0,0,0,1,0]$	0	$[0,0,0,0,1]$

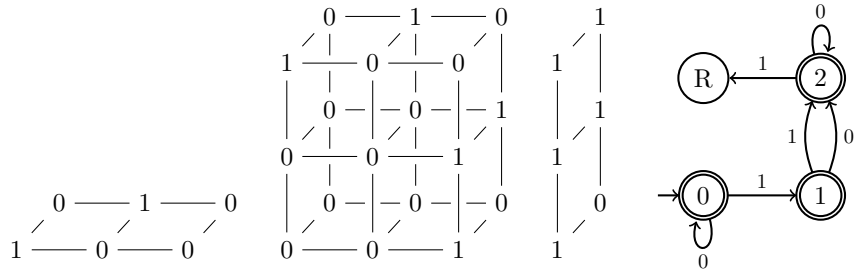

Fig. 6: The length mask tensor $\mathcal{L}^{\leq n}$ for $N = 3$ and $n = 2$ (left) and an equivalent DFA that reads 3 symbols (right). It contains a 1 at every position $\boldsymbol{\alpha}$ where $\text{len}(\boldsymbol{\alpha}) \leq n$, and 0 otherwise. It uses the same ranks and number of elements as $\mathcal{M}^{\leq n}$.

5. Computing Sensitivity Metrics. In this section we show how the proposed tensors can be used to extract various sensitivity analysis metrics from any Sobol TT via efficient operations in the TT format (most importantly, tensor dot products and global optimization).

5.1. Mean Dimension. Let us consider the formula for the mean dimension $D_S := \sum_{\boldsymbol{\alpha}} S_{\boldsymbol{\alpha}} \cdot |\boldsymbol{\alpha}|$, and let our Sobol TT $\mathcal{S}$ contain all variance components S. We observe that, being a weighted summation over all $\boldsymbol{\alpha}$, the formula for D_S equals the tensor dot product $\langle \mathcal{S}, \mathcal{W} \rangle$. We compute it by successively contracting the TT cores of $\mathcal{S}$ and $\mathcal{W}$ together (Alg. 1, see also [33] and Alg. 3 from [13]). Supposing that $\mathcal{S}$ has a constant TT rank $R = R_1 = \cdots = R_{N-1}$, this has a total asymptotic cost of $O(NR^2)$ operations.

Algorithm 1 Given a Sobol TT $\mathcal{S}$, compute the mean dimension D_S using fast TT dot product via tensor contractions [13]. We use Einstein notation, i.e. tensors are contracted along the indices that appear both as subscripts and superscripts.

1: Assemble $\mathcal{W}$ with ranks $Q_0, Q_1, \ldots, Q_{N-1}, Q_N = 1, 2, \ldots, 2, 1$ (Sec. 4.1)
2: $\mathcal{C} := \text{ones}(1 \times 1)$ ▷ $\mathcal{C}$ has size $1 \times 1 = R_0 \times Q_0$
3: **for** $n = 1, \ldots, N$ **do**
4: $\quad \mathcal{C}_{ijk} := \mathcal{C}_{lk} \cdot \mathcal{S}^{l\ (n)}_{ji}$ ▷ $\mathcal{C}$ has now size $R_n \times 2 \times Q_{n-1}$
5: $\quad \mathcal{C}_{ij} := \mathcal{C}_{imk} \cdot \mathcal{W}^{m\ (n)}_{k\ \ j}$ ▷ $\mathcal{C}$ has now size $R_n \times Q_n$
6: **end for** ▷ $\mathcal{C}$ has now size $R_N \times Q_N = 1 \times 1$
7: **return** $D_S = \mathcal{C}[0, 0]$

5.2. Dimension Distribution. We can extract the complete dimension distribution histogram in one go via the Hamming state tensor, namely by contracting $\mathcal{M}^S$ with $\mathcal{S}$ along all physical dimensions (Fig. 7). The array we seek results from the remaining free edge; see also Alg. 2.

Algorithm 2 Given a Sobol TT $\mathcal{S}$, compute the dimension distribution ν.

1: Assemble $\mathcal{M}^S$ with ranks $Q_0, Q_1, \ldots, Q_{N-1}, Q_N = 1, N+1, \ldots, N+1$ (Sec. 5.1)
2: Contract $\mathcal{S}$ with $\mathcal{M}^S$ as in Alg. 1
3: **return** $\nu :=$ resulting vector of N elements

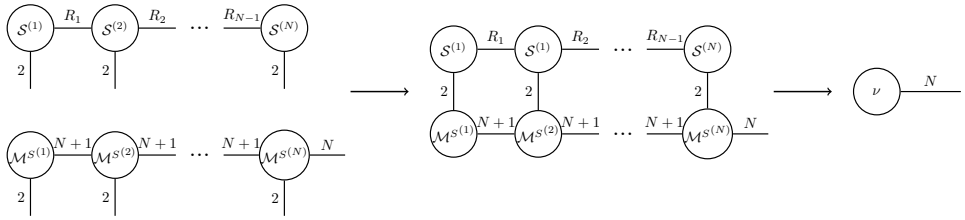

Fig. 7: We compute the dimension distribution vector ν, which has N elements, by contracting together the Sobol TT $\mathcal{S}$ (ranks $1, R_1, \ldots, R_{N-1}, 1$) with our proposed mask state tensor $\mathcal{M}^S = [[\mathcal{M}^{S(1)}, \ldots, \mathcal{M}^{S(N)}]]$ (ranks $1, N+1, \ldots, N+1, 1$) along their spatial and rank dimensions. All TT cores are collapsed together, along Alg. 1; this can be accomplished in $O(N^2R^2) + O(N^3R)$ operations. Only the N-sized edge from $\mathcal{M}^S$ remains, and after the contraction it gathers all variance contributions according to their order as expected.

5.3. Effective Dimension (Superposition Sense). The variance due to order n and below is

$$\sum_{\boldsymbol{\alpha} \,\middle|\, |\boldsymbol{\alpha}| \leq n} \mathcal{S}_{\boldsymbol{\alpha}} = \sum_{\boldsymbol{\alpha}} (\mathcal{S} \circ \mathcal{M}^{\leq n})_{\boldsymbol{\alpha}} = \langle \mathcal{S}, \mathcal{M}^{\leq n} \rangle. \tag{14}$$

With Eq. 14 we can easily extract the superposed effective dimension: it is sufficient to iteratively find the smallest k that yields a relative variance above the given threshold $1 - \epsilon$; see also Alg. 3.

Algorithm 3 Given a Sobol TT $\mathcal{S}$, compute its effective dimension (in the superposition sense) d_S w.r.t. threshold ϵ.

```
Compute dimension distribution ν as in Alg. 2
ν̄ := cumulativeSum(ν)
for n = 1, ..., N do
    if ν̄[n] ≥ 1 − ε then
        return d_S := n
    end if
end for
```

5.4. Effective Dimension (Truncation Sense). The truncated effective dimension d_T depends on the ordering of variables [16]. However, the formula

$$\arg\min_{n} \left\{ n \;\middle|\; \max_{\boldsymbol{\alpha}} \left\{ (\mathcal{S} \circ \mathcal{M}^{\leq n})_{\boldsymbol{\alpha}} \right\} \geq 1 - \epsilon \right\} \tag{15}$$

gives us the truncated effective dimension w.r.t. the best possible ordering $\boldsymbol{\alpha}$ of variables. The n can be found iteratively (Alg. 4). Note that the best (largest variance) order-n tuple is given by finding the maximum of a 2^N tensor. Although this is an

NP-hard problem, we exploit an existing global optimization heuristic based on cross-approximation [35, 13] that only evaluates a subset of "promising" tensor entries. The method works well in practice and is fast for tensors of moderate rank; see also [32].

Algorithm 4 Given a Sobol TT $\mathcal{S}$, compute its effective dimension (in the truncation sense) d_T w.r.t. threshold ϵ and best possible ordering of variables.

```
Compute dimension distribution ν as in Alg. 2
for n = 1, ..., N do
    Assemble M^{≤n} (Sec. 4.2)
    v := max_α{S ∘ M^{≤n}}          ▷ Element-wise product using cross-approximation [35], tensor maximum found as in [13, 32]
    if v ≥ 1 − ε then
        return d_s := n
    end if
end for
```

5.5. Effective Dimension (Successive Sense). The following dot product is the total variance contributed by all tuples whose length is n or less:

$$0 \leq \langle \mathcal{S}, \mathcal{L}^{\leq n} \rangle \leq 1. \tag{16}$$

The effective dimension in the successive sense is therefore the smallest n (Alg. 5):

$$d_s = \arg\min_n \left\{ n \;\middle|\; \langle \mathcal{S}, \mathcal{L}^{\leq n} \rangle \geq 1 - \epsilon \right\}. \tag{17}$$

Algorithm 5 Given a Sobol TT $\mathcal{S}$, compute its effective dimension (in the successive sense) d_s w.r.t. threshold ϵ.

```
l := the contraction between S and L^S as in Alg. 2
d_s := proceed as in Alg. 3 but using l instead of ν
return d_s
```

5.6. Shapley Values. The Shapley values can be interpreted as variance contributions in a SA context and computed in terms of the SI [39]. They are considered a challenging QoI from a computational point of view [39, 40]: the n-th value relies on 2^{N-1} terms, each weighted by a combinatorial number. Let the superscript $\mathcal{T}^C$ denote the closed version of a tensor $\mathcal{T}$, i.e. for each $\boldsymbol{\alpha}$, $\mathcal{T}^C_{\boldsymbol{\alpha}}$ is the sum of all $\mathcal{T}_{\boldsymbol{\beta}}$ where $\boldsymbol{\beta} \subseteq \boldsymbol{\alpha}$ (recall Eq. 3). We observe that, in tensor form, the Shapley formula is equivalent to:

$$\phi_n = \sum_{\boldsymbol{\alpha} | n \in \boldsymbol{\alpha}} \frac{S_{\boldsymbol{\alpha}}}{|\boldsymbol{\alpha}|} = (\mathcal{S} \circ (1/\mathcal{W}))^C_{\{n\}}. \tag{18}$$

This represents the closed version of the weighted variance components, evaluated at tuples $\{1\}, ..., \{N\}$. Note that $1/\mathcal{W}$ is a so-called *Hilbert tensor*; such tensors are

known to have high-rank but are extremely well compressible via low-rank expansions [35]. We have confirmed this experimentally for this particular case in which the tensor has size 2^N. For $N = 20$, for example, compressing $1/\mathcal{W}$ using only $R_1 = \cdots = R_{19} = 7$ ranks is enough to achieve a relative error of 2.16×10^{-8} over the full set of 2^{20} groundtruth values (we set $\mathcal{W}_\emptyset := \infty$ formally, i.e. $1/\mathcal{W}_\emptyset = 0$, to prevent division by 0). In Alg. 6 we show how to compute all Shapley values $\phi_1, \ldots, \phi_N$ using Eq. 18.

Algorithm 6 Given a Sobol TT $\mathcal{S}$, compute all N Shapley values.

Compute $1/\mathcal{W}$ ▷ Using cross-approximation. Can be precomputed
$\widehat{\mathcal{S}} = [[\widehat{S}^{(1)}, \ldots, \widehat{S}^{(N)}]] := \mathcal{S} \circ (1/\mathcal{W})$ ▷ Denote its ranks as $1, Q_1, \ldots, Q_{N-1}, 1$
for $n = 1, \ldots, N$ **do** ▷ Compute the closed tensor of a tensor [3]
    $\widehat{\mathcal{S}}^{(n)^C} := \text{zeros}(Q_{n-1}, 2, Q_n)$
    $\widehat{\mathcal{S}}^{(n)^C}[:, 0, :] := \widehat{\mathcal{S}}^{(n)}[:, 0, :]$ ▷ The first slice stays the same
    $\widehat{\mathcal{S}}^{(n)^C}[:, 1, :] := \widehat{\mathcal{S}}^{(n)}[:, 0, :] + \widehat{\mathcal{S}}^{(n)}[:, 1, :]$ ▷ The second slice becomes the sum of both slices
end for
$\widehat{\mathcal{S}}^C := [[\widehat{\mathcal{S}}^{(1)^C}, \ldots, \widehat{\mathcal{S}}^{(N)^C}]]$
for $n = 1, \ldots, N$ **do**
    $\phi_n := \widehat{\mathcal{S}}^C_{\{n\}} = \widehat{\mathcal{S}}^C[\overbrace{0, \ldots, 0}^{n-1}, 1, \overbrace{0, \ldots, 0}^{N-n}]$
end for
return $(\phi_1, \ldots, \phi_N)$

6. Experimental Results. We have tested our metric computation algorithms[1] with three models of dimensionalities 10 and 20 on a desktop workstation (3.2GHz Intel i5). For each model, a TT surrogate with 100 points per axis is built using adaptive cross-approximation [35, 50] as released in the Python library *ttpy* [15], from which we extract its Sobol TT using the method from [3]. In all cases we compare all our resulting single-variable SI with the quasi-MC algorithm by Saltelli et al. [46] implemented in the Sensitivity Analysis Library for Python (*SALib*) [23], and our Shapley values with the state-of-the-art MC method by Song et al. [53]. We also assess the accuracy of the proposed algorithm for the mean dimension D_S by using the identity $D_S = \sum_n S^T_n$ due to Liu and Owen [31]. We report the number of function evaluations for each method.

6.1. Sobol "G" function. Our first model is synthetic and widely used for benchmarking purposes in the SA literature:

$$f(\boldsymbol{x}) := \prod_{n=1}^{N} \frac{|4x_n - 2| + a_n}{1 + a_n} \tag{19}$$

with $x_n \sim \mathcal{U}(0, 1)$ for all n. Coefficients a_n are customizable; we have chosen $a_n := (n - 2)/2$ for $n = 1, \ldots, N$ as in [11]. This model is partly provided as a sanity check: since f is separable (i.e. has TT-rank 1), we can expect to obtain an exact TT

[1] Our algorithms are provided within the Python package https://github.com/rballester/ttrecipes (the code for all models is available in the folder `examples/sensitivity_analysis/`).

interpolator over the discretized grid. We took $N = 40$ dimensions, i.e. the tensor grid has 100^{40} elements and the resulting Sobol TT encodes $2^{40} \approx 1.1 \times 10^{12}$ elements. We furthermore know the analytical values for the SI from [28]: $S_n^C = D_n/D, S_n^T = D_n^T/D$ for all n, where

$$D_n = \frac{1}{3(1+a_n)^2},\ D_n^T = D_n \cdot \prod_{i \neq n}(1+D_i),\ D = -1 + \prod_n (1+D_n). \tag{20}$$

Albeit separable, this model is very high-dimensional and therefore challenging for MC- and QMC-based approaches. Adaptive cross-approximation here produced $R_1 = \cdots = R_{39} = 1$ as TT-ranks, as is expected for a separable function. It took $123,400$ samples in 2.18s and achieved a relative error under 10^{-15}. Since its Sobol TT cannot have a rank larger than $R^2 = 1$ [3] it is free of compression error (this applies to all product functions). Computing the Sobol TT took 0.16s, whereas extracting all metrics took 9.69s. The most expensive metric is the effective dimension in the truncation sense d_T, since it requires several runs of a global optimization algorithm (Alg. 4). Note also that since the ranks are all equal, all slices along dimensions $1, \ldots, N$ are visited the same number of times during sampling. For this separable function, a minimum of $N \cdot I = 4,000$ samples would be needed. However, the practical algorithm does not know R a priori, so it does take extra samples and stops after realizing that the error does not improve for $R > 1$.

For comparison we run methods [46] (QMC for SI) and [53] (MC for Shapley values) with 5×10^6 and 40×10^6 samples, respectively. Note that the QMC method uses an extension of the Sobol sequence that does not require a power of two as the number of samples. Numerical results are reported in Tab. 2 (effective and mean dimensions), Tab. 3 (Sobol/Shapley indices), and Fig. 8 (dimension distribution). We always use the threshold $\epsilon = 0.05$ for the effective dimensions, i.e. the proportion of variance preserved must be at least 95% (in Tab. 2 we list the percentage achieved in each case). Note that the SI coincide with the groundtruth by more than three decimal digits; specifically, the mean absolute error was 3.2×10^{-6} for the S_n and 5.7×10^{-6} for the S_n^T. Note also that the QMC [46] only provides single-variable SI, whereas the Sobol TT gives indices for interactions of all possible order (only the first order is shown).

Table 2: Sobol "G" function: effective and mean dimensions using our method. The mean dimension is compared with the analytical value, which is available for this model, and an estimation using the sum of total effects [31] computed from QMC.

Dimension metric	Value
Effective dimension ($\epsilon = 0.05$)	
Superposition sense (d_S)	3 (98.1% variance)
Truncation sense (d_T) $[\ X_1\ \cdots\ X_{23}\ ]$	23 (97.1% variance)
Successive sense (d_s)	15 (95.2% variance)
Mean dimension (D_s)	
Exact	1.603
Ours	1.603
$\sum_n S_n^T$ (QMC [46])	1.596

Table 3: Sobol "G" function: Shapley values and Sobol indices for the first and last 5 variables.

	Shapley value		Closed Sobol index			Total Sobol index		
	Ours	MC [53]	Ours	Exact	QMC [46]	Ours	Exact	QMC [46]
X_1	0.5036	0.5042	0.3322	0.3322	0.3324	0.7138	0.7138	0.7076
X_2	0.1835	0.1750	0.0831	0.0830	0.0826	0.3122	0.3123	0.3144
X_3	0.0898	0.0866	0.0369	0.0369	0.0373	0.1611	0.1612	0.1607
X_4	0.0524	0.0499	0.0208	0.0208	0.0218	0.0961	0.0961	0.0963
X_5	0.0341	0.0375	0.0133	0.0133	0.0131	0.0632	0.0632	0.0625
...	...			...			...	
X_{36}	0.0007	0.0024	0.0003	0.0003	0.0003	0.0013	0.0013	0.0013
X_{37}	0.0006	0.0051	0.0002	0.0002	0.0002	0.0012	0.0012	0.0012
X_{38}	0.0006	-0.0070	0.0002	0.0002	0.0002	0.0012	0.0012	0.0011
X_{39}	0.0006	-0.0017	0.0002	0.0002	0.0002	0.0011	0.0011	0.0011
X_{40}	0.0006	-0.0017	0.0002	0.0002	0.0003	0.0010	0.0010	0.0010

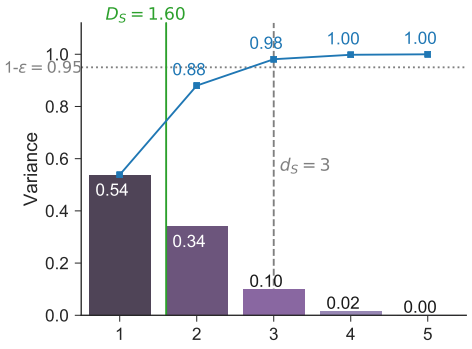

Fig. 8: Sobol "G" function: dimension distribution (truncated after order 5), mean dimension D_S, and effective dimension in the superposition sense d_S for $\epsilon = 0.05$.

6.2. Simulated decay chain. The second example simulates a radioactive decay chain that concatenates Poisson processes for 11 chemical species. It is a linear *Jackson network*, i.e. each species (except the last one) can decay into the next species in the chain. In the beginning the material belongs entirely to the first species. We model 10 parameters, namely the decay rates λ_n of the 10 first species. The simulation result $f_T(\lambda_1, \ldots, \lambda_{10})$ is the amount of stable material (last node in the chain) measured after a certain time span T. To evaluate each sample of f_T we simulate the chain by discretizing the span T into timesteps of one day. Each λ_n is constant throughout all time steps and represents the fraction of each material that decays every day. They are independent and uniformly distributed in the interval $[0.00063281, 0.00756736]$, which corresponds to half-lives from 3 years down to 3 months.

The resulting effective and mean dimension metrics for a time span of $T = 2$ years are reported in Tab. 4, and the Sobol and Shapley sensitivity indices in Tab. 5. The cross-approximation sampling stage used 134,400 function evaluations (8.69s)

and stopped after achieving a relative error of 1.5×10^{-7}. All TT-ranks equal 6, so again the sampling plan is evenly distributed among all slices. Building the Sobol TT took 0.06s and extracting all metrics, 0.76s. Methods [53] and [46] were run with 1.4×10^7 and 600,000 samples respectively. These were the minimal numbers such that their suggested confidence intervals (one standard error for [46], two for [53]) were on average the 10% of their estimated indices. Note that our results converge unanimously to $S_1 = \cdots = S_{10}$, $S_1^T = \cdots = S_{10}^T$, and $\phi_1 = \cdots = \phi_{10} = 1/10$. This follows from the fact that the model is symmetric. We verified experimentally that this equality still holds when we decrease I or R, although results start to deviate for small enough values of these hyperparameters. From this we learn that although restricting the sampling budget affects the final metric accuracy, cross-approximation stays robust at reflecting the symmetry of the model sampled.

Table 4: Decay chain: effective and mean dimensions (time span = 2 years)

Dimension metric	Value
Effective dimension ($\epsilon = 0.05$)	
Superposition sense (d_S)	3 (95.9% variance)
Truncation sense (d_T) [$\lambda_1 \ \cdots \ \lambda_{10}$]	10 (100.0% variance)
Successive sense (d_s)	9 (97.8% variance)
Mean dimension (D_s)	
Ours	1.728
$\sum_n S_n^T$ (QMC [46])	1.722

Table 5: Decay chain: Shapley values and SI (time span = 2 years)

Variable	Shapley value		Sobol index		Total Sobol index	
	Ours	MC [53]	Ours	QMC [46]	Ours	QMC [46]
λ_1	0.100	0.096	0.049	0.049	0.173	0.171
λ_2	0.100	0.099	0.049	0.052	0.173	0.171
λ_3	0.100	0.093	0.049	0.050	0.173	0.170
λ_4	0.100	0.102	0.049	0.049	0.173	0.171
λ_5	0.100	0.100	0.049	0.048	0.173	0.167
λ_6	0.100	0.099	0.049	0.050	0.173	0.184
λ_7	0.100	0.111	0.049	0.051	0.173	0.171
λ_8	0.100	0.109	0.049	0.051	0.173	0.170
λ_9	0.100	0.096	0.049	0.048	0.173	0.172
λ_{10}	0.100	0.097	0.049	0.050	0.173	0.176

We have also studied the sensitivity behavior when varying the simulated time span: $0.5 \leq T \leq 30.5$ years (see dimension distributions in Fig. 9). Fig. 10 depicts the evolution of the Shapley values, the SI and the dimension metrics within the same range. Note how the average interaction order is higher for extreme values of T and lower in the center (≈ 1 at around 12 years). This behavior hints that, for very short or long time spans, several decay rates need to have specific values to affect

the output of the function. In the former case, for example, no amount of the final species is detected unless many decay rates are fast enough. On the other hand, after a very long time span T, all materials have decayed completely unless several rates are slow enough. The successive dimension d_s (right chart in Fig. 10) strongly reflects this behavior as well: the metric is highly sensitive to such high-order consecutive interactions, and is therefore useful for this kind of time-series models.

Finally, this example illustrates how the Shapley values ϕ_n are able to recognize the equal importance of all variables and act as the best overall summary of their influence. This is consistent with Owen [39], their original proponent in the context of sensitivity analysis.

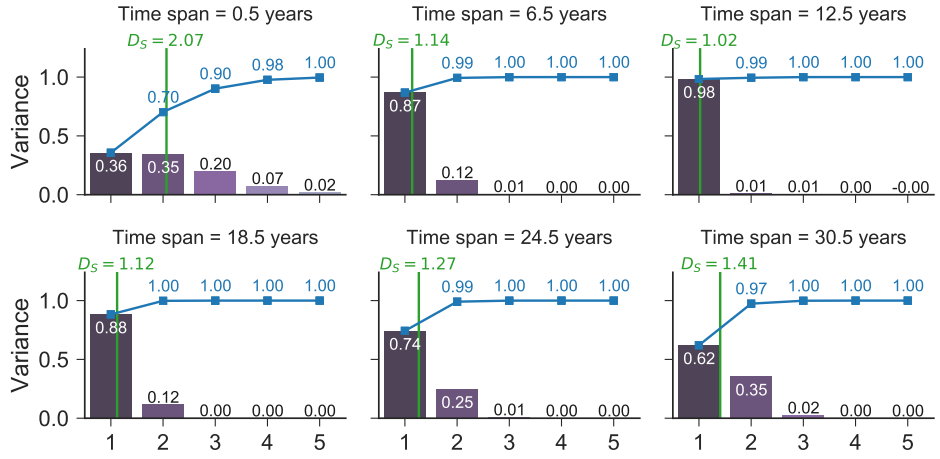

Fig. 9: Decay chain: mean dimension and dimension distribution (truncated after order 5) for 6 different time spans $0.5 \leq T \leq 30$. Very short and very long time spans result in higher-order interactions.

6.3. Fire-spread Model. Last we model the rate of fire spread in the Mediterranean shrublands according to 10 variables that are fed into Rothermel's equations [45]. Both [48] and [53] have analyzed this model from the sensitivity analysis perspective. We compare our results with the latter work for the case of independently (but non-uniformly) distributed parameters. We use the updated equations as in [53], which incorporates the modifications by Albini [1] and Catchpole and Catchpole [9], with the marginal distributions shown in Tab. 6.

Cross-approximation took 4.72×10^6 samples in 8.28s and achieved a relative error of 5.3×10^{-5} with TT-ranks $(11, 17, 17, 7, 43, 16, 16, 11, 6)$. The sixth core has size $43 \times 100 \times 16$ and is the largest we encountered so far; this means that relatively many sample fibers were needed along variable m_d. Building the Sobol TT took 0.28s and extracting all metrics, 0.52s. Method [53] was run with 4.6×10^7 samples (just as in the original paper), while we again run method [46] so as to attain a 10% confidence interval (6.96×10^6 samples). Results are reported in Tab. 7 (effective and mean dimensions) and Tab. 8 (Shapley values and SI) as well as Fig. 11 (dimension distribution). The analytical metrics for this model are unknown, but our results are in all cases within the other two methods' intervals of confidence.

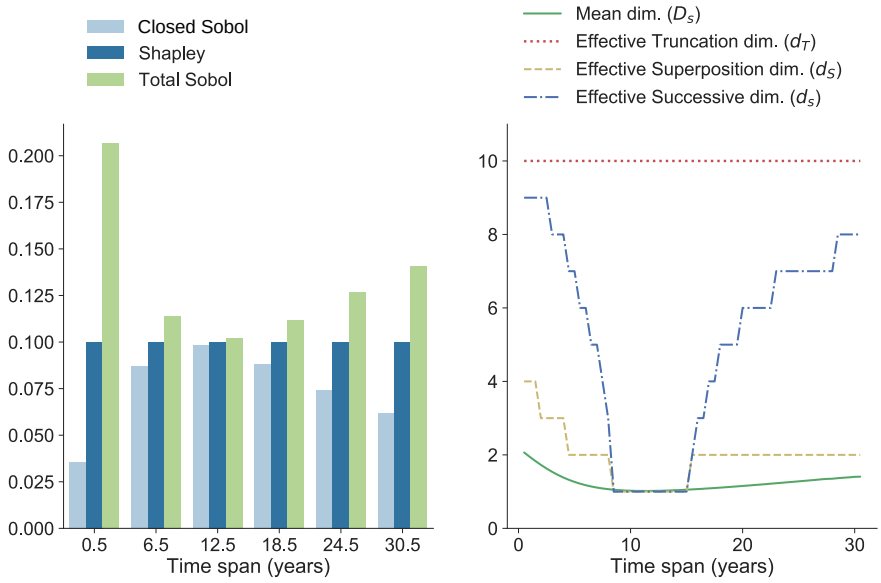

Fig. 10: Decay chain: all metrics as a function of the time span. In the left plot, each index is averaged across all variables. Note how the closed and total SI follow an inverse relationship with one another and meet at about $T = 12$ years, whereas the Shapley values are constant.

Table 6: Fire spread: parameter description and marginal distributions [53].

Variable	Description	Distribution
δ	Fuel depth (cm)	$Log\mathcal{N}(2.19, 0.517) \cap [5, \infty)$
σ	Fuel particle area-to-volume ratio (1/cm)	$Log\mathcal{N}(3.31, 0.294)$
h	Fuel particle low heat content (Kcal/kg)	$Log\mathcal{N}(8.48, 0.063)$
ρ_p	Oven-dry particle density (D.W.g/cm^3)	$Log\mathcal{N}(-0.592, 0.219)$
m_l	Moisture content of the live fuel (H_2O g/D.W.g)	$\mathcal{N}(1.18, 0.377) \cap [0, \infty)$
m_d	Moisture content of the dead fuel (H_2O g/D.W.g)	$\mathcal{N}(0.19, 0.047)$
S_T	Fuel particle total mineral content (MIN.g/D.W.g)	$\mathcal{N}(0.049, 0.011) \cap [0, \infty)$
U	Wind speed at midflame height (km/h)	$6.9 \cdot Log\mathcal{N}(1.0174, 0.5569)$
$\tan\phi$	Slope	$\mathcal{N}(0.38, 0.186) \cap [0, \infty)$
P	Dead to total fuel loading ratio	$Log\mathcal{N}(-2.19, 0.64) \cap (-\infty, 1]$

7. Discussion and Conclusions. The Sobol indices have long been in the spotlight of the variance-based sensitivity analysis and uncertainty quantification communities. In recent years, several other sensitivity metrics are becoming popular that capitalize on these building blocks and are combinatorial in nature. State-of-the-art algorithms and sampling schemes that estimate these advanced metrics often require rather large numbers of samples, i.e. in the order of millions of function evaluations. Therefore, modern methods are growingly dependent on fast analytical or surrogate models in order to keep overall computation times low. We found that, in many cases, similar or lower sampling budgets are actually sufficient to build highly accurate TT surrogates. Once such a tensor representation is available, its advantages become ev-

Table 7: Fire spread: effective and mean dimensions.

Dimension metric	Value
Effective dimension ($\epsilon = 0.05$)	
Superposition sense (d_S)	3 (97.6% variance)
Truncation sense (d_T) [δ, σ, m_l, m_d, U, P]	6 (96.1% variance)
Successive sense (d_s)	8 (96.5% variance)
Mean dimension (D_s)	
Ours	1.650
$\sum_n S_n^T$ (QMC [46])	1.714

Table 8: Fire spread: Shapley values and Sobol indices.

Variable	Shapley value		Closed Sobol index		Total Sobol index	
	Ours	MC [53]	Ours	QMC [46]	Ours	QMC [46]
δ	0.203	0.217	0.106	0.106	0.331	0.348
σ	0.125	0.132	0.048	0.048	0.235	0.228
h	0.003	-0.020	0.001	0.001	0.005	0.006
ρ_p	0.012	0.013	0.004	0.005	0.023	0.023
m_l	0.231	0.231	0.142	0.143	0.346	0.364
m_d	0.165	0.183	0.095	0.097	0.259	0.262
S_T	0.002	-0.006	0.001	0.001	0.005	0.004
U	0.202	0.210	0.090	0.093	0.354	0.387
$\tan\phi$	0.004	-0.014	0.002	0.002	0.006	0.006
P	0.053	0.054	0.029	0.029	0.086	0.086

ident: a wide range of operations can be computed expeditiously in the compressed domain, including differentiation, integration, optimization, element-wise dot products and functions, and in particular fast evaluation of many sensitivity metrics as proposed in this paper.

All proposed TT automata are error-free and highly compact, since they can compress masks of size 2^N using only $O(N^3)$ (and often fewer) elements. Given a high-quality surrogate, they allow us to extract metrics that are close to their analytical values by several decimal digits of precision. Constructing the tensor masks is a fast procedure: one first "compiles" an algorithm (automaton) into a compressed tensor by organizing sparse arrays (TT cores) in a certain configuration that is determined by the automaton's type and dimensionality. The algorithm is then "executed" by computing its tensor dot product with the target Sobol TT. This dot product is an iterative algorithm on its own, namely a sequence of tensor contractions that boil down to simple matrix-matrix products and array reshaping operations.

To summarize, we have contributed algorithms for advanced variance-based sensitivity analysis that leverage tensor decompositions heavily, in particular the TT model. Such decompositions can represent exponential numbers of indices and QoIs in an extremely compact manner. Throughout our paper we exploited three crucial tools. First, the Sobol TT is able to gather all variance components. While most

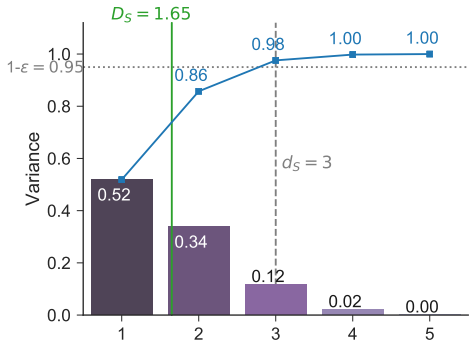

Fig. 11: Fire spread: dimension distribution (truncated after order 5).

surrogate-based SA approaches use a metamodel as a departure point for metric computation, we go one step further and work with a data structure that already contains all elementary indices of interest. Second, the automaton-inspired TT tensors contributed here are in charge of selecting and weighing index tuples as needed, and they all have a moderate rank. Last, tensor-tensor contractions (i.e. dot products) in the TT format have a polynomial cost w.r.t. number of dimensions and, combined with the previous elements, can produce sophisticated QoIs in a matter of few sequential steps. Other auxiliary (but still important) tools include adaptive cross-approximation, which is useful for building TT surrogates and auxiliary tensors, and global optimization in the compressed domain.

Limitations and Future Work. Our main limitation is the critical assumption that the model of interest admits a low- or moderate-rank TT representation. It is known that various classes of tensors may be better compressed via alternative tensor networks, and we are actually confident that the proposed methods and automata are adaptable to other network topologies. Other techniques that remain promising to combine and improve TT compression include dimension reordering [6] and active subspace transformations [4].

On another note, the particular case remains of how to extract reliable sensitivity metrics when limited by modest sampling budgets, e.g. of a few hundreds or thousands of samples. Using an intermediate surrogate is often the method of choice in these cases, also for the other state-of-the-art methods [41]. On the TT side, one may either use cross-approximation on the intermediate model or build directly the tensor from a limited set of samples using low-rank tensor completion techniques; see e.g. the positive recent results by Gorodetsky and Jakeman [18]. Research on TT completion usually strives for a generalization error as low as possible, as estimated for example by cross-validation strategies. Here we deal with derived quantities of interest whose computation does not add additional error, but that will be affected nonetheless by the surrogate's generalization error. Future research efforts will be directed towards better characterizing how the inaccuracy due to a small sampling budget can have an impact on derived sensitivity metrics.

Acknowledgments. This work was partially supported by the University of Zurich's Forschungskredit "Candoc" (grant number FK-16-012). The authors wish to thank Eunhye Song for providing code for the fire-spread model [53].

REFERENCES

[1] F. A. Albini, *Estimating wildfire behavior and effects*, tech. report, U.S. Department of Agriculture, Intermountain Forest and Range Experiment Station, 1976.

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

[3] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola, *Sobol tensor trains for global sensitivity analysis*, ArXiv e-prints, (2017), https://arxiv.org/abs/1712.00233.

[4] V. Baranov and I. V. Oseledets, *Fitting high-dimensional potential energy surface using active subspace and tensor train (as+tt) method*, The Journal of Chemical Physics, 143 (2015), p. 17417.

[5] M. Bianchetti, S. Kucherenko, and S. Scoleri, *Pricing and risk management with high-dimensional quasi Monte Carlo and global sensitivity analysis*, ArXiv e-prints, (2015), https://arxiv.org/abs/1504.02896.

[6] D. Bigoni and A. Engsig-Karup, *Uncertainty Quantification With Applications to Engineering Problems*, PhD thesis, 2015.

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

[8] R. E. Caflisch, W. J. Morokoff, and A. B. Owen, *Valuation of Mortgage Backed Securities Using Brownian Bridges to Reduce Effective Dimension*, CAM report, Department of Mathematics, University of California, Los Angeles, 1997.

[9] E. Catchpole and W. Catchpole, *Modelling moisture damping for fire spread in a mixture of live and dead fuels*, International Journal of Wildland Fire, 1 (1991), pp. 101–106.

[10] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, and D. P. Mandic, *Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions*, Foundations and Trends in Machine Learning, 9 (2016), pp. 249–429.

[11] T. Crestaux, O. P. L. Maître, and J. Martinez, *Polynomial chaos expansion for sensitivity analysis*, Reliability Engineering and System Safety, 94 (2009), pp. 1161–1172.

[12] G. M. Crosswhite and D. Bacon, *Finite automata for caching in matrix product algorithms*, ArXiv e-prints, 78 (2008), https://arxiv.org/abs/0708.1221.

[13] S. V. Dolgov, B. N. Khoromskij, A. Litvinenko, and H. G. Matthies, *Computation of the response surface in the tensor train data format*, CoRR, (2014), http://arxiv.org/abs/1406.2816.

[14] S. V. Dolgov and D. V. Savostyanov, *Alternating minimal energy methods for linear systems in higher dimensions*, SIAM Journal on Scientific Computing, 36 (2014), pp. A2248–A2271.

[15] I. V. O. et al., *ttpy: Python implementation of the TT-toolbox*. http://github.com/oseledets/ttpy.

[16] G. Geraci, P. Congedo, R. Abgrall, and G. Iaccarino, *High-order statistics in global sensitivity analysis: Decomposition and model reduction*, Computer Methods in Applied Mechanics and Engineering, 301 (2016), pp. 80 – 115.

[17] S. A. Goreinov and E. E. Tyrtyshnikov, *The maximal-volume concept in approximation by low-rank matrices*, Contemporary Mathematics, 280 (2001), pp. 47–51.

[18] A. A. Gorodetsky and J. D. Jakeman, *Gradient-based Optimization for Regression in the Functional Tensor-Train Format*, ArXiv e-prints, (2018), https://arxiv.org/abs/1801.00885, https://arxiv.org/abs/1801.00885.

[19] L. Grasedyck, M. Kluge, and S. Krämer, *Variants of alternating least squares tensor completion in the tensor train format*, SIAM Journal on Scientific Computing, 37 (2015), pp. A2424–A2450.

[20] L. Grasedyck and S. Krämer, *Stable ALS approximation in the TT-format for rank-adaptive tensor completion*, ArXiv e-prints, (2017), https://arxiv.org/abs/1701.08045.

[21] L. Grasedyck, D. Kressner, and C. Tobler, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78.

[22] S. Handschuh, *Numerical Methods in Tensor Networks*, Doctoral dissertation, Universität Leipzig, Leipzig, 2015, http://nbn-resolving.de/urn:nbn:de:bsz:15-qucosa-159672.

[23] J. Herman and W. Usher, *SALib: An open-source python library for sensitivity analysis*, The Journal of Open Source Software, 2 (2017), https://doi.org/10.21105/joss.00097.

[24] W. Hoeffding, *A class of statistics with asymptotically normal distribution*, The Annals of Mathematical Statistics, 19 (1948), pp. 293–325.

[25] T. Homma and A. Saltelli, *Importance measures in global sensitivity analysis of nonlinear models*, Reliability Engineering & System Safety, 52 (1996), pp. 1–17.

[26] B. Iooss and C. Prieur, *Shapley effects for sensitivity analysis with dependent inputs: Com-*

parisons with sobol' indices, numerical estimation and applications. working paper or preprint, July 2017, https://hal.inria.fr/hal-01556303.

[27] K. Konakli and B. Sudret, *Low-rank tensor approximations for reliability analysis*, in Proceedings International Conference on Applications of Statistics and Probability in Civil Engineering, July 2015.

[28] S. Kucherenko, B. Feil, N. Shah, and W. Mauntz, *The identification of model effective dimensions using global sensitivity analysis*, Reliability Engineering and System Safety, 96 (2011), pp. 440–449.

[29] L. Le Gratiet, S. Marelli, and B. Sudret, *Metamodel-Based Sensitivity Analysis: Polynomial Chaos Expansions and Gaussian Processes*, Springer International Publishing, Cham, 2017, pp. 1289–1325, https://doi.org/10.1007/978-3-319-12385-1_38.

[30] P. L'Ecuyer and C. Lemieux, *Variance reduction via lattice rules*, Management Science, 46 (2000), pp. 1214–1235.

[31] R. Liu and A. B. Owen, *Estimating mean dimensionality of analysis of variance decompositions*, Journal of the American Statistical Association, 101 (2006), pp. 712–721.

[32] A. Mikhalev and I. V. Oseledets, *Rectangular maximum-volume submatrices and their applications*, Linear Algebra and its Applications, 538 (2018), pp. 187–211, https://doi.org/10.1016/j.laa.2017.10.014.

[33] I. V. Oseledets, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317.

[34] I. V. Oseledets, *Constructive representation of functions in low-rank tensor formats*, Constructive Approximation, 37 (2013), pp. 1–18.

[35] I. V. Oseledets and E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra and its Applications, 432 (2010), pp. 70–88.

[36] I. V. Oseledets and E. Tyrtyshnikov, *Algebraic wavelet transform via quantics tensor train decomposition.*, SIAM Journal on Scientific Computing, 33 (2011), pp. 1315–1328.

[37] A. B. Owen, *The dimension distribution and quadrature test functions*, Statistica Sinica, 13 (2003), pp. 1–17.

[38] A. B. Owen, *Variance components and generalized Sobol' indices*, SIAM Journal on Uncertainty Quantification, 1 (2013), pp. 19–41.

[39] A. B. Owen, *Sobol' indices and Shapley value*, SIAM/ASA Journal on Uncertainty Quantification, 2 (2014), pp. 245–251.

[40] A. B. Owen and C. Prieur, *On Shapley value for measuring importance of dependent inputs*, SIAM/ASA Journal on Uncertainty Quantification (to appear), (2017), https://arxiv.org/abs/1610.02080.

[41] N. E. Owen, P. Challenor, P. P. Menon, and S. Bennani, *Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators*, ArXiv e-prints, (2015), https://arxiv.org/abs/1511.00926.

[42] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. K. Tucher, *Surrogate-based analysis and optimization*, Progress in Aerospace Sciences, 41 (2005), pp. 1–28.

[43] G. Rabusseau, *A Tensor Perspective on Weighted Automata, Low-Rank Regression and Algebraic Mixtures*, PhD thesis, University of Aix-Marseille, 2016.

[44] P. Rai, *Sparse Low Rank Approximation of Multivariate Functions – Applications in Uncertainty Quantification*, doctoral thesis, Ecole Centrale Nantes, Nov. 2014, https://tel.archives-ouvertes.fr/tel-01143694.

[45] R. C. Rothermel, *A mathematical model for predicting fire spread in wildland fuels*, tech. report, U.S. Department of Agriculture, Intermountain Forest and Range Experiment Station, 1972.

[46] A. Saltelli, P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola, *Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index*, Computer Physics Communications, 181 (2010), pp. 259–270.

[47] A. Saltelli, S. Tarantola, F. Campolongo, and M. Ratto, *Sensitivity Analysis in Practice: A Guide to Assessing Scientific Models*, Halsted Press, New York, NY, USA, 2004.

[48] R. Salvador, J. Piñol, S. Tarantola, and E. Pla, *Global sensitivity analysis and scale effects of a fire propagation model used over Mediterranean shrublands*, Ecological Modelling, 136 (2001), pp. 175 – 189.

[49] D. V. Savostyanov, *Quasioptimality of maximum-volume cross interpolation of tensors*, Linear Algebra and its Applications, 458 (2014), pp. 217 – 244.

[50] D. V. Savostyanov and I. V. Oseledets, *Fast adaptive interpolation of multi-dimensional arrays in tensor train format*, in Proceedings International Workshop on Multidimensional Systems, 2011.

[51] L. S. Shapley, *A value for n-person games*, Contributions to the theory of games, 2 (1953), pp. 307–317.
[52] I. M. Sobol', *Sensitivity estimates for nonlinear mathematical models (in Russian)*, Mathematical Models, 2 (1990), pp. 112–118.
[53] E. Song, B. L. Nelson, and J. Staum, *Shapley effects for global sensitivity analysis: Theory and computation*, SIAM/ASA Journal on Uncertainty Quantification, 4 (2016), pp. 1060–1083.
[54] Q. Song, H. Ge, J. Caverlee, and X. Hu, *Tensor Completion Algorithms in Big Data Analytics*, ArXiv e-prints, (2017), https://arxiv.org/abs/1711.10105, https://arxiv.org/abs/1711.10105.
[55] M. Steinlechner, *Riemannian optimization for high-dimensional tensor completion*, Tech. Report 05.2015, Mathematics Institute of Computational Science and Engineering, EPFL, March 2015.
[56] E. E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing, 64 (2000), pp. 367–380.
[57] N. Vervliet, O. Debals, L. Sorber, and L. D. Lathauwer, *Breaking the curse of dimensionality using decompositions of incomplete tensors: Tensor-based scientific computing in big data analysis.*, IEEE Signal Processing Magazine, 31 (2014), pp. 71–79.
[58] A. Yankov, *Analysis of Reactor Simulations Using Surrogate Models*, PhD thesis, University of Michigan, 2015.
[59] Z. Zhang, X. Yang, I. V. Oseledets, G. E. Karniadakis, and L. Daniel, *Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34 (2015), pp. 63–76.