Algorithm 1040: The Sparse Grids Matlab Kit - a Matlab implementation of sparse grids for high-dimensional function approximation and uncertainty quantification

CHIARA PIAZZOLA, Technische Universität München, Germany and Consiglio Nazionale delle Ricerche, Italy
LORENZO TAMELLINI, Consiglio Nazionale delle Ricerche, Italy

The Sparse Grids Matlab Kit provides a Matlab implementation of sparse grids, and can be used for approximating high-dimensional functions and, in particular, for surrogate-model-based uncertainty quantification. It is lightweight, high-level and easy to use, good for quick prototyping and teaching; however, it is equipped with some features that allow its use also in realistic applications. The goal of this paper is to provide an overview of the data structure and of the mathematical aspects forming the basis of the software, as well as comparing the current release of our package to similar available software.

CCS Concepts: • **Mathematics of computing** → *Quadrature*; *Interpolation*; *Approximation*; **Mathematical software;**

Additional Key Words and Phrases: High-dimensional functions, uncertainty quantification, sparse grids, reduced order modeling, surrogate modeling

ACM Reference format:
Chiara Piazzola and Lorenzo Tamellini. 2024. Algorithm 1040: The Sparse Grids Matlab Kit - a Matlab implementation of sparse grids for high-dimensional function approximation and uncertainty quantification. *ACM Trans. Math. Softw.* 50, 1, Article 7 (March 2024), 22 pages.
https://doi.org/10.1145/3630023

1 INTRODUCTION

The reliability of computer simulations in science and engineering crucially depends on having a precise knowledge of the input parameters of such simulations, such as coefficients, forcing terms, initial and boundary conditions, shape of the computational domain, and so on. However,

Lorenzo Tamellini and Chiara Piazzola have been supported by the PRIN 2017 project 201752HKH8 "Numerical Analysis for Full and Reduced Order Methods for the efficient and accurate solution of complex systems governed by Partial Differential Equations (NA-FROM-PDEs)". Lorenzo Tamellini has been partially supported by ICSC—Centro Nazionale di Ricerca in High Performance Computing, Big Data, and Quantum Computing funded by European Union—NextGenerationEU. Chiara Piazzola has been also supported by the Alexander von Humboldt Foundation.
Authors' addresses: C. Piazzola, Technische Universität München, Department of Mathematics, Boltzmannstraß, 3, 85748, Garching bei München, Germany and Consiglio Nazionale delle Ricerche, Istituto di Matematica Applicata e Tecnologie Informatiche E. Magenes", Via Ferrata, 5/A, 27100, Pavia, Italy; e-mail: chiara.piazzola@tum.de; L. Tamellini, Consiglio Nazionale delle Ricerche, Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Via Ferrata, 5/A, 27100, Pavia, Italy; e-mail: tamellini@imati.cnr.it.

0098-3500/2024/03-ART7 $15.00
https://doi.org/10.1145/3630023

in practical scenarios it is quite common to have only a partial knowledge of said parameters, due e.g. to measurement errors, temporal/monetary/technological constraints on running an experimental campaign, or intrinsic randomness of certain quantities. It is, therefore, useful to model these uncertain parameters as random variables, and to think of a computer simulation as a function f that associates the corresponding simulation outputs to each possible value of such uncertain input parameters. The values of f (i.e., the simulation outputs) are then also random/uncertain quantities, whose statistical properties should be assessed. This kind of analysis is called Uncertainty Quantification (UQ).

UQ analyses pose significant theoretical and computational challenges, mainly due to two facts: (1) the number of uncertain parameters could be very large i.e., f could be a high-dimensional function; (2) evaluating f requires running a computer simulation, which could be computationally expensive. One major research direction that has been explored in literature to deal with these issues consists in a two-step procedure: (1) deriving an approximation of f, say $\widehat{f}$ (that can go under different names, each carrying different meaning nuances also depending on the scientific community: surrogate model, proxy model, response function, reduced order model), that is ideally both cheap to obtain and much faster to evaluate than f; (2) computing the statistical properties of $\widehat{f}$ instead of those of f. Among the several surrogate modeling techniques available in literature (see e.g. [29, 65]), we focus here on the so-called sparse-grids method.

More specifically, the aim of this manuscript is to introduce the Sparse Grids Matlab Kit (SGMK) as a tool for high-dimensional function approximation and UQ. The SGMK is freely available under the BSD2 license on Github [54], and full resources (past and current releases, user manual [56], and other release-related material including source code from selected publications that have used the SGMK) are available on a dedicated website [55]; the first version was released in 2014 (14-4 "Ritchie"), and the current version was released in 2023 (23-5 "Robert"). It is written in Matlab, and its compatibility with Octave has been tested; it is extensively documented and commented (release 23-5 has about 9,800 lines of code and 5,300 lines of comment). The release contains several tutorials and a testing unit is also available.

From a mathematical point of view, the package implements the combination technique form of sparse grids. It is a high-level package with a syntax quite close to the mathematical description of sparse grids which makes it (hopefully) easy to use and, therefore, suitable for quick prototyping and didactic purposes (for example, it has been used to write the codes of the book [40]). However, we will point out a number of functionalities that make the SGMK usable for realistic UQ problems, as well as for problems with hundreds of random variables: an interface with the Matlab Parallel Toolbox; a strategy to recycle evaluations of f that might be already available from previous computations; a so-called buffering strategy that improves the work allocation when f depends on a very large number of random variables; full compatibility with the UM-Bridge protocol [62] for HTTP communication with external software for evaluating f. These features have been used for example to obtain the results shown in [10, 13, 21, 46, 57, 63].

Although being general enough to be used for manipulation of high-dimensional functions in many frameworks, the SGMK is geared towards UQ, as already mentioned. In particular, it provides the following UQ tools: collocation points for several random variables (uniform, normal, exponential, gamma, beta, triangular); computation of generalized Polynomial Chaos Expansions [20, 71, 76] for such random variables; computation of Sobol sensitivity indices [2, 25, 66, 71]; approximation of gradients and Hessians of a function, which could be useful for a number of UQ tasks, such as local sensitivity analysis [9], detection of active subspaces [15], Maximum-A-Posteriori estimation of uncertain parameters [7, 34, 58, 70], and Markov-Chain Monte-Carlo (MCMC) sampling [34, 51, 70]. It is also straightforward to connect the SGMK functions with built-in Matlab functions for minimization and MCMC sampling, as well as for

Table 1. List of High-dimensional Approximation / UQ-related Software

Name	Language	Ref.	Webpage
Dakota	C++	[1]	https://dakota.sandia.gov
PyApprox	Python	[33]	https://pypi.org/project/pyapprox
MUQ	C++, Python	[50]	https://mituq.bitbucket.io
UQLab	Matlab	[39]	https://uqlab.com
ChaosPy	Python	[22, 23]	https://chaospy.readthedocs.io
SG++	Python, Matlab, Java, C++	[52]	https://sgpp.sparsegrids.org/
Spinterp	Matlab	[35–37]	http://calgo.acm.org/847.zip
UQTk	C++, Python	[16, 17]	https://sandia.gov/uqtoolkit
Tasmanian	C++, Python, Matlab, Fortran 90/95	[67–69]	https://github.com/ORNL/TASMANIAN
OpenTURNS	C++, Python	[4]	https://openturns.github.io/www/index.html
URANIE	C++, Python	[5]	https://www.salome-platform.org/?page_id=2019
UncertainSCI	Python	[43]	https://www.sci.utah.edu/cibc-software/uncertainsci.html

Note that the webpage reported for Spinterp corresponds to the last officially released version, to the best of knowledge of the authors of this manuscript. A later version can be found at https://people.sc.fsu.edu/~jburkardt/m_src/spinterp/spinterp.html

standard UQ tasks such as computation of histograms and approximation of probability density functions.

The SGMK belongs to the same niche as a number of other packages for surrogate modeling and UQ purposes; we provide a (knowingly incomplete) list in Table 1. The package in the table closest to the SGMK (in terms of language, functionalities, and usability) is probably Spinterp, which is however no longer maintained and does not implement any UQ function. A deeper discussion is reported in Section 5, where a closer comparison in terms of functionalities is given between the SGMK and the other Matlab-based sparse-grids/UQ software in Table 1 (either natively implemented in Matlab or providing interfaces to software written in C++/Python), i.e., SG++, Spinterp, Tasmanian.

The rest of the paper is organized as follows. Section 2 introduces the minimal mathematical background necessary to understand the entities implemented in the SGMK. Section 3 covers how sparse grids are generated in the SGMK and the data structure used to store them. Note, in particular, that the SGMK provides two mechanisms to generate sparse grids: a-priori and adaptive a-posteriori. Section 4 discusses the main functionalities available in the SGMK, with special emphasis on evaluation recycling, parallelization, interface with the UM-Bridge protocol, and computation of polynomial chaos expansions. Section 5 contains the comparison with the other Matlab sparse-grids/UQ software available in the literature. Finally, Section 6 draws some conclusions. In general, we will keep the discussion on a high-level, language-independent tone. However, from time to time, we will give references to specific parts of the user manual [56], where interested readers can find implementation details.

2 MATHEMATICAL BASICS OF SPARSE GRIDS

We consider the two problems of (a) approximating and (b) computing weighted integrals, i.e., expected values and higher-order moments, of (the components of) a function $f : \mathbb{R}^N \to \mathbb{R}^V$ given some samples of f whose location we are free to choose. More specifically, we assume that f depends on N random variables $\mathbf{y} = (y_1, \ldots, y_N) \in \Gamma$, with $\Gamma = \Gamma_1 \times \ldots \times \Gamma_N \subset \mathbb{R}^N$ being the set of all possible values of $\mathbf{y}$. We denote by $\rho_n : \Gamma_n \to \mathbb{R}^+$ the probability density function (pdf) of each variable y_n, $n = 1, \ldots, N$ and assume independence of $y_1, \ldots, y_N$, such that the joint pdf of $\mathbf{y}$ is $\rho(\mathbf{y}) = \prod_{n=1}^{N} \rho_n(y_n), \forall \mathbf{y} \in \Gamma$. Note that the assumption of independence is kept for simplicity

but is actually not needed. Indeed, approximations of f can be built based only on the marginal probability density functions of $y_1, \dots, y_N$, and, while independence is needed for quadrature, in case $y_1, \dots, y_N$ are correlated it is possible to use the theory of copulas [44] to introduce a change of variables where the new random variables are independent uniform random variables, and to set the sparse grid in this new set of variables.

The first step in building a sparse grid is to define a set of collocation knots for each variable y_n. We thus introduce the univariate discretization level $i_n \in \mathbb{N}_+$ and a non-decreasing function, called "level-to-knots function", that specifies the number of collocation knots to be used for each random variable at discretization level i_n, i.e.,

$$m : \mathbb{N}_+ \to \mathbb{N}_+, \ i_n \mapsto m(i_n). \tag{1}$$

Then, we denote by $\mathcal{T}_{n,i_n}$ the set of $m(i_n)$ knots along y_n, i.e.,

$$\mathcal{T}_{n,i_n} = \left\{ y^{(j_n)}_{n,m(i_n)} : j_n = 1, \dots, m(i_n) \right\} \quad \text{for } n = 1, \dots, N. \tag{2}$$

Typical examples of level-to-knots functions are

$$m(i) = i \quad (\textit{linear}), \tag{3}$$

$$m(i) = 2(i-1) + 1 \quad (\textit{2-step}), \tag{4}$$

$$m(1) = 1, \ m(i) = 2^{i-1} + 1 \text{ for } i > 1 \quad (\textit{doubling}). \tag{5}$$

The knots of $\mathcal{T}_{n,i_n}$ are usually chosen according to the probability distribution of the random variables ρ_n, to guarantee a good convergence rate of the resulting sparse-grid approximation, see for example [21, 47, 48]. For efficiency reasons, it is beneficial if the sequences of knots are nested, i.e., if $\mathcal{T}_{n,i_n} \subset \mathcal{T}_{n,j_n}$ with $j_n \geq i_n$. However, the SGMK does not require these sequences to be nested, contrary to other software (see Section 5 for details).

Next, we introduce N-dimensional tensor grids obtained by taking the Cartesian product of the N univariate sets of knots just introduced. For this purpose, we collect the discretization levels i_n in a multi-index $\mathbf{i} = [i_1, \dots, i_N] \in \mathbb{N}_+^N$ and denote the corresponding tensor grid by $\mathcal{T}_{\mathbf{i}} = \bigotimes_{n=1}^{N} \mathcal{T}_{n,i_n}$. Using standard multi-index notation, we can then write

$$\mathcal{T}_{\mathbf{i}} = \left\{ \mathbf{y}^{(\mathbf{j})}_{m(\mathbf{i})} \right\}_{\mathbf{j} \leq m(\mathbf{i})}, \quad \text{with} \quad \mathbf{y}^{(\mathbf{j})}_{m(\mathbf{i})} = \left[y^{(j_1)}_{1,m(i_1)}, \dots, y^{(j_N)}_{N,m(i_N)} \right] \text{ and } \mathbf{j} \in \mathbb{N}_+^N,$$

where $m(\mathbf{i}) = [m(i_1), \ m(i_2), \dots, m(i_N)]$ and $\mathbf{j} \leq m(\mathbf{i})$ means that $j_n \leq m(i_n)$ for every $n = 1, \dots, N$.

A tensor grid approximation of $f(\mathbf{y})$ based on global Lagrangian interpolants collocated at these grid knots can then be written in the following form

$$\mathcal{U}_{\mathbf{i}}(\mathbf{y}) := \sum_{\mathbf{j} \leq m(\mathbf{i})} f\left(\mathbf{y}^{(\mathbf{j})}_{m(\mathbf{i})} \right) \mathcal{L}^{(\mathbf{j})}_{m(\mathbf{i})}(\mathbf{y}), \tag{6}$$

where $\left\{ \mathcal{L}^{(\mathbf{j})}_{m(\mathbf{i})}(\mathbf{y}) \right\}_{\mathbf{j} \leq m(\mathbf{i})}$ are N-variate Lagrange basis polynomials, defined as tensor products of univariate Lagrange polynomials, i.e.,

$$\mathcal{L}^{(\mathbf{j})}_{m(\mathbf{i})}(\mathbf{y}) = \prod_{n=1}^{N} l^{(j_n)}_{n,m(i_n)}(y_n) \quad \text{with} \quad l^{(j_n)}_{n,m(i_n)}(y_n) = \prod_{k=1, k \neq j_n}^{m(i_n)} \frac{y_n - y^{(k)}_{n,m(i_n)}}{y^{(j_n)}_{n,m(i_n)} - y^{(k)}_{n,m(i_n)}}. \tag{7}$$

Note that other choices could be made here, namely replacing Lagrange polynomials in (6) by, for example, splines as in [60], or trigonometric polynomials as in [41]; we discuss this issue further in Section 5 when comparing different sparse-grids software.

Similarly, the tensor grid quadrature of $f(\mathbf{y})$, i.e., the approximation of its weighted integral (expected value), can be computed by taking the integral of the Lagrangian interpolant in (6):

$$\begin{aligned} Q_{\mathbf{i}} := \int_{\Gamma} \mathcal{U}_{\mathbf{i}}(\mathbf{y})\rho(\mathbf{y})\,\mathrm{d}\mathbf{y} &= \sum_{\mathbf{j}\le m(\mathbf{i})} f\left(\mathbf{y}^{(\mathbf{j})}_{m(\mathbf{i})}\right)\left(\prod_{n=1}^{N}\int_{\Gamma_n} l^{(j_n)}_{n,m(i_n)}(y_n)\rho(y_n)\,\mathrm{d}y_n\right) \\ &= \sum_{\mathbf{j}\le m(\mathbf{i})} f\left(\mathbf{y}^{(\mathbf{j})}_{m(\mathbf{i})}\right)\left(\prod_{i=1}^{N}\omega^{(j_n)}_{n,m(i_n)}\right) = \sum_{\mathbf{j}\le m(\mathbf{i})} f\left(\mathbf{y}^{(\mathbf{j})}_{m(\mathbf{i})}\right)\omega^{(\mathbf{j})}_{m(\mathbf{i})}, \end{aligned} \tag{8}$$

where $\omega^{(j_n)}_{n,m(i_n)}$ are the standard quadrature weights obtained by computing the integrals of the associated univariate Lagrange polynomials, and $\omega^{(\mathbf{j})}_{m(\mathbf{i})}$ are their multivariate counterparts.

Naturally, the approximations $\mathcal{U}_{\mathbf{i}}$ and $Q_{\mathbf{i}}$ are more and more accurate the higher the number of collocation knots in each random variable and, therefore, one would ideally choose all the components of $\mathbf{i}$ to be large, say $\mathbf{i} = \mathbf{i}^{\star}$ with $i^{\star}_n \gg 1, \forall n = 1,\ldots,N$. However, obtaining these approximations could be too computationally expensive even for N moderately large, due to fact that they would require $\prod_{n=1}^{N} m(i^{\star}_n)$ evaluations of f, i.e., their cost grows exponentially in N.

To circumvent this issue, the sparse-grids method replaces $\mathcal{U}_{\mathbf{i}^{\star}}$ with a linear combination of multiple coarser $\mathcal{U}_{\mathbf{i}}$, and similarly for $Q_{\mathbf{i}^{\star}}$ (from now on we use the generic symbol $\mathcal{F}_{\mathbf{i}}$ to denote both $\mathcal{U}_{\mathbf{i}}$ and $Q_{\mathbf{i}}$). To this aim, we introduce the so-called "detail operators" (univariate and multivariate). They are defined as follows, with the understanding that $\mathcal{F}_{\mathbf{i}}(\mathbf{y}) = 0$ when at least one component of $\mathbf{i}$ is zero. Thus, let $\mathbf{e}_n$ the nth canonical multi-index, i.e., $(\mathbf{e}_n)_k = 1$ if $n = k$ and 0 otherwise, and define

$$\begin{aligned} &\textbf{Univariate detail:} \quad \Delta_n[\mathcal{F}_{\mathbf{i}}] = \mathcal{F}_{\mathbf{i}} - \mathcal{F}_{\mathbf{i}-\mathbf{e}_n}, \text{ with } 1 \le n \le N, \\ &\textbf{Multivariate detail: } \Delta[\mathcal{F}_{\mathbf{i}}] = \bigotimes_{n=1}^{N} \Delta_n[\mathcal{F}_{\mathbf{i}}], \end{aligned} \tag{9}$$

where taking tensor products of univariate details amounts to composing their actions, i.e.,

$$\Delta[\mathcal{F}_{\mathbf{i}}] = \bigotimes_{n=1}^{N} \Delta_n[\mathcal{F}_{\mathbf{i}}] = \Delta_1\left[\cdots[\Delta_N\left[\mathcal{F}_{\mathbf{i}}\right]]\right].$$

By replacing the univariate details with their definitions, we can then see that this implies that the multivariate operators can be evaluated by evaluating certain full-tensor approximations $\mathcal{F}_{\mathbf{i}}$, and then taking linear combinations:

$$\Delta[\mathcal{F}_{\mathbf{i}}] = \Delta_1\left[\cdots[\Delta_N\left[\mathcal{F}_{\mathbf{i}}\right]]\right] = \sum_{\mathbf{j}\in\{0,1\}^N} (-1)^{\|\mathbf{j}\|_1}\mathcal{F}_{\mathbf{i}-\mathbf{j}}.$$

Observe that by introducing these detail operators a hierarchical decomposition of $\mathcal{F}_{\mathbf{i}}$ can be obtained; indeed, the following telescopic identity holds true:

$$\mathcal{F}_{\mathbf{i}} = \sum_{\mathbf{j}\le\mathbf{i}} \Delta[\mathcal{F}_{\mathbf{j}}]. \tag{10}$$

Example 1 (Telescopic Identity). As an example of (10), consider the case $N = 2$. Recalling that by definition $\mathcal{F}_{[j_1,j_2]} = 0$ when either $j_1 = 0$ or $j_2 = 0$, it can be seen that

$$\begin{aligned} \sum_{[j_1,j_2]\le[2,2]} \Delta[\mathcal{F}_{[j_1,j_2]}] =& \Delta[\mathcal{F}_{[1,1]}] + \Delta[\mathcal{F}_{[1,2]}] + \Delta[\mathcal{F}_{[2,1]}] + \Delta[\mathcal{F}_{[2,2]}] \\ =& \mathcal{F}_{[1,1]} + (\mathcal{F}_{[1,2]} - \mathcal{F}_{[1,1]}) + (\mathcal{F}_{[2,1]} - \mathcal{F}_{[1,1]}) + (\mathcal{F}_{[2,2]} - \mathcal{F}_{[2,1]} - \mathcal{F}_{[1,2]} + \mathcal{F}_{[1,1]}) \\ =& \mathcal{F}_{[2,2]}. \end{aligned}$$

The crucial observation that allows us to obtain a sparse grid is that, under suitable regularity assumptions for $f(\mathbf{y})$, not all of the details in the hierarchical decomposition in (10) contribute equally to the approximation, i.e., some of them can be discarded and the resulting formula will retain good approximation properties at a fraction of the computational cost: roughly speaking, the multi-indices to be discarded are those corresponding to "high-order" details, i.e., those for which $\|\mathbf{j}\|_1$ is sufficiently large, see, for example, [8]. A simple way to discard high-order details would be for instance to replace the summation set $\{\mathbf{j} \leq \mathbf{i}\}$ in (10) with $\{\|\mathbf{j}\|_1 \leq w\}$ for $w \in \mathbb{N}_+$ small enough, as shown in the next Example 2.

Example 2 (Discarding High-order Details). Following Example 1 and replacing the constraint $[j_1, j_2] \leq [2, 2]$ with $\|\mathbf{j}\|_1 \leq 3$, we obtain the following approximation:

$$\mathcal{F}_{[2,2]} \approx \sum_{j_1+j_2 \leq 3} \Delta[\mathcal{F}_{[j_1,j_2]}] = \Delta[\mathcal{F}_{[1,1]}] + \Delta[\mathcal{F}_{[1,2]}] + \Delta[\mathcal{F}_{[2,1]}] = -\mathcal{F}_{[1,1]} + \mathcal{F}_{[1,2]} + \mathcal{F}_{[2,1]}.$$

In general, upon collecting the multi-indices to be retained in the sum in a multi-index set $\mathcal{I} \subset \mathbb{N}_+^N$ the sparse-grid approximation of f and of its weighted integral can finally be written as (see, for example, [74]):

$$f(\mathbf{y}) \approx \mathcal{U}_{\mathcal{I}}(\mathbf{y}) = \sum_{\mathbf{i} \in \mathcal{I}} \Delta[\mathcal{U}_{\mathbf{i}}(\mathbf{y})] = \sum_{\mathbf{i} \in \mathcal{I}} c_{\mathbf{i}} \, \mathcal{U}_{\mathbf{i}}, \quad c_{\mathbf{i}} := \sum_{\substack{\mathbf{j} \in \{0,1\}^N \\ \mathbf{i}+\mathbf{j} \in \mathcal{I}}} (-1)^{\|\mathbf{j}\|_1} \tag{11}$$

$$\int_\Gamma f(\mathbf{y}) \rho(\mathbf{y}) \, \mathrm{d}\mathbf{y} \approx Q_{\mathcal{I}}(\mathbf{y}) = \sum_{\mathbf{i} \in \mathcal{I}} \Delta[Q_{\mathbf{i}}(\mathbf{y})] = \sum_{\mathbf{i} \in \mathcal{I}} c_{\mathbf{i}} Q_{\mathbf{i}}, \tag{12}$$

and the sparse grid is defined as

$$\mathcal{T}_{\mathcal{I}} = \bigcup_{\substack{\mathbf{i} \in \mathcal{I} \\ c_{\mathbf{i}} \neq 0}} \mathcal{T}_{\mathbf{i}}. \tag{13}$$

Remark 1. The sparse-grid approximation in (11) is not necessarily an interpolant operator, i.e., the equality $f(\mathbf{y}_k) = \mathcal{U}_{\mathcal{I}}(\mathbf{y}_k)$ for $\mathbf{y}_k \in \mathcal{T}_{\mathcal{I}}$ does not necessarily hold true. More specifically, a sparse grid is interpolatory only if it is built with nested knots. In the following, with a slight abuse of terminology, we will nonetheless refer to the operation of evaluating $\mathcal{U}_{\mathcal{I}}(\mathbf{y})$ by means of (11) as sparse-grid interpolation. Another commonly used terminology, especially in the field of uncertainty quantification, is to refer to $\mathcal{U}_{\mathcal{I}}(\mathbf{y})$ as the sparse-grid surrogate model of f.

The right-most equalities in (11) and (12) are known as the combination technique form of the sparse-grid approximation and quadrature, respectively (see [30]), which is the form implemented in the SGMK. Another possibility would be to implement the first form, i.e., the sum of detail operators $\sum_{\mathbf{i} \in \mathcal{I}} \Delta[\mathcal{U}_{\mathbf{i}}(\mathbf{y})]$, which is known as the hierarchical form of sparse grids. In particular, implementing the hierarchical form requires introducing a basis for the detail operators $\Delta[\mathcal{F}_{\mathbf{i}}]$ in (9) rather than for the tensor interpolants $\mathcal{U}_{\mathbf{i}}$: this request naturally suggests using the hierarchical form of piecewise polynomials as a basis (for example, the classical hat-functions). In turn, this opens the way to the so-called local adaptivity of sparse grids, as discussed in [18, 38, 49, 53]. Using piecewise polynomials to introduce a basis for the detail operators $\Delta[\mathcal{F}_{\mathbf{i}}]$ is not mandatory though and it would be possible to use global Lagrange polynomials even in this context, using the hierarchical form of Lagrange polynomials described, for example, in [12, 19]. Note that the equivalence between the two forms is true only if $\mathcal{I}$ is chosen as downward closed, i.e.,

$$\forall \mathbf{k} \in \mathcal{I}, \quad \mathbf{k} - \mathbf{e}_n \in \mathcal{I} \text{ for every } n = 1, \ldots, N \text{ such that } k_n > 1, \tag{14}$$

see Figure 1. The choice of implementing the combination technique instead of the hierarchical form allows to keep the data structure to a minimum, and guarantees ease of use and high-level,

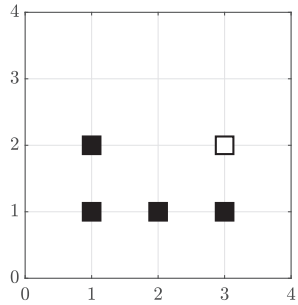


Fig. 1. Downward closedness of a multi-index set. The set of the multi-indices marked in black is downward closed. Instead, the multi-index $[3, 2]$ (in white) violates the rule in (14): the multi-index $[2, 2] = [3, 2] - \mathbf{e}_1$ is not contained in the multi-index set and hence the set of black and white multi-indices is not downward closed.

"close-to-the-math" coding. In Section 5 we examine a number of existing software packages and specify whether they implement the combination technique or the hierarchical form of sparse grids (or both).

Coming back to the choice of the set $\mathcal{I}$, the optimal choice depends on the decay rate of the norm of the detail operators, which in turn depends on the regularity of f, see, for example, [8, 12, 45]. One classical choice of the set $\mathcal{I}$ is the following one:

$$\mathcal{I}_{\text{sum}}(w) = \left\{ \mathbf{i} \in \mathbb{N}_+^N : \sum_{n=1}^{N} (i_n - 1) \leq w \right\}, \tag{15}$$

for some $w \in \mathbb{N}$. In particular, together with the doubling level-to-knots, (5), this results in the famous Smolyak grid. Examples of other choices are

$$\text{Tensor product set: } \mathcal{I} = \left\{ \mathbf{i} \in \mathbb{N}_+^N : \max_{n=1}^{N} g_n(i_n - 1) \leq w \right\}, \tag{16}$$

$$\text{Total degree set: } \mathcal{I} = \left\{ \mathbf{i} \in \mathbb{N}_+^N : \sum_{n=1}^{N} g_n(i_n - 1) \leq w \right\}, \tag{17}$$

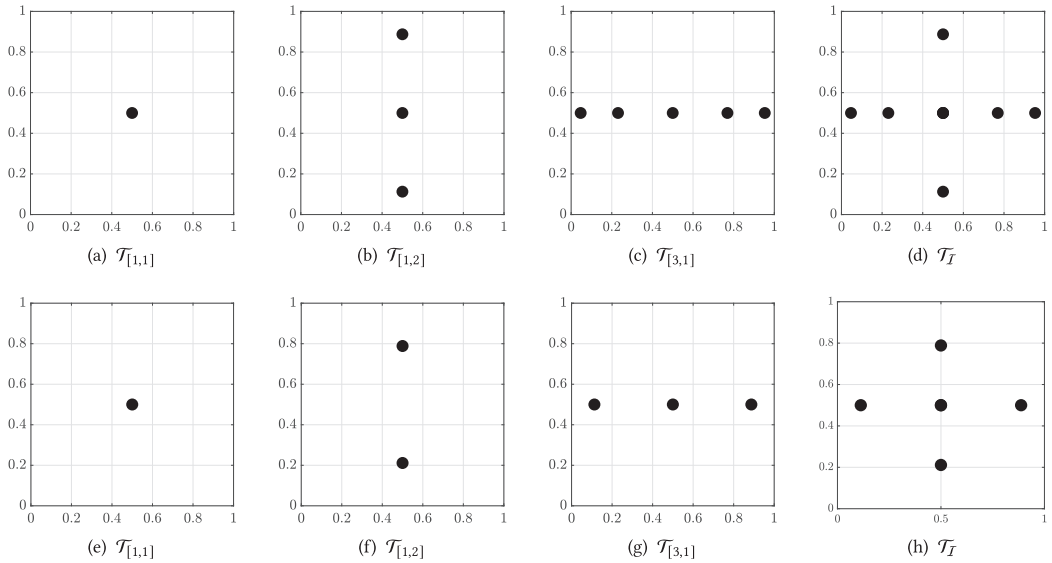
$$\text{Hyperbolic cross set: } \mathcal{I} = \left\{ \mathbf{i} \in \mathbb{N}_+^N : \prod_{n=1}^{N} (i_n)^{g_n} \leq w \right\}, \tag{18}$$

$$\text{Multi-index box set: } \mathcal{I} = \left\{ \mathbf{i} \in \mathbb{N}_+^N : i_n \leq b_n \right\}, \tag{19}$$

for $g_1, \ldots, g_N \in \mathbb{R}, w, b_1 \ldots, b_N \in \mathbb{N}$, where, in particular, $g_1, \ldots, g_N$ are typically called anisotropy weights and w sparse-grid level. For a thorough discussion of the motivations for introducing the sets above, see, for example, [3]. The set $\mathcal{I}$ could also be built adaptively based on suitable profit indicators; this leads to adaptive sparse-grid algorithms, which will be discussed in Section 3.1.

Example 3 (Sparse-Grid Construction). Let $N = 2$ and consider the downward closed multi-index set reported in Figure 1, i.e., $\mathcal{I} = \{[1, 1], [1, 2], [2, 1], [3, 1]\}$. We first show the combination technique form of the sparse-grid approximation and quadrature in (11) and (12), respectively. We use again the generic symbol $\mathcal{F}_\mathbf{i}$ to denote both $\mathcal{U}_\mathbf{i}$ and $\mathcal{Q}_\mathbf{i}$ and obtain,

$$\mathcal{F}_\mathcal{I}(\mathbf{y}) = c_{[1,1]}\mathcal{F}_{[1,1]}(\mathbf{y}) + c_{[1,2]}\mathcal{F}_{[1,2]}(\mathbf{y}) + c_{[2,1]}\mathcal{F}_{[2,1]}(\mathbf{y}) + c_{[3,1]}\mathcal{F}_{[3,1]}(\mathbf{y}),$$



(a) $\mathcal{T}_{[1,1]}$ (b) $\mathcal{T}_{[1,2]}$ (c) $\mathcal{T}_{[3,1]}$ (d) $\mathcal{T}_{\mathcal{I}}$

(e) $\mathcal{T}_{[1,1]}$ (f) $\mathcal{T}_{[1,2]}$ (g) $\mathcal{T}_{[3,1]}$ (h) $\mathcal{T}_{\mathcal{I}}$

Fig. 2. Top row: tensor grids (panels a, b, c) and the sparse grid (panel d) for Example 3 with level-to-knots doubling. Bottom row: tensor grids (panels e, f, g) and the sparse grid (panel h) for the same Example with level-to-knots linear.

with

$$
\begin{aligned}
c_{[1,1]} &= (-1)^{\|[0,0]\|_1} + (-1)^{\|[1,0]\|_1} + (-1)^{\|[0,1]\|_1} = -1,\\
c_{[1,2]} &= (-1)^{\|[0,0]\|_1} = +1,\\
c_{[2,1]} &= (-1)^{\|[0,0]\|_1} + (-1)^{\|[1,0]\|_1} = 0,\\
c_{[3,1]} &= (-1)^{\|[0,0]\|_1} = +1.
\end{aligned}
$$

Since $c_{[2,1]} = 0$, only three Lagrangian interpolant/quadrature operators explicitly appear in the combination technique formulas (11) and (12), i.e.,

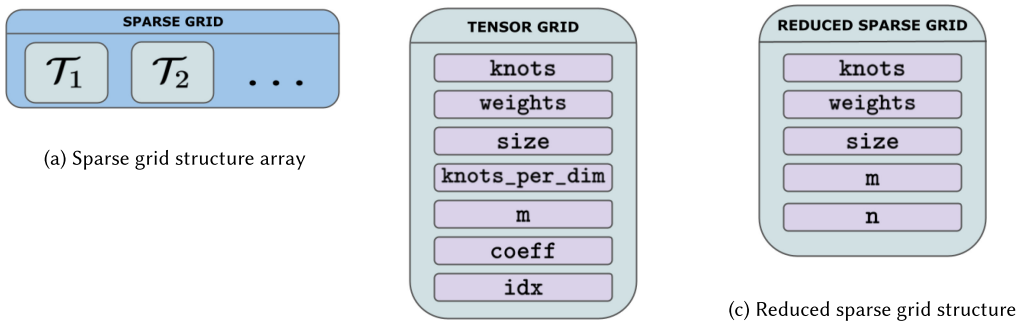
$$\mathcal{F}_{\mathcal{I}}(\mathbf{y}) = -\mathcal{F}_{[1,1]}(\mathbf{y}) + \mathcal{F}_{[1,2]}(\mathbf{y}) + \mathcal{F}_{[3,1]}(\mathbf{y}),$$

and only the corresponding three tensor grids contribute to the sparse grid (cf. (13)). Then, considering the doubling level-to-knots function (5), the resulting tensor grids consist of one, three, and five grid knots, respectively:

$$
\begin{aligned}
\mathcal{T}_{[1,1]} &= \left\{ \begin{bmatrix} y^1_{1,1} & y^1_{2,1} \end{bmatrix} \right\},\\
\mathcal{T}_{[1,2]} &= \left\{ \begin{bmatrix} y^1_{1,1} & y^1_{2,3} \end{bmatrix}, \begin{bmatrix} y^1_{1,1} & y^2_{2,3} \end{bmatrix}, \begin{bmatrix} y^1_{1,1} & y^3_{2,3} \end{bmatrix} \right\},\\
\mathcal{T}_{[3,1]} &= \left\{ \begin{bmatrix} y^1_{1,5} & y^1_{2,1} \end{bmatrix}, \begin{bmatrix} y^2_{1,5} & y^1_{2,1} \end{bmatrix}, \begin{bmatrix} y^3_{1,5} & y^1_{2,1} \end{bmatrix}, \begin{bmatrix} y^4_{1,5} & y^1_{2,1} \end{bmatrix}, \begin{bmatrix} y^5_{1,5} & y^1_{2,1} \end{bmatrix} \right\}.
\end{aligned}
$$

Choosing Gauss-Legendre knots as univariate collocation knots on $\Gamma_1 = \Gamma_2 = [0,1]$ leads to the tensor grids displayed in Figure 2(a), (b), (c) and the sparse grid of Figure 2(d). If instead, we consider the linear level-to-knots function (3), the three tensor grids will consist of one, two, and three knots

$$
\begin{aligned}
\mathcal{T}_{[1,1]} &= \left\{ \begin{bmatrix} y^1_{1,1} & y^1_{2,1} \end{bmatrix} \right\},\\
\mathcal{T}_{[1,2]} &= \left\{ \begin{bmatrix} y^1_{1,1} & y^1_{2,2} \end{bmatrix}, \begin{bmatrix} y^1_{1,1} & y^2_{2,2} \end{bmatrix} \right\},\\
\mathcal{T}_{[3,1]} &= \left\{ \begin{bmatrix} y^1_{1,3} & y^1_{2,1} \end{bmatrix}, \begin{bmatrix} y^2_{1,3} & y^1_{2,1} \end{bmatrix}, \begin{bmatrix} y^3_{1,3} & y^1_{2,1} \end{bmatrix} \right\}.
\end{aligned}
$$



(a) Sparse grid structure array

(b) Tensor grid structure

(c) Reduced sparse grid structure

Fig. 3. Sparse-grid data structure: a sparse grid is stored in extended format in a structure array (panel a), each structure corresponding to one tensor grid (see panel b) that contains seven fields that identify the current grid. The data structure for the reduced format is instead shown in panel c.

and the same choice of Gauss–Legendre knots as univariate collocation knots on $\Gamma_1 = \Gamma_2 = [0, 1]$ leads to the tensor grids displayed in Figure 2(e), (f), (g) and the sparse grid of Figure 2(h).

3 THE SPARSE GRIDS MATLAB KIT: SPARSE-GRID DATA STRUCTURE

As described in the previous section, defining a sparse grid requires choosing a family $\mathcal{T}_{n,i_n}$ of collocation knots for each variable y_n, a level-to-knots function $m(\cdot)$ and a multi-index set $\mathcal{I}$ to be input in the combination technique formulas of (11) and (12). In the SGMK, the same steps are to be followed to define a sparse grid, as can be seen in the SGMK user manual, Listing 1 in Section 3 and the subsequent discussion. As already discussed, the knots should be chosen according to the distribution of each random variable y_n. The SGMK supports uniform, normal, exponential, gamma, beta and triangular distributions, and provides at least two choices of collocation knots (namely, Gaussian and weighted Leja knots, see [42, 59] respectively) for almost all of them; more options are provided for specific choices of random variables, such as midpoint, equispaced and Clenshaw–Curtis knots [73] for uniform random variables, and Genz–Keister [27] for normal distributions; the user manual reports some discussion on the algorithms used to compute the knots and quadrature weights for the various choices in Section 3.1. Several choices are also available for the level-to-knots functions, including but not limited to the ones reported in (3)–(5), as well as for generating the multi-index sets in (16)–(19). We refer to Section 3 of the user manual for a thorough discussion on the various options available.

The data structure chosen to store a sparse grid is also quite close to the mathematical formalism: following the definition of a sparse grid in (13) as a union of tensor grids, the SGMK stores a sparse grid as a structure array, where each component of the array stores a single tensor grid (see Figure 3(a)). Of course, only tensor grids whose coefficient in the combination technique formula is non-zero are stored, see (11) and (12).

The structures storing a tensor grid are also quite minimal and contain only seven fields. Following the ordering in Figure 3(b), the fields of the structure are: a matrix storing the knots of the tensor grid $\mathcal{T}_{\mathbf{i}}$ (each knot being a column), a vector collecting the corresponding quadrature weights multiplied by the combination technique coefficient of the current tensor grid, i.e., $\omega_{m(\mathbf{i})}c_{\mathbf{i}}$, an integer recording the number of knots/weights, a cell array containing the the univariate sets of collocation knots $\mathcal{T}_{n,i_n}$, a row vector containing the number of knots in each dimension, an integer

storing the coefficient c_i of the combination technique formula, see (11), and finally another row vector containing the multi-index from which the tensor grid is generated.

Note, however, that there is some redundancy in the information stored in the structure array. Specifically, the same knot might appear in more than one tensor grid and thus be stored more than once (for instance, the knot $(0.5, 0.5)$ appears in all of the tensor grids forming the sparse grids on both rows of Figure 2); this phenomenon is even more pronounced (actually, desired!) when nested sequences of knots are used. Therefore, it is useful to have at disposal a compact representation of a sparse grid, where duplicate knots are stored only once, together with their "lumped" quadrature weights, obtained by taking the linear combination of the quadrature weights of each instance of the repeated knot with the combination coefficient weights in (11). This representation is called a reduced sparse grid and is obtained by detecting (up to a certain tolerance, tunable by the user) the identical knots, deleting the possible repetitions, and computing the corresponding quadrature weights. The resulting data structure is a single structure (see Figure 3(c)), with five fields: a matrix where each collocation knot of the sparse grid appears only once (each knot being a column), a vector for their corresponding "lumped" quadrature weights, an integer containing the number of knots/weights, and finally two vectors of indices mapping from the list of knots with repetitions to the reduced version and vice-versa.

Note that both extended and reduced formats are useful for working with sparse grids and should always be stored in memory. This implies a certain redundancy in memory storage, and reducing a sparse grid takes a non-negligible computational time, as we further discuss in Example 4 below. However, having the two structures at hand considerably simplifies coding operations on sparse grids such as interpolation or conversion to Polynomial Chaos Expansion (these operations are described in details in Section 4), and hides a lot of complexity from the final user.

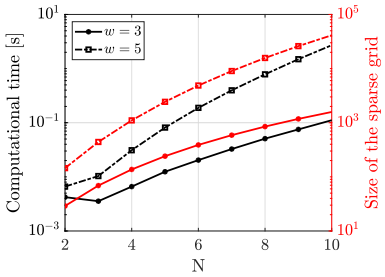
Remark 2. Once the family of knots and the level-to-knots function are known, it would be in principle possible to perform the sparse-grid reduction comparing indices of knots rather than coordinates, which is faster and does not need to use a tolerance. However, we decided to go for the admittedly slower alternative and compare coordinates rather than indices, as this makes it easier for a generic user to introduce their own family of collocation knots.

Indeed, especially for non-nested knots, determining if sets of knots at different levels (not just consecutive ones) have knots in common, i.e., assessing $\mathcal{T}_{n,k} \cap \mathcal{T}_{n,j}$ for generic choices of $k, j \in \mathbb{N}$ is not straightforward. This information would however be needed if we were to use only indices when reducing a grid, thus the user would need to provide such information in a suitable format (and possibly precompute it offline – again up to a tolerance), which might be not easy to do.

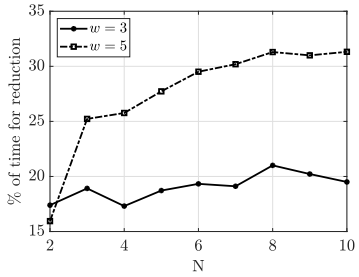
Comparing coordinates instead allows much more flexibility in the way knots are provided by the user. In particular, the user does not even need to precompute knots offline, but can just provide a function that computes them at each call, knowing that the reduction operation is robust to differences in coordinates induced by floating point arithmetic. This is actually what happens for Clenshaw–Curtis and Gaussian knots that are not precomputed and tabulated but rather computed at each call.

Example 4 (Computational Cost). Here we measure memory usage and CPU time for generating Smolyak sparse grids (i.e., using (15) to generate the multi-index set and the level-to-knots function in (5)) in reduced format, for increasing $N = 2, \dots, 10$ and for fixed $w = 3$ or $w = 5$. In this example, we use the so-called Clenshaw–Curtis knots, a choice that ensures that the grids generated will be nested. We display the growth of the size of the resulting sparse grid and the computational time in Figure 4(a), and the percentage of computational time taken by the reduction step in Figure 4(b).

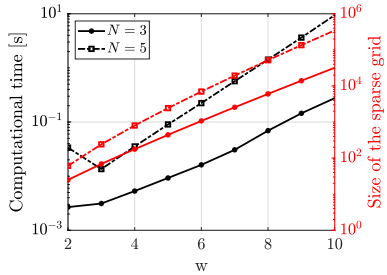
We then swap the role of w and N and perform the dual experiment, i.e., we measure memory usage and CPU time for generating reduced Smolyak sparse grids for fixed $N = 3$ or $N = 5$ and



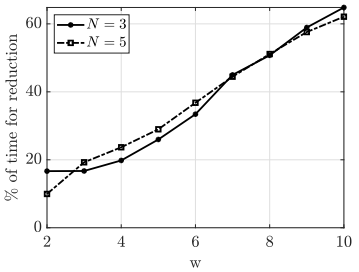
(a) Sparse grid size and total computational time for increasing N (logarithmic scale in the vertical axis)



(b) Percentage of computational time taken by the reduction step for increasing N



(c) Sparse grid size and computational time for increasing w (logarithmic scale in the vertical axis)



(d) Percentage of computational time taken by the reduction step for increasing w

Fig. 4. Computational cost and size of sparse grids for different values of N and w. This test was carried out in Matlab 2019b on a standard laptop with processor Intel(R) Core(TM) i7-8665U CPU 2.10/4.80 GHz and 16 GB RAM.

increasing $w = 2, \ldots, 10$. Sparse-grid size and total CPU time are now reported in Figure 4(c) while Figure 4(d) shows the percentage of time taken by the reduction step.

Both the computational time and the sparse-grid size can be seen to grow faster with respect to w than N. Moreover, the time taken by the reduction step is only slightly increasing the total time when keeping w to small values and increasing N (panel b), whereas steadily increasing with w (panel d). This phenomenon is partially due to the chosen level-to-knots function which doubles the number of points each time w is increased, and using another type of level-to-knots function, which grows points more slowly, could result in a lower percentage of time being spent on reduction.

3.1 Adaptive Sparse-Grid Generation

The straightforward way of generating a sparse grid is by specifying a-priori the multi-index set $\mathcal{I}$, i.e., before sampling the function f. However, as already mentioned, a greedy adaptive approach in which the multi-index set (and hence the approximation of f) is constructed in an iterative way, relying on some heuristic criteria based on the values of the function f obtained so far, is often beneficial.

The adaptive algorithm implemented in the SGMK is described in details in [46] and extends the original by Gerstner and Griebel in [28] in several ways, as listed below; an alternative approach was proposed by Stoyanov and Webster in [41, 69].

Roughly speaking, the Gerstner–Griebel algorithm starts with the trivial multi-index set $\mathcal{I} = \{[1, 1, \ldots, 1]\}$ and iteratively adds to $\mathcal{I}$ the multi-index $\mathbf{i}$ with the largest heuristic profit indicator

chosen from a set of candidates, called the reduced margin of $\mathcal{I}$ and defined as follows:

$$\mathcal{R}_{\mathcal{I}} = \{\mathbf{i} \in \mathbb{N}_+^N \text{ s.t. } \mathbf{i} \notin \mathcal{I} \text{ and } \mathbf{i} - \mathbf{e}_n \in \mathcal{I} \ \ \forall n \in \{1, \dots, N\} \text{ s.t. } i_n > 1\}.$$

Note that the condition $\mathbf{i} \in \mathcal{R}_{\mathcal{I}}$ is required to guarantee that $\mathcal{I} \cup \{\mathbf{i}\}$ is downward closed, cf. (14). The role of the profit indicator is to balance error reduction and additional computational costs brought in by each multi-index $\mathbf{i}$ (where the cost is measured as the number of new evaluations of f needed to add $\mathbf{i}$ to $\mathcal{I}$); in other words, it quantifies the fact that ideally, we would like to add to the sparse grid multi-indices that carry a large reduction in interpolation/quadrature error for a minimal extra cost. Convergence of this algorithm was recently proved for certain classes of problems, see [19, 24].

The computation of the profit of a multi-index $\mathbf{i}$ actually requires evaluating f at the new collocation knots that would be added to the sparse grid. This explains why the Gerstner–Griebel algorithm is typically referred to as an a-posteriori adaptive algorithm, and it is, in some sense, a sub-optimal procedure, since computational work is invested in assessing the profit of multi-indices which might then turn out to be "useless". A variant of the algorithm, where the computation of the profit does not require evaluating f at the new knots, is proposed in [31]. Note however that such a profit estimator is tailored to the specific f considered in [31]. Therefore, this variant of the adaptive algorithm cannot be applied verbatim to any f but must be suitably adjusted to each case, which is a highly non-trivial task.

We also mention that the Gerstner–Griebel algorithm is sometimes referred to as dimension-adaptive: indeed, in the framework introduced so far, the algorithm will add more knots in the variables that are deemed more important, but these knots are spread throughout the whole support of the random variables (any clustering being a consequence only of the marginal pdfs $\rho_1, \dots, \rho_N$) rather than localized in certain regions of the support where the algorithm has detected local features of f. The latter algorithm would be the locally-adaptive one that we already mentioned in the previous section [18, 38, 49, 53], and that is not supported in the current release of the SGMK.

As already hinted, the version of the adaptive sparse-grid algorithm implemented in the SGMK extends the original by Gerstner and Griebel in several ways, see also Section 3.4.2 of the user manual for a more implementation-oriented discussion and customization options:

- it can use several profit definitions, (see user manual, Table 8);
- it can use non-nested knots. This requires introducing suitable profit indicators, as discussed in [46]. Furthermore, note that in this case, it will generally happen that some of the evaluations of f that have been requested at intermediate iterations will not be used in the final sparse-grid approximation, because they are associated to multi-indices whose combination coefficient have dropped to zero after having been non-zero for a few iterations. In this case however, the evaluations of f on these "removed" points are not deleted from memory but rather stored in a suitable list of evaluations, from which they will be recovered should they be needed in a later iteration (see also the later Section 4.1 about "evaluation recycling – recycling from an unstructured list of points"). The software allows also to specify whether such search should be done or if the software should just evaluate again f (which could be faster in case the evaluation of f is very fast and the list of points very large). This complexity is however invisible to the end user, who is only in charge of setting correctly the input flags that control the execution of the algorithm, namely to specify that the implementation for non-nested knots must be used, and to indicate whether the algorithm should look for knots in the list of "removed points" or not (see user manual, Table 7).
- it can operate on vector-valued functions, which also requires suitable definitions for the profit indicators, see Equations (12)–(13) of the user manual and discussion thereafter;

— it implements the so-called dimension-buffering, see [46] and user manual, Table 7; this improves the performance of the Gerstner–Griebel algorithm when the function f is "very high-dimensional", $N \gg 1$. Indeed, when $N \gg 1$, the size of the reduced margin $\mathcal{R}_{\mathcal{I}}$ grows very quickly, which in turn implies a quick growth of the number of evaluations of f. In this case, if we know that $y_1, \ldots, y_N$ are "sorted decreasingly according to their importance" (this might be, for example, the case when f is the solution of a partial differential equation (PDE) with uncertain coefficients represented by a Karhunen–Loève expansion), the SGMK adaptive algorithm starts by exploring only an initial subset of dimensions (the most relevant ones) and then gradually adds more dimensions to the approximation, thus limiting the number of indices in the candidate set. More specifically, the algorithm splits the random variables in three groups: activated, buffered (or non-activated), and neglected. A variable y_k is said to be buffered if the algorithm has computed the profit of the "first non-trivial multi-index" in the random variable y_k, i.e., of $\mathbf{b}_k = [1\,1\,1\,\cdots] + \mathbf{e}_k$, but $\mathbf{b}_k$ has not been selected yet (i.e., its profit is not the highest one in the list of candidates); when $\mathbf{b}_k$ is selected, it is moved from the list of candidates to $\mathcal{I}$ and the variable y_k becomes activated. Then, when the adaptive algorithm is run in "buffered mode", with N_{buf} variables:
  — it begins considering $N_{cur} = N_{buf}$ variables, i.e. the set of candidate multi-indices is $\mathcal{R}_{\mathcal{I}} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{N_{cur}}\}$ and all other variables $y_{N_{cur}+1}, y_{N_{cur}+2}, \ldots$ are neglected;
  — as soon as $\mathbf{b}_k$ gets selected (i.e., y_k becomes activated) for some $k \in \{1, \ldots, N_{buf}\}$, the number of current variables N_{cur} is increased by 1 and the first neglected variable $y_{N_{cur}+1}$ becomes a buffered variable.

  In this way, at each iteration the algorithm is forced to explore candidates with at most N_{buf} buffered variables. This approach is also discussed in [12, 61], but only for the special case $N_{buf} = 1$.

4 THE SPARSE GRIDS MATLAB KIT: OPERATIONS ON SPARSE GRIDS

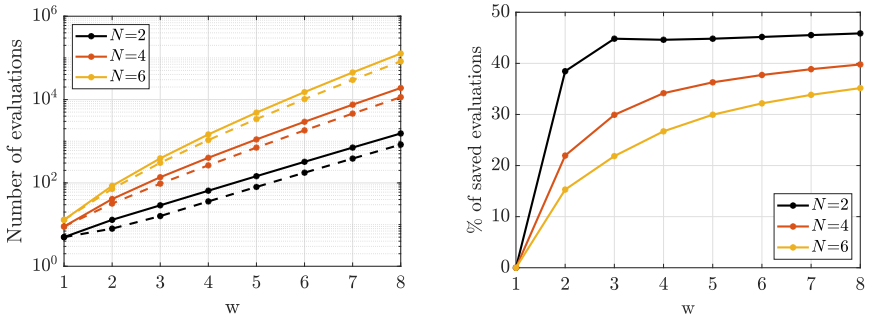
The SGMK implements a number of operations on sparse grids:

— evaluation of f over the collocation knots of the sparse grid;
— interpolation (cf. Remark 1) and quadrature; these operations in practice are the solution to the problems of approximating and integrating f, cf. beginning of Section 2;
— computation of gradients and Hessians (by finite differences) of the sparse-grid interpolant;
— conversion to generalized Polynomial Chaos Expansions (gPCE) and computation of PCE-based Sobol indices for sensitivity analysis.

We discuss below the most noteworthy features implemented in the code, and refer the reader to the Section 4 of the user manual for a thorough discussion of each functionality with practical examples.

4.1 Evaluation Recycling

Evaluation of f on the collocation knots of a sparse grid is of course conceptually straightforward – it is simply a matter of looping through the knots of a sparse grid and calling the evaluation of f on each of them. To this end, the SGMK provides a convenience wrapper function, to which f is passed as anonymous function, see Section 4.1 of the user manual. This wrapper can take as input a list of collocation knots where the function has already been evaluated (either another sparse grid or, say, coming from a Monte Carlo sampling) and will detect which of these evaluations can be "recycled", to reduce the amount of calls to f (see user manual, Section 4.2.1). This algorithm proceeds in two different ways depending on whether the list of knots is another sparse grid or not.



(a) Number of evaluations without recycling (solid lines) and with recycling (dashed lines) (logarithmic scale on the vertical axis)

(b) Percentage of saved evaluation in case of recycling with respect to the number of evaluations without recycling

Fig. 5. Assessment of the recycling functionality: test for different values of w.

— If the list of knots is actually another sparse grid (which needs to be passed both in reduced and non-reduced format), and both sparse grids are using the same family of univariate collocation knots: the search for knots in common is speeded up by comparing essentially the multi-indices in the two grids rather than comparing the actual coordinates of the knots (i.e., comparing mostly integer numbers, which is of course faster than comparing floating point numbers). Of course, larger savings are obtained if nested collocation knots are used when generating the two grids.
— If the list of knots is unstructured (say, a simple list of knots): the same algorithm used when reducing a sparse grid is employed to compare the coordinates of the knots in the list with those in the sparse grid, cf. Section 3.

Example 5 (Recycling). In Figure 5(a), we compare the number of evaluations of f over a Smolyak grid with Clenshaw–Curtis (i.e., nested) knots for $N = 2, 4, 6$ and increasing values of the level w, with and without recycling from the previous grid (i.e., recycling the evaluations at level $w - 1$ to evaluate f over the grid at level w). The same information is shown in Figure 5(b), where we display the percentage of evaluations saved when making use of the recycling functionality, and indeed observe that in this case, the saving is considerable ($30\% - 50\%$ of the evaluations). However, note that this amount depends on the type of multi-index set, the level-to-knots function, and the type of knots used (nested/non-nested).

4.2 Parallelization

The use of a dedicated wrapper function for evaluating f also allows the evaluations of f to be performed in serial or parallel in a way that is transparent to the user (see user manual, Section 4.2.2). As soon as N_{par} or more evaluations are required, the code switches on the Matlab Parallel Toolbox, which allocates the requested evaluations of f on all available workers. The user controls only the value of N_{par}, which should be set depending on the CPU time required by a single evaluation of f. Indeed, for very fast evaluations of f the communication time between the parallel workers and the central Matlab instance might exceed the evaluation time. We conclude this paragraph by mentioning some technical aspects:

— The allocation of the evaluations of f on the workers is completely delegated to Matlab's built-in scheduler, and, in particular, this means that we are assuming that evaluating f requires the same CPU time for every value of $\mathbf{y}$ (which is not necessarily the case when f is the result of a complex PDE solver).

— The only part of the SGMK that makes explicit use of the Matlab parallel environment is the evaluation of f over the grid knots. All other operations (sparse-grid generation and reduction, other operations on sparse grid mentioned at the beginning of this section) do not have an explicit parallel implementation.
— If the adaptive algorithm is used, at each iteration several candidate indices are tested by adding them to the current sparse grid. The question that then arises is whether to consider candidate indices sequentially and then execute in parallel only on the knots requested by each index or conversely to gather all new knots requested by all candidates and parallelize them all in a single loop. We implement the first strategy, since (a) in this way we can use the fast version of the evaluation recycling (indeed, the union of all the new knots needed is not a sparse grid, whereas by testing one index at a time we can compare two sparse grids: the one with and the one without such index) and (b) it improves code modularity.

4.3 Interface with External Software by UM-Bridge Protocol

A further advantage of using a wrapper function for evaluations of f is that external software for evaluating f can be simply connected to the SGMK: to this end, it is enough to encapsulate calls to such external software in an anonymous function, and to pass the latter to the wrapper routine.

A particularly efficient way is by using the UM-Bridge software [62], which implements a standardized HTTP protocol (available in several languages, including Matlab) for calling external software to evaluate f from within outer loop software, such as UQ and optimization software. Using UM-Bridge to call external software from SGMK can take as little as 3 lines of code, see Section 4.5.2 of the user manual.

Communication via HTTP messages also allows for the interfacing of the SGMK with software running on remote servers. In particular, if these servers allow parallel requests, activating parallel execution from within Matlab as discussed above will automatically enable parallelism on the servers. More specifically, the SGMK would activate one or more Matlab workers, whose only task would be to control a corresponding processor on the remote server, to which the actual evaluation of f is delegated. Since the workload of the Matlab workers is minimal (just sending an HTTP message) one can activate a large number of workers, opening the door to large-scale UQ applications, see [63] for a naval engineering UQ application, in which the solver runs remotely in parallel on the Google Cloud Platform.

4.4 Interpolation and Quadrature

The SGMK provides an interpolation function that evaluates (11), see Section 4.1 of the user manual for details. The implementation follows closely the mathematical formulation:

— it loops through the tensor grids, creating a Lagrange interpolant of f on each tensor grid as in (6), and evaluating it at the requested knots (the standard form of Lagrange polynomials (7) is implemented in the SGMK – another possibility would be to implement their barycentric form);
— the evaluations on each tensor grid are combined with the combination technique coefficients $c_{\mathbf{i}}$.

Note that for this procedure both the extended and reduced versions of the sparse grid are needed. This is due to the fact that the information about tensor grids needed for the above-listed operations (knots, combination technique coefficients) is stored in the extended format, whereas the values of f on the sparse-grid knots are stored in compressed format, i.e., in vectors (matrices for vector-valued f) whose number of elements (columns for vector-valued f) is the number of knots without repetitions in the reduced grid. The index information stored in the reduced grid

(see Figure 3(c)) is therefore needed to be able to assign the correct values of f at each node of each tensor grid.

Similarly, the SGMK also provides an implementation for the quadrature formula in (12), see again Section 4.1. of the user manual for details. Note however that this implementation actually does not require looping through the tensor grids, therefore it does not need both the extended and reduced format, but only the reduced one. Indeed, the quadrature weights in the reduced format of the sparse grid already take into account the possible occurrences of the knots in multiple grids and the combination technique coefficients. Therefore the only operation required is a linear combination of the evaluations of f with the quadrature weights stored in the reduced grid format.

4.5 Polynomial Chaos Expansion and Sobol Indices Computation

The sparse-grid approximation of a function f is based on Lagrange interpolation polynomials, and hence is a nodal approximation. However, sometimes it is interesting to work with modal approximations instead, specifically with the gPCE [20, 71, 76], i.e., an expansion of f over multivariate ρ-orthonormal polynomials [26, 72], that gets typically truncated as

$$f(\mathbf{y}) \approx \sum_{\mathbf{p}\in\Lambda} d_{\mathbf{p}} \mathcal{P}_{\mathbf{p}}(\mathbf{y}). \tag{20}$$

Here $\Lambda \subset \mathbb{N}^N$ is a multi-index set (note that this time multi-indices can have zero entries) and $\mathcal{P}_{\mathbf{p}} = \prod_{n=1}^{N} P_{p_n}(y_n)$ are products of N univariate ρ_n-orthonormal polynomials of degree p_n. The multi-index set Λ can be prescribed either a-priori based on the regularity of f [3, 64] or adaptively [6, 11] and the coefficients $d_{\mathbf{p}}$ can be computed in several ways, for example, by quadrature [75], least squares fitting [6], or compressed sensing approaches [32].

The strategy provided by the SGMK to obtain a gPCE expansion consists of computing both the multi-index set Λ and the coefficients $d_{\mathbf{p}}$ by re-expressing the sparse-grid interpolant $\mathcal{U}_I(\mathbf{y})$ over the ρ-orthogonal basis of choice; in other words, the SGMK performs a change of basis to represent the same polynomial from a linear combination of Lagrange polynomials (the sparse-grid interpolant) to a linear combination of ρ-orthogonal polynomials (the gPCE). The algorithm that performs the conversion was introduced in [25] (see [14] for a similar approach) and proceeds in two steps:

— each tensor interpolant $\mathcal{U}_{\mathbf{i}}$ in the sparse-grid approximation (11), is converted into a linear combination of N-variate ρ-orthogonal polynomials. This requires solving the following Vandermonde-like linear system for each tensor interpolant:

$$\sum_{\substack{\mathbf{p}\in\mathbb{N}^N:\\ \mathbf{p}\le m(\mathbf{i})-\mathbf{1}}} \widetilde{d}_{\mathbf{p}} \mathcal{P}_{\mathbf{p}}(\mathbf{y}_k) = \mathcal{U}_{\mathbf{i}}(\mathbf{y}_k) \quad \forall \mathbf{y}_k \in \mathcal{T}_{\mathbf{i}}; \tag{21}$$

— if the same polynomial $\mathcal{P}_{\mathbf{p}}$ is generated by more than one tensor interpolant, the corresponding coefficient $d_{\mathbf{p}}$ in the final gPCE expansion (20) is the linear combination of the partial coefficients $\widetilde{d}_{\mathbf{p}}$ with coefficients $c_{\mathbf{i}}$ of the combination technique, see again (11).

Also in this case, we need to loop over the tensor grids, therefore both the extended and reduced format of a sparse grid are needed in the implementation. Note that the algorithm works in such a way that Λ is completely determined by the choices of the level-to-knots function and of the multi-index set of the sparse grid from which the conversion procedure begins. The condition number of the Vandermonde-like linear systems (21) depends on the choice of the families of collocation knots used to build the tensor grids. In particular, the matrix becomes orthogonal if the tensor grids are built using ρ-Gaussian collocation knots and we want to compute the gPCE expansion over

the corresponding ρ-orthogonal polynomials (for example, Gauss–Legendre knots and Legendre polynomials, Gauss–Hermite knots and Hermite polynomials).

The SGMK supports conversion to Legendre, Hermite, Laguerre, generalized Laguerre, and probabilistic Jacobi polynomials, which are, respectively, the ρ-orthogonal polynomials for uniform, normal, exponential, gamma, and beta probability density functions (all random variables for which the SGMK provides collocation knots for generation of the corresponding sparse grids), see user manual, Section 4.4. Conversion to Chebyshev polynomials is also available, since they are a valid alternative to Legendre polynomials for expanding functions with respect to the uniform measure, even though they are not orthogonal with respect to it. The evaluation of the above listed polynomials is obtained by means of the well-known three-term recurrence formulas, see [26].

An example of a situation when having the gPCE expansion of f is helpful is the computation of the Sobol indices for global sensitivity analysis of f [2, 66]. An efficient way to compute such Sobol indices is to perform some algebraic manipulations on the coefficients of the gPCE, $d_{\mathbf{p}}$, see [25, 71]; to this end, the SGMK provides a wrapper function (see Section 4.4. of the user manual) in order to perform the required algebraic manipulations. Another reason to perform the conversion to gPCE is to inspect the spectral content of the sparse-grid approximation, to verify how much the nodal representation is storing "redundant information", see, for example, [21].

5 COMPARISON WITH OTHER SOFTWARE

In this section, we provide a comparison of the SGMK with the packages for sparse grids and sparse-grids-based UQ in Table 1 that either are natively written in Matlab or provide an interface to Matlab. We thus compare the SGMK with SG++, Tasmanian and Spinterp; we choose to neglect UQLab, since it does not provide full sparse-grids functionalities (more precisely, it provides sparse-grids quadrature but not sparse-grid interpolation). We focus on a comparison in terms of functionalities rather than on computational efficiency since the typical CPU-intensive utilization scenario of this kind of software is the construction of surrogate models for UQ purposes, in which case the computational cost is largely dominated by the evaluation of the function f, and only a small fraction of cost may be ascribed to the actual sparse-grid functionalities.

The results of the comparison are reported in Table 2, which is divided into several "thematic" blocks. We begin by providing the essential software information: native Matlab/interface and whether each package is currently maintained or not. We then move to comparing the basic features of sparse grids provided by each package, as discussed in Section 2: the sparse-grid forms implemented (combination technique/hierarchical), the supported basis functions and knots and, more specifically, whether non-nested knots can be used. The third block focuses on adaptive algorithms: what dimension-adaptive is provided, whether locally-adaptivity is implemented (cf. Sections 2 and 3.1), and some connected features, most notably the dimension buffering discussed in Section 3.1. Next, we consider features connected to evaluations of f: evaluation recycling (Section 4.1), parallel evaluation (Section 4.2), and the possibility of connecting to external software to evaluate f (cf. Section 4.3). The last two blocks deal with additional features: derivatives (i.e., gradients and Hessians) and UQ functionalities.

The main take-away point of this comparison is that the four packages considered have very little overlap: they all provide Lagrange-based interpolation and gradient computation, and support evaluation recycling (which in a way can be considered the minimum requirements of a piece of software that wants to be of any practical use) and dimension-adpativity. Other than this, they all come with their own set of unique features. In general, Tasmanian and the SGMK have more UQ functionalities, whereas SG++ can be thought of as a more generalistic sparse-grids software, since it provides additional modules that implement functionalities for PDE solving and data mining. Spinterp provides an interesting implementation of the dimension-adaptive algorithm, that

Table 2. Comparative Table of Matlab Software for Sparse Grids and Sparse-grids-based Uncertainty Quantification

Feature	Tasmanian	SG++	spinterp	SGMK
Native Matlab / Interface	interface	interface	native	native
Currently maintained	yes	yes	no	yes
Comb. tec. / hierarchical	both	mainly hierarchical[♭]	hierarchical	comb. tec.
Lagrange basis	yes	yes	yes	yes
Piecewise pol. basis	yes	yes	yes	no
Splines basis	no	yes	no	no
Trigonometric basis	yes	no	no	no
Non-nested knots?	yes	no	no	yes
Dimension-adaptive	Webster–Stoyanov	Gerstner–Griebel	Gerstner–Griebel[♮]	Gerstner–Griebel
with non-nested knots	no	no	no	yes
Buffering	implicit[†]	no	no	yes
Local adaptivity	yes	yes	no	no
Parallel eval. of f	OpenMP, CUDA/Hip	OpenMP	no	Matlab Parallel Toolbox
knot recycling	yes	yes	yes	yes
Connection to external f	yes, through LibEnsamble[♠]	no[♯]	no[◇]	yes, through UM-Bridge
Gradients	yes	yes, in optimization module	yes (exact)	yes
Hessian	no	no	no	yes
Random vars. beyond uniform?	yes	no[♯]	no	yes
Computation of PCE	no	no[♯]	no	yes
Computation of Sobol idx	no	yes	no	yes

Annotations:
[♭] : limited support for combination technique provided by the combination technique module
[♮] : with blending of dimension-adaptive and Smolyak construction, and dropping of multi-indices with small profit [36]
[†] : once the anisotropy estimate in the Webster–Stoyanov algorithm is reliable
[♠] : https://libensemble.readthedocs.io
[♯] : supported through interface with Dakota [1], but not in the Matlab interface
[◇] : only through Matlab system calls.

(a) blends the Gerstner–Griebel and the Smolyak a-priori grid construction by setting a so-called balancing parameter that can range from 1 (100% of sparse-grid knots generated by the adaptive algorithm) to 0 (Smolyak construction) and (b) can drop multi-indices added to the approximation with little profit, see [36]. Tasmanian provides the largest choice of basis functions, whereas SGMK has some unique functionalities for UQ: dimension buffering for adaptivity, support for PCE and Sobol indices, as well as computation of Hessians which could be useful, for example, for Bayesian inversion tasks (see, for example, [7, 34, 51, 58, 70] for more details).

6 CONCLUSIONS

In this manuscript, we have introduced the combination technique form of the sparse-grid methodology for approximating and computing integrals of high-dimensional functions, in particular for UQ purposes. The SGMK is a Matlab package that can be used to these ends. We have discussed the data structure of the software and the mathematical aspects of the functionalities implemented in it, and we have compared it with other Matlab software for sparse grids and UQ (Spinterp, Tasmanian, SG++). Compared to alternative software, the SGMK provides the most tools for UQ, for example dimension-buffering for adaptivity, and support for PCE and Sobol indices.

ACKNOWLEDGMENTS

The authors gratefully acknowledge several people who contributed to the development of the package either by providing implementations for some functions or by using the software and reporting success cases, bugs and missing features. In particular we would like to thank: Fabio Nobile (early version of the code and continued support throughout the development of the project), Alessandra Sordi and Maria Luisa Viticchiè (early contributions to the code), Francesco Tesei and Diane Guignard (adaptive sparse grids), Giovanni Porta (Sobol indices and conversion to PCE), Björn Sprungk (adaptive sparse grids and weighted Leja knots), Francesca Bonizzoni (compatibility with Octave). Finally, we thank Miroslav Stoyanov and Dirk Pflüger for the help in crafting Table 2.

REFERENCES

[1] B. M. Adams, W. J. Bohnhoff, K. R. Dalbey, M. S. Ebeida, J. P. Eddy, M. S. Eldred, R. W. Hooper, P. D. Hough, K. T. Hu, J. D. Jakeman, M. Khalil, K. A. Maupin, J. A. Monschke, E. M. Ridgway, A. A. Rushdi, D. T. Seidl, J. A. Stephens, and J. G. Winokur. 2021. *Dakota, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.15 User's Manual.* Technical Report. Sandia National Laboratiories. DOI: https://doi.org/10.2172/1829573

[2] G. E. B. Archer, A. Saltelli, and I. M. Sobol. 1997. Sensitivity measures, ANOVA-like techniques and the use of bootstrap. *Journal of Statistical Computation and Simulation* 58, 2 (1997), 99–120. DOI: https://doi.org/10.1080/00949659708811825

[3] J. Bäck, F. Nobile, L. Tamellini, and R. Tempone. 2011. Stochastic spectral Galerkin and collocation methods for PDEs with random coefficients: A numerical comparison. In *Proceedings of the Spectral and High Order Methods for Partial Differential Equations.* Lecture Notes in Computational Science and Engineering, Vol. 76. Springer, Berlin, 43–62. DOI: https://doi.org/10.1007/978-3-642-15337-2_3

[4] M. Baudin, A. Dutfoy, B. Iooss, and A.-L. Popelin. 2017. *OpenTURNS: An Industrial Software for Uncertainty Quantification in Simulation.* Springer International Publishing, Cham, 1–38. DOI: https://doi.org/10.1007/978-3-319-11259-6_64-1

[5] J.-B. Blanchard, G. Damblin, J.-M. Martinez, G. Arnaud, and F. Gaudier. 2019. The URANIE platform: An open-source software for optimisation, meta-modelling and uncertainty analysis. *EPJ Nuclear Sci. Technol.* 5 (2019), 4. DOI: https://doi.org/10.1051/epjn/2018050

[6] B. Blatman, G. Sudret. 2011. Adaptive sparse polynomial chaos expansion based on least angle regression. *Journal of Computational Physics* 230, 6 (2011), 2345 – 2367. DOI: https://doi.org/10.1016/j.jcp.2010.12.021

[7] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler. 2013. A computational framework for infinite-dimensional bayesian inverse problems part I: The linearized case, with application to global seismic inversion. *SIAM Journal on Scientific Computing* 35, 6 (2013), A2494–A2523. DOI: https://doi.org/10.1137/12089586X

[8] H. J. Bungartz and M. Griebel. 2004. Sparse grids. *Acta Numer.* 13 (2004), 147–269. DOI: https://doi.org/10.1017/S0962492904000182

[9] D.G. Cacuci. 2003. *Sensitivity & Uncertainty Analysis, Volume 1: Theory.* Chapman and Hall/CRC. DOI: https://doi.org/10.1201/9780203498798

[10] M. Chiappetta, C. Piazzola, M. Carraturo, L. Tamellini, A. Reali, and F. Auricchio. 2023. Sparse-grids uncertainty quantification of part-scale additive manufacturing processes. *International Journal of Mechanical Sciences* 256 (2023), 108476. DOI: https://doi.org/10.1016/j.ijmecsci.2023.108476

[11] A. Chkifa, A. Cohen, R. Devore, and C. Schwab. 2013. Sparse adaptive Taylor approximation algorithms for parametric and stochastic elliptic PDEs. *ESAIM: Mathematical Modelling and Numerical Analysis* 47, 1 (2013), 253–280. DOI: https://doi.org/10.1051/m2an/2012027

[12] A. Chkifa, A. Cohen, and C. Schwab. 2014. High-dimensional adaptive sparse polynomial interpolation and applications to parametric PDEs. *Foundations of Computational Mathematics* 14, 4 (2014), 601–633. DOI: https://doi.org/10.1007/s10208-013-9154-z

[13] I. Colombo, F. Nobile, G. Porta, A. Scotti, and L. Tamellini. 2018. Uncertainty quantification of geochemical and mechanical compaction in layered sedimentary basins. *Computer Methods in Applied Mechanics and Engineering* 328 (2018), 122–146. DOI: https://doi.org/10.1016/j.cma.2017.08.049

[14] P.G. Constantine, M. S. Eldred, and E. T. Phipps. 2012. Sparse pseudospectral approximation method. *Comput. Methods Appl. Mech. Engrg.* 229/232 (2012), 1–12. DOI: https://doi.org/10.1016/j.cma.2012.03.019

[15] P. G. Constantine. 2015. *Active Subspaces: Emerging Ideas for Dimension Reduction in Parameter Studies.* Society for Industrial and Applied Mathematics, Philadelphia, PA. https://epubs.siam.org/doi/book/10.1137/1.9781611973860

[16] B. Debusschere, H. N. Najm, P. P. Pébay, O. M. Knio, R. G. Ghanem, and O. P. Le Maître. 2004. Numerical challenges in the use of polynomial chaos representations for stochastic processes. *SIAM Journal on Scientific Computing* 26, 2 (2004), 698–719. DOI: https://doi.org/10.1137/S1064827503427741

[17] B. Debusschere, K. Sargsyan, C. Safta, and K. Chowdhary. 2017. Uncertainty quantification toolkit (UQTk). In *Proceedings of the Handbook of Uncertainty Quantification.* R. Ghanem, D. Higdon, and H. Owhadi (Eds.), Springer International Publishing, Cham, 1807–1827. DOI: https://doi.org/10.1007/978-3-319-12385-1_56

[18] A. Eftekhari and S. Scheidegger. 2022. High-dimensional dynamic stochastic model representation. *SIAM Journal on Scientific Computing* 44, 3 (2022), C210–C236. DOI: https://doi.org/10.1137/21M1392231

[19] M. Eigel, O. G. Ernst, B. Sprungk, and L. Tamellini. 2022. On the convergence of adaptive stochastic collocation for elliptic partial differential equations with affine diffusion. *SIAM Journal on Numerical Analysis* 60, 2 (2022), 659–687. DOI: https://doi.org/10.1137/20M1364722

[20] O. G. Ernst, A. Mugler, H.-J. Starkloff, and E. Ullmann. 2012. On the convergence of generalized polynomial chaos expansions. *ESAIM: Mathematical Modelling and Numerical Analysis* 46, 02 (2012), 317–339. DOI: https://doi.org/10.1051/m2an/2011045

[21] O. G. Ernst, B. Sprungk, and L. Tamellini. 2018. Convergence of sparse collocation for functions of countably many gaussian random variables (with application to lognormal elliptic diffusion problems). *SIAM Journal on Numerical Analysis* 56, 2 (2018), 877–905. DOI: https://doi.org/10.1137/17M1123079

[22] J. Feinberg, V. G. Eck, and H. P. Langtangen. 2018. Multivariate polynomial chaos expansions with dependent variables. *SIAM Journal on Scientific Computing* 40, 1 (2018), 199–223. DOI: https://doi.org/10.1137/15M1020447

[23] J. Feinberg and H. P. Langtangen. 2015. Chaospy: An open source tool for designing methods of uncertainty quantification. *Journal of Computational Science* 11 (2015), 46–57. DOI: https://doi.org/10.1016/j.jocs.2015.08.008

[24] M. Feischl and A. Scaglioni. 2021. Convergence of adaptive stochastic collocation with finite elements. *Computers & Mathematics with Applications* 98 (2021), 139–156. DOI: https://doi.org/10.1016/j.camwa.2021.07.001

[25] L. Formaggia, A. Guadagnini, I. Imperiali, V. Lever, G. Porta, M. Riva, A. Scotti, and L. Tamellini. 2013. Global sensitivity analysis through polynomial chaos expansion of a basin-scale geochemical compaction model. *Computational Geosciences* 17, 1 (2013), 25–42. DOI: https://doi.org/10.1007/s10596-012-9311-5

[26] W. Gautschi. 2004. *Orthogonal Polynomials: Computation and Approximation.* Oxford University Press, Oxford. DOI: https://doi.org/10.1093/oso/9780198506720.001.0001

[27] A. Genz and B. D. Keister. 1996. Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight. *Journal of Computational and Applied Mathematics* 71, 2 (1996), 299–309. DOI: https://doi.org/10.1016/0377-0427(95)00232-4

[28] T. Gerstner and M. Griebel. 2003. Dimension-adaptive tensor-product quadrature. *Computing* 71, 1 (2003), 65–87. DOI: https://doi.org/10.1007/s00607-003-0015-5

[29] R. Ghanem, D. Higdon, and H. Owhadi. 2016. *Handbook of Uncertainty Quantification.* Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-12385-1

[30] M. Griebel, M. Schneider, and C. Zenger. 1992. A combination technique for the solution of sparse grid problems. In *Proceedings of the Iterative Methods in Linear Algebra.* P. de Groen and R. Beauwens (Eds.), IMACS, Elsevier, North Holland, 263–281.

[31] D. Guignard and F. Nobile. 2018. A posteriori error estimation for the stochastic collocation finite element method. *SIAM Journal on Numerical Analysis* 56, 5 (2018), 3121–3143. DOI: https://doi.org/10.1137/17M1155454

[32] J. Hampton and A. Doostan. 2015. Compressive sampling of polynomial chaos expansions: Convergence analysis and sampling strategies. *Journal of Computational Physics* 280 (2015), 363 – 386. DOI: https://doi.org/10.1016/j.jcp.2014.09.019

[33] J. D. Jakeman. 2023. PyApprox: A software package for sensitivity analysis, Bayesian inference, optimal experimental design, and multi-fidelity uncertainty quantification and surrogate modeling. *Environmental Modelling and Software* 170 (2023), 105825. DOI: https://doi.org/10.1016/j.envsoft.2023.105825

[34] K.-T. Kim, U. Villa, M. Parno, Y. Marzouk, O. Ghattas, and N. Petra. 2023. HIPPYlib-MUQ: A Bayesian inference software framework for integration of data with complex predictive models under uncertainty. *ACM Transactions on Mathematical Software* 49, 2 (2023), 1–31. DOI: https://doi.org/10.1145/3580278

[35] A. Klimke. 2006. *Uncertainty Modeling Using Fuzzy Arithmetic and Sparse Grids*. Ph.D. Dissertation. Universität Stuttgart, Shaker Verlag, Aachen.

[36] A. Klimke. 2008. *Sparse Grid Interpolation Toolbox User's Guide V. 5.1*. Technical Report 2007/17. Universität Suttgart.

[37] A. Klimke and B. Wohlmuth. 2005. Algorithm 847: Spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB. *ACM Transactions on Mathematical Software* 31, 4 (2005), 561579. DOI: https://doi.org/10.1145/1114268.1114275

[38] X. Ma and N. Zabaras. 2010. An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations. *Journal of Computational Physics* 229, 10 (2010), 3884–3915. DOI: https://doi.org/10.1016/j.jcp.2010.01.033

[39] S. Marelli and B. Sudret. 2014. UQLab: A framework for uncertainty quantification in matlab. In *Proceedings of the Vulnerability, Uncertainty, and Risk*. Michael Beer, Siu-Kui Au, and Jim W. Hall (Eds.), American Society of Civil Engineers, 2554–2563. DOI: https://doi.org/10.1061/9780784413609.257

[40] J. Martínez-Frutos and F. Periago. 2018. *Optimal Control of PDEs under Uncertainty: An Introduction with Application to Optimal Shape Design of Structures*. Springer International Publishing, Cham. DOI: https://doi.org/10.1007/978-3-319-98210-6

[41] Z. Morrow and M. Stoyanov. 2020. A method for dimensionally adaptive sparse trigonometric interpolation of periodic functions. *SIAM Journal on Scientific Computing* 42, 4 (2020), A2436–A2460. DOI: https://doi.org/10.1137/19M1283483

[42] A. Narayan and J. D. Jakeman. 2014. Adaptive Leja Sparse Grid Constructions for Stochastic Collocation and High-Dimensional Approximation. *SIAM Journal on Scientific Computing* 36, 6 (2014), A2952–A2983. DOI: https://doi.org/10.1137/140966368

[43] A. Narayan, Z. Liu, J. A. Bergquist, C. Charlebois, S. Rampersad, L. Rupp, D. Brooks, D. White, J. Tate, and R. S. MacLeod. 2023. UncertainSCI: Uncertainty quantification for computational models in biomedicine and bioengineering. *Computers in Biology and Medicine* 152 (2023), 106407. DOI: https://doi.org/10.1016/j.compbiomed.2022.106407

[44] R. B. Nelsen. 2006. *An Introduction to Copulas* (2nd. ed.). Springer, New York. xiv+269 pages. DOI: https://doi.org/10.1007/0-387-28678-0

[45] F. Nobile, L. Tamellini, and R. Tempone. 2016. Convergence of quasi-optimal sparse-grid approximation of Hilbert-space-valued functions: Application to random elliptic PDEs. *Numerische Mathematik* 134, 2 (2016), 343–388. DOI: https://doi.org/10.1007/s00211-015-0773-y

[46] F. Nobile, L. Tamellini, F. Tesei, and R. Tempone. 2016. An adaptive sparse grid algorithm for elliptic PDEs with lognormal diffusion coefficient. In *Proceedings of the Sparse Grids and Applications – Stuttgart 2014*. J. Garcke and D. Pflüger (Eds.), Lecture Notes in Computational Science and Engineering, Vol. 109, Springer International Publishing Switzerland, Cham, 191–220. DOI: https://doi.org/10.1007/978-3-319-28262-6_8

[47] F. Nobile, R. Tempone, and C. G. Webster. 2008. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *Journal on Numerical Analysis* 46, 5 (2008), 2411–2442. DOI: https://doi.org/10.1137/070680540

[48] F. Nobile, R. Tempone, and C. G. Webster. 2008. A sparse grid stochastic collocation method for partial differential equations with random input data. *Journal on Numerical Analysis* 46, 5 (2008), 2309–2345. DOI: https://doi.org/10.1137/060663660

[49] M. Obersteiner and H.-J. Bungartz. 2021. A generalized spatially adaptive sparse grid combination technique with dimension-wise refinement. *SIAM Journal on Scientific Computing* 43, 4 (2021), A2381–A2403. DOI: https://doi.org/10.1137/20M1325885

[50] M. Parno, A. Davis, and L. Seelinger. 2021. MUQ: The MIT uncertainty quantification library. *Journal of Open Source Software* 6, 68 (2021), 3076. DOI: https://doi.org/10.21105/joss.03076

[51] N. Petra, J. Martin, G. Stadler, and O. Ghattas. 2014. A computational framework for infinite-dimensional bayesian inverse problems, part II: Stochastic newton MCMC with application to ice sheet flow inverse problems. *SIAM Journal on Scientific Computing* 36, 4 (2014), A1525–A1555. DOI: https://doi.org/10.1137/130934805

[52] Dirk Pflüger. 2010. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München.
[53] D. Pflüger, B. Peherstorfer, and H.-J. Bungartz. 2010. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity* 26, 5 (2010), 508–522. DOI: https://doi.org/10.1016/j.jco.2010.04.001
[54] C. Piazzola and L. Tamellini. 2023. (2023). Retrieved from https://github.com/lorenzo-tamellini/sparse-grids-matlab-kit. Accessed 20 November 2023.
[55] C. Piazzola and L. Tamellini. 2023. (2023). Retrieved from https://sites.google.com/view/sparse-grids-kit. Accessed 20 November 2023.
[56] C. Piazzola and L. Tamellini. 2023. The Sparse Grids Matlab Kit user manual - v. 23-5 Robert. (2023). Retrieved from https://sites.google.com/view/sparse-grids-kit
[57] C. Piazzola, L. Tamellini, R. Pellegrini, R. Broglia, A. Serani, and M. Diez. 2022. Comparing multi-index stochastic collocation and multi-fidelity stochastic radial basis functions for forward uncertainty quantification of ship resistance. *Engineering with Computers* 39 (2022), 2209–2237. DOI: https://doi.org/10.1007/s00366-021-01588-0
[58] C. Piazzola, L. Tamellini, and R. Tempone. 2021. A note on tools for prediction under uncertainty and identifiability of SIR-like dynamical systems for epidemiology. *Mathematical Biosciences* 332 (2021), 108514. DOI: https://doi.org/10.1016/j.mbs.2020.108514
[59] A. Quarteroni, R. Sacco, and F. Saleri. 2007. *Numerical Mathematics* (2nd. ed.). Texts in Applied Mathematics, Vol. 37. Springer-Verlag, Berlin. xviii+655 pages. DOI: https://doi.org/10.1007/b98885
[60] M. F. Rehme, F. Franzelin, and D. Pflüger. 2021. B-splines on sparse grids for surrogates in uncertainty quantification. *Reliability Engineering and System Safety* 209 (2021), 107430. DOI: https://doi.org/10.1016/j.ress.2021.107430
[61] C. Schillings and C. Schwab. 2013. Sparse, adaptive Smolyak quadratures for Bayesian inverse problems. *Inverse Problems* 29, 6 (2013), 065011. DOI: https://doi.org/10.1088/0266-5611/29/6/065011
[62] L. Seelinger, V. Cheng-Seelinger, A. Davis, M. Parno, and A. Reinarz. 2023. UM-bridge: Uncertainty quantification and modeling bridge. *Journal of Open Source Software* 8, 83 (2023), 4748. DOI: https://doi.org/10.21105/joss.04748
[63] L. Seelinger, A. Reinarz, J. Benezech, M. B. Lykkegaard, L. Tamellini, and R. Scheichl. 2023. Lowering the entry bar to HPC-scale uncertainty quantification. arXiv:2304.14087. Retrieved from https://arxiv.org/abs/2304.14087
[64] J. Shen and L.-L. Wang. 2010. Sparse spectral approximations of high-dimensional problems based on hyperbolic cross. *SIAM Journal on Numerical Analysis* 48, 3 (2010), 1087–1109. DOI: https://doi.org/10.1137/090765547
[65] R. C. Smith. 2013. *Uncertainty Quantification: Theory, Implementation, and Applications*. Society for Industrial and Applied Mathematics. DOI: https://doi.org/10.1137/1.9781611973228
[66] I. M. Sobol. 2001. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation* 55, 1-3 (2001), 271 280. DOI: https://doi.org/10.1016/S0378-4754(00)00270-6
[67] M. Stoyanov. 2015. *User Manual: TASMANIAN Sparse Grids*. Technical Report ORNL/TM-2015/596. Oak Ridge National Laboratory, One Bethel Valley Road, Oak Ridge, TN.
[68] M. Stoyanov. 2018. Adaptive sparse grid construction in a context of local anisotropy and multiple hierarchical parents. In *Proceedings of the Sparse Grids and Applications - Miami 2016*. J. Garcke, D. Pflüger, C. G. Webster, and G. Zhang (Eds.), Springer International Publishing, 175–199. DOI: https://doi.org/10.1007/978-3-319-75426-0_8
[69] M. Stoyanov and C. G Webster. 2016. A dynamically adaptive sparse grids method for quasi-optimal interpolation of multidimensional functions. *Computers and Mathematics with Applications* 71, 11 (2016), 2449–2465. DOI: https://doi.org/10.1016/j.camwa.2015.12.045
[70] A. M. Stuart. 2010. Inverse problems: A Bayesian perspective. *Acta Numerica* 19, 5 (2010), 451–559. DOI: https://doi.org/10.1017/S0962492910000061
[71] B. Sudret. 2008. Global sensitivity analysis using polynomial chaos expansions. *Reliability Engineering and System Safety* 93, 7 (2008), 964 – 979. DOI: https://doi.org/10.1016/j.ress.2007.04.002
[72] G. Szegö. 1939. *Orthogonal Polynomials*. American Mathematical Society. DOI: https://doi.org/10.1090/coll/023
[73] L. N. Trefethen. 2008. Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Review* 50, 1 (2008), 67–87. DOI: https://doi.org/10.1137/060659831
[74] G. W. Wasilkowski and H. Wozniakowski. 1995. Explicit cost bounds of algorithms for multivariate tensor product problems. *Journal of Complexity* 11, 1 (1995), 1–56. DOI: https://doi.org/10.1006/jcom.1995.1001
[75] D. Xiu. 2007. Efficient collocational approach for parametric uncertainty analysis. *Communications in Computational Physics* 2, 2 (2007), 293–309.
[76] D. Xiu and G.E. Karniadakis. 2002. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing* 24, 2 (2002), 619–644.

Received 18 March 2022; revised 9 October 2023; accepted 15 September 2023