DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Using the Spatially Adaptive Combination Technique for Efficient Quantification of Uncertainty in Hydrological Models

Markus Englberger

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Using the Spatially Adaptive Combination Technique for Efficient Quantification of Uncertainty in Hydrological Models

Verwendung der räumlich-adaptiven Kombinationstechnik zur effizienten Quantifizierung von Unsicherheiten in hydrologischen Modellen

Author:	Markus Englberger
Supervisor:	Prof. Dr. rer. nat. habil. Hans-Joachim Bungartz
Advisor:	M.Sc. (hons) Ivana Jovanovic Buha, Dr. rer. nat. Michael Obersteiner
Submission Date:	15.03.2022

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.03.2022 Markus Englberger

Abstract

Constructing polynomial chaos expansions in the context of uncertainty quantification requires the computation of inner products that are typically high dimensional integrals involving an expensive to evaluate model and a set of orthogonal polynomials. This thesis explores the approach of first interpolating the model and then either performing pseudospectral projection (PSP) on this surrogate or, by exploiting the simpler structure of the interpolant, deriving analytical formulas for the inner products. To keep the number of model evaluations low, several variants of the sparse grid combination technique are proposed for interpolation, in particular the single dimension spatially adaptive refinement strategy. Comparisons for several test functions and the hydrological model HBV, with the number of model evaluations as reference, show that this approach often produces better results than direct PSP.

Contents

1 Introduction

In uncertainty quantification (UQ), one considers a model that has, besides deterministic input parameters, uncertain input parameters with known independent distributions. The model propagates these uncertainties to the output domain. Statistical properties of this output distribution, like expectation or variance are often of interest. Additionally, one wants to quantify the impact that single or combinations of input parameters have on the output. Such sensitivity coefficients can be specified with so-called *Sobol' indices*. These properties of the output distribution can be obtained by constructing a general polynomial chaos expansion (gPCE), which is a linear combination of polynomials orthogonal in regard to the joint density functions of the uncertain input distributions. Expectation, variance, and Sobol' indices can be easily obtained having the gPCE coefficients. Constructing such a polynomial approximation of the model function involves computing inner products which are, depending on the number of input distributions, high dimensional integrals over the model function and the orthogonal polynomials, weighted with the density function. Computing high dimensional integrals is computationally challenging. Additionally, the model is often expensive to evaluate and might show local phenomena such as discontinuities, requiring adaptive methods. The idea examined in this thesis is to first interpolate the model and then compute the inner products on the interpolant. To keep the number of model evaluations low, different variants of the sparse grid combination technique can be applied for interpolation, in particular the single dimension spatially adaptive refinement method. One way to compute the gPCE coefficients of the surrogate is *pseudospectral projection* (PSP), applying numerical quadrature to obtain the inner products. The quadrature rule is chosen in accordance with the gPCE truncation and, to avoid internal aliasing errors, by integrating all combinations of polynomials existing in the truncation scheme exactly. Besides performing PSP on the interpolant, a second way to compute the gPCE coefficients is to derive analytical formulas for the inner products by exploiting the simpler structure of the surrogate.

The thesis first summarizes several interpolation and numerical quadrature methods in chapter 2 and describes how these methods can be applied in higher dimensions in an efficient way using sparse grids. To cope with local phenomena, a spatially adaptive combination technique with dimension-wise refinement is presented. Chapter 3 summarizes relevant aspects of uncertainty quantification and the general polynomial

chaos expansion. The approach of first interpolating the model function using the combination technique, and in particular the single-dimension spatially adaptive sparse grid refinement strategy, and then obtaining the gPCE coefficients either with PSP or analytical formulas, is explored. In order to test out theses different variants, chapter 4 presents aspects of the libraries SparseSpACE, which implements many combination technique algorithms, and chaospy, a toolbox supporting uncertainty quantification using polynomial chaos expansions. Finally, numerical results for different test functions and the hydrological model HBV are compared in chapter 5.

2 Numerical Quadrature and Interpolation

This chapter covers, considering a function $f : X \to \mathbb{R}$ on the domain $X \subset \mathbb{R}^d, d \in \mathbb{N}$, the problems of approximating f via interpolation and of approximating the integral $\int_X f(x)dx$. Numerical quadrature is necessary if the integral cannot be solved analytically. For several quadrature rules, f is approximated with an interpolant $\tilde{f}$ for which the integral $\int_X \tilde{f}(x)dx$ can be computed analytically, i.e.

$$\int_X \tilde{f}(x) \approx \int_X f(x)dx,$$

connecting the problems of numerical quadrature and interpolation.

The first section of this chapter describes several interpolation and quadrature rules in a one-dimensional setting. These rules are generalized for higher dimensions in the second section. To mitigate the *curse of dimensionality* arising from full grids in higher dimensions, sparse grids are introduced. Furthermore, adaptive sparse grids and in particular the single dimension spatially adaptive refinement strategy [15] are presented.

2.1 Single dimension

In the one-dimensional case $d = 1$, we consider a function $f : [a, b] \to \mathbb{R},\ a, b \in \mathbb{R}$. Most numerical quadrature and interpolation methods require f to be sufficiently smooth. Otherwise, if f has non-smooth regions or even discontinuities, adaptive grids can be a remedy. For the following quadrature methods, the approximate integral is of the form $\sum_{i=1}^{n} w_i f(x_i)$ where w_i are the quadrature weights and $(x_i)_{i \leq n}$ are the grid points. The weights are the integral of basis functions $(\Phi_i)_{i \leq n}$. The linear combination of these basis functions with coefficients $(f(x_i))_{i \leq n}$ is the corresponding interpolant, i.e.

$$\int_a^b f(x)dx \approx \int_a^b \tilde{f}(x)dx = \int_a^b \sum_{i=1}^{n} f(x_i)\Phi_i(x)dx = \sum_{i=1}^{n} f(x_i) \int_a^b \Phi_i(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

2.1.1 Piecewise linear interpolation and quadrature

A simple quadrature rule is the trapezoidal rule. The approximate integral is computed as

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b}{2}$$

which is the result of linearly interpolating f on the boundary points a and b and then computing the exact integral of the interpolant.

Instead of computing only one trapezoid, we can split up the domain over n intervals of length $h := \frac{b-a}{n}$ and then sum up all these approximate integrals:

$$\int_a^b f(x)dx = h \cdot \sum_{k=0}^{n} f(a+k\cdot h) + f(a+(k+1)\cdot h).$$

This so-called trapezoidal sum is the result of first piecewise linearly interpolating f on the grid points $\{a, a+\frac{b-a}{n}, a+2\cdot\frac{b-a}{n}, ..., b\}$ and then integrating this piecewise linear function.

In order to introduce spatial adaptivity and later sparse grids in section 2.2.3, we have a closer look at piecewise linear interpolation [1]. We first assume that f lives on the domain $[0,1]$ and evaluates to zero on the boundaries. The basis functions

$$\varphi_{l,i}(x) := \varphi(2^l x - i), i \in \mathcal{I}_l,$$

where

$$\varphi(x) := max(1-|x|, 0)$$

is the *standard hat function* and l the discretization level, are placed equidistantly on the domain, formalised by the index set $\mathcal{I}_l := \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1\}$. Scaling all basis functions with $(f(x_i))_{i\in\mathcal{I}_l}$ gives the piecewise linear interpolant

$$\tilde{f} = \sum_{i\in\mathcal{I}_l} f(x_i)\cdot\varphi_{l,i}.$$

The function space that is spanned by the basis functions of level l, called *nodal basis*, is denoted as

$$V_l = span\{\varphi_{l,i} : i \in I_l\}.$$

In order to perform adaptivity and to introduce sparse grids, it is necessary to hierarchize the nodal basis. Instead of only having hat functions of the same support corresponding to the discretization level, the hierarchical set of basis functions keeps the basis functions of previous levels and adds new hat functions only at new grid points. Assuming some smoothness condition, the surpluses are getting smaller for

higher levels and are therefore useful as error estimates. Formalizing this hierarchical approach, hat functions added at level l form the hierarchical subspace

$$W_l = span\{\varphi_{l,i} : i \in I_l\}$$

with hierarchical index set

$$\mathcal{I}_l := \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1, i \text{ odd}\}.$$

Summing up all these subspaces up a to discretization level L, gives again the function space

$$V_L = \bigoplus_{l \leq L} W_l.$$

The interpolant is

$$\tilde{f} = \sum_{l \leq L} \sum_{i \in \mathcal{I}_l} \alpha_{l,i} \cdot \varphi_{l,i}(x)$$

with surpluses $\{\alpha_{l,i}, l \leq L, i \in \mathcal{I}_l\}$. Since both the nodal set and the hierarchical basis set span the same function space for a given discretization level, the surpluses can be translated in both directions via (de)hierarchization. If f is not zero on the boundaries, one may either add hat functions on the two boundary points or modify the basis functions such that those hat functions adjacent to the boundary extrapolate. Figure 2.1 shows the one-dimensional piecewise linear hierarchical basis functions, both with standard as well as modified hat functions.

2.1.2 B-Splines

Instead of using hat functions leading to a piecewise linear interpolant, other basis functions may result in a smoother interpolant. One such family of different basis functions are B-Splines [19], using piecewise polynomials of arbitrary degree as basis functions.

2.1.3 Leja sequence

Another idea is to interpolate the function with one global polynomial. For any set of $n+1$ interpolation points, there always exists a unique polynomial of minimal degree, but maximal n, that interpolates the function in those points. However, using an equidistant grid leads to an ill-conditioned interpolant for higher polynomial degrees. A set of points that has better approximation properties is the Leja-sequence $(\theta_i)_{j \in \mathbb{N}}$ [13], which is recursively defined as

$$\theta_i = \underset{\theta \in X}{\operatorname{argmax}} \prod_{j=0}^{i-1} |\theta - \theta_j|.$$

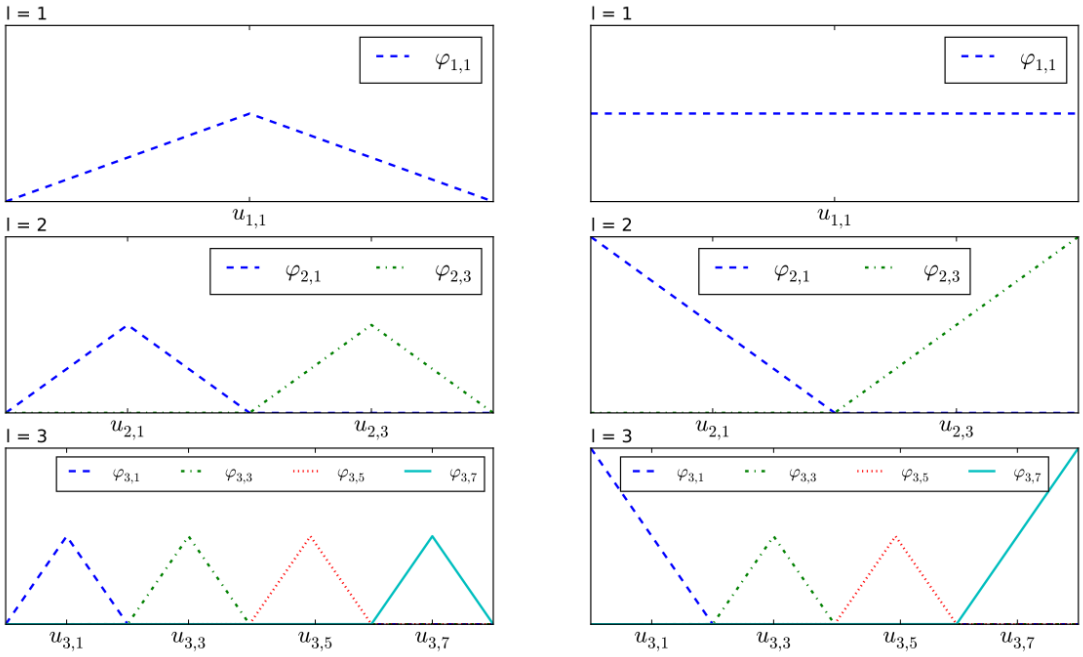


Figure 2.1: The standard hierarchical hat functions and the modified version, up to discretization level 3, taken from [3]

As the Leja-points form a sequence, the grid points are nested, i.e. the set of Leja points of level l contains all Leja points of levels $l' \leq l$, a property which is useful for adaptivity and sparse grids, introduced in section 2.2.3. Generalizing to weighted integration, i.e. if the integrand is multiplied with a weight function w, which is $w = 1$ in the unweighted case, the definition includes this weight function:

$$\theta_i = \underset{\theta \in X}{\operatorname{argmax}} \prod_{j=0}^{i-1} w(\theta)|\theta - \theta_i|$$

2.1.4 Clenshaw-Curtis quadrature

In a similar fashion to Leja, Clenshaw-Curtis quadrature [2] is also the exact integration of an interpolated polynomial. Here, the interpolation points are roots of Chebyshev polynomials. Again, using $n+1$ points integrates polynomials of maximal degree n exactly. Additionally, Clenshaw-Curtis grids are nested.

2.1.5 Gaussian quadrature

The quadrature rule which integrates the highest degree of polynomials exactly is the Gaussian quadrature. Using n quadrature points, this method allows to integrate polynomials of degree $2n-1$ exactly. For the derivation, we consider a polynomial p of degree $2n-1, n \in \mathbb{N}$ on the interval $[-1,1]$, which can be generalized by scaling. The integrand p can be decomposed into

$$p(x) = q(x)L_n(x) + r(x),$$

where $q(x)$ and $r(x)$ are polynomials of degree $n-1$ and L_n is the n-th Legendre polynomial. The Legendre polynomials are a set of orthonormal polynomials over the interval $[-1,1]$, i.e. $\langle L_i, L_j \rangle = \delta_{ij}$. With this decomposition the integral becomes

$$\int_{-1}^{1} p(x)dx = \int_{-1}^{1} q(x)L_n(x)dx + \int_{-1}^{1} r(x)dx.$$

As the n-th Legendre polynomial L_n is orthogonal to any linear combination of lower degree Legendre polynomials, in particular to $q(x)$, the integral simplifies to

$$\int_{-1}^{1} p(x)dx = \int_{-1}^{1} r(x)dx.$$

To obtain a quadrature rule $\sum_{i=0}^{n-1} w_i f(x_i)$ which correctly computes $\int_{-1}^{1} q(x)L_n(x)dx = 0$, the quadrature nodes are chosen to be the roots of L_n. The weights on the other hand are chosen such that r is integrated exactly, which is possible as r is of degree $n-1$ and the n weights can be chosen freely. Differently to Leja and Clenshaw-Curtis grids, the Gaussian quadrature points are not nested.

2.2 Higher dimensions

This section explores where the grid points for numerical quadrature or interpolation should be placed in higher dimensions. Although one can approximately integrate a function in arbitrary dimension via tensor products of one-dimensional interpolation/ quadrature rules, the number of points in such a full tensor grid grows exponentially with the dimension. Therefore, methods that reduce the number of points without much loss of accuracy are needed. After a description of full tensor grids, this chapter introduces sparse grids and the related combination technique, a combination of full grids resulting in a sparse grid, which mitigate the *curse of dimensionality*. A spatially adaptive version of the combination technique is presented.

2.2.1 Generalization of one-dimensional interpolation/ quadrature rules to an arbitrary dimension

We can generalize the seen one-dimensional quadrature and interpolation rules to an arbitrary dimension d by taking their tensor products. Let $\mathcal{L}^i$ describe the continuous linear operator of dimension i, then the tensor product operator

$$\mathcal{L}^{\vec{d}} := \mathcal{L}^1 \oplus ... \oplus \mathcal{L}^d$$

is the corresponding $d-$dimensional operator. For numerical quadrature and interpolation problems, these exact operators are not available, so we use one-dimensional approximate operators $\mathcal{L}_l^i$ that converge to the true operator $\mathcal{L}^i$, i.e. $\lim_{l\to\infty} ||\mathcal{L}^i - \mathcal{L}_l^i|| = 0$ which holds true for all of the presented interpolation/ quadrature rules. The resulting tensor product approximation converges to $\mathcal{L}^{\vec{d}}$ for $\vec{l} \to \infty$.

To get a more concrete result, let the *exact set* of an approximate operator $\mathcal{L}_l$ be defined as

$$\mathcal{E}(\mathcal{L}_l) := \{f : \mathcal{L}(f) = \mathcal{L}_l(f)\},$$

i.e. the set of functions that are integrated exactly by $\mathcal{L}_l$. It can be shown that the tensor approximation $\mathcal{L}_{\vec{l}}^{\vec{d}} = \mathcal{L}_{l_1}^1 \oplus ... \oplus \mathcal{L}_{l_d}^d$ integrates all functions in the tensor product of the one-dimensional exact sets:

$$\mathcal{E}(\mathcal{L}_{l_1}^1) \oplus ... \oplus \mathcal{E}(\mathcal{L}_{l_d}^d) \subseteq \mathcal{E}_{\vec{l}}^{\vec{d}}(\mathcal{L}_{\vec{l}}^{\vec{d}}).$$

Therefore, one-dimensional quadrature rules can be easily generalized to higher dimensions. [12]

2.2.2 Full grids

Although d-dimensional quadrature rules can be obtained by taking the full tensor grid of one-dimensional quadrature rules, these full grids are problematic in higher dimensions as the number of points grows exponentially with the dimension.

We consider the example of a function $f : \Omega = [0,1]^d \to \mathbb{R}$ that lives on the d-dimensional unit cube and evaluates to zero on the boundary on which piecewise linear interpolation is performed. For this purpose, we extend the nodal basis to an arbitrary dimension. The basis functions are constructed via tensor products of the one-dimensional nodal basis

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j)$$

for a levelvector $\vec{l} = (l_1, ..., l_d) \in \mathbb{N}^d$ and multi-index $\vec{i} = (i_1, ..., i_d) \in \mathbb{N}^d$. The set of indices in the nodal basis is $\mathcal{I}_{\vec{l}} := \{\vec{i} : 1 \leq i_j \leq 2^{l_j}, 1 \leq j \leq d\}$. Applying the same discretization level L in every dimension, i.e. $l_1 = l_2 = ... = L$, the grid has $N := 2^{ld}$ grid points. This exponential growth with dimensionality, the so-called *curse of dimensionality*, makes full grid interpolation/quadrature infeasible in high dimensions. The accuracy for full grid piecewise linear interpolation can be shown to be in $\mathcal{O}(N^{-2})$, assuming mixed bounded derivatives up to order two. [1]

2.2.3 Sparse grids

In order to mitigate the curse of dimensionality, the idea introduced by Smolyak [17] is to hierarchize the $d-$dimensional tensor product and then leave out subspaces that represent high coupling between dimensions.

We again consider the tensor product of one-dimensional operators $\mathcal{L}^{\vec{d}}_{\vec{l}} = \mathcal{L}^1_{l_1} \oplus ... \oplus \mathcal{L}^d_{l_d}$. Writing the operator of dimension i as the series $\mathcal{L}^i = \sum_{l=0}^{\infty}(\mathcal{L}^i_l - \mathcal{L}^i_{l-1})$ where $\mathcal{L}^i_{-1} := 0$ and defining $\triangle^i_0 := \mathcal{L}^i_0, \triangle^i_l := \mathcal{L}^i_l - \mathcal{L}^i_{l-1}$ gives

$$\mathcal{L}^i = \sum_{l=0}^{\infty} \triangle^i_l.$$

The tensor product of the operators is then written as

$$\mathcal{L}^{\vec{d}} = \mathcal{L}^1 \oplus ... \oplus \mathcal{L}^d = \sum_{|\vec{l}|=1}^{\infty} \triangle^1_{l_1} \oplus ... \oplus \triangle^d_{l_d}$$

As the exact quadrature/ interpolation operators are not available, the series needs to be truncated. One way to truncate the series is the L_∞-norm, i.e. only including

multi-indices $\{\vec{l} \in \mathbb{N}^d : |\vec{l}|_\infty \leq L\}$ which leads to a full grid with truncation at L. The idea of leaving out subspaces representing high coupling between dimensions is realized if instead the L_1-norm is used, resulting in the indices

$$\mathcal{K} := \{\vec{l} \in \mathbb{N}^d : |\vec{l}|_1 \leq L + d - 1\}$$

where L is the maximum level in any dimension. The truncated d-dimensional operator becomes

$$\mathcal{L}_{\mathcal{K}}^{\vec{d}} = \sum_{\vec{l} \in \mathcal{K}} \triangle_{l_1}^1 \oplus ... \oplus \triangle_{l_d}^d \tag{2.1}$$

An important property, when implementing the the formula eq. (2.1), is nestedness, fulfilled for the trapezoidal sum, Leja or Clenshaw-Curtis grids. Let G_n^i be the set of grid points belonging to $\mathcal{L}_n^i$. If the grid points are nested, i.e. $G_{n-1}^i \subset G_n^i$, evaluating $\triangle_n^i$ only requires the points $G_n^i \backslash G_{n-1}^i$.

Benefits of sparse grids can be seen for piecewise linear interpolation. Unlike for the nodal basis, for sparse grids hierarchization is necessary. The basis functions are again constructed via tensor products of the one-dimensional hierarchical basis functions

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j)$$

with multi-indices $\vec{l} = (l_1, ..., l_d) \in \mathbb{N}^d$ and $\vec{i} = (i_1, ..., i_d) \in \mathbb{N}^d$. The set of hierarchical indices at a given level $\vec{l}$ is defined as

$$I_{\vec{l}} := \{\vec{i} : 1 \leq i_j \leq 2^{l_j}, i_j \text{ odd}, 1 \leq j \leq d\}.$$

Again, we obtain subspaces $W_{\vec{l}}$ spanned by all basis functions at a set level

$$W_{\vec{l}} = span\{\varphi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}.$$

Applying the Smolyak method, summing over all subspaces where $|\vec{l}|_1 \leq L + d - 1$ results in the sparse grid function space

$$V_L^1 = \bigoplus_{|\vec{l}|_1 \leq L+d-1} W_{\vec{l}}$$

with discretization level L. The interpolant $\tilde{f}$ is written as

$$\tilde{f}(\vec{x}) = \sum_{|\vec{l}|_1 \leq L+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x}).$$

Denoting the number of grid points in one dimension as $N := 2^L$, the number of points reduces to $\mathcal{O}(N \cdot \log_2(N)^{d-1})$, and the accuracy considering the L_∞-norm and L_2-norm only declines slightly to $\mathcal{O}(N^{-2} \cdot (\log N)^{d-1})$, if again bounded mixed second derivatives are assumed. Figure 2.2 shows the two- and three-dimensional sparse grids of discretization level 5 with equidistant subgrids. [1]

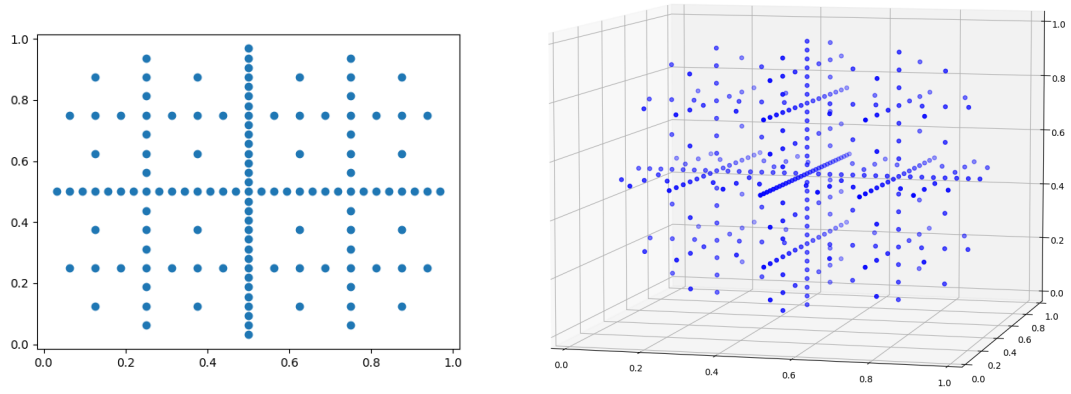


Figure 2.2: the two- and three-dimensional sparse grids of discretization level 5 with equidistant subgrids

2.2.4 Combination Technique

Direct implementations of sparse grids often require complicated datastructures and existing solvers may only exist for full grids in many applications. Full grid implementations can still be used for sparse grids, as it is possible to decompose sparse grids into a combination of full grids [7]. For this purpose, one can show that eq. (2.1) can be written as

$$\mathcal{L}_L^{\vec{d}} = \sum_{\vec{l} \in \mathcal{K}} c_{\vec{l}} \mathcal{L}_{l_1}^1 \oplus ... \oplus \mathcal{L}_{l_d}^d \tag{2.2}$$

with grid coefficients $c_{\vec{l}}$ determined as

$$c_{\vec{l}} = \sum_{\vec{z}=0}^{(1,...,1)} (-1)^{|\vec{z}|_1} \cdot \chi^{\mathcal{K}}(\vec{l} + \vec{z})$$

with

$$\chi^{\mathcal{K}}(\vec{l} + \vec{z}) = \begin{cases} 1 & \text{if } \vec{l} + \vec{z} \in \mathcal{K} \\ 0 & \text{otherwise} \end{cases}.$$

That means that the Smolyak grid can be achieved by a linear combination of full grids. Although many grid points appear on different grids making the approach more computationally expensive than direct sparse grids, caching avoids reevaluating the same points. The combination technique (CT) allows parallelization as the result for different subgrids can be computed independently. So far, the index set $\mathcal{K}$ has been either defined including all multi-indices with L_∞-norm smaller than some level L

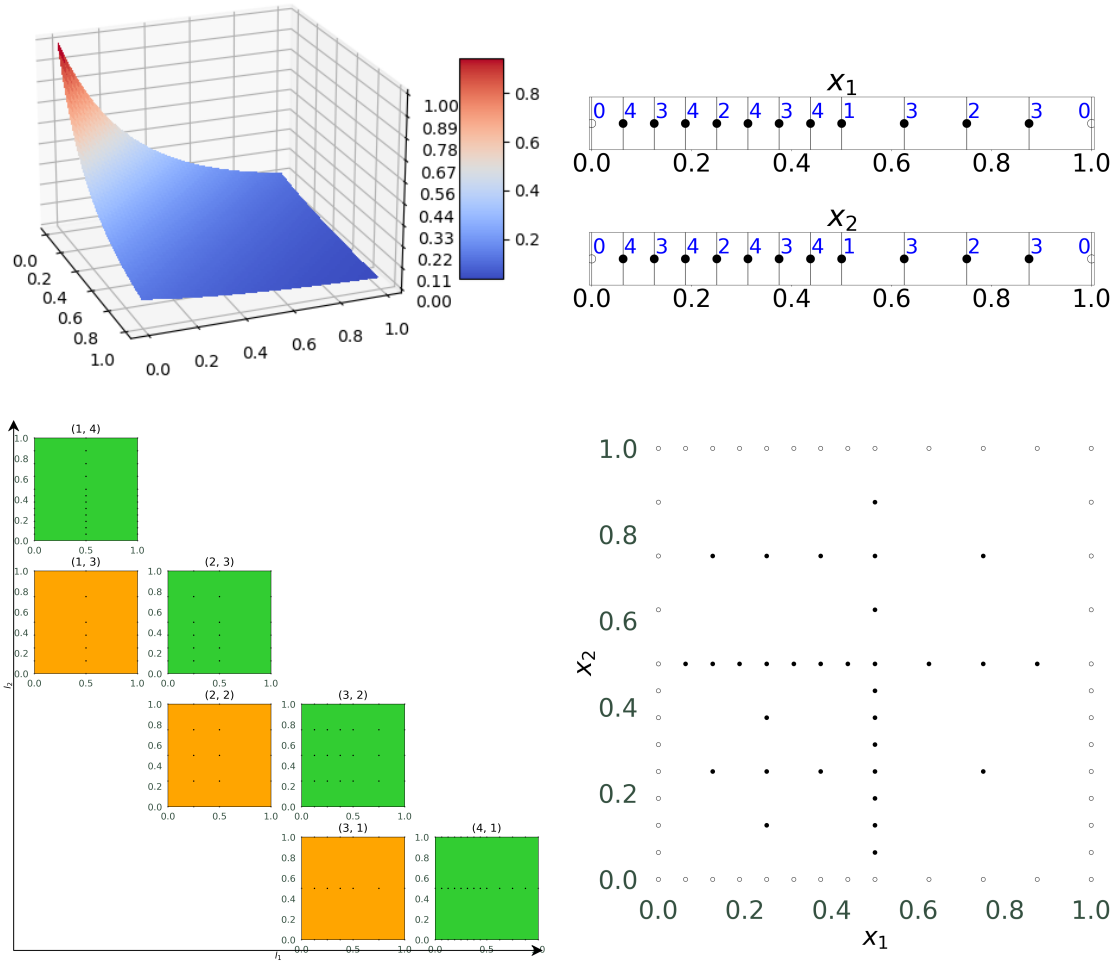


Figure 2.3: interpolation of the *Genz corner peak* function with the dimension wise refinement strategy, showing the points sets P^1 and P^2, the resulting combination scheme and the corresponding sparse grid

resulting in a full tensor grid, or instead by using the L_1-norm resulting in the Smolyak approach. However, the index set may be chosen arbitrarily as long as the following admissibility condition holds:

$$\vec{l} - \vec{e}_i \in \mathcal{K}, \text{for } 1 \leq i \leq d, l_i > 1, \forall \vec{l} \in \mathcal{K} \tag{2.3}$$

with $\vec{e}_i$ being the unit vector in direction i.

2.2.5 Adaptivity

Typical problem functions may be complicated only in some dimension or interaction between dimensions. The flexibility of eq. (2.3) allows to adjust the index set $\mathcal{K}$ accordingly. Since these properties are usually not known beforehand, adjusting the set of levelvectors $\mathcal{K}$ to the problem function may be done in an adaptive fashion, so called *dimension adaptivity*, using an appropriate error estimator.

For functions exhibiting local phenomena, grids that place more points in these regions, so called *spatially adaptive* grids [16], are more beneficial. While spatial adaptivity can be performed in a straightforward way for direct sparse grid implementations with a hierarchical set of basis functions such as the introduced hierarchical set of hat functions by adding children points of nodes that have high surpluses, it is more difficult for the combination technique. This is because spatially refined sparse grids may usually not be decomposed into an admissible set of full grids with equidistant nodes. The following section presents one way spatial adaptivity can be achieved with the combination technique.

2.2.6 Dimension-wise spatial refinement with the sparse grid combination technique

In [15], an algorithm to perform spatial adaptivity with the combination technique using a set of basis functions that can be hierarchized was introduced. It creates a valid combination scheme with full grids that are, unlike for the standard combination technique, not equidistant. The algorithms keeps a set of points for every dimension, out of which a valid combination scheme can be constructed, and refines these one-dimensional point sets by means of a reasonable error estimator.

The strategy to construct a valid combination scheme out of the one-dimensional sets of points $\{P^i, i = 1, ..., d\}$ has to determine for some level vector $\vec{l}$ which of the points are included in the full tensor subgrid $\vec{l}$. Any point P^i_j in a set P^i can be attributed to some grid of level L^i_j and their hierarchical parent also has to be included in P^i. Therefore, the point sets can be represented by a binary tree. Since only points in P^i can

be included that have a level $L_j^i \leq l_i$, the strategy that includes the maximum amount of points is the following choice of points for dimension i:

$$P^{i,\vec{l}} = \{P_j^i \in P^i | L_j^i \leq l_i\}$$

As this strategy includes points as early as possible with zero delay, the resulting sparse grid may show full-grid like structures. A second strategy, going in the other direction of delaying the inclusion points to higher levels, is the following strategy:

$$P^{i,\vec{l}} = \{P_j^i \in P^i | L_j^i \leq l_i - c_j^i\}$$

where $c_j^i = l_i^{max} - D_j^i$ and D_j^i is the the maximum of the levels of the hierarchical descendants of P_j^i in the hierarchical tree corresponding to P^j. This strategy delays the inclusion of points in coarse regions of the one-dimensional strips, i.e. points which do not have high-level descendants, by using a delay of c_j^i for point P_j^i. Although this strategy avoids full grid structures in coarse regions, the high delay may get rid of too many points missing relevant interactions between dimensions. The final strategy actually used in the algorithm is more complicated but is a trade-off between the two discussed strategies, i.e. between a delay of zero and of c_j^i.

To perform refinement, an error estimator has to be selected to choose the points in the point sets $\{P^i, i = 1, ..., d\}$ added to the combination scheme. To maintain a valid combination scheme, refinement candidates have to be children of points already existing in the set, so-called *leaf points*. Error estimates are calculated for every grid $\vec{l}$ in the combination scheme. The error estimation loops through the corresponding strips $P^{i,\vec{l}}$ and calculates error estimation via hierarchization of the one-dimensional strip while fixating the coordinates of all other dimension and adding these errors up for every combination of coordinates of the other dimensions present in the subgrid. If the global error is bigger then a chosen threshold, the grid adds children of leaf points whose error is above a threshold depending on the maximum error *max-error* of all leaf nodes, i.e. all leaf nodes whose error is larger than $\gamma \cdot$ *max-error* are refined. The choice of $\gamma \in (0, 1)$ determines how many nodes are refined in every iteration, the closer to zero the broader the refinement.

The algorithm rebalances the hierarchical trees belonging to the point sets in every iteration. Without rebalancing, the hierarchical trees could become very unbalanced if during the refinement points gather densely around some region. As it is necessary for valid combination schemes that all parent nodes exist in the combination scheme, unbalanced hierarchical trees would enforce points to be included that may otherwise not be necessary.

For more details on the algorithm, refer to [15]. Figure 2.3 shows the example of

interpolating the *Genz corner peak* [6] function with the single dimension refinement strategy.

3 Forward Uncertainty Quantification with the Adaptive Combination Technique

In forward uncertainty quantification, one considers a model $f : \vec{\Omega} \to \mathbb{R}$, where $\vec{\Omega}$ is a d-dimensional probability space with density function w. For the purpose of this thesis, we assume the ideal case of the one-dimensional distributions being known and independent. To keep the notation simple, this chapter omits possible deterministic inputs. By propagating the model, statistical properties of the output distribution are gained, such as expectation, variance, and Sobol' indices which measure the impact that single or combinations of input parameters have on the output distribution. These properties of the output distribution can be obtained by constructing a general polynomial chaos expansion (gPCE), which is a linear combination of polynomials, orthogonal in regard to the joint density functions of the uncertain input distributions.

This chapter first explains general polynomials chaos expansions and how expectation, variance and Sobol' indices are determined from the gPCE coefficients. Then, the approach of interpolating the model and constructing the gPCE on the interpolant is introduced, discussing the application of several variants of adaptive and non-adaptive sparse grids.

3.1 Polynomial chaos expansion (gPCE)

The goal of constructing a gPCE is to obtain a surrogate that may be cheaper to evaluate than the model and out of whose coefficients statistical properties can be easily extracted [20]. First we consider the one-dimensional case where the model $f : \Omega \to \mathbb{R}$ has only one random variable as input. Under the assumption that the model is square-integrable, f lies in the separable Hilbert space $\mathcal{H} := L^2(X, w)$ with inner products as the weighted integral $\langle f, g \rangle = \int_X fgwdx$ where X is the support of Ω. It is possible to construct an orthonormal set $\{\phi_j(x) : j \in \mathbb{N}_0\}$ of polynomials that is dense in $\mathcal{H}$ where ϕ_j is of degree j. For example, the Legendre polynomials represent such a set for the uniform distribution on the domain $[-1, 1]$. Due to the density of this

set of functions in $\mathcal{H}$, the model can be written as the series

$$f = \sum_{j=0}^{\infty} f_j \phi_j. \tag{3.1}$$

with inner products $f_j := \langle f, \phi_j \rangle$ Since only finitely many coefficients can be computed, we truncate the series introducing a projection operator

$$P_n(f) = \sum_{j=0}^{n} \langle f(x), \phi_j \rangle \phi_j(x) = \sum_{i=0}^{n} f_j \phi_j(x), \tag{3.2}$$

i.e. P_n projects f orthogonaly onto the subspace containing all polynomials of degree n or less. The error regarding the L_2-norm of the actual function and the projection at degree n is $||f - P_n^{(i)}||_2^2 = \sum_{n+1}^{\infty} f_j^2 < \infty$, since $f \in \mathcal{H}$.

Extending the gPCE method to higher dimensions, we consider a model that has several uncertain input parameters, i.e. $f : \vec{\Omega} \to \mathbb{R}$ where $\vec{\Omega} = (\Omega_1, ..., \Omega_d)^T$ is a random vector consisting of independent random variables, and $H^1, ..., H^d$ are the single Hilbert spaces with dense sets of functions $\{\{\phi_j^{(1)}, j \in \mathbb{N}\}, ..., \{\phi_j^{(d)}, j \in \mathbb{N}\}\}$. The Hilbert space $\mathcal{H}^{\vec{d}} := \mathcal{H}^{(1)} \oplus ... \oplus \mathcal{H}^d$ corresponds to the d-dimensional stochastic input space. For simplicity, we assume having the same polynomial truncation n for all dimensions. The according d-dimensional orthogonal polynomials are then $\phi_{\vec{j}}(\vec{x}) := \prod_{i=1}^{d} \phi_{j_i}(x_i)$. Following the *Smolyak* approach in the d-dimensional truncation scheme, we may want to disregard contributions of highly coupled inner products. For that purpose, instead of taking the full tensor product that arises by using the L_∞-norm, we apply the L_1-norm leading to the d-dimensional projection operator

$$P_n^{\vec{d}}(f) = \sum_{\vec{j} \in \mathcal{K}} \langle f, \phi_{\vec{j}} \rangle \phi_{\vec{j}}$$

with multi-indices

$$\mathcal{K} = \{\vec{j} : |j|_1 \leq n + d - 1\} \tag{3.3}$$

3.2 Mean and variance

Besides having a surrogate that is cheaper to evaluate than the original model, constructing the gPCE function also has the advantage that statistical properties of the output distribution, like mean and variance, can simply be calculated having the gPCE coefficients. For simplicity, we again consider a model f with only one stochastic parameter and the expansion P_n for a maximal polynomial degree n. The orthonormality

of the basis functions and ϕ_0 being a constant implies $\phi_0 = 1$:

$$1 = \langle \phi_0, \phi_0 \rangle = \int_X \phi_0^2 w dx = \phi_0^2.$$

This allows to approximate the expectation

$$\mathbb{E}[f] \approx \mathbb{E}\left[\sum_{j=0}^{n} f_j \phi_j\right] = \sum_{j=0}^{n} f_j \mathbb{E}[\phi_j] = \sum_{j=0}^{n} f_j \mathbb{E}[\phi_0 \phi_j] = \sum_{j=0}^{n} f_j \int_X \phi_0 \phi_j w dx = \sum_{j=0}^{n} f_j \langle \phi_0, \phi_j \rangle = f_0$$

and variance

$$Var[f] = \mathbb{E}\left[(f - \mathbb{E}[f])^2\right] \approx \mathbb{E}\left[\left(\left(\sum_{j=0}^{n} f_j \phi_j\right) - f_0\right)^2\right] = \mathbb{E}\left[\left(\sum_{j=1}^{n} f_j \phi_j\right)^2\right] =$$
$$\sum_{j_1=1}^{n}\sum_{j_2=1}^{n} f_{j_1} f_{j_2} \mathbb{E}[\phi_{j_1}\phi_{j_2}] = \sum_{j_1=1}^{n}\sum_{j_2=1}^{n} f_{j_1} f_{j_2} \langle \phi_{j_1}\phi_{j_2} \rangle = \sum_{j=1}^{n} f_j^2$$

3.3 Global sensitivity analysis

It is often of interest to quantify the impact that single or combinations of input parameters have on the output distribution, called *global sensitivity analysis*. To simplify the notation, we assume in the following that the independent input random variables $\Omega_1, ..., \Omega_d$ are identically distributed. It is possible to decompose the model in the so-called *Sobol' decomposition* [18]:

$$f(x_1, ..., x_n) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{1 \le i < j \le n} f_{ij}(x_i, x_j) + ... f_{1,...,d}(x_1, ..., x_d) \tag{3.4}$$

where the summands are defined recursively:

$$f_i(x_i) = \int_{X^{d-1}} f(\vec{x}) d\vec{x}_{\sim i} - f_0,$$
$$f_{ij}(x_i, x_j) = \int_{X^{d-2}} f(\vec{x}) d\vec{x}_{\sim ij} - f_i(x_i) - f_j(x_j) - f_0$$
$$...$$

with $\int_{X^{d-|I|}} (\cdot) dx_{\sim I}$ denoting the integration over all variables except those in the index set I. This construction results in the following property:

$$\int_{X_k} f_{i_1,...,i_s}(x_1, ..., x_s) dx_k = 0, 1 \le k \le s. \tag{3.5}$$

By eq. (3.5), it is possible to write the variance as the sum

$$Var[f] = \int_{X^d} f^2(\vec{x})w(\vec{x})d\vec{x} - f_0^2 = \sum_{i=1}^{n} D_i + \sum_{1\leq i\leq j\leq n} D_{ij} + ... + D_{1,...,n} \tag{3.6}$$

with *partial variances*

$$D_{i_1,...,i_s} = \int_{X^s} f^2_{i_1...i_s}(x_{i_1},...,x_{i_s})w_1 \cdot ... \cdot w_s dx_1,...,dx_s, 1 \leq i_1 < ..., i_s \leq n.$$

The first order Sobol indices, measuring the contribution of an input alone, are the normalized partial variances

$$S_i := \frac{D_i}{Var[f]}.$$

To measure the total effect that a single input variable has on the uncertainty of the random variable $f(\vec{\Omega})$, considering both its individual contribution and interaction with other variables, the *total Sobol' indices* are defined as:

$$S_i^T(t) := \sum_{i\in\{i_1,...,i_s\}} S_{i_1,...,i_s}(t)$$

3.4 Pseudospectral projection

The method of performing numerical quadrature to obtain the inner products $\langle f, \phi_{\vec{i}}\rangle$, which are usually not possible to calculate analytically, is called *pseudospectral projection*.

In [12], for a given projection level n of an expansion $\mathcal{P}_n$, an appropriate quadrature level $q(n)$ has been derived:

To simplify notation, let $d = 1$ in the following. The error of PSP compared to an exact projection is

$$||P_n(f) - \sum_{j=0}^{n} \tilde{f}_j\phi_j(x)||_2^2 = \sum_{j=0}^{n}(f_j - \tilde{f}_j)^2 \tag{3.7}$$

where $\tilde{f}_j$ is the approximate inner product

$$f_j = \langle f, \phi_j\rangle \approx \tilde{f}_j := Q_{q(n)}(f\phi_j) = \sum_{k=0}^{n} f_k Q_{q(n)}(\phi_j\phi_k) + \sum_{k=n+1}^{\infty} f_k Q_{q(n)}(\phi_j\phi_k) \tag{3.8}$$

with Q_n as the operator for weighted quadrature on level m. Inserting 3.8 into 3.7 gives the error estimation

$$\sum_{j=0}^{n}(f_j - \tilde{f}_j)^2 = \sum_{k=0}^{n}\left(f_j - \sum_{k=0}^{n} f_k Q_{q(n)}(\phi_j\phi_k) - \sum_{k=n+1}^{\infty} f_k Q_{q(m)}(\phi_j\phi_k)\right)^2 \tag{3.9}$$

The first two terms on in the parentheses on the right of 3.9 are responsible for *internal aliasing* error and the third term for the *external aliasing* error. The external aliasing error goes to zero as the polynomial truncation level increases, the internal aliasing error however can be constant regarding the function f. Therefore, internal aliasing should be avoided, requiring the quadrature level $q(n)$ to be chosen, if possible, such that the inner product of polynomials existing in the truncation scheme can be computed exactly. This result connects the two necessary choices of a suitable Smolyak grid for quadrature and for the orthogonal projection.

3.5 Computing the gPCE coefficients with the adaptive combination technique

This section now explores how the (adaptive) combination technique can be applied when computing the gPCE coefficients. We can differentiate between three approaches. The first one is to apply the single dimension refinement strategy with the trapezoidal sum for every inner product, i.e. to perform PSP directly on the model. A second approach is to first use an adaptive or non-adaptive sparse grid to interpolate the model and then use a separate quadrature rule to compute the inner products now with the cheaper to evaluate sparse grid surrogate instead of the original model, i.e. to perform PSP not on the model but on the interpolant. A third related approach is to again interpolate the model and then, by exploiting the simpler structure of the interpolant, derive analytical formulas for the inner products. The following subsections explore these three variants:

3.5.1 Spatial refinement for every inner product

The spatially refinement strategy may be used to calculate the inner products $\langle f, \phi_i \rangle$. In [9], such an approach has been examined. The method computes all integrals with one sparse grid that adapts to all integrands $f \cdot \phi_i$. It uses the trapezoidal sum with the single dimension refinement and incorporates the density function by applying weighted integration. In this method, the choice of the polynomial truncation and the number of grid points are independent, because internal aliasing errors are inevitable as exact integration of the orthogonal gPCE polynomials is not possible using trapezoidal quadrature.

3.5.2 PSP on an interpolated surrogate

An approach that avoids internal aliasing errors but still supports adaptive sparse grids is to use a sparse grid variant for interpolation and then perform pseudospectral

projection on this interpolant instead of the model. The quadrature rule that is used for the calculation of the inner products may be chosen such that inner products of the polynomials existing in the truncation are integrated exactly to avoid internal aliasing errors, for example via Gaussian quadrature. As the surrogate is typically cheaper to evaluate than the model, the quadrature rule may be more generous with the number of evaluations compared to computing the inner products directly on the model. Unlike the adaptive grid of section 3.5.1 which adapts to all integrands $f \cdot \phi_i$, here the grid only adapts to the model f, which may be beneficial as the first gPCE coefficient $\langle f, \phi_0 \rangle$ is in general the most important one.

As there exist different sparse grid variants in regard to the underlying interpolation rules and other implementation details and also different quadrature rules to compute the inner products, this method has many variants. For once, one may chose hat functions as basis functions and then use either the non-adaptive standard combination technique or the spatially adaptive combination technique for interpolation. In higher dimensions, points on the boundary may get too expensive and instead, modified basis functions that extrapolate to the boundaries should be used. One may also exchange the hat functions with smoother B-Splines. If Lagrange polynomials are used as basis functions, the surrogate is a polynomial for which all maximal combined and single degrees are known. Therefore, all inner products where one argument is a polynomial of a degree that is not in the Smolyak grid of the interpolation, the inner product is zero. This allows to choose the quadrature rule with the according gPCE truncation such that only non-zero inner products are included. We can use Leja points for the Lagrange polynomials, which have good interpolation properties and are nested. However, Leja interpolation does not allow for spatial adaptivity.

3.5.3 Analytical formulas for interpolants

Interpolating the model also allows, for certain interpolation rules, to derive analytical formulas for computing the inner products by exploiting the simpler structure of the surrogate. In the work of [3] such a formula has been derived for sparse grids: The n-th gPCE coefficient c_n may be computed as

$$c_n = \sum_{\vec{l} \in \mathcal{K}, \vec{i} \in \mathcal{I}_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \prod_{j=1}^{d} \int_0^1 \phi_n(F^{-1}(x_j)) \varphi_{l_j,i_j} dx_j \tag{3.10}$$

where $F^{-1}(x_j)$ is the inverse cumulative distribution function (cdf) of the i-th parameter and the $\alpha_{\vec{l},\vec{i}}$ are the hierarchical surpluses, however not of the original model, but the model transformed into the d-dimensional unit cube via the inverse cdfs, i.e. the function f^{nonlin} to be interpolated is $f^{nonlin}(\vec{x}) = f(F^{-1}(x_1), ..., F^{-1}(x_d))$. Such a

nonlinear transformation may be beneficial for interpolation, as the grid points are selected in accordance with the weight function. For a more detailed derivation, refer to [3]. For the function tested in this thesis, however, only uniform distribution are considered, therefore the transformation is only a linear scaling of the model into the unit cube. For uniform distributions, using piecewise linear hat functions, the integrand appearing in the one-dimensional integrals in 3.10 are piecewise polynomials, because the inverse cdfs are linear. Therefore, these integrals can be analytically integrated.

4 Implementation

This chapter presents the libraries chaospy and sparseSpACE and their application in implementing the proposed gPCE methods of section 3.5, i.e. to first interpolate the model and then either perform PSP on that interpolant or compute the gPCE coefficients analytically. For the comparisons in this thesis, sparseSpACE serves for interpolation and then, in the case of PSP, chaospy is used for computing the gPCE coefficients.

4.1 sparseSpACE

The library sparseSpACE [14], which is an acronym for *Sparse Grid Spatially Adaptive Combination Environment*, implements several variants of the combination technique, in particular several adaptive versions. Besides dimension adaptivity, the library implements several spatially adaptive techniques, amongst which the single dimension adaptive scheme has shown compelling results, documented in [15]. Since the single dimension refinement strategy requires the basis functions to be hierarchizable, the trapezoidal sum or B-Splines are possible, however not Leja interpolation/ quadrature. For the spatially adaptive technique, the user also has to specify the refinement threshold γ, see 3.5, and a *maximum_level* determining the initial grid on the basis of which refinement is performed, a tolerance and a maximum number of evaluations. If either the error estimate is below the tolerance or the grid contains more points than the maximum specified, the refinement stops.

Additionally, sparseSpACE provides functionality to apply the combination technique in application areas such as density estimations and uncertainty quantification. The UQ method that already existed in sparseSpACE is to construct a gPCE surrogate with PSP by applying the single dimension adaptive scheme as quadrature rule.

4.2 Chaospy

Chaospy [4] is a toolbox for uncertainty quantification using polynomial chaos expansions. Relevant for this thesis, it supports PSP using different quadrature methods both with full and sparse grids. In order to construct a gPCE, the user specifies the

independent input distributions, such as uniform or normal, a quadrature method and the polynomial truncation. Possible quadrature methods include Gaussian, Clenhsaw-Curtis or Leja grids. Given a polynomial truncation n, chaospy includes all orthogonal polynomials in the scheme eq. (3.3). The statistical properties of expectation, variance and Sobol' indices can then be extracted from the gPCE surrogate.

4.3 Implementation of proposed UQ methods

This section explains the implementation of the gPCE methods proposed in 3.5.

Interpolation is performed with sparseSpACE. Interpolation methods used are the spatially adaptive CT with hat functions and the non-adaptive standard combination technique with either hat functions or Leja interpolation. For this purpose, Leja interpolation with Lagrange polynomials was added to sparseSpACE.

Considering the approach of section 3.5.2, PSP is performed on the interpolant with full Gaussian grids.

Analytical computation the inner products on an interpolant has been implemented for the basis set of hat functions for both the standard CT and the spatially adaptive CT and also supports modified basis functions. For this purpose the formula to compute the n-th gPCE coefficient on a sparse grid

$$c_n = \sum_{\vec{l}\in\mathcal{K},\vec{i}\in\mathcal{I}_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \prod_{j=1}^{d} \int_0^1 \phi_n(F^{-1}(x_j))\varphi_{l_j,i_j} dx_j,$$

compare eq. (3.10), has to be implemented for the (adaptive) combination technique. Algorithm 1 shows the pseudocode. To obtain the n-th gPCE coefficient, the algorithm loops through all subgrids that are included in the combination scheme. For a given subgrid with levelvector $\vec{l}$, the algorithm goes through all points $\vec{i} \in \mathcal{I}_{\vec{l}}$ and every dimension $j \in \{1, ..., d\}$ to compute the one-dimensional integrals $\int_0^1 \phi_n(F^{-1}(x_j))\varphi_{l_j,i_j}(x_j)dx_j$. To compute these integrals analytically, the integrand which is a piecewise polynomial for uniform distributions is split into these polynomials. The integrand evaluates to zero outside the support of φ_{l_j,i_j}. For the standard CT, the support only depends on the levelvector, whereas for the spatially adaptive CT the support can be determined by finding the adjacent points in the one-dimensional coordinate stripe corresponding to the subgrid and dimension. The integrand can then be split in two halves on the point's coordinate in the respective dimension. If a modified basis is used, it has to be checked whether the point is adjacent to the boundary and extrapolation is required. As the subgrids are full grids with no hierarchization, the surplus is just the model function evaluated at the given point, i.e. $\alpha_{\vec{l},\vec{i}} = f^{nonlinear}(x_{\vec{l},\vec{i}})$. This way, eq. (3.10) is

computed for every subgrid. The linear combination of the results via the component grid coefficients gives the sparse grid solution.

Algorithm 1 Pseudocode for computing the gPCE coefficients of a piecewise linear interpolant, adaptive or non-adaptive

procedure COMPUTE GPCE_COEFFICIENTS(combination_technique, truncation_scheme, $f^{nonlinear}$)
    gPCE_coeffs ← []
    for `poly in truncation_scheme` **do**
        for `component_grid in combination_technique` **do** ▷ solve for every included subgrid
            integral_component_grid ← 0
            for `point in component_grid` **do**
                integral_point ← 1
                for `d in 0,...,` $f^{nonlinear}$`.dimension` **do** ▷ compute the one-dimensional integrals
                    integral_point ← integral_point · $\int_0^1 \varphi_{\text{componentgrid.levelvector[d], point[d]}}(x_d)$ ·poly[d]$(F_d^{-1}(x_d))dx_d$
                end for
                integral_component_grid ← integral_component_grid + integral point · $f^{nonlinear}$(point)
            end for
            gPCE_coeffs[poly] ← gPCE_coeffs[poly] + component_grid_coefficient · integral_component_grid
        end for
    end for
end procedure

For those methods included in the comparisons in the next chapter that do not first interpolate the model but perform PSP directly on the model, implementation already exists in sparseSpACE or chaospy. The included PSP quadrature methods are weighted trapezoidal sum with the spatially adaptive CT, implemented in sparseSpACE, and full Gaussian and sparse Leja quadrature, implemented in chaospy.

Results for all of these methods will be compared in the following chapter.

5 Results

After introducing several gPCE methods in section 3.5 and presenting the implementation of these methods in chapter 4, this chapter compares test results for the proposed methods for several 3-dimensional test functions and the hydrological model HBV with 6 stochastic parameters. The test functions are chosen to represent a variety of properties: non-differentiability, discontinuities, localized problems, and smooth functions in C^∞ are included.

5.1 3-dimensional test functions

The first three methods used to compare results on the 3-dimensional test functions perform PSP directly on the model with the following quadrature rules:

- full Gaussian, using chaospy
- sparse Leja, using chaospy
- the single dimension refinement strategy with the weighted trapezoidal rule, using sparseSpACE

Since the Gaussian quadrature points are not nested and in practice the quadrature may be done in an adaptive fashion, two lines are plotted: one counting only points in the current grid and another one that includes the number of points from previous grids. For both Gaussian and Leja quadrature, the gPCE truncation can be chosen depending on the quadrature order, such that the truncation is maximal while still avoiding internal aliasing errors. The maximum truncation is at degree 10. For the weighted trapezoidal sum on the other hand, internal aliasing errors cannot be avoided and therefore the gPCE truncation is fixed at a degree of 10.

The approach of first interpolating the model and then performing PSP on the interpolant is compared with three different interpolation methods:

- the standard CT with Leja interpolation
- the standard CT with hat functions
- the spatially adaptive CT with hat functions

The chaos expansion is also truncated at degree 10. To avoid internal aliasing errors, a Gaussian grid with 11 points per dimension is employed. Therefore, the interpolant has to be evaluated 11^3 times, additional to the model evaluations during interpolation whereas for direct only the model has to be evaluated. For all figures in this chapter, only the number of model evaluations are considered which can be justified as for actual hydrological models, the model is usually more expensive to evaluate as the interpolant. Furthermore, in high dimensions the PSP quadrature method should also use sparse grids to reduce the number of interpolant evaluations.

The idea of first interpolating the model and then computing the gPCE coefficients analytically is compared with two variants:

- standard CT with hat functions
- the spatially adaptive CT with hat functions.

The last method included in the comparisons does not construct a chaos expansion but computes the mean and variance by computing the integrals $\mathbb{E}(\Omega) = \int_X f w dx$ and $Var[\Omega] = \int_X f^2 w dx - \left(\int_X f w dx\right)$ applying the single dimension refinement strategy with the weighted trapezoidal rule.

Table 5.1 summarizes the methods included in the comparisons.

name	interpolation method	quadrature method	gPCE truncation
A1	/	Gaussian	adaptive, maximum 10
A2	/	sparse Leja	adaptive, maximum 10
A3	/	weighted trapezoidal	10
B1	piecewise linear, standard CT	Gaussian	10
B2	piecewise linear, spatially adaptive CT	Gaussian	10
B3	sparse Leja-Lagrange, standard CT	Gaussian	10
C1	piecewise linear, standard CT	analytical	10
C2	piecewise linear, spatially adaptive CT	analytical	10
D	/	weighted trapezoidal	no gPCE

All sparseSpACE grids are chosen to include points on the boundary. Therefore, the basis functions adjacent to the boundaries do not have to be modified. For all spatially adaptive grids, the initial maximum level is set to 3 and the refinement threshold is set to $\gamma = 0.8$.

The plots in this section show the relative error of expectation, variance, and, provided that analytical solutions exist, also Sobol' indices, depending on the number of model evaluations.

5.1.1 Ishigami

The *Ishigami* function [10]

$$f(x,y,z) = \sin(x) + 7\sin^2(y) + 0.1 \cdot z^4 \sin(x)$$

has three identical stochastic inputs that are uniformly distributed: $X, Y, Z \sim U(-\pi, \pi)$. Analytical solutions exist for mean, variance, and Sobol' indices. The results, see fig. 5.1 and fig. 5.2, do not show a clear winner. For the expectation, the spatially adaptive methods perform better Gaussian, Leja, and the standard CT. However, for the variance and the Sobol indices', it seems beneficial to compute the inner products directly on the model, either with Gaussian or Leja grids. Gaussian or Leja quadrature may perform well, because the Ishigami function is in C^∞ which makes it a good candidate for approximation with a polynomial. In general, one can expect worse results when first interpolating the model with the adaptive or non-adaptive trapezoidal rule and then performing PSP on the model, compared to computing the inner products analytically for the same interpolation. PSP with the spatially adaptive trapezoidal rule as quadrature method performs worse for the inner products, possibly because internal aliasing errors are not avoided.

5.1.2 G-function

The second test function is the three-dimensional G-function [11]

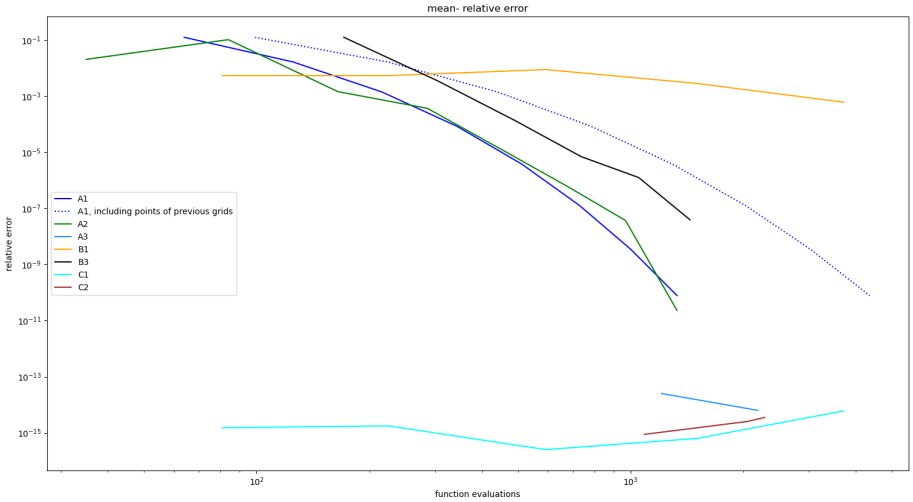
$$f(\vec{x}) = \Pi_{i=0}^{d-1} \frac{|4x_i - 2| + 0.5 \cdot i}{1 + 0.5 \cdot i}, d = 3$$

with identical distributions $X_i \sim U(0,1), i = 0,1,2$. Analytical solutions are available for expectation, variance, and Sobol' indices. The function is continuous but is not differentiable everywhere. Figure 5.3 plots the 2-dimensional G-function. Results, see fig. 5.13, show that analytically computing the inner products on an interpolant constructed via adaptive or non-adaptive CT with hat functions performs best. The reason is probably that the function is piecewise linear and the piecewise linear interpolant is the exact model. Therefore the remaining error is only due to the gPCE truncation. The inaccuracies for methods that use Gaussian or Leja grids may be attributed to the non-differentiability.

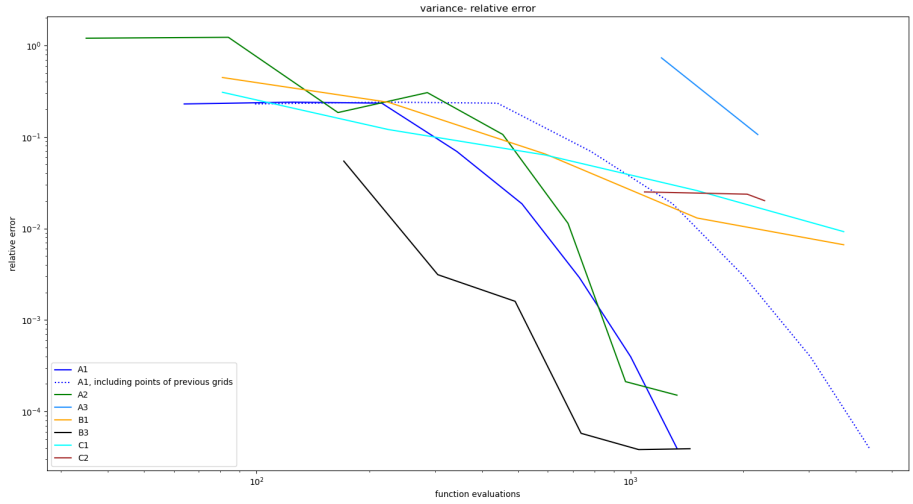
5.1.3 UQ-function

The third test function is the UQ-function [5]

$$f(x,y,z) = e^{-x^2 + 2\text{sign}(y)} + z$$

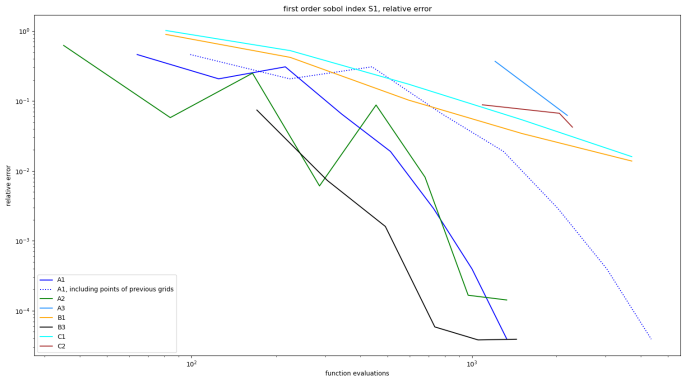


(a) mean *Ishigami* function, relative error

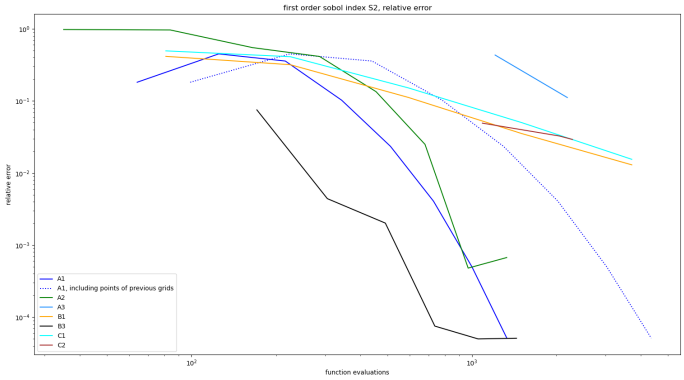


(b) variance *Ishigami* function, relative error

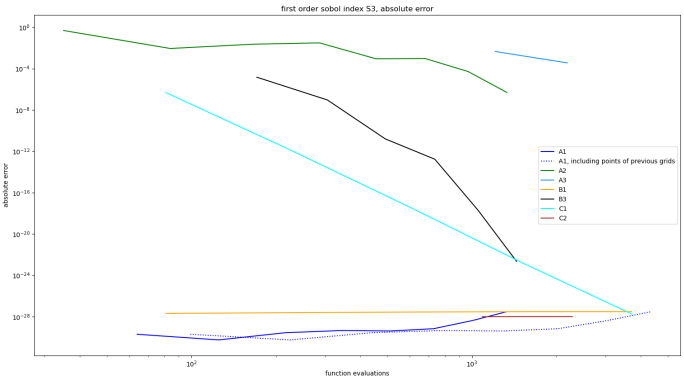
Figure 5.1: *Ishigami* function, relative error of mean and variance



(a) first order Sobol index for parameter x, relative error



(b) first order Sobol index for parameter y, relative error



(c) first order Sobol index for parameter z, relative error

Figure 5.2: *ishigami* function, relative error of first order Sobol indices

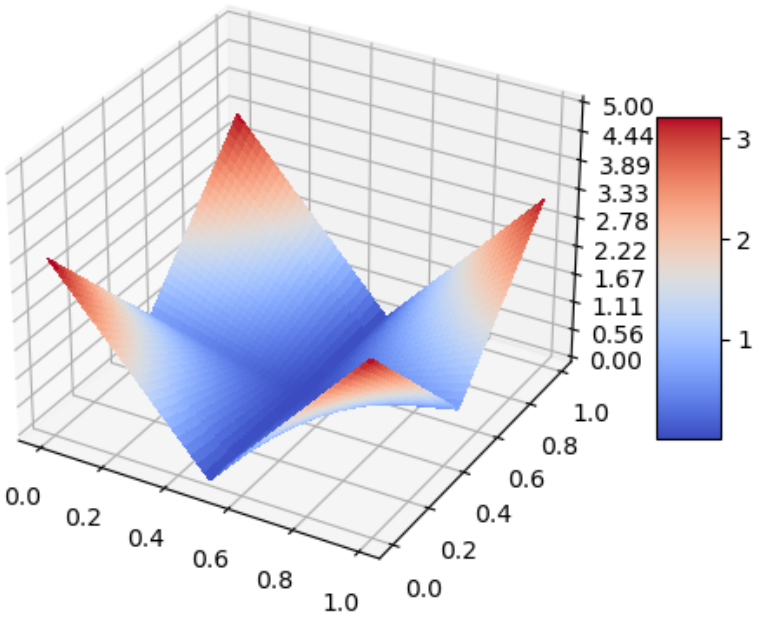
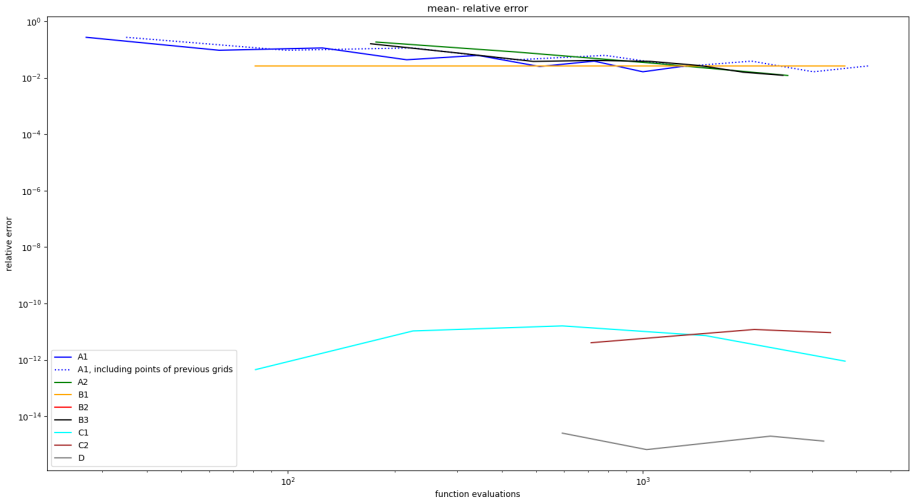
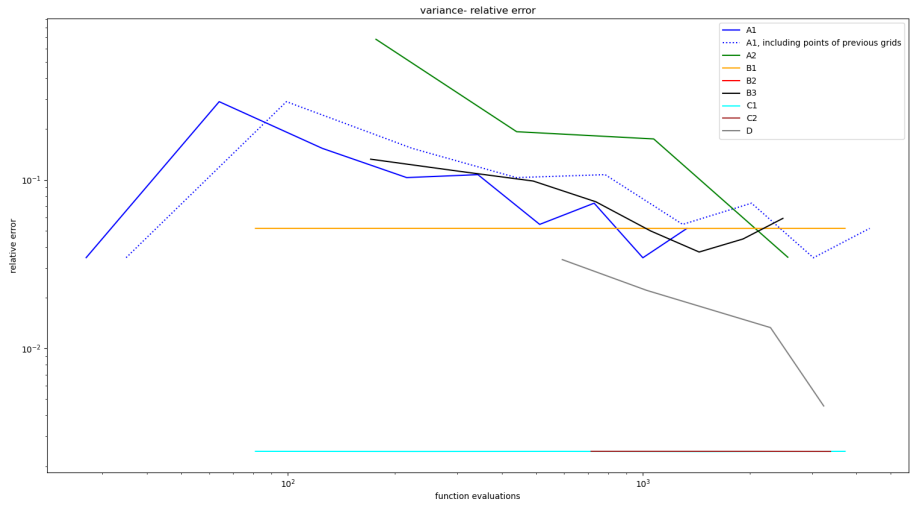


Figure 5.3: plot of the two-dimensional *GFunction*

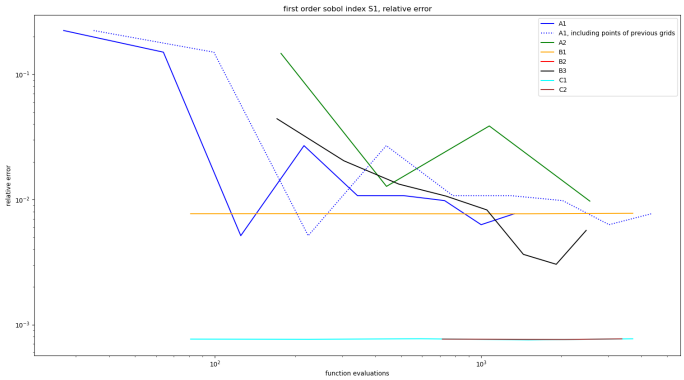


(a) mean *GFunction* function, relative error

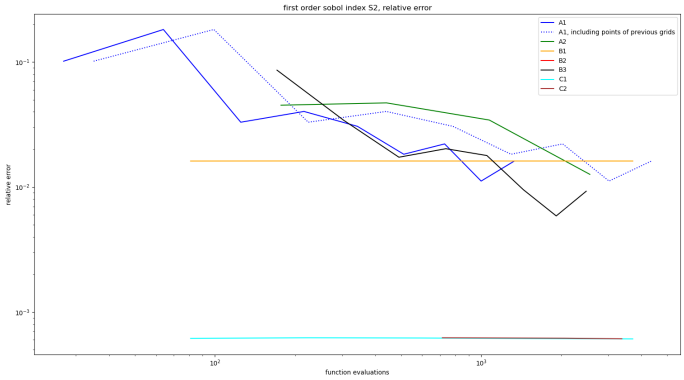


(b) variance *GFunction* function, relative error

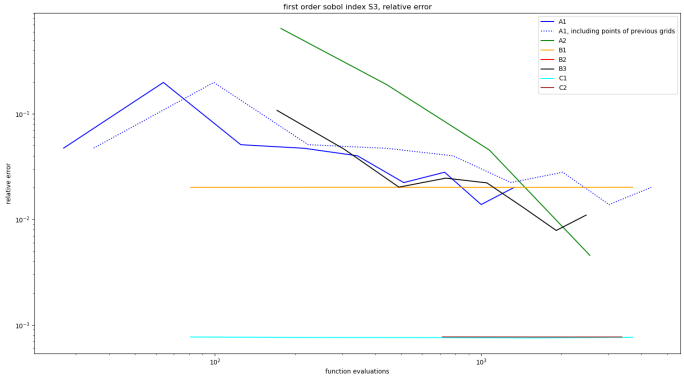
Figure 5.4: 3 dimensional *GFunction*, relative error of mean and variance



(a) first order Sobol index for parameter x_1, relative error



(b) first order Sobol index for parameter x_2, relative error



(c) first order Sobol index for parameter x_3, relative error

Figure 5.5: 3 dimensional *GFunction*, relative error of Sobol indices

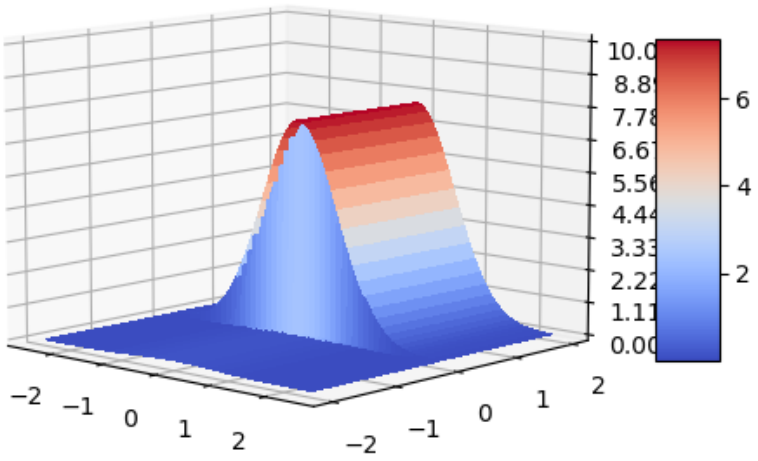
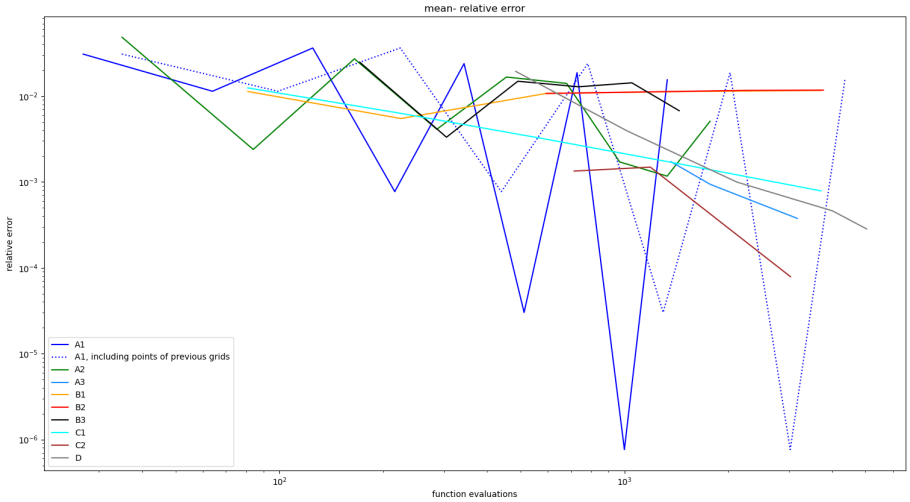
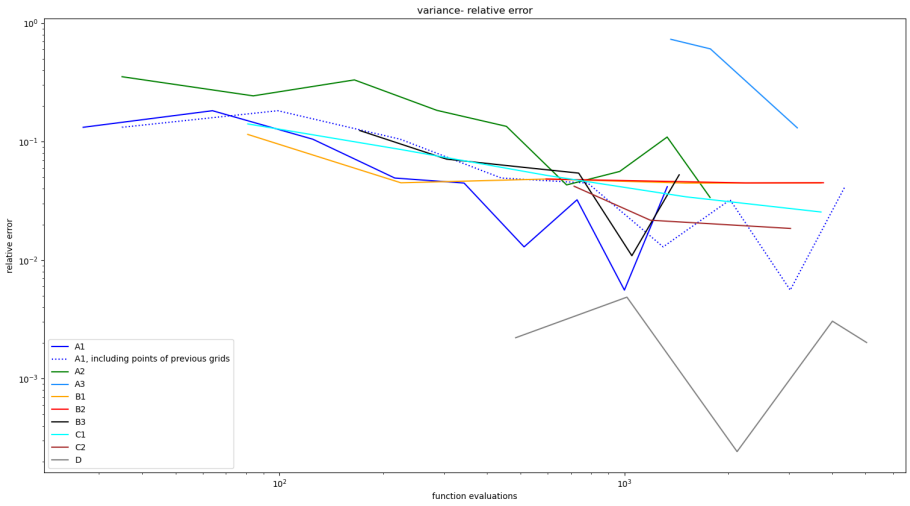


Figure 5.6: plot of *UQFunction*, with parameter z=0

with uniform distributions $X \sim U(-2.5, 2.5), Y \sim U(-2, 2), Z \sim U(5, 15)$. Analytical solutions are available for expectation and variance. The function has a discontinuity at $y = 0$. **??** plots the 2-dimensional UQ-Function where $z = 0$. Results, fig. 5.13, show that analytically computing the inner products on an interpolant constructed via the adaptive or non-adaptive combination technique using hat functions performs best. The discontinuity explains why spatial adaptivity shows good result for the expectation. Using Gaussian quadrature directly on the model results in high oscillations in the error for both expectation and variance; for an even amount of points per dimensions, the results are far better. The reason may be that whenever the number of points per dimension is odd, a point is placed directly on the discontinuity which distorts the global polynomial. We can also observe, that no method which constructs a gPCE truncation, produces good results for the variance. Possibly, the polynomial truncation at 10 is too low to produce a decent surrogate for the model.



(a) mean *UQ-Function* function, relative error



(b) variance *UQ-function*, relative error

Figure 5.7: *UQ-Function*, relative error of mean and variance

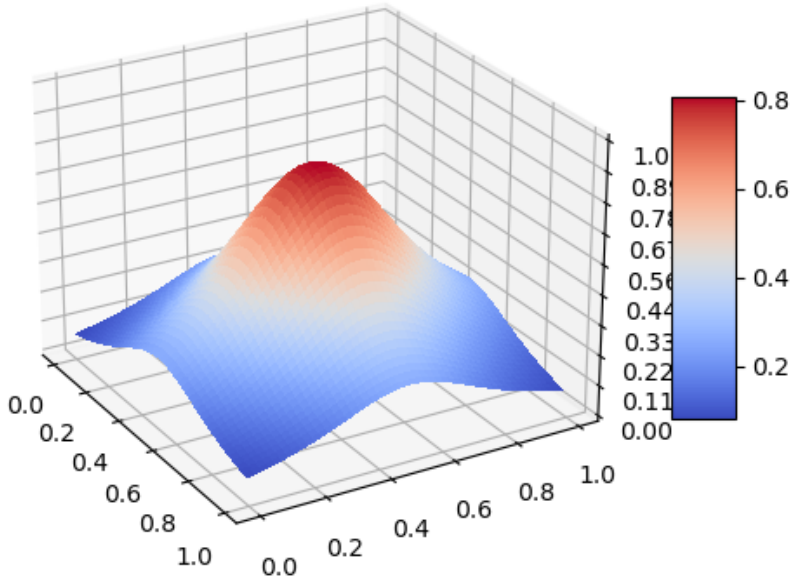


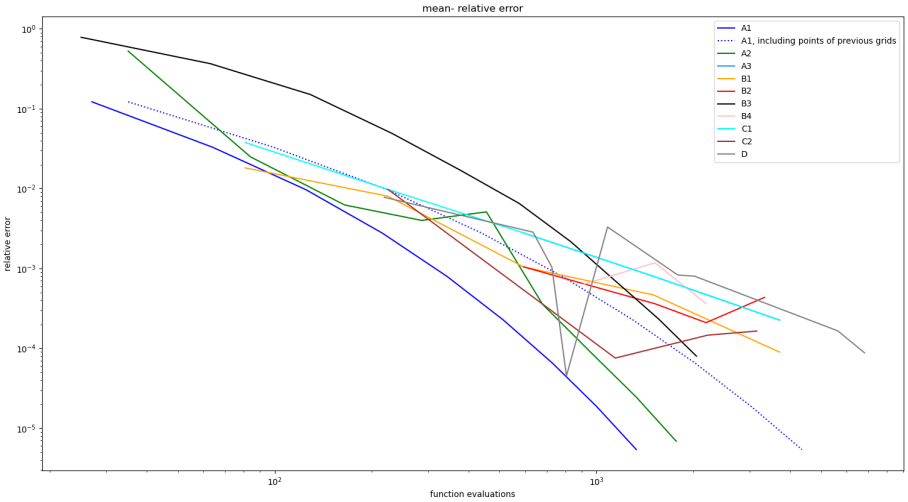
Figure 5.8: plot of the two-dimensional *product peak* function

5.1.4 Product-Peak

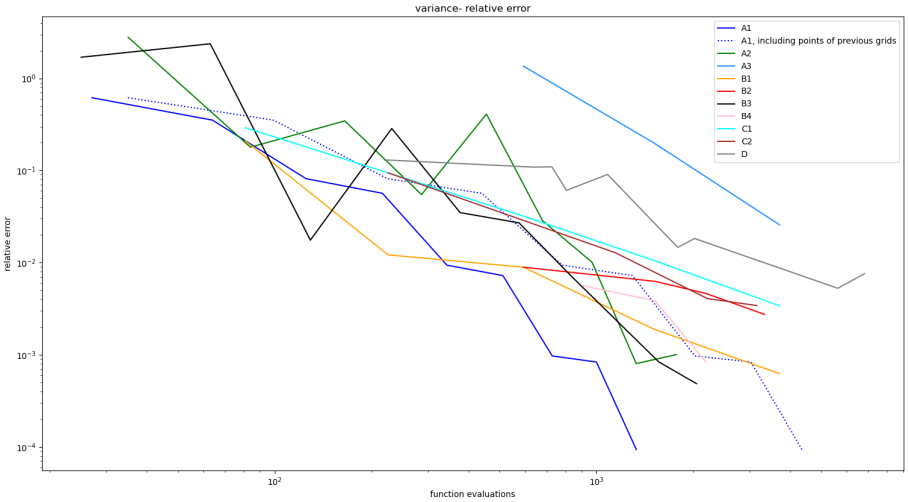
The next test function is the 3-dimensional *product-peak* function [6]

$$f(\vec{x}) = \frac{10^{-d}}{\Pi_{i=1}^{d}}(10i)^{-2} + (x_i - 0.99)^2)$$

with identical uniform distributions $X_i \sim U(0,1), i = 0,1,2$. Analytical solutions are available for expectation and variance. Figure 5.8 plots the 2-dimensional case. Results, see fig. 5.9, show that Gaussian and Leja perform well, which may be attributed to the smoothness. This smoothness and lack of local phenomena could also explain why spatial adaptivity is not advantageous in this case. Again, using the spatially adaptive trapezoidal rule as quadrature method, which does not avoid internal aliasing errors, performs worst.



(a) mean *product peak* function, relative error



(b) variance *product peak* function, relative error

Figure 5.9: 3 dimensional *product peak* function, relative error of mean and variance

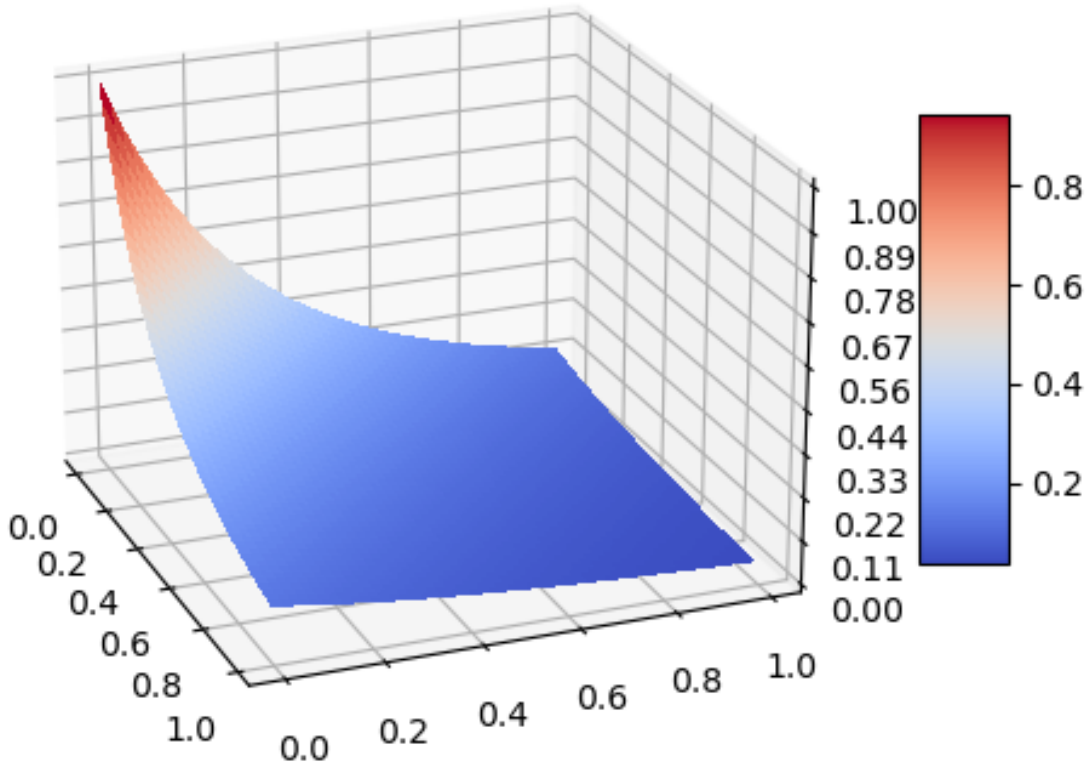


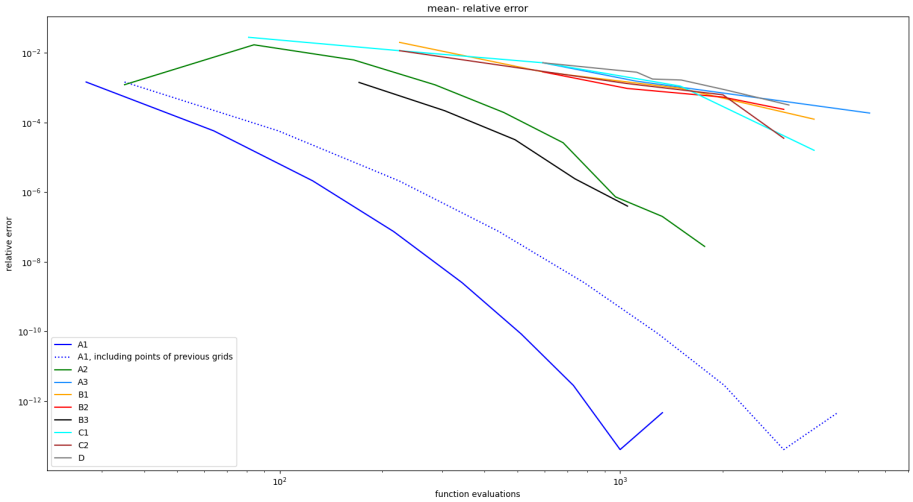
Figure 5.10: plot of the two-dimensional Genz Corner Peak function

5.1.5 Corner Peak

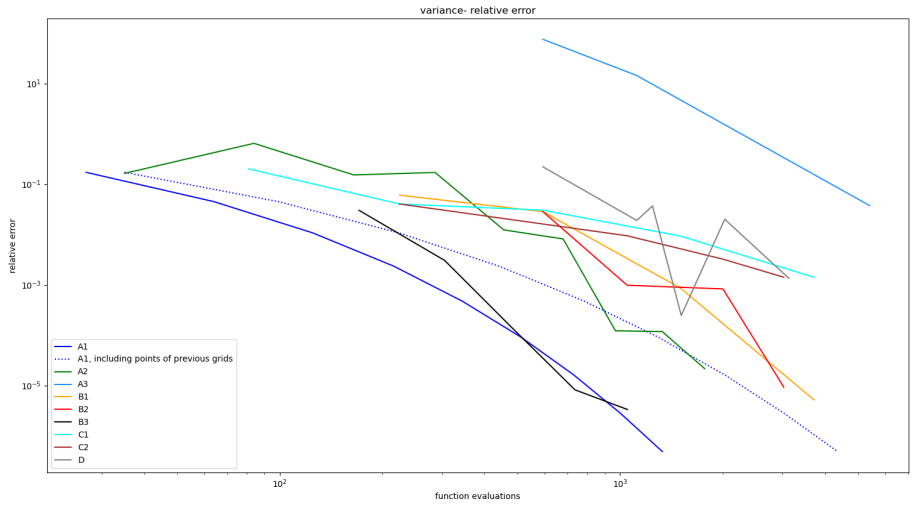
The next test function is the 3-dimensional *corner peak* function [6]

$$f(\vec{x}) = (1 + \sum_{i=1}^{d} x_i)^{-d-1}$$

with identical uniform distributions $X_i \sim U(0,1), i = 0,1,2$. Analytical solutions are available for expectation and variance. Figure 5.10 plots the 2-dimensional case. Like the product peak function, the corner peak function is fairly smooth and results, see fig. 5.11, are similar. Gaussian and Leja again perform well. The method of first interpolating the model via sparse Leja interpolation and then perform PSP with Gaussian quadrature seems to produce better results than performing PSP with sparse Leja quadrature. Again, spatial adaptivity is not advantageous in this case, possibly because of the smoothness.



(a) mean *corner peak* function, relative error



(b) variance *corner peak* function, relative error

Figure 5.11: three dimensional *corner peak* function, relative error of mean and variance

5.1.6 discontinuous Genz function

The last 3-dimensional test function is the discontinuous Genz function [6]

$$f(\vec{x}) = \begin{cases} 0 & \vec{x} \geq 0.5 \\ e^{-\sum_{i=1}^{d} x_i} & \text{otherwise} \end{cases}$$

with identical uniform distributions $X_i \sim U(0,1), i = 0,1,2$. Analytical solutions are available for expectation and variance. fig. 5.12 plots the 2-dimensional function. Due to the discontinuity, method that apply Gaussian and Leja quadrature/ interpolation do not perform well, see fig. 5.13. When applying Gaussian quadrature directly on the model, for every odd number of points per dimension, a point is placed on the discontinuity at $x_i = 0.5, i = 0,1,2$ resulting in oscillations, like for the discontinuous UQ-function. The discontinuity and the very localized problem, as the function is constant on most of the domain, makes the discontinuous Genz function a good candidate for spatial adaptivity. We observe that the relative error for the variance stops converging towards zero for all methods, suggesting that a gPCE truncation at degree 10 is not sufficient for constructing a good surrogate and that further increases in model evaluations brings little benefit.

5.2 HBV

This section compares results for the hydrological model HBV. The model has 12 stochastic inputs, but for the comparisons, 6 parameters were fixed, reducing the problem to 6 dimensions. The model returns values for the streamflow in the Oldman river basin over a period of several decades; we only consider the period from 01.10.2005 until 30.09.2006. Alternatively, it can also return a goodness of fit value which measures how good the simulated streamflow fits to the measured data. This value can be used to adjust the input distributions, by minimizing the goodness of fit. As goodness of fit, we use the root mean squared error:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{N-1}(m_i - s_i)^2}{N}}$$

where m_i is the i-th measurement, s_i the i-th simulated value and N the number of measurements, in our case $N = 365$, one per day. Analytical solutions for expectation, variance or Sobol indices are not available. To obtain reference solutions for expectation and variance, Quasi Monte Carlo with the Halton sequence and 50000 evaluations was performed. For all methods in this section, the maximum gPCE truncation is set at degree 3 and no points on the boundary are included.

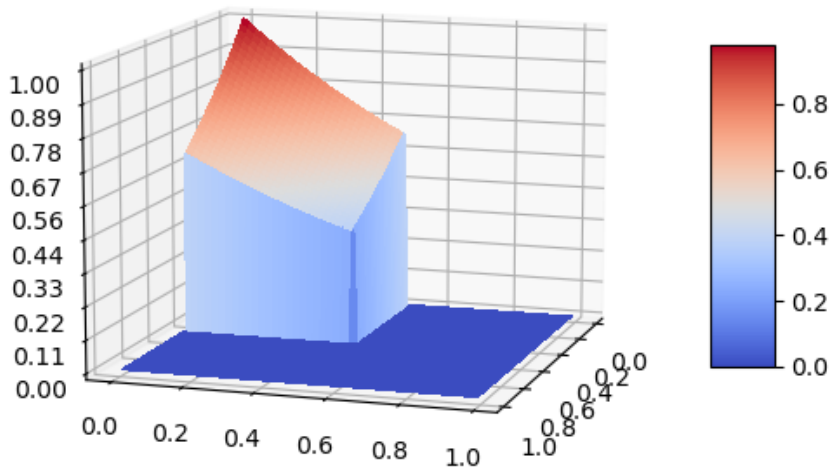
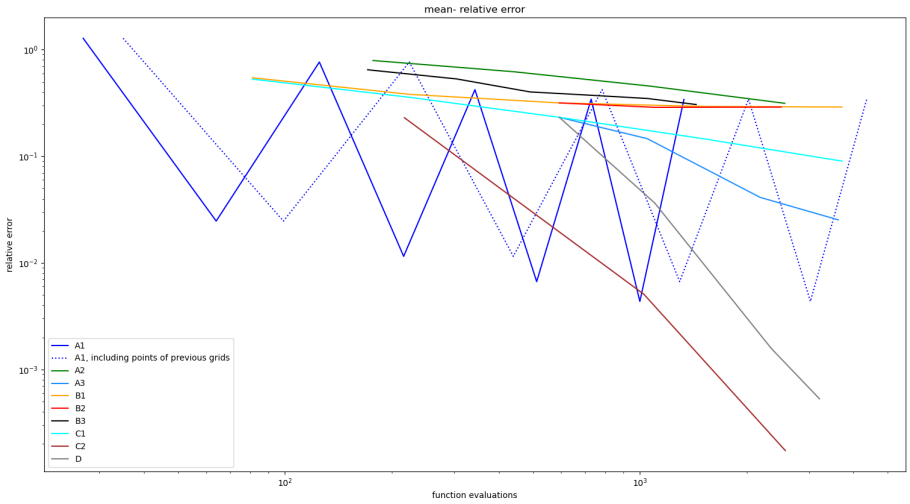
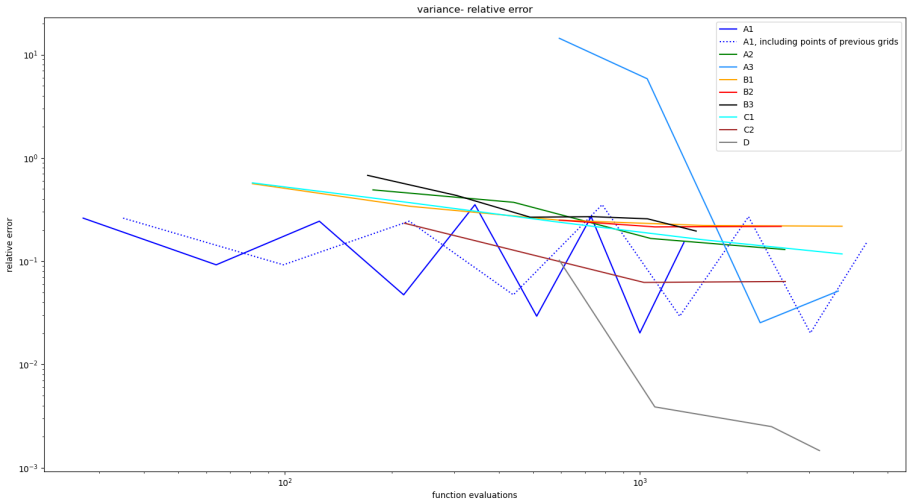


Figure 5.12: plot of the two-dimensional Genz-Discontinuous function



(a) mean discontinuous Genz function, relative error



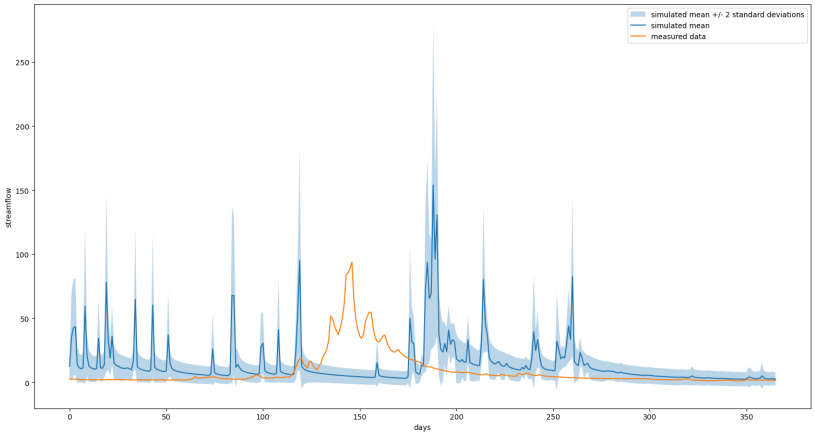
(b) variance discontinuous Genz function, relative error

Figure 5.13: 3 dimensional discontinuous Genz function, relative error of mean and variance

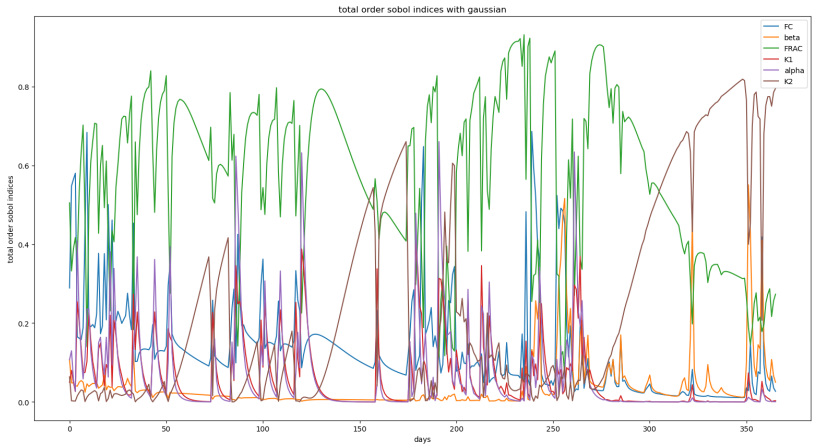
First we compare the results with time series data as output. Figure 5.14 and fig. 5.15 show the simulations for PSP with Gaussian quadrature and for the analytical computation of the gPCE coefficients on a piecewise linear interpolant constructed with the single dimension refinement strategy, respectively. We observe that for both methods the measured data deviates significantly from the expectation suggesting that the model or the input distributions may not fit too well. For around 4000 model evaluations invested in both methods, the results are already quite similar. Concluding from the total order Sobol' indices, the parameter *FRAC* has the most impact on the output most days.

Going on with the RMSE, fig. 5.16 shows the relative errors for mean and variance. Included are the gPCE methods which produced the best results for the 3-dimensional test functions: A1, A2, B3, C1, C2, see table 5.1. Even in 6 dimensions, the full Gaussian quadrature performs well and produces the most accurate result for the mean while the analytical computation of the gPCE via the standard CT using linear interpolation gives better results for the variance. For the Ishigami function, the opposite was observed, which could be investigated in future work. Perhaps the good results of linear interpolation with standard CT could be further improved by employing the Simpson's rule instead of the trapezoidal rule. The spatially adaptive CT performs worse than the Standard CT with piecewise linear interpolation. In general, the final one-dimensional point sets for the spatially adaptive combination technique can give insights on properties of the function, e.g. where it has local problems, and may also give hints whether the refinement went wrong. In this case, however, the final point sets, see fig. 5.17, at first glance do not explain why spatial adaptivity seems to perform worse than the standard combination technique. Another observation is that Leja interpolation/ quadrature does not give good results. One could examine if leaving out boundary points is, in general, reasonable for the sparseSpACE implementation.

Figure 5.18 show the computed total order Sobol' indices. All methods indicate that the parameter *FRAC* has the most impact on the RMSE, followed by *FC*. Knowing which parameter has the biggest impact on the chosen error is useful when adjusting the input parameters to better fit to the measured data.

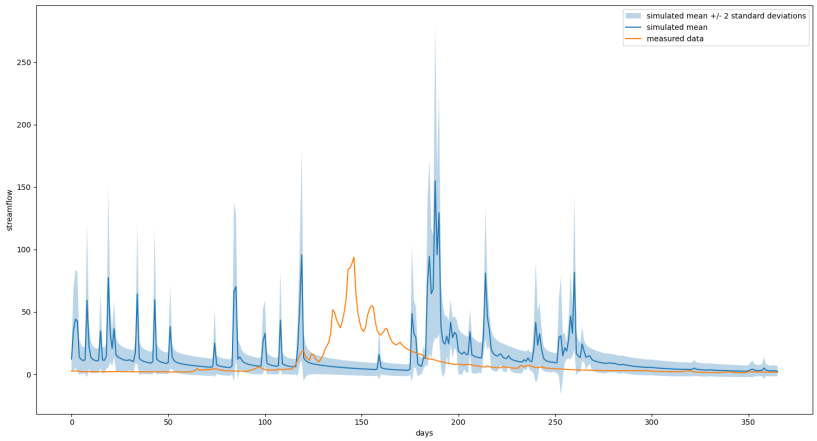


(a) simulated mean and standard deviation

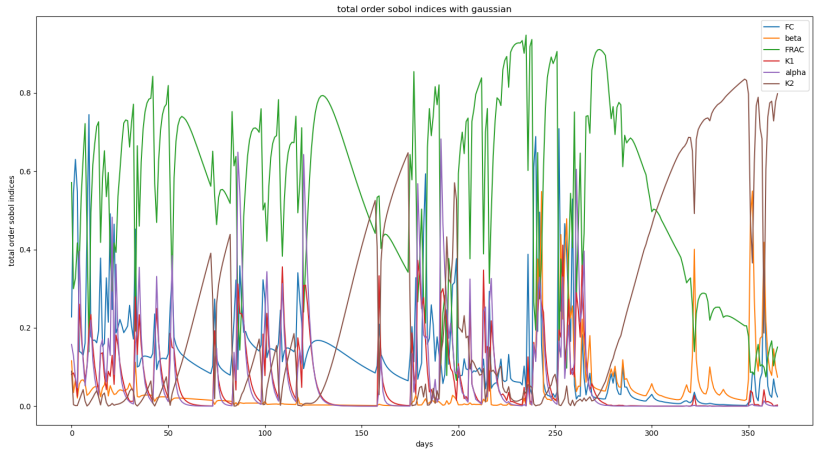


(b) total order Sobol' indices

Figure 5.14: simulations for the HBV model via PSP with Gaussian quadrature and 4096 model evaluations

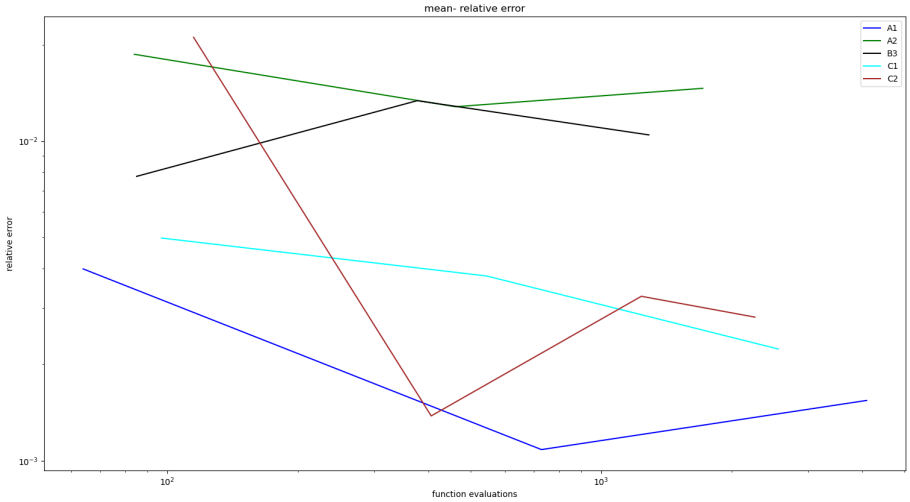


(a) simulated mean and standard deviation

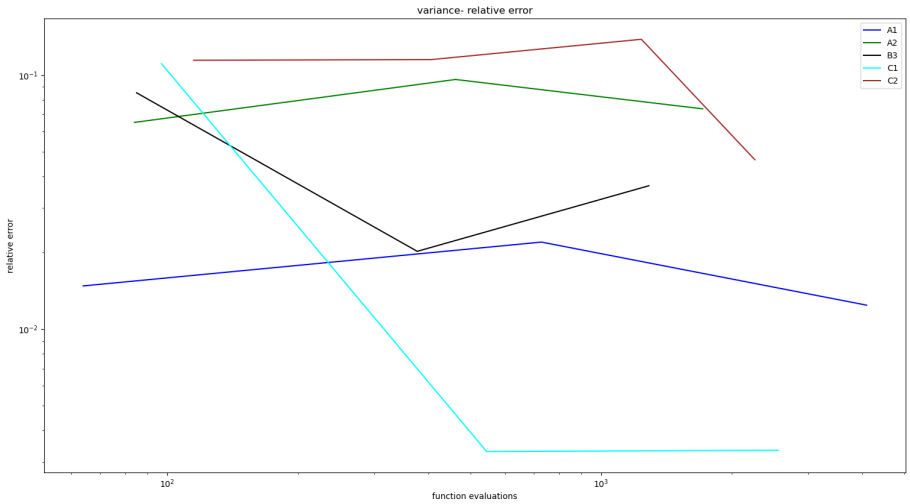


(b) total order Sobol' indices

Figure 5.15: simulations for the HBV model, with analytical gPCE computation on a piecewise linear interpolant constructed with the single dimension refinement strategy



(a) mean HBV, relative error



(b) variance HBV, relative error

Figure 5.16: HBV-RSME, relative error of mean and variance

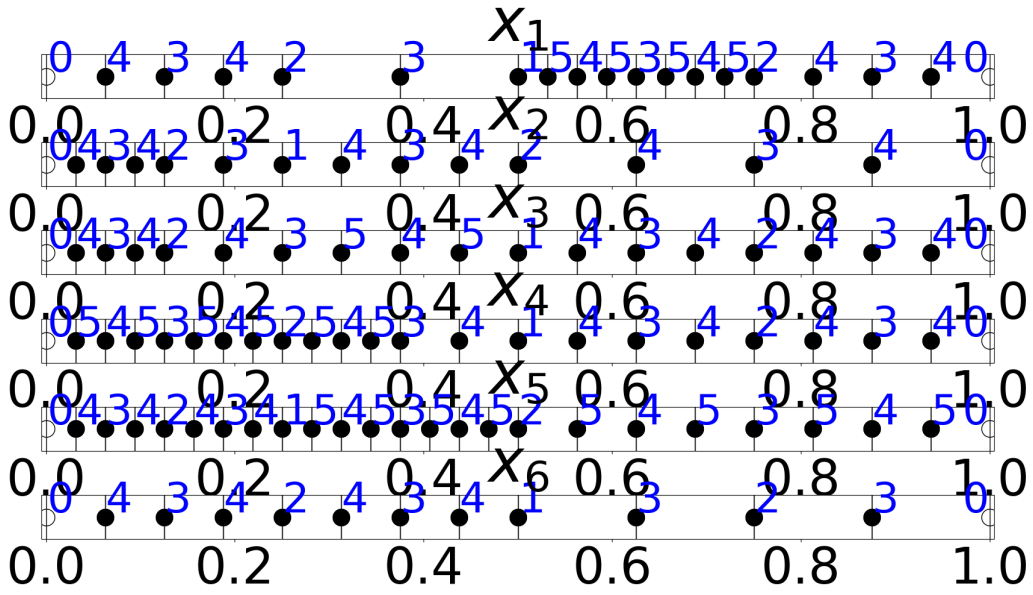


Figure 5.17: refinement for HBV-RMSE with 2263 interpolation points

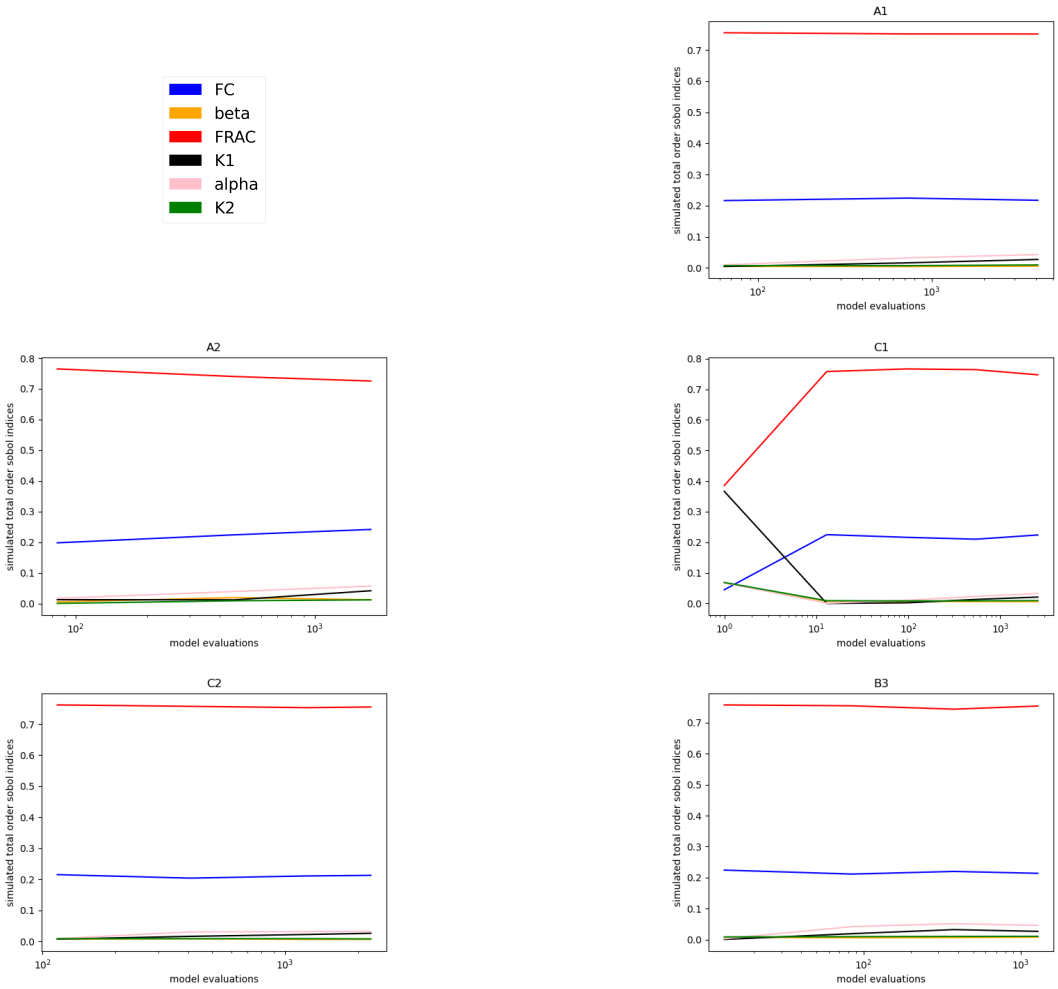


Figure 5.18: total order Sobol' indices of the HBV-RSME

6 Conclusion and Outlook

This thesis has presented and compared results for several gPCE methods: performing PSP directly on the model, performing PSP on an interpolant, and computing the gPCE coefficients of an interpolant analytically. Computing the inner products on a piecewise linear interpolant constructed with the single dimension refinement strategy allowed to apply spatial adaptivity while also avoiding internal aliasing errors that arise when applying the spatially adaptive trapezoidal sum as PSP quadrature method. Results for models with three to six stochastic inputs show that for functions that are not differentiable or have discontinuities, the single dimension refinement strategy performs well while for smoother functions, higher order interpolation with the Leja grid or direct PSP with Gaussian quadrature produces better results.

In future work, one could try out models of higher dimensionality which may produce better results for the spatially adaptive combination technique in comparison to the full Gaussian quadrature. For higher dimensions, the PSP quadrature rule should also use sparse grids. A sparse PSP quadrature rule that integrates polynomials exactly, for example sparse Clenshaw-Curtis or Leja, can be chosen such that internal aliasing errors are avoided without including any points only necessary for higher polynomials than those in the gPCE truncation scheme, because the set of polynomials in the truncation scheme also follows the Smolyak approach.

Another variant one could try out is to apply Simpson's sum instead of the trapezoidal sum, for which spatial adaptivity is also possible.

The gPCE truncation has not been chosen depending on the number of interpolation points. For a piecewise linear interpolant, a link between these two choices is not obvious and depends on the properties of the model: for the compared non-differentiable functions, i.e. the G-function, the UQ-function, and the discontinuous Genz function, increasing the number of interpolation points soon stops increasing the accuracy for a truncation at degree 10, while for the smoother functions, i.e. Ishigami, corner-peak, and product-peak, the gPCE truncation at degree 10 was sufficient to allow longer convergence of the error twoards zero. One could think of heuristics to combine the two choices.

While eq. (3.10), i.e. the formula to compute the gPCE coefficients on a sparse grid interpolant, has been implemented for interpolation with hat functions, it could also be done for the Leja interpolant with Lagrange polynomials. Since the Leja interpolant is

a global polynomial, the gPCE truncation and the order of Leja interpolation would not have to be chosen separately because inner products for polynomials with degree higher than the Leja interpolant are zero. While spatial adaptivity is not available for Leja grids, dimension adaptivity could be applied. Having a Leja interpolant constructed with dimension adaptivity, the choice of the truncation scheme would again be obvious as the adapted Smolyak set of levelvectors included in the interpolation translates to a Smolyak set for the gPCE truncation scheme.

Furthermore, while in this thesis only uniform dsitributions were considered, comparing results for other distributions may show benefits of the nonlinear transformation of the model into the unitcube.

List of Figures

Bibliography

[1] Hans-Joachim Bungartz and Michael Griebel. "Sparse Grids." In: *In: Acta Numerica. Vol. 13, pp. 147-269* 13 (May 2004). DOI: 10.1017/S0962492904000182.

[2] C.W. CLENSHAW and A.R. CURTIS. "A method for numerical integration on an automatic computer." In: *Numerische Mathematik* 2 (1960), pp. 197–205.

[3] Ionut-Gabriel Farcas et al. "Nonintrusive Uncertainty Analysis of Fluid-Structure Interaction with Spatially Adaptive Sparse Grids and Polynomial Chaos Expansion." In: *SIAM Journal on Scientific Computing* 40 (2018).

[4] Jonathan Feinberg and Hans Petter Langtangen. "Chaospy: an open source tool for designing methods of uncertainty quantification." In: *SIAM Journal on Scientific Computing* (2015).

[5] Baskar Ganapathysubramanian and Nicholas Zabaras. "Sparse grid collocation schemes for stochastic natural convection problems." In: *Journal of Computational Physics* 225 (July 2007), pp. 652–685. DOI: 10.1016/j.jcp.2006.12.014.

[6] Alan Genz. "A Package for Testing Multiple Integration Subroutines." In: *Numerical Integration: Recent Developments, Software and Applications*. Ed. by Patrick Keast and Graeme Fairweather. Dordrecht: Springer Netherlands, 1987, pp. 337–340. ISBN: 978-94-009-3889-2. DOI: 10.1007/978-94-009-3889-2_33.

[7] Michael Griebel, Michael Schneider, and Christoph Zenger. "A combination technique for the solution of sparse grid problems." In: (1992).

[8] Mario Heene. "A massively parallel combination technique for the solution of high-dimensional PDEs." PhD thesis. Universitaet Stuttgart, 2019.

[9] Fritz Hofmeier. "Applying the Spatially Adaptive Combination Technique to Uncertainty Quantification." Technical University of Munich, 2019.

[10] T. Ishigami and T. Homma. "An importance quantification technique in uncertainty analysis for computer models." In: *the First International Symposium on Uncertainty Modeling and Analysis* (1990).

[11] Amandine Marrel et al. "Calculations of Sobol indices for the Gaussian process metamodel." In: *Reliability Engineering and System Safety* 94.3 (2009), pp. 742–751. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2008.07.008.

[12] Youssef M. Marzouk and Patrick R. Conrad. "Adaptive smolyak pseudospectral approximations." In: *SIAM Journal on Scientific Computing* 35 (2013).

[13] Akil Narayan and John Jakeman Jakeman. "Adaptive Leja sparse grid constructions for stochastic collocation and high-dimensional approximation." In: *SIAM Journal on Scientific Computing* (2014).

[14] Michael Obersteiner. *sparseSpACE-The Sparse Grid Spatially Adaptive Combination Environment*. `https://github.com/obersteiner/sparseSpACE`.

[15] Michael Obersteiner and Hans-joachim Bungartz. "A generalized spatially adaptive sparse grid combination technique with dimension-wise refinement." In: 43.4 (2021).

[16] Dirk Pflüger. "Spatially Adaptive Sparse Grids for High-Dimensional Problems." PhD thesis. Technical University of Munich, 2010. ISBN: 9783868535556.

[17] Smolyak S. "Quadrature and interpolation formulas for tensor products of certain classes of functions." In: *Svoiet Mathematics, Doklady* (1963).

[18] Bruno Sudret. "Global sensitivity analysis using polynomial chaos expansions." In: *Reliability Engineering & System Safety* 93.7 (2008). Bayesian Networks in Dependability, pp. 964–979. ISSN: 0951-8320. DOI: `https://doi.org/10.1016/j.ress.2007.04.002`.

[19] Julian Valentin. "B-Splines for Sparse Grids: Algorithms and Application to Higher-Dimensional Optimization." PhD thesis. Universitaet Stuttgart, 2019.

[20] D. Xiu and G. E. Karniadakis. "The Wiener-Askey polynomial chaos for stochatic differential equations." In: *SIAM Journal on Scientific Computing* (2002).