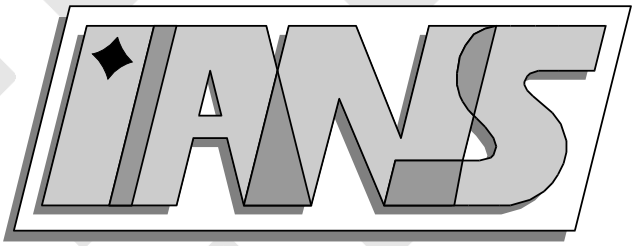
Universität Stuttgart



Computing expensive multivariate functions of fuzzy numbers using sparse grids

Andreas Klimke, Barbara Wohlmuth

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Preprint 2004/002

Universität Stuttgart

Computing expensive multivariate functions of fuzzy numbers using sparse grids

Andreas Klimke, Barbara Wohlmuth

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Preprint 2004/002

Institut für Angewandte Analysis und Numerische Simulation (IANS)
Fakultät Mathematik und Physik
Fachbereich Mathematik
Pfaffenwaldring 57
D-70 569 Stuttgart

E-Mail: ians-preprints@mathematik.uni-stuttgart.de
WWW: http://preprints.ians.uni-stuttgart.de

ISSN **1611-4176**

Computing expensive multivariate functions of fuzzy numbers using sparse grids

Andreas Klimke, Barbara Wohlmuth

Abstract

Fuzzy arithmetic provides a powerful tool to introduce uncertainty into mathematical models. With Zadeh's extension principle, one can obtain a fuzzy extension of any objective function. Computing expensive multivariate functions of fuzzy numbers, however, often poses a difficult problem due to non-applicability of common fuzzy arithmetic algorithms, severe overestimation, or very high computational complexity. This paper proposes a new approach based on sparse grids, consisting of two parts: First, we compute a surrogate function using sparse grid interpolation. Second, we perform the fuzzy-valued evaluation of the surrogate function by a suitable implementation of the extension principle based on real or interval arithmetic. The new approach gives accurate results and requires only few function evaluations.

Keywords: fuzzy arithmetic, interval analysis, sparse grids, uncertainty modeling

---

1 Introduction

Fuzzy sets [32] provide a widely appreciated tool to introduce uncertain parameters into mathematical models. Especially in engineering applications (e.g. risk assessment, manufacturing tolerances), a great interest in modeling uncertain input data and the effects on the output of the model can be observed.

Zadeh's well-known extension principle forms the theoretical basis of fuzzy arithmetic, extending real-valued functions to functions of fuzzy numbers. In practice, however, the implementation of fuzzy arithmetic turns out to be a problem of nonlinear programming, which is commonly difficult to solve and also computationally expensive.

In the following, we give a brief review of existing methods implementing fuzzy arithmetic. We can classify these methods into two categories, which we will refer to as *classical* fuzzy arithmetic and *constrained* fuzzy arithmetic. The classical category includes fuzzy arithmetic based on LR-fuzzy numbers according to Dubois and Prade [6], fuzzy arithmetic based on interval arithmetic according to Kaufmann and Gupta [13], or similar variations (e.g. [7]). They provide a way to compute fuzzy-valued expressions in a manageable way in terms of computational complexity. These methods, however, have one important common characteristic, which turns out to be a serious drawback: Each variable is considered independently in

each occurrence, even if the same variable occurs multiple times in a given expression. This can lead to a large overestimation of the result, which has been rigorously demonstrated (see, for instance, [4, 9, 18, 30, 31]).

Methods of the constrained category attempt to avoid the overestimation effect of the classical methods. In interval arithmetic, *branch-and-bound* algorithms are employed to adaptively reduce overestimation [8, 12, 20], which can also be used in fuzzy arithmetic. Dong and Wong [5] suggested a non-overestimating approach called FWA algorithm. In its initial version, the algorithm was only applicable to monotonic functions. An improved algorithm was presented by Wood et al. [30]. In addition to a reduction of the complexity of the FWA algorithm in certain cases, it included an enhancement to correctly compute non-monotonic functions. Wood's algorithm requires an additional routine to locate internal extrema, which can be done either analytically or numerically with an algorithm suited for global optimization of non-linear functions. More recently, the transformation method was proposed by Hanss [9, 10] as a practical approach to evaluate fuzzy-parameterized models without requiring an external optimization routine. Additionally, Hanss' method provides a sensitivity analysis framework. Further implementations of constrained fuzzy arithmetic were presented in [22, 31].

Let us now consider the case of expensive multivariate functions, which usually exhibit at least one of the following characteristics:

1. A single real-valued function evaluation requires a significant amount of computation time in the order of seconds, minutes or even hours,
2. interval-valued evaluation leads to a severe overestimation due to multiple variable occurrences, wrapping, or cancellation (classification by Neumaier [23]).
3. an extension of the function to perform fuzzy or interval arithmetic directly instead of real-valued arithmetic is hardly possible (e.g. the evaluation involves sophisticated numerical algorithms), or
4. the function is implemented as a separate, black box routine permitting real-valued function evaluations only.

In the cases (2) to (4), classical fuzzy arithmetic cannot be applied due to a severe overestimation of the results or non-applicability of the specific modified arithmetic of fuzzy numbers. Constrained fuzzy arithmetic will also not be applicable in most cases (depending on the actual implementation), or be extremely slow due to a high number of required function evaluations.

In the following, we will show that sparse grid interpolation can be used to obtain a surrogate function with a low number of function evaluations, even in higher dimensions. We then apply constrained fuzzy arithmetic on the surrogate function to obtain a good approximation of the actual fuzzy result.

The rest of the paper is organized as follows: In Section 2, we briefly review the concepts of sparse grid interpolation. Section 3 describes how to perform fuzzy arithmetic on the sparse grid. Section 4 provides numerical results for several nonlinear test functions of various typical characteristics. We conclude with an overview of the benefits of the new method and its limitations in Section 5.

2 Sparse grid interpolation

Sparse grids have been successfully applied to the solution of partial differential equations, initiated by Zenger in 1991 [33]. This is still the most active research field involving sparse grids, while other important application areas are being studied as well (e.g. numerical quadrature, image processing, and data compression – see [2] for references).

2.1 Characteristics of sparse grid interpolation

The interpolation problem considered with sparse grid interpolation is an optimal recovery problem (i.e. the selection of points such that a smooth function can be approximated with a suitable interpolation formula). Depending on the characteristics of the function to interpolate (degree of smoothness, periodicity), various interpolation techniques based on sparse grids exist (e.g. [1, 25, 27, 28]). All of them employ Smolyak's construction [26], which forms the basis of all sparse grid methods. With Smolyak's method, well-known univariate interpolation formulas are extended to the multivariate case by using tensor products in a special way. As a result, one obtains a powerful interpolation method that requires significantly less support nodes than conventional interpolation on a full grid. The points comprising the multidimensional sparse grid are hereby selected in a predefined fashion. The difference in the number of required points can be several orders of magnitude with increasing problem dimension. The most important property of the method constitutes the fact that the asymptotic quadratic error decay of full grid interpolation with increasing grid resolution is preserved up to a logarithmic factor. An additional benefit of the method is its hierarchical structure, which can be used to obtain an estimate of the current approximation error. Thus, one can easily develop an interpolation algorithm that aborts automatically when a desired (estimated) accuracy is reached.

2.2 Smolyak's algorithm

In the following, we briefly review Smolyak's construction for multivariate interpolation. The notation adheres to [1]. We would like to approximate functions $f : [0,1]^d \to \mathbb{R}$ using a finite number of support nodes. For simplicity, we have restricted the domain to the d-dimensional unit cube, however, we can easily handle arbitrary axis-aligned boxes by rescaling later on. In the one-dimensional case, an interpolation formula is given by

$$U^i(f) = \sum_{j=1}^{m_i} a_j^i \cdot f(x_j^i),$$

with $i \in \mathbb{N}$, the basis functions $a_j^i \in C([0,1])$, $a_j^i(x_l^i) = \delta_{jl}$, $l \in \mathbb{N}$, and $x_j^i \in X^i = \{x_1^i, \ldots, x_{m_i}^i\}$, $x_k^i \in [0,1], 1 \le k \le m_i$. To obtain an interpolation formula for the multivariate case, one can use the tensor product formula

$$(U^{i_1} \otimes \cdots \otimes U^{i_d})(f) = \sum_{j_1=1}^{m_{i_1}} \cdots \sum_{j_d=1}^{m_{i_d}} (a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_d}^{i_d}) \cdot f(x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d}). \tag{1}$$

However, the above formula requires a very high number of $m_{i_1} \cdots m_{i_d}$ support nodes, which are sampled on the full grid. To reduce the number of support nodes while maintaining

the properties of the interpolation formula for $d = 1$, Smolyak's construction is used. With $U^0 = 0$, $\Delta^i = U^i - U^{i-1}$, $|\mathbf{i}| = i_1 + \cdots + i_d$ for $\mathbf{i} \in \mathbb{N}^d$, and $q \geq d, q \in \mathbb{N}$, the Smolyak algorithm is given by

$$\begin{aligned} A_{q,d}(f) &= \sum_{|\mathbf{i}| \leq q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f) \\ &= A_{q-1,d}(f) + \sum_{|\mathbf{i}|=q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f) \qquad (2) \\ &= \sum_{k=0}^{n:=q-d} \underbrace{\sum_{|\mathbf{i}|=k+d} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f)}_{\Delta A_{k+d,d}(f)}, \qquad (3) \end{aligned}$$

with

$$\Delta A_{k+d,d}(f) = \sum_{|\mathbf{i}|=k+d} \sum_{\mathbf{j}} (a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_d}^{i_d}) \cdot \left(f(\mathbf{x}_\mathbf{j}^\mathbf{i}) - A_{k+d-1,d}(f)(\mathbf{x}_\mathbf{j}^\mathbf{i}) \right), \qquad (4)$$

$\Delta A_{d-1,d} = 0$, and $\mathbf{j}$ denoting the multi-index $(j_1, \ldots, j_d)$, $j_l = 1, \ldots, m_{i_l}^\Delta$, $l = 1, \ldots, d$, and the points $\mathbf{x}_\mathbf{j}^\mathbf{i} = (x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d})$ with $x_{j_l}^{i_l}$ denoting the j_lth element of $X_\Delta^{i_l} = X^{i_l} \setminus X^{i_l - 1}$, $X^0 = \emptyset$, and $m_{i_l}^\Delta = \# X_\Delta^{i_l}$. Furthermore, it is useful to define

$$w_\mathbf{j}^{k,\mathbf{i}} = f(\mathbf{x}_\mathbf{j}^\mathbf{i}) - A_{k+d-1,d}(f)(\mathbf{x}_\mathbf{j}^\mathbf{i})$$

as the *hierarchical surpluses* [2] at level k with $k = |\mathbf{i}| - d$, since $\Delta A_{k+d,d}$ corrects $A_{k+d-1,d}$ at the points $\mathbf{x}_\mathbf{j}^\mathbf{i}$ to the actual value of $f(\mathbf{x}_\mathbf{j}^\mathbf{i})$. For continuous functions, the hierarchical surpluses tend to zero as the level k tends to infinity. This makes the hierarchical surpluses a natural candidate for error estimation and control.

From (2), we observe that we can increase the accuracy of the interpolation based on Smolyak's construction without having to discard previous results. Most importantly, to compute $A_{q,d}(f)$, only the function values at the sparse grid

$$H_{q,d} = \bigcup_{q-d+1 \leq |\mathbf{i}| \leq q} (X^{i_1} \times \cdots \times X^{i_d}) \qquad (5)$$

are needed. One should select the sets X^i in a nested fashion such that $X^i \subset X^{i+1}$ to obtain many recurring points with increasing q. With $X^0 = \emptyset$, $X_\Delta^i = X^i \setminus X^{i-1}$, one can rewrite (5) to

$$H_{q,d} = \bigcup_{|\mathbf{i}| \leq q} (X_\Delta^{i_1} \times \cdots \times X_\Delta^{i_d}) = H_{q-1,d} \cup \underbrace{\bigcup_{|\mathbf{i}|=q} (X_\Delta^{i_1} \times \cdots \times X_\Delta^{i_d})}_{\Delta H_{q,d}}, \qquad (6)$$

$H_{d-1,d} = \emptyset$, which is more convenient for a successive refinement of the grid with increasing parameter q. For comparison, we also introduce the full grid $H_{\text{full},q,d}$ as the full grid with the smallest number of points still containing the sparse grid $H_{q,d}$ with

$$H_{\text{full},q,d} = X^{q-d+1} \times \cdots \times X^{q-d+1}. \qquad (7)$$

2.3 Definition of the grid and the basis functions

To implement sparse grid interpolation, it is required to select suitable basis functions and an according sparse grid structure. A straightforward approach is obtained by using piecewise linear basis functions with sets X^i of equidistant nodes. For a detailed introduction to piecewise multilinear sparse grid interpolation, see [16]. This introduction also serves as a reference for an easy-to-use MATLAB package for multilinear sparse grid interpolation, which is available for free at [11].

In case of piecewise linear basis functions, the Clenshaw-Curtis type sparse grid H^{CC} [24, 25] with equidistant nodes seems to perform best (for a comparison of various grid types, see [16]).

The grid points can be computed using the sets X^i with x_j^i defined as follows:

$$\begin{aligned} X^i &= \{x_1^i, \ldots, x_{m_i}^i\} \\ m_i &= \begin{cases} 1 & \text{if } i = 1, \\ 2^{i-1} + 1 & \text{if } i > 1. \end{cases} \\ x_j^i &= \begin{cases} (j-1)/(m_i - 1) & \text{for } j = 1, \ldots, m_i \text{ if } m_i > 1, \\ 0.5 & \text{for } j = 1 \text{ if } m_i = 1. \end{cases} \end{aligned} \tag{8}$$

The basis functions are given by

$$a_1^1(x) = 1 \quad \text{for } i = 1, \text{ and}$$

$$a_j^i(x) = \begin{cases} 1 - (m_i - 1) \cdot \left| x - x_j^i \right|, & \text{if } \left| x - x_j^i \right| < 1/(m_i - 1), \\ 0, & \text{otherwise,} \end{cases}$$

for $i > 1$ and $j = 1, \ldots, m_i$. Figure 1 shows the grid structure in two and three dimensions.

2.4 Accuracy of piecewise multilinear sparse grid interpolation

To guarantee a good approximation quality, the function f to recover must satisfy smoothness properties. An a priori error estimate can be obtained for a d-variate function f if bounded mixed derivatives

$$D^\beta f = \frac{\partial^{|\beta|} f}{\partial x_1^{\beta_1} \cdots x_d^{\beta_d}},$$

with $\beta \in \mathbb{N}_0^d$, $|\beta| = \sum_{i=1}^{d} \beta_i$, and $\beta_1, \ldots, \beta_d \leq 2$, exist, i.e. $f \in F$ with

$$F := \left\{ f : [0,1]^d \to \mathbb{R},\ D^\beta f \in C^0\big([0,1]^d\big),\ \beta_1, \ldots, \beta_d \leq 2) \right\}. \tag{9}$$

According to [1], the order of the interpolation error in the maximum norm is then given by

$$\| f - A_{q,d}(f) \|_\infty = \mathcal{O}(N^{-2} \cdot (\log_2 N)^{3 \cdot (d-1)}), \tag{10}$$

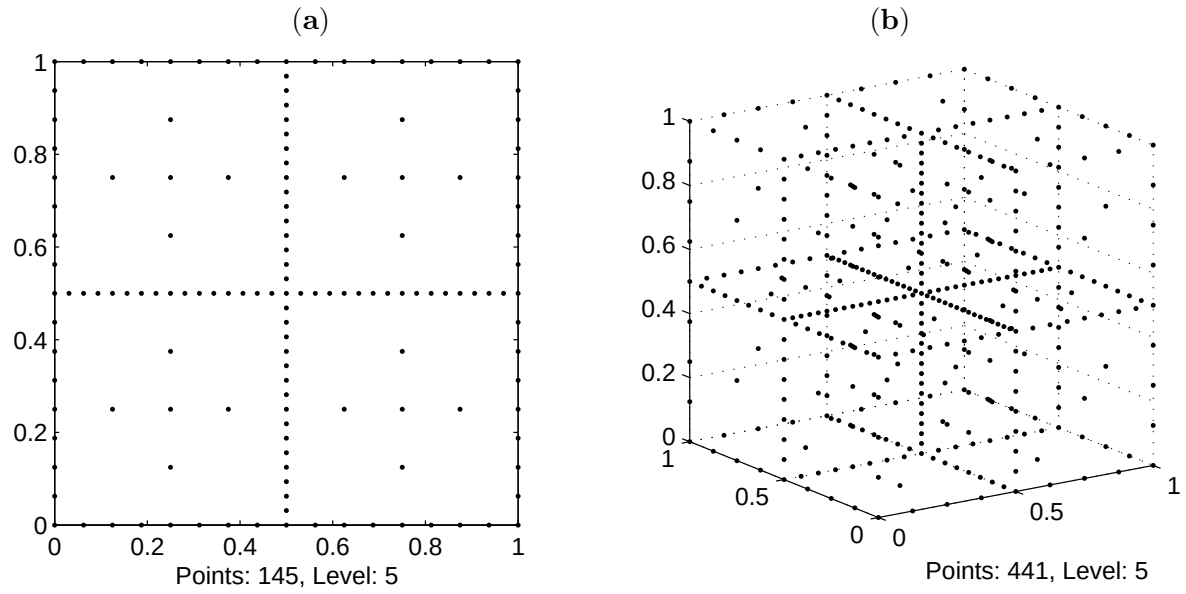


Figure 1: Sparse grids $H^{\mathrm{CC}}_{(7,2)}$ (**a**), $H^{\mathrm{CC}}_{(8,3)}$ (**b**).

with N denoting the number of sparse grid points of $H_{q,d}$. Piecewise multilinear approximation on a full grid with $\hat{N}$ grid points is much less efficient, i.e. the error in the maximum norm is of order $\mathcal{O}(\hat{N}^{-\frac{2}{d}})$.

Numerical tests [16] show that even for continuous functions, a satisfactory approximation is achieved, although the convergence rate is slower. Only discontinuous functions cannot be successfully approximated.

3 Fuzzy arithmetic using sparse grids

Having provided a tool to obtain a surrogate function (that can be made arbitrarily accurate with increasing sparse grid resolution), we will now show how to use a sparse grid interpolant $A_{q,d}(f)$ to efficiently perform fuzzy arithmetic. First, we will introduce the notation for the required discretization of the fuzzy numbers. Then, we will describe the new algorithm. It is composed of two parts:

- **Part 1.** Determining the sparse grid surrogate function, and
- **Part 2.** Performing constrained fuzzy arithmetic by a suitable method.

There are several possibilities of implementing the second part of the algorithm. We will review some of the most commonly used methods, and provide the means to use them within our approach in Section 3.3.

3.1 Discretization of the fuzzy numbers

Let $\tilde{p}_i$, $i = 1, 2, \ldots, d$ denote the fuzzy-valued input parameters with the membership functions $\mu_{\tilde{p}_i}(x_i)$, $\mu_i : \mathbb{R} \to [0, 1]$. We obtain an arbitrarily accurate discrete formulation of the fuzzy numbers with $m \to \infty$ by decomposing them into sets of $m + 1$ intervals [9]

$$P_i = \left\{ I_i^{(0)}, I_i^{(1)}, \ldots, I_i^{(m)} \right\}$$

with the intervals of confidence

$$I_i^{(j)} = \left[a_i^{(j)}, b_i^{(j)} \right], \quad a_i^{(j)} < b_i^{(j)}, \quad j = 0, \ldots, m. \tag{11}$$

The intervals of confidence [13] are often called α-*cuts* with the α-level $\alpha = j/m \in [0, 1]$. Thus, this discretization is usually referred to as α-cut representation. Note that by definition, a fuzzy number is convex, i.e. any fuzzy number can be decomposed into its α-cuts with the property

$$I_i^{(j+1)} \subseteq I_i^{(j)}, \ \forall \ j \in 0, \ldots, m-1. \tag{12}$$

In the following, we refer to $I_i^{(0)}$ as the *base* of the fuzzy number $\tilde{p}_d$. $I_i^{(0)}$ must be bounded, i.e. must be a closed interval. We define the Cartesian product of the bases as the *base box*

$$\Omega = I_1^{(0)} \times I_2^{(0)} \times \cdots \times I_p^{(0)},$$

which is a d-dimensional interval vector.

3.2 The algorithm

We first compute the sparse grid interpolant $A_{q,d}(f)$ with sufficient accuracy for the base box Ω of the fuzzy input parameters, using only a low number of real-valued function evaluations. This holds even in higher dimensions, as illustrated in Table 1, due to the very moderate growth of the number of sparse grid points with increasing problem dimension. The hierarchical structure of the sparse grid interpolation scheme allows us to subsequently increase the interpolation depth until a sufficient estimated relative or absolute accuracy is reached. We can then replace all interval- or real-valued function evaluations that become necessary with the application of a suitable constrained fuzzy arithmetic algorithm by evaluations of the interpolant $A_{q,d}(f)$, our surrogate function. The entire procedure is summarized in Algorithm 3.1.

It is well known that the implementation of the extension principle leads to a global optimization problem that must be solved for each considered α-cut, i.e. in Part 2 of the algorithm, by replacing the objective function f with the sparse grid interpolant $A_{q,d}(f)$, one must solve

$$\begin{aligned} c^{(j)} &= \min_{\mathbf{x} \in I_1^{(j)} \times \cdots \times I_d^{(j)}} A_{q,d}(f)(\mathbf{x}), \\ d^{(j)} &= \max_{\mathbf{x} \in I_1^{(j)} \times \cdots \times I_d^{(j)}} A_{q,d}(f)(\mathbf{x}), \end{aligned} \tag{13}$$

where $I_1^{(j)} \times \cdots \times I_d^{(j)}$ are d-dimensional boxes formed by the Cartesian product of the intervals of confidence $I_i^{(j)}$ of each α-cut with index j of the fuzzy input parameters $\tilde{p}_i$, $i = 1, \ldots, d$, $j =$

$0, \ldots, m$. We denote the resulting intervals of confidence forming the discrete representation of the fuzzy result by $J^{(j)} = [c^{(j)}, d^{(j)}]$.

The accuracy of the obtained fuzzy result depends on both parts of the algorithm. The initial contributor to the quality of the fuzzy result is of course the accuracy of the interpolant, which can be easily monitored during its hierarchical construction. The order of convergence in the maximum norm is well known, as stated in Eq. (10). We thus have a solid basis for the second part of the algorithm. The quality of the final fuzzy result, however, can still vary depending

---

Algorithm 3.1 Fuzzy arithmetic on sparse grids

Initialization.

Let $f : \mathbb{R}^d \to \mathbb{R}$ be the objective function.

Let $\tilde{p}_i$, $i = 1, \ldots, d$ be the fuzzy-valued input parameters.

Let Ω be the base box of the fuzzy parameters $\tilde{p}_i$.

Let δ_{rel} be a relative, δ_{abs} an absolute error tolerance.

Let $k_{\max} \in \mathbb{N}$ be the maximum interpolation depth.

Let $w_{\max}^{-1} = \infty,\ y_{\min} = \infty,\ y_{\max} = -\infty$.

Algorithm.

Part 1. Determine the sparse grid surrogate function $A_{q,d}(f)$:

Let $k = 0$.

While $w_{\max}^{k-1} \geq \max[\delta_{\text{rel}} \cdot (y_{\max} - y_{\min}),\ \delta_{\text{abs}}]$ **And** $k \leq k_{\max}$ **Do**

Compute the sparse grid points $\Delta H_{k+d,d}([0,1]^d)$.

Re-scale $\Delta H_{k+d,d}([0,1]^d)$ to $\Delta H_{k+d,d}(\Omega)$.

$Z_k = \{w^k \mid w^k = f(\mathbf{x}) - A_{k+d-1,d}(f)(\mathbf{x}),\ \mathbf{x} \in \Delta H_{k+d,d}(\Omega)\}$.
If $k = 0$, use simply $Z_k = \{w^k \mid w^k = f(\mathbf{x})\}$. Otherwise, use Eq. (3) to compute the values $A_{k+d-1,d}(f)(\mathbf{x})$.

$y_{\max} = \max\big[y_{\max}, f(\mathbf{x})\ \forall\ \mathbf{x} \in \Delta H_{k+d,d}(\Omega)\big]$.

$y_{\min} = \min\big[y_{\min}, f(\mathbf{x})\ \forall\ \mathbf{x} \in \Delta H_{k+d,d}(\Omega)\big]$.

$w_{\max}^k = \max[w^k\ \forall\ w^k \in Z_k]$.

$k = k + 1$.

End

Let $q = k + d - 1$.

Construct $A_{q,d}(f)$ using Eq. (3) and the sets of hierar. surpluses Z_k.

Part 2. Perform constrained fuzzy arithmetic by a suitable method, e.g.

– the transformation method [9],

– direct evaluation of the interpolant for each set of intervals of confidence using a tight inclusion function,

– a direct global search algorithm for each set of intervals of confidence,

– a branch-and-bound algorithm based on interval arithmetic.

Hereby, replace all real-valued or interval-valued function evaluations of f by evaluations of the surrogate function $A_{q,d}(f)$.

Table 1: Comparison: number of grid points for refinement level $n = q - d$.

	full grid $H^{\mathrm{CC}}_{\mathrm{full},n+d,d}$				sparse grid $H^{\mathrm{CC}}_{n+d,d}$			
n	$d=2$	$d=4$	$d=8$	$d=16$	$d=2$	$d=4$	$d=8$	$d=16$
1	9	81	6561	4.3e7	5	9	17	33
2	25	625	41553	1.5e11	13	41	145	545
3	81	6561	4.3e7	1.9e15	29	137	849	6049
4	289	83521	7.0e9	4.9e19	65	401	3937	51137
5	1089	1.2e6	1.4e12	2.0e24	145	1105	15713	3.5e5
6	4225	1.8e7	3.2e14	1.0e29	321	2561	31745	2.1e6
7	16641	2.8e8	7.7e16	5.9e33	705	7537	1.9e5	1.1e7

on the actual fuzzy arithmetic algorithm chosen to implement Zadeh's extension principle. If one uses a method that computes the global minimum and the global maximum of the surrogate function for each α cut exactly, one can eliminate the error resulting from Part 2. On the other hand, if one uses an approximative method to solve the global optimization problem, a second error source is introduced.

3.3 Performing constrained fuzzy arithmetic on the interpolant

In general, one is free to select any suitable algorithm to solve the optimization problem given by Eq. (13). Since the evaluation of $A_{q,d}(f)$ is explicitly given for single points in Eq. (2), constrained fuzzy arithmetic algorithms requiring only real-valued function evaluations are applicable without any further additions or modifications. The transformation method [9, 10], for example, is based on real-valued function evaluations, we can therefore directly apply it by just replacing the function evaluations of f by evaluations of the surrogate function $A_{q,d}(f)$. However, computationally more efficient and usually more accurate are incomplete global search methods specifically designed for sparse grids. We therefore provide a modified coordinate search algorithm in Section 3.3.1. It is also possible to implement complete search algorithms based on interval extensions of $A_{q,d}(f)$, i.e. implement inclusion functions $[A_{q,d}(f)]$ of $A_{q,d}(f)$ that take interval vectors (also called boxes) instead of single points as arguments. By doing so, we can apply powerful branch-and-bound optimization algorithms based on interval arithmetic. This approach will be discussed in Section 3.3.2.

3.3.1 Applying an incomplete global search method

A straightforward way to solve the global optimization problem for each α-cut (13) represent incomplete global search methods. The simplest incomplete search method, for example, is the multiple random start method consisting of picking random start points and performing local optimizations from these points [21]. In the context of this paper, as mentioned before, all real-valued function evaluations should be replaced by evaluations of the interpolant $A_{q,d}(f)$. In general, a drawback of incomplete global search methods is the fact that they cannot

guarantee that the global optimum has been found, since the search might have been aborted prematurely according to the applied stopping criteria. However, in many cases, the objective function does not exhibit multiple local extrema, and the results obtained using incomplete global search are sufficiently accurate.

As an example of an incomplete global search algorithm, we have implemented a variation of a coordinate search algorithm (often also known as compass search) [19]. The smallest search step-length to be used can be limited to the minimum sparse grid step width $\Delta_{\min} = 2^{q-d}$, since the optimum can only lie at the points of the full grid $H_{\text{full},q,d}$, or at the boundary of the search box (see Section 3.3.2).

A standard coordinate search algorithm can only guarantee convergence to a local minimizer. We thus have to modify the method such that with increasing accuracy of the sparse grid interpolant, it becomes globally convergent. From the construction of the interpolant $A_{q,d}(f)$, we have a set of grid points $H_{q,d}$ and corresponding function values available. As the parameter q goes to infinity, the set $H_{q,d}$ forms a dense subset of the unit hypercube, i.e. given any point $\mathbf{x}$ in the hypercube and any $\delta > 0$, a grid point lies within a distance δ of $\mathbf{x}$. $H_{q,d}$ is dense since it contains the regular full grid $X^{i_f} \times \cdots \times X^{i_f}$ with $i_f = \text{floor}(q/d)$, which can be shown by induction. The grid step width is given by $w_{i_f} = 1/(2^{i_f - 1})$. Thus, we have $\delta \leq \sqrt{w_{i_f} d}$. Clearly, $i_f \to \infty$ for $q \to \infty$, and $\delta \to 0$ for $i_f \to \infty$. If we select the optimum over all sparse grid points contained in the search box (which have already been computed in Part 1 of Algorithm 3.1) as the start point of our search algorithm, we will eventually use a start point within the attraction of a global optimizer, and the method will converge to the global optimum for $q \to \infty$. The modified coordinate search algorithm for sparse grids is given in Algorithm 3.2.

3.3.2 Applying global search methods based on interval arithmetic

In contrast to the approach described in Section 3.3.1, we will now formulate methods to perform a *complete* search. Thus, we will solve the global optimization problem given by (13) without introducing an additional possible source of error. To do this, we use the tools provided by interval analysis.

To apply fuzzy arithmetic based on interval arithmetic, we must define an interval inclusion $[A_{q,d}(f)]$ of the sparse grid interpolant. An interval inclusion function takes interval boxes as input arguments, and gives a guaranteed inclusion of the output range of the function as result. Different types of inclusion functions can be implemented with varying computational cost. In the following, we will discuss the minimal inclusion function, the natural inclusion function, and the centered inclusion function. We will also show how to obtain an inclusion of the gradient vector of $A_{q,d}(f)$, since it is required for the centered inclusion function, and it is also often used to perform a monotonicity test in branch-and-bound algorithms.

In case of the minimal inclusion function, we can directly give an algorithm to compute the fuzzy result. Its complexity is similar to the one of the transformation method if $m \simeq 2^{q-d}$. The natural and the centered inclusion function can be obtained with significantly lower computational effort, but at the price of overestimation.

Algorithm 3.2 Coordinate search on piecewise multilinear sparse grids.

Initialization.

Let $\Omega = [a_1^{(0)}, b_1^{(0)}] \times \cdots \times [a_d^{(0)}, b_d^{(0)}]$ be the base box.

Let $A_{q,d}(f)$ be the sparse grid interpolant of $f : \Omega \to \mathbb{R}$, with $q > d$.

Let $H_{q,d}^{\mathrm{CC}}(\Omega)$ be the set of sparse grid points.

Let $\mathbf{I} = [a_1, b_1] \times \cdots \times [a_d, b_d]$ be the search box.

Let $y_{\min} = \min(A_{q,d}(f)(\mathbf{x}) \ \forall \ \mathbf{x} \in H_{q,d}^{\mathrm{CC}} \cap \mathbf{I})$.

Store the grid point $\hat{\mathbf{x}}$ where $A_{q,d}(f)(\hat{\mathbf{x}}) = y_{\min}$.

Let $\Delta = 1/2^{\mathrm{floor}(q/d)}$ be the initial value of the step-length control parameter.

Let $\mathcal{E}_{\oplus}$ be the set of coordinate directions $\{\mathbf{e}_i \mid i = 1, \ldots, d\}$, where $\mathbf{e}_i$ is the ith unit vector in $\mathbb{R}^d$.

Algorithm.

While $\Delta \geq 1/2^{q-d}$

If there exists $\mathbf{e}_i \in \mathcal{E}_{\oplus}$ such that $A_{q,d}(f)(\mathbf{x}) < y_{\min}$, with

$$\mathbf{x} \in \{\mathbf{x}_i^l, \mathbf{x}_i^r\},$$
$$\mathbf{x}_i^l = \begin{cases} \hat{\mathbf{x}} - \Delta \mathbf{e}_i (b_i^{(0)} - a_i^{(0)}) = \mathbf{x}^l, & \mathbf{x}^l \in \mathbf{I}, \\ \hat{\mathbf{x}} + \mathbf{e}_i (a_i - (\hat{\mathbf{x}})_i), & \mathbf{x}^l \notin \mathbf{I}, \end{cases} \quad \textbf{Then}$$
$$\mathbf{x}_i^r = \begin{cases} \hat{\mathbf{x}} + \Delta \mathbf{e}_i (b_i^{(0)} - a_i^{(0)}) = \mathbf{x}^r, & \mathbf{x}^r \in \mathbf{I}, \\ \hat{\mathbf{x}} + \mathbf{e}_i (b_i - (\hat{\mathbf{x}})_i), & \mathbf{x}^r \notin \mathbf{I}, \end{cases}$$

$y_{\min} = \min\left(A_{q,d}(f)(\mathbf{x}) \ \forall \ \mathbf{x} \in \{\mathbf{x}_i^l, \mathbf{x}_i^r \mid i = 1, \ldots, d\}\right)$.

Let $\hat{\mathbf{x}} = \mathbf{x}$ where $A_{q,d}(f)(\mathbf{x}) = y_{\min}$.

Else contract the step-length control parameter: Let $\Delta = \frac{1}{2}\Delta$.

End

The minimal inclusion function. According to the definition by Jaulin et. al. [12], an inclusion function $[f]$ is minimal if for any box $\mathbf{I}$, $[f](\mathbf{I})$ is the smallest box that contains $f(\mathbf{I})$. The minimal inclusion function $[A_{q,d}(f)]^*$ can be computed by sampling all points at the full grid with a mesh width chosen according to the sparse grid interpolation depth, evaluating these points using $A_{q,d}(f)$, and computing the maximum and minimum over all results. This results in a minimal interval, since piecewise multilinear interpolation, and thus, a piecewise monotonic interpolant implies that the maximum and minimum function value over a search box $\mathbf{I}$ can only lie at the full grid points $H_{\mathrm{full},q,d} \cap \mathbf{I}$, or at a point on the boundary of $\mathbf{I}$. We define

$$\begin{aligned} [A_{q,d}(f)]^*(\mathbf{I}) &= [a^*, b^*], \quad \text{with} \\ \mathbf{I} &= I_1 \times \ldots \times I_d, \quad I_i = [a_i, \ b_i], \quad i = 1, \ldots, d, \\ Y^i &= \left(\{x \mid x = a_i^{(0)} + \hat{x}(b_i^{(0)} - a_i^{(0)}), \hat{x} \in X^{q-d+1}\} \cap I_i\right) \cup \{a_i, b_i\}, \\ \hat{H}_{\mathrm{full}} &= Y^1 \times \ldots \times Y^d, \\ a^* &= \min(A_{q,d}(f)(\mathbf{x}) \ \forall \ \mathbf{x} \in \hat{H}_{\mathrm{full}}), \\ b^* &= \max(A_{q,d}(f)(\mathbf{x}) \ \forall \ \mathbf{x} \in \hat{H}_{\mathrm{full}}). \end{aligned} \tag{14}$$

In the above equations, we use $a_i^{(0)}$, $b_i^{(0)}$ as defined in (11) and the sets X as defined in (8). With this simple method, we can compute the minimal resulting interval (i.e. without overestimation) for any interval $\mathbf{I} \subseteq \Omega$, using $n^*_{\text{sp}} = \#\hat{H}_{\text{full}} \leq (2^{q-d}+1)^d$, $q > d$ evaluations of $A_{q,d}(f)$.

To compute the discrete fuzzy result $Q = \{J^{(0)}, \ldots, J^{(m)}\}$ we could proceed by applying the minimal inclusion function $[A_{p,q}(f)]^*$ to each α-cut separately. However, by using the convexity property (12), we eliminate recurring points in $\hat{H}_{\text{full}}$ for subsequent α-cuts, similar to the idea presented in [14], and arrive at Algorithm 3.3.

Algorithm 3.3 Minimal fuzzy inclusion $[A_{q,d}(f)]^*(\tilde{p}_1, \ldots, \tilde{p}_d)$.

Initialization.

Let $A_{q,d}(f)$ be the sparse grid interpolant of f.

Let $m+1$ be the number of α-cuts.

Let $P_i = \{I_i^{(0)}, \ldots, I_i^{(m)}\}$, $i = 1, \ldots, d$ be the discretized fuzzy input parameters, with $I_i^{(j)} = [a_i^{(j)}, b_i^{(j)}]$, $j = 0, \ldots, m$.

Let $Q = \{J^{(0)}, \ldots, J^{(m)}\}$ be the discretized fuzzy result, $J^{(j)} = [c^{(j)}, d^{(j)}]$.

Let $\hat{Y}^i = \emptyset$, $i = 1, \ldots, d$.

Algorithm.

For $j = m$ **DownTo** 0

  For $i = 1$ **To** d

    $\hat{Y}^i_{\text{old}} = \hat{Y}^i$.

    $\hat{Y}^i = \{x \mid x = a_i^{(0)} + \hat{x}(b_i^{(0)} - a_i^{(0)}), \hat{x} \in X^{q-d+1}\} \cap I_i$.

    $Y^i = \hat{Y}^i \cup \{a_i^{(j)}, b_i^{(j)}\}$.

  End

  $\Delta\hat{H}^j_{\text{full}} = Y^1 \times \ldots \times Y^d \setminus \hat{Y}^1_{\text{old}} \times \ldots \times \hat{Y}^d_{\text{old}}$.

  $c^{(j)} = \min(A_{q,d}(f)(\mathbf{x}) \;\forall\; \mathbf{x} \in \Delta\hat{H}_{\text{full}})$. $\quad d^{(j)} = \max(A_{q,d}(f)(\mathbf{x}) \;\forall\; \mathbf{x} \in \Delta\hat{H}_{\text{full}})$.

  If $j < m$ **Then** $c^{(j)} = \min(c^{(j+1)}, c^{(j)})$. $\quad d^{(j)} = \max(d^{(j+1)}, d^{(j)})$.

End

The natural inclusion function. The natural inclusion function $[A_{q,d}(f)]_n$ is immediately obtained by replacing each real variable by an interval variable and each real operator by its interval counterpart (for various examples, see [12, pp. 29–32]). Thus, by replacing x with the interval variable I, the natural inclusion function of the basis functions a_i^j becomes

$$[a_1^1]_n(I) = 1 \quad \text{for } i = 1, \text{ and}$$

$$[a_j^i]_n(I) = \begin{cases} 1 - (m_i - 1) \cdot \left| I - x_j^i \right|, & \text{if } \left| I - x_j^i \right| < 1/(m_i - 1), \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

for $i > 1$ and $j = 1, \ldots, m_i$. We have to make a minor adjustment to the above equation for $i > 1$ and $1/(m_i - 1) \in \left| I - x_j^i \right|$, since in this case, the interval boolean operator $<$

does not give a unique result of true (1) or false (0) (see [12, p. 41] for the extension of boolean operators to intervals). Taking this special case into account, and defining $I = [a, b]$, $r = 1/(m_i - 1)$, we arrive at

$$[c,d] := \left|I - x_j^i\right| = \begin{cases} \left[a - x_j^i, b - x_j^i\right], & \text{if } a - x_j^i \geq 0, \\ \left[-b + x_j^i, -a + x_j^i\right], & \text{if } b - x_j^i \leq 0, \\ \left[0,\ \max\left(\left|a - x_j^i\right|, \left|b - x_j^i\right|\right)\right], & \text{otherwise,} \end{cases}$$

$$[a_j^i]_n(I) = \begin{cases} 1 - r \cdot [c,d], & \text{if } d < 1/r, \\ 0, & \text{if } c \geq 1/r, \\ [0, 1 - cr], & \text{otherwise.} \end{cases}$$

Note that the natural inclusion $[a_j^i]_n$ of the basis functions a_j^i is minimal (see Fig 2 (**a**)). By replacing the real basis functions in the Smolyak tensor product formula (2) by its interval-valued counterparts, we obtain the natural inclusion function $[A_{q,d}(f)]_n$(f) with

$$[\Delta A_{k+d,d}]_n(f) = \sum_{|\mathbf{i}|=k+d} \sum_{\mathbf{j}} ([a_{j_1}^{i_1}]_n \otimes \cdots \otimes [a_{j_d}^{i_d})]_n) \cdot \left(f(\mathbf{x_j^i}) - A_{k+d-1,d}(f)(\mathbf{x_j^i})\right) \tag{16}$$

from (4). Due to the non-overlapping support of the piecewise multilinear tensor product basis functions for equal $\mathbf{i}$ in Eq. (16), we can further improve this natural inclusion function by replacing the inner sum by the interval hull operator $\sqcup$ [12, p. 18]. We call this inclusion function the reduced natural inclusion function $[A_{q,d}]_{n,\mathrm{red}}(f)$ with

$$\begin{aligned}[\Delta A_{k+d,d}]_{n,\mathrm{red}}(f) = \sum_{|\mathbf{i}|=k+d} \bigsqcup_{\mathbf{j}} \Big[&([a_{j_1}^{i_1}]_n \otimes \cdots \otimes [a_{j_d}^{i_d})]_n) \\ &\cdot \left(f(\mathbf{x_j^i}) - A_{k+d-1,d}(f)(\mathbf{x_j^i})\right)\Big].\end{aligned}$$

In comparison to the minimal inclusion function $[A_{q,d}(f)]^*$, computing $[A_{q,d}(f)]_n(\mathbf{I})$ and $[A_{q,d}(f)]_{n,\mathrm{red}}(\mathbf{I})$ requires only a single interval-valued evaluation composed of at most [25, 29]

$$n_{\mathrm{sp},n} = \#H_{q,d}^{\mathrm{CC}} \leq 2^{q-d+1} \binom{q-1}{d-1} \tag{17}$$

inclusion functions of the multidimensional basis functions $([a_{j_1}^{i_1}]_n \otimes \cdots \otimes [a_{j_d}^{i_d}]_n)$. Thus, the natural inclusion functions are much cheaper to obtain than $[A_{q,d}(f)]^*$, but at the cost of overestimation.

The centered inclusion function. One can also construct higher order inclusion functions, such as the mixed centered inclusion function [8, 12]

$$[A_{q,d}(f)]_c(\mathbf{I}) = A_{q,d}(f)(\mathbf{c}) + \sum_{l=1}^{d} [g(A_{q,d}(f))_l](I_1, \ldots, I_l, c_{l+1}, \ldots, c_d)(I_l - c_l),$$

where $[g(A_{q,d}(f))_l]$ is an inclusion function of the lth component of the gradient vector of $A_{q,d}(f)$, and $\mathbf{c} = (c_1, \ldots, c_d)$ is the midpoint of the interval box $\mathbf{I}$. One cannot expect a

tighter interval inclusion of the basis function itself, regardless of the input box size, since the inclusions of the basis functions are already minimal for the natural inclusion function. In fact, the centered inclusion of the basis function even produces a wider result if the peak of the basis function is contained in I (see Fig. 2 (**b**)). However, once the hierarchical structure of the interpolant is considered, the additional derivative information often translates into tighter interval bounds with decreasing box size.

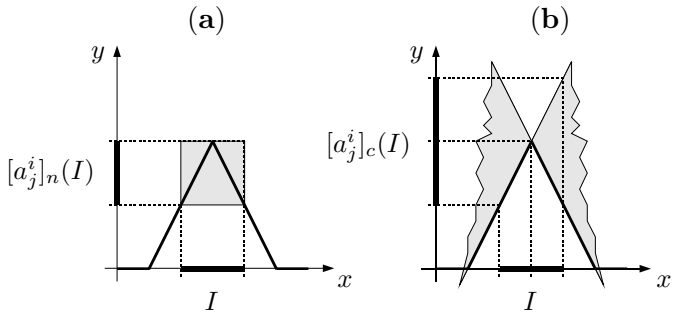


Figure 2: Comparison of natural (**a**) and centered **b**) inclusion function for an interval with $x_j^i \in I$.

The hierarchical, additive structure of $A_{q,d}(f)$ permits an interesting additional inclusion function constructed from the mixed centered and the natural inclusion function, which we call the *nc* inclusion function $[A_{q,d}(f)]_{nc}$. In a nutshell, we use the centered inclusion function, but replace the contribution $[\Delta A_{k,d}(f)]_c$ at a level k by its natural counterpart $[\Delta A_{k,d}(f)]_{n,\text{red}}$ if the resulting interval is smaller. As a result, the combined inclusion function *nc* gives an improved inclusion compared to both the natural inclusion function and the centered inclusion function.

Computing the inclusion of the gradient vector. In addition to using an inclusion of the gradient vector to compute the centered inclusion function, one can also conduct a monotonicity test based on an interval-valued evaluation of the gradient to eliminate boxes, or at least reduce the dimension of boxes. This test is commonly performed in branch-and-bound algorithms [8] of global optimization to reduce the computational effort. We therefore show how to obtain an inclusion of the (weak) partial derivatives of the interpolant (weak, since the piecewise multilinear interpolant is only continuous, but not continuously differentiable).

Similar to the construction of the previous inclusion functions, we need the derivatives of the basis functions a_j^i to construct the gradient vector. Computing the natural interval inclusion of the derivative $(a_j^i)'$ yields

$$[a_1^1]'_n(I) = 0 \quad \text{for } i = 1, \text{ and}$$

$$[a_j^i]'_n(I) = \begin{cases} -(m_i - 1) \cdot [u]'_n(I), & \text{if } \left| I - x_j^i \right| < 1/(m_i - 1), \\ 0, & \text{otherwise,} \end{cases} \tag{18}$$

with $u'(x) = \frac{\mathrm{d}|x - x_j^i|}{\mathrm{d}x}$. To resolve the ambiguous inequality condition $\left| I - x_j^i \right| < 1/(m_i - 1)$, and to correctly compute the interval-valued inclusion of the derivatives, we have to consider seven cases, which are depicted in Figure 3.

case	conditions	$[a_j^i]'(I)$
1	$b < x_j^i - r$	$[0, 0]$
2	$a < x_j^i + r$	$[0, 0]$
3	$a > x_j^i - r$ and $b < x_j^i$	$[s, s]$
4	$a > x_j^i$ and $b < x_j^i + r$	$[-s, -s]$
5	$a \leq x_j^i$ and $b \geq x_j^i$	$[-s, s]$
6	$a \leq x_j^i - r$ and $b \geq x_j^i - r$ and $b < x_j^i$	$[0, s]$
7	$b \geq x_j^i + r$ and $a \leq x_j^i + r$ and $a > x_j^i$	$[-s, 0]$

with $s = m_i - 1$.

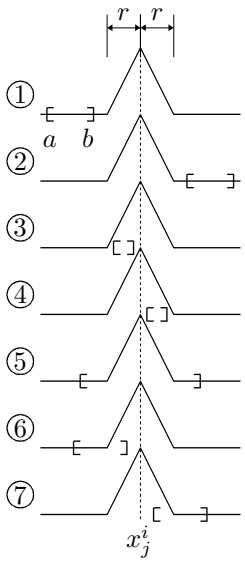


Figure 3: Basis function shape and different resulting derivatives for different positions of the interval $I = [a, b]$ w.r.t. the peak x_j^i and the support with $r = 1/(m_i - 1)$.

The partial derivatives of (4) are obtained to

$$\frac{\partial(\Delta A_{k+d,d}(f))}{\partial x_l} = \sum_{|\mathbf{i}|=k+d} \sum_{\mathbf{j}} \Big[(a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_l-1}^{i_l-1} \otimes (a_{j_l}^{i_l})' \otimes a_{j_l+1}^{i_l+1} \otimes \cdots \otimes a_{j_d}^{i_d}) \cdot \Big(f(\mathbf{x}_\mathbf{j}^\mathbf{i}) - A_{k+d-1,d}(f)(\mathbf{x}_\mathbf{j}^\mathbf{i}) \Big) \Big], \quad (19)$$

$l = 1, \ldots, d$. Thus, we can compute the partial derivatives of $A_{q,d}(f)$ composing the gradient vector $g(A_{q,d}(f))$ by applying the Smolyak formula (2), replacing the $\Delta A_{k+d,d}(f)$ by (19). Analogously, we get its natural interval inclusion $[g(A_{q,d}(f))]_n$ by replacing the real basis functions by its interval counterparts, as defined in (15) and (18).

In terms of computational complexity, each component of the gradient vector is composed of at most $\#H_{q,d}^{\mathrm{CC}}$ partial derivatives of the multidimensional basis functions $(a_{j_1}^{i_1} \otimes \cdots \otimes a_{j_l-1}^{i_l-1} \otimes (a_{j_l}^{i_l})' \otimes a_{j_l+1}^{i_l+1} \otimes \cdots \otimes a_{j_d}^{i_d})$. The complete gradient vector can thus be obtained at a cost proportional to $n_{\mathrm{sp},g} = d \cdot n_{\mathrm{sp},n}$.

A branch-and-bound method. Branch-and-bound algorithms split a problem recursively into subproblems. A subproblem is eliminated as soon as it is shown that it cannot contain the solution [21]. Interval branch-and-bound methods for global optimization usually start with an initial box, and subdivide it repeatedly into sub-boxes. Sub-boxes that are guaranteed not to contain a global minimizer are removed while the remaining boxes are split further until a specified accuracy is achieved. To determine whether a box does not contain a global minimum, various tests are applied that utilize interval inclusion functions.

With the previously defined interval inclusion functions, we can use an interval branch-and-bound method (Algorithm 3.4) to solve the optimization problem involving the sparse grid interpolant, as stated in Eq. (13). For simplicity, we restrict the domain of f to the unit

cube $[0,1]^d$. We use two tests to eliminate boxes, a cut-off test and a monotonicity test. An important difference compared to standard interval branch-and-bound algorithms lies in the bisection rule. We denote the width of an interval $I = [a, b]$ by $w(I) = b - a$, and the width of a box $\mathbf{I} = I_1 \times \cdots \times I_d$ by $w(\mathbf{I}) = \max_{i=1}^{d} w(I_i)$. Since the minimum function value can only be attained at the full grid points $H_{\text{full},q,d} \cap \mathbf{I}$ or at a boundary point of the box, we bisect along the coordinate direction i with $w(I_i) = w(\mathbf{I})$ such that the boundaries along the cut are shifted to the nearest hyperplanes of grid points towards the inside of the two new boxes. This results in the boxes being disjoint by a gap of $\epsilon = 1/2^{q-d}$ (in case of the domain of f being the unit cube), it also guarantees that the boxes will eventually degenerate to points lying on full grid or the boundary and become removed.

By applying Algorithm 3.4 repeatedly to the boxes formed by the intervals of confidence of subsequent α-cuts of the fuzzy input parameters, and by computing the maxima as well, we can compute the exact range of the interpolant. Thus, we obtain an approximation of the discrete representation of the fuzzy result with no error source but the interpolation error. If one proceeds from the highest α-cut down to the base box, one can easily add another test

---

Algorithm 3.4 Interval branch-and-bound on sparse grids.

Initialization.

Let $A_{q,d}(f)$ be the sparse grid interpolant of $f : [0,1]^d \to \mathbb{R}$.

Let $[A_{q,d}(f)]$ be an interval inclusion function of $A_{q,d}(f)$.

Let $[g_i(A_{q,d}(f))]$ be an interval inclusion of the ith component of the gradient vector of $A_{q,d}(f)$.

Let $\mathbf{I} = [a_1, b_1] \times \cdots \times [a_d, b_d]$ be the initial box.

Let $y_{\min} = \min(A_{q,d}(f)(\mathbf{x}) \ \forall \ \mathbf{x} \in H_{q,d}^{\text{CC}} \cap \mathbf{I})$.

Let $\mathcal{L} = \{\mathbf{I}\}$ be a list.

Algorithm.

While $\mathcal{L} \neq \emptyset$

Pop first box out of $\mathcal{L}$ into $\mathbf{I}$.

$[\underline{y}, \overline{y}] = [A_{q,d}(f)](\mathbf{I})$.

If $\underline{y} \geq y_{\min}$ **Then Continue**. {cut-off test}

For $i = 1$ **To** d {inside the loop: monotonicity test}

$[\underline{\mathrm{d}y}, \overline{\mathrm{d}y}] = [g_i(A_{q,d}(f))](\mathbf{I})$.

If $\underline{\mathrm{d}y} \geq 0$ **Then** $(\mathbf{I})_i = [a_i, a_i]$.

If $\overline{\mathrm{d}y} \leq 0$ **Then** $(\mathbf{I})_i = [b_i, b_i]$.

End

If $w(\mathbf{I}) = \mathbf{0}$ **Then** $\hat{\mathbf{x}} = (a_1, \ldots, a_d)$. {remove degenerate boxes}

Else

$\hat{\mathbf{x}} = ((a_1 + b_1)/2, \ldots, (a_d + b_d)/2)$.

Bisect $\mathbf{I}$ into $\mathbf{I}_1$ and $\mathbf{I}_2$ and put $\mathbf{I}_1$ and $\mathbf{I}_2$ into $\mathcal{L}$.

$y_{\min} = \min(y_{\min}, A_{q,d}(f)(\hat{\mathbf{x}}))$.

End

---

to the optimization algorithm removing boxes as soon as they lie entirely in the box formed by the intervals of the next-higher α-cut.

4 Numerical results

In Section 4.1, we will briefly take a look at the performance of the sparse grid optimization algorithms introduced in Sections 3.3.1–3.3.2, which may be used for Part 2 of Algorithm 3.1. To allow a comparison of the different optimization algorithms, we express the computing times in standard time units. All numerical tests were carried out using a MATLAB V6.5 implementation of the algorithms, running on a Linux i686 1.6 GHz PC. The standard time unit, 1000 real evaluations of the Shekel 5 function at the point $(4, 4, 4, 4)$, was 0.11 sec. on this platform.

In the second subsection, we will illustrate the overall performance of our new sparse grid-based fuzzy arithmetic algorithm for test functions with different distinct properties. We assess the performance by relating the relative error of the fuzzy result to the number of required function evaluations. For a definition of the relative error of a fuzzy number, see [14]. For comparison, we have included the results obtained with the general transformation method [9] and the general transformation method considering recurring permutations [15].

Finally, in Section 4.3, we give examples for objective functions where the proposed method cannot be applied successfully. These cases are classified into three categories.

4.1 Performance of the optimization algorithms

The aim of this section is to provide some information on the performance of the global optimization algorithms discussed in Section 3.3. It is important to efficiently perform the optimization part (Eq. 13) of the proposed fuzzy arithmetic algorithm, since it may affect the overall performance, depending on the cost of a function evaluation (the cheaper the function evaluation, the more weighs the optimization part, especially with increasing problem dimension). The algorithms should also be reliable, i.e. compute the correct or at least nearly correct global minimum and maximum of $A_{q,d}(f)$ for each considered α-cut, and thus, compute the best-possible approximation of the fuzzy result. Recall that we can chose an arbitrarily high number of α-cuts without affecting the number evaluations of the objective function f, since the process of obtaining the interpolant is decoupled from the optimization process, i.e. only evaluations of the interpolant $A_{q,d}(f)$ are performed during the optimization process.

We decided to use some well-known test problems [3] to analyze the proposed optimization algorithms, namely Branin's function

$$f_{\mathrm{BR}}(\mathbf{x}) = \left(\frac{5}{\pi}x_1 - \frac{5.1}{4\pi^2}x_1^2 + x_2 - 6\right) + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10,$$

with $f_{\mathrm{BR}} : [-5, 10] \times [0, 15] \to \mathbb{R}$, the six-hump camel-back function

$$f_{\mathrm{C6}}(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4,$$

with $f_{\mathrm{C6}} : [-2,2]^2 \to \mathbb{R}$, Schwefel's function

$$f_{\mathrm{SW}}(\mathbf{x}) = \sum_{i=1}^{3} \left((x_1 - x_i^2)^2 + (x_i - 1)^2 \right),$$

with $f_{\mathrm{SW}} : [-10,10]^3 \to \mathbb{R}$, and Hartman's functions

$$f_{\mathrm{H}d}(\mathbf{x}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{d} (A)_{ij} (x_j - (P)_{ij})^2 \right),$$

where $d = 3$ or $d = 6$, $A, P \in \mathbb{R}^{4\times d}$ and $c \in \mathbb{R}^d$, with $f_{\mathrm{H}d} : [0,1]^d \to \mathbb{R}$.

We have computed both the minimum and the maximum value of the interpolant for a considered interval box Ω, i.e. the range $R(A_{q,d}(f), \Omega)$. As the refinement parameter q increases, the range of the interpolant approaches the actual range $R(f, \Omega) = [f_{\min}, f_{\max}]$ of the objective function f. We have underlined the correct digits to illustrate the convergence. Additionally, we have measured the approximation quality of the range $R(A_{q,d}(f), \Omega)$ by the relative distance

$$\Delta_{\mathrm{rel}} = \frac{|A_{q,d,\min}(f) - f_{\min}| + |A_{q,d,\max}(f) - f_{\max}|}{f_{\max} - f_{\min}}. \tag{20}$$

Since the modified coordinate search algorithm (CS, Algorithm 3.2) is an incomplete search method only, it might underestimate the actual range of $A_{q,d}(f)$. Therefore, we also provide the relative distance $\Delta_{\mathrm{rel}}^{\mathrm{CS}}$ of the range computed with CS denoted by $R^{\mathrm{CS}}(A_{q,d}(f), \Omega) = [A_{q,d,\min}^{\mathrm{CS}}(f), A_{q,d,\max}^{\mathrm{CS}}(f)]$ from the exact range $R(A_{q,d}(f), \Omega)$ according to

$$\Delta_{\mathrm{rel}}^{\mathrm{CS}} = \frac{\left| A_{q,d,\min}^{\mathrm{CS}}(f) - A_{q,d,\min}(f) \right| + \left| A_{q,d,\max}^{\mathrm{CS}}(f) - A_{q,d,\max}(f) \right|}{A_{q,d,\max}(f) - A_{q,d,\min}(f)}. \tag{21}$$

Table 2 presents the results. The coordinate search algorithm is by far the fastest of the proposed global optimization algorithms, but may not give the exact global optimum. However, the distances $\Delta_{\mathrm{rel}}^{\mathrm{CS}}$ indicate that the error is often zero, or otherwise of similar order as the distance Δ_{rel} resulting from the interpolation error. The minimal inclusion function (MI, Eq. (14)) represents a complete search algorithm considering all full grid points, but is considerably more expensive for large d, q. The interval branch-and-bound method (BB, Algorithm 3.4) is well suited for higher values of q and low d. We suggest to use the combined inclusion function as discussed in Section 3.3.2, which provides a tighter inclusion than the mere natural inclusion regardless of the input interval width, and also exhibits the higher-order convergence of a centered inclusion. Figure 4 gives a visual impression of the performance of the branch-and-bound algorithm for different inclusion functions for the minimization of the interpolant $A_{9,2}(f_{\mathrm{BR}})$ of Branin's function.

Unfortunately, interval-based methods seem unsuited for solving difficult sparse grid optimization problems of higher dimension, since the number of sub-boxes to search may still be large due to the pessimism of the inclusion functions and/or the objective function characteristics (e.g. many regions potentially containing the global optimum). In most cases, using the modified coordinate search algorithm is therefore the best choice.

Table 2: Accuracy Δ_{rel} of range R and standard time units of the optimization algorithms.

n	$R(A_{n+d,d}(f_{\mathrm{BR}}), \Omega_{\mathrm{BR}})$	Δ_{rel}	$\Delta_{\mathrm{rel}}^{\mathrm{CS}}$	t_{CS}	t_{MI}	t_{BB}
2	[2.5012, 308.13]	7e-3	0	0.1	0.02	1.0
4	[0.40148, 308.13]	1e-5	6e-3	0.2	0.06	3.4
6	[0.40148, 308.13]	1e-5	0	0.2	1	9.9
8	[0.39846, 308.13]	2e-6	3e-6	0.4	27	24
10	[0.39767, 308.13]	7e-7	0	0.7	676	50
n	$R(A_{n+d,d}(f_{\mathrm{C6}}), [-2,2]^2)$	Δ_{rel}	$\Delta_{\mathrm{rel}}^{\mathrm{CS}}$	t_{CS}	t_{MI}	t_{BB}
2	[0.0000, 55.733]	0.02	0	0.1	0.02	0.8
4	[−0.9844, 55.733]	8e-4	0	0.2	0.06	3.8
6	[−1.0244, 55.733]	1e-4	0	0.4	1.0	11
8	[−1.0313, 55.733]	6e-6	0	0.5	28	22
10	[−1.0316, 55.733]	4e-7	0	0.8	680	44
n	$R(A_{n+d,d}(f_{\mathrm{SW}}), [-10,10]^3)$	Δ_{rel}	$\Delta_{\mathrm{rel}}^{\mathrm{CS}}$	t_{CS}	t_{MI}	t_{BB}
2	[3.0000, 36663]	8e-5	0	0.3	0.04	1.3
4	[−7.3320, 36663]	2e-4	3e-4	0.2	1.5	20
6	[−0.3442, 36663]	9e-6	0	0.6	248	188
8	[−0.0233, 36663]	6e-7	0	1.2	–	878
10	[−0.0014, 36663]	4e-8	0	2.7	–	3158
n	$R(A_{n+d,d}(f_{\mathrm{H3}}), [0,1]^3)$	Δ_{rel}	$\Delta_{\mathrm{rel}}^{\mathrm{CS}}$	t_{CS}	t_{MI}	t_{BB}
2	[−3.6219, 0.8763]	0.3	0	0.2	0.04	1.6
4	[−3.7059, 0.2333]	0.1	0	0.5	1.5	44
6	[−3.7955, 0.0291]	0.02	4e-3	0.8	247	940
8	[−3.8551, 0.0006]*	2e-3	–	1.1	–	–
10	[−3.8619, 0.0000]*	2e-4	–	2.3	–	–
n	$R(A_{n+d,d}(f_{\mathrm{H6}}), [0,1]^6)$	Δ_{rel}	$\Delta_{\mathrm{rel}}^{\mathrm{CS}}$	t_{CS}	t_{MI}	t_{BB}
2	[−3.6367, 1.6699]	0.6	0.03	0.4	4.8	27
4	[−2.5543, 0.4121]*	0.4	–	1.4	–	–
6	[−2.9036, 0.0476]*	0.1	–	4.6	–	–
8	[−3.1179, 0.0038]*	0.06	–	25	–	–
10	[−3.1835, 0.0000]*	0.04	–	123	–	–

*: $R^{\mathrm{CS}}(A_{n+d,d}, \Omega)$ −: not considered/available

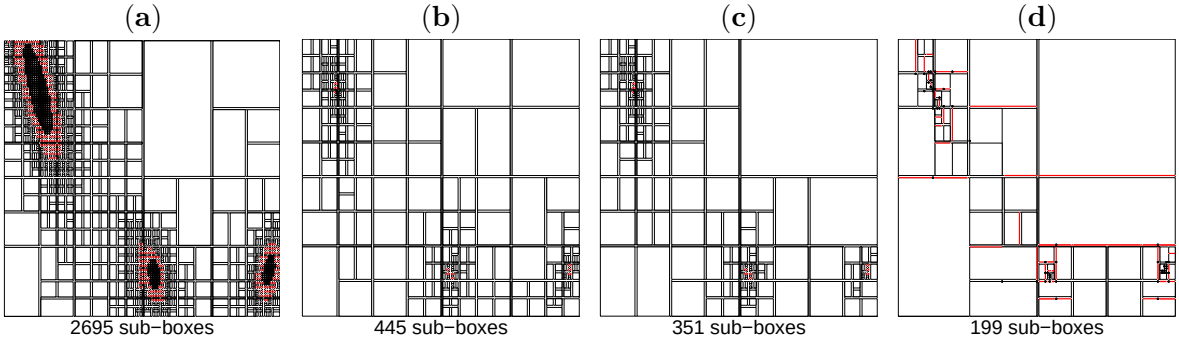


Figure 4: Comparison of pavings obtained by the branch-and-bound optimization algorithm for the Branin function with different inclusion functions. (**a**): natural, (**b**): centered, (**c**): *nc* (all three without the application of the monotonicity test based on the inclusion of the gradient), (**d**): *nc* including the monotonicity test. Boxes degenerated to a line are marked in red color. Boxes degenerated to a single point are indicated by a black dot.

4.2 Overall performance results

We now take a look at the overall performance of the proposed fuzzy arithmetic algorithm based on sparse grids. The test functions we have used are listed in Table 3. We have taken symmetric triangular fuzzy numbers as input parameters for all test functions. However, the method is not restricted to this type of shape function, any convex fuzzy set, i.e. any fuzzy number or fuzzy interval could be used. Also provided are the base box of the fuzzy input parameters taken for each test function, i.e. the input fuzzy-valued inputs of f_1 are given by the triangular fuzzy numbers $\tilde{p}_1 = \langle 0, 2.5, 5\rangle_{\text{TFN}}$ and $\tilde{p}_2 = \langle 1, 3, 5\rangle_{\text{TFN}}$ since $\Omega_1 = [0, 5] \times [1, 5]$. The functions f_1–f_3 are taken from [14], the functions f_4–f_7 are taken from [1], f_8 from [3]. Figure 5 shows the fuzzy-valued outputs of the test functions. We would like to add that the test functions presented here could obviously be treated in a much simpler way than within the framework of sparse grids, since the functions are inexpensive to evaluate. But we have intentionally kept the numerical examples simple for mainly two reasons: (1) the exact result can be easily obtained, and (2) the behavior of the method with different characteristics of the objective function becomes apparent. For more complex applications of the new method, the interested reader may refer to [17] dealing with the simulation of dynamic systems under uncertainty.

Figure 6 shows convergence plots for the fuzzy-valued results of the seven test functions. One can observe the excellent asymptotic error decay according to Eq. (10) with increasing number of function evaluations for the smooth test functions of two variables f_1 to f_3 (Fig. 6 (**a**)-(**c**)), which all satisfy the smoothness properties defined in Eq. (9). As the dimension of the test functions increases, the efficiency of the sparse grid scheme becomes even more apparent (Fig. 6 (**d**),(**e**)), especially compared to a method sampling points at the full grid, such as the transformation method.

In the numerical examples considered so far, the bases of the fuzzy numbers have been rather large considering the fact that the fuzzy numbers usually represent uncertainties. If we look at the bases of the input parameters taken for the function f_4, for example, the width of the base corresponds to an uncertainty range of $\pm 100\%$ of the peak value 0.5. The input parameters

Table 3: Test functions.

i	*function* $f_i(\mathbf{x}) : \Omega_i \to \mathbb{R}$	*base box* Ω_i	*characteristics*
1	$\cos(\pi x_1)x_2$	$[0,5] \times [1,5]$	Non-monotonic w.r.t. x_1
2	$\big[(x_1-2)^4 + (x_2-2)^2 + 0.2\big]^{-1}$	$[0,5] \times [1,5]$	One local extremum
3	$3(1-x_1)^2 \exp[-x_1^2 - (x_2+1)^2]$ $-10(\frac{x_1}{5} - x_1^3 - x_2^5) \exp[-x_1^2$ $-x_2^2] - \frac{1}{3}\exp[-(x_1+1)^2 - x_2^2]$	$[-3,3] \times [-2,2]$	Multiple local optima
4	$\prod_{i=1}^{4} \big(c_i^{-1} + (x_i - w_i)^2\big)^{-1}$	$[0,1]^4$	Product peak (Genz)
5	$\cos\big(2\pi w_1 + \sum_{i=1}^{d} c_i x_i\big)$	$[0,1]^8$	Oscillatory (Genz)
6	$\exp\big(-\sum_{i=1}^{d} c_i \cdot \lvert x_i - w_i\rvert\big)$	$[0,1]^2$	Continuous (Genz)
7	$\begin{cases} 0, & \text{if } x_1 > w_1 \\ & \text{or } x_2 > w_2, \\ \exp\big(\sum_{i=1}^{d} c_i x_i\big), & \text{otherwise} \end{cases}$	$[0,1]^2$	Discontinuous (Genz)
8	$-\sum_{i=1}^{5} \frac{1}{(\mathbf{x}-(A)_i)(\mathbf{x}-(A)_i)^T + c_i}$, where $A \in \mathbb{R}^{5\times 3}$, $c \in \mathbb{R}^5$	$[0,10]^4$	5 sharp peaks (Shekel)

The constants for the test functions of Genz (f_4 to f_7) have been taken to:
f_4: $c = (0.8, 0.5, 1.3, 1.4)$, $w = (0.2, 0.4, 0.3, 0.1)$,
f_5: $c = (0.4, 0.2, 0.3, 0.2, 0.6, 0.2, 0.1, 0.2)$, $w_1 = 0.2$,
f_6: $c = (8, 12.4)$, $w = (0.3, 0.4)$, f_7: $c = (2.1, 2.2)$, $w = (0.3, 0.7)$.

taken for the other test functions exhibit similar ranges of uncertainty. Uncertainties of this magnitude are rather uncommon in engineering applications. Instead, much smaller uncertainty ranges are of practical interest, say 5–20% or less. Clearly, with decreasing range of uncertainty, and thus, decreasing size of the base box, a smaller part of the function needs to be approximated by the sparse grid interpolant. Furthermore, as the uncertainty ranges become smaller, the characteristics of the function will become increasingly linear within the according box, meaning that increasingly less sparse grid support nodes are required to accurately approximate the objective function. In other words, the proposed method will perform even better for small ranges of uncertainty.

To illustrate this claim, we have again used the test functions from Table 3, but this time, we have varied the fuzzy input parameters regarding the degree of uncertainty and the membership function shape. Apart from symmetric triangular membership functions, we have used quasi-Gaussian [9] membership functions. For both membership function types, we left the peak value of the fuzzy numbers unchanged while setting the width of the bases to 100%, 20%, and 5%, respectively, of its original size according to Table 3. The quasi-Gaussian

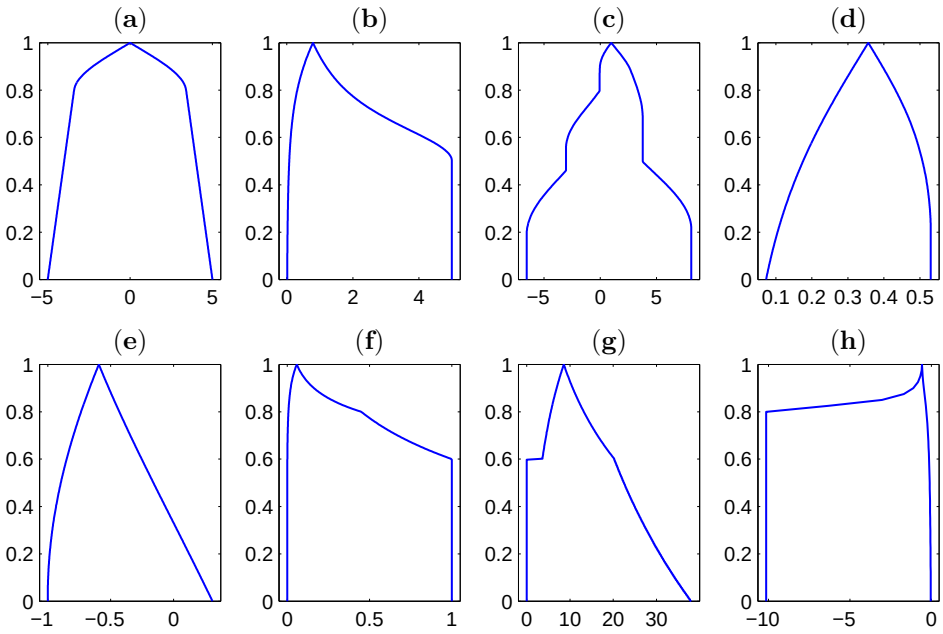


Figure 5: (**a**)–(**h**): Fuzzy-valued results of f_1–f_8 (from left to right, top to bottom) for the given fuzzy-valued input parameters of symmetric triangular shape.

membership functions are given by

$$\mu(x) = \begin{cases} \exp\left(-\frac{(x-\overline{m})^2}{2\sigma^2}\right), & \text{if } |x-\overline{m}| \leq 3\sigma, \\ 0, & \text{otherwise,} \end{cases} \tag{22}$$

where we chose the standard deviation σ and the mean value $\overline{m}$ such that the base and the peak value of the resulting fuzzy number remained the same as for the according symmetric triangular fuzzy number. Table 4 summarizes the results. We give the number of function evaluations required for the fuzzy-valued result to reach a relative error below $e_{\text{tol}} = 10^{-2}$. With decreasing uncertainty, much less function evaluations are needed to attain the desired accuracy, regardless of the shape of the membership function.

Table 4: Number of function evaluations N required to reach a relative error $e_{\text{tol}} < 10^{-2}$. $e \cdot 10^{-3}$ is the actually achieved relative error.

	symmetric triangular					
	100%		20%		5%	
	N	e	N	e	N	e
f_1	321	4.9	65	4.0	13	3.4
f_2	705	7.3	65	5.4	13	8.8
f_3	1537	4.2	65	6.8	13	7.2
f_4	1105	4.6	41	8.3	41	1.8
f_5	849	4.4	145	1.8	17	8.7

	symmetric quasi-Gaussian					
	100%		20%		5%	
	N	e	N	e	N	e
f_1	321	5.3	65	3.9	13	3.6
f_2	1537	4.1	65	6.5	13	1.3
f_3	1537	5.8	65	8.4	13	7.2
f_4	1105	6.7	137	4.8	41	4.7
f_5	849	5.0	145	4.0	17	7.9

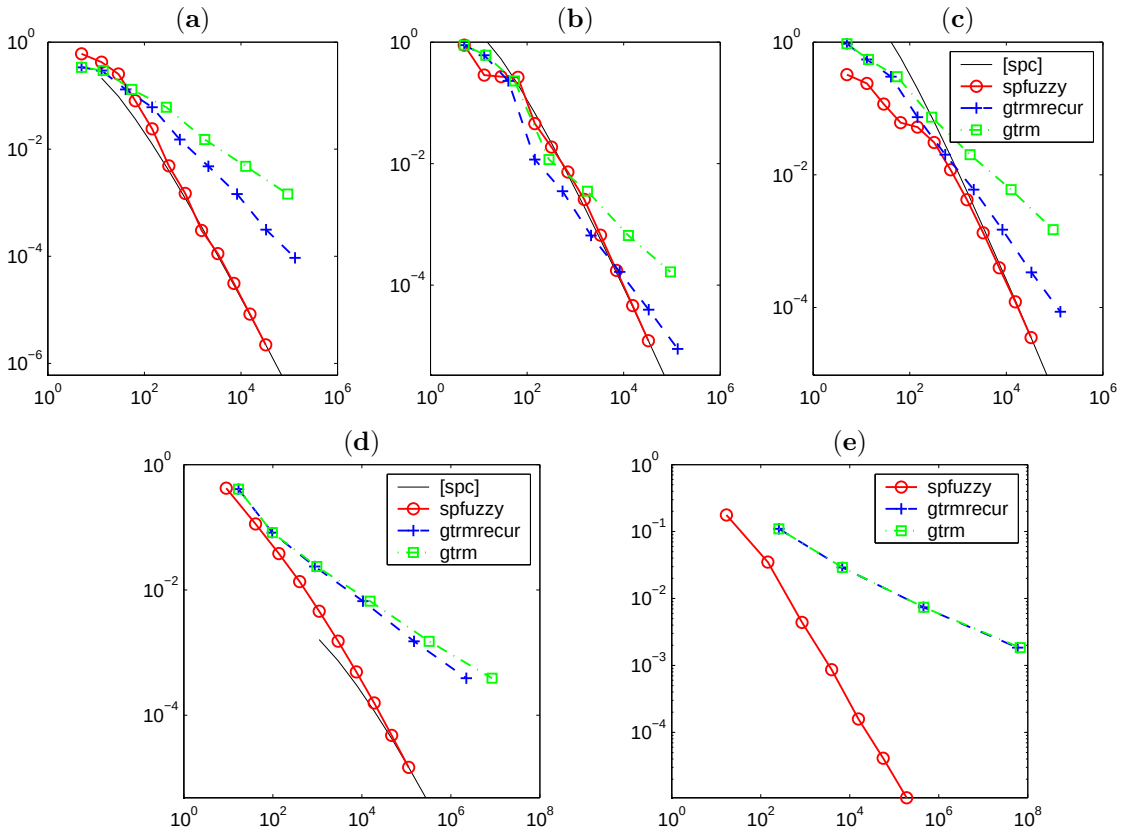


Figure 6: (**a**)–(**e**): Relative error of the fuzzy-valued results of f_1–f_5 w.r.t. the number of real-valued function evaluations. *[spc]* indicates the expected convergence rate according to Eq. (10); the position of the curve is chosen for reasons of clarity. *spfuzzy* depicts the curve obtained for the proposed algorithm, *gtrm* represents the general transformation method, and *gtrmrecur* is the general transformation method without recurring permutations.

4.3 Limitations

There are some cases when the proposed sparse grid approach cannot be applied successfully. These cases can be categorized as follows.

1. Non-smooth objective functions $f \notin F$ (F according to Eq. (9)). For the continuous function f_6, the method only converges slowly (Fig. 7 (**a**)). As expected, the method fails completely for the discontinuous test function f_7, since a discontinuous function cannot be accurately approximated by a continuous interpolant (Fig. 7 (**b**)).

2. Smooth objective functions $f : \Omega \to \mathbb{R}$, $f \in F$ with strong local features within the considered box Ω, e.g. a function with sharp peaks. In this case, many support nodes are required until the local features are sufficiently approximated and the high order asymptotic convergence rate of sparse grid interpolation arises. An example of such a function is the function f_8 as shown in Fig. 7 (**c**). The transformation method performs quite satisfactory in this particular case, since the locations of the internal extrema match the sampled full grid points of the transformation method.

3. Smooth objective functions $f : \Omega \to \mathbb{R}$, $f \in F$ with strong oscillatory behavior within the objective box Ω. Again, a high number of support nodes is required until the oscillation is sufficiently resolved.

The uncertainty ranges considered with fuzzy arithmetic are usually rather small, thus, the above cases will not occur often. In the unlikely event, however, the sparse grid interpolant will not converge within a reasonable refinement depth, which can be easily detected by a careful implementation.

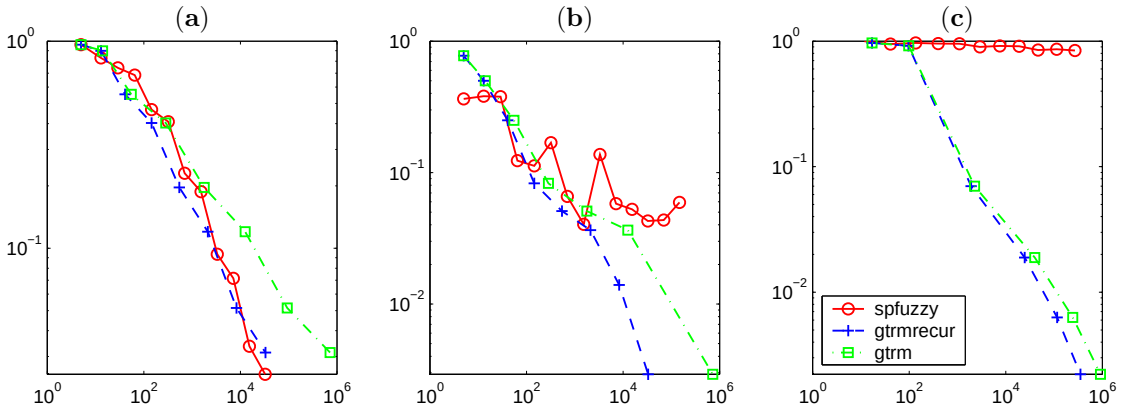


Figure 7: (**a**)–(**c**): Relative error of the fuzzy-valued results of f_6–f_8 w.r.t. the number of real-valued function evaluations. *spfuzzy* depicts the curve obtained for the proposed algorithm, *gtrm* represents the general transformation method, and *gtrmrecur* is the general transformation method without recurring permutations.

5 Conclusions

We have shown that sparse grid interpolation offers a powerful tool to accurately perform fuzzy arithmetic. Due to the excellent ratio of the number of required function evaluations compared to the interpolation quality, the method is especially appealing when the objective function is expensive to evaluate. Furthermore, we have discussed several ways to solve the subsequent sparse grid optimization problems in a reliable and efficient way.

Since the interpolant is only computed for the base box of the fuzzy parameters, any types of fuzzy numbers or intervals (or, in general, any convex fuzzy sets) may be used as inputs. We have provided numerical results illustrating that the theoretical order of convergence known from piecewise multilinear sparse grid interpolation can be reliably achieved for the convergence of the fuzzy output parameters. In case of smaller uncertainty ranges, the method becomes even more efficient, since less support nodes are required to construct the sparse grid interpolant. As the dimension of the function increases, the proposed method shows its full potential, especially compared to discrete methods sampling points at a full grid, since the number of required support nodes grows only moderately with the dimension.

Finally, we have shown the limitations of the new method, which may eventually occur in case of large uncertainty ranges, and/or certain functions exhibiting either non-smoothness or strong local features within the considered base box.

Acknowledgements

The authors are thankful to Kai Willner for many valuable discussions and for useful comments to improve this paper.

References

[1] V. Barthelmann, E. Novak, K. Ritter, High dimensional polynomial interpolation on sparse grids, Adv. Comput. Math. 12 (4) (2000) 273–288.

[2] H.-J. Bungartz, Finite Elements of Higher Order on Sparse Grids, Shaker Verlag, Aachen, 1998.

[3] L. Dixon, G. Szego, The Global Optimization Problem: An Introduction, North Holland, New York, NY, 1978, Ch. Toward global optimization 2, pp. 1–15.

[4] W. M. Dong, H. C. Shah, Vertex method for computing functions of fuzzy variables, Fuzzy Sets and Systems 24 (1987) 65–78.

[5] W. M. Dong, F. S. Wong, Fuzzy weighted averages and implementation of the extension principle, Fuzzy Sets and Systems 21 (1987) 183–199.

[6] D. Dubois, H. Prade, Fuzzy Sets and Systems, Theory and Applications, Vol. 144 of Mathematics in Science and Engineering, Academic Press, Inc., New York, 1980.

[7] R. E. Giachetti, R. E. Young, A parametric representation of fuzzy numbers and their arithmetic operators, Fuzzy Sets and Systems 91 (2) (1997) 185–202.

[8] E. H. Hansen, Global Optimization Using Interval Analysis, Marcel Dekker, New York, NY, 1992.

[9] M. Hanss, The transformation method for the simulation and analysis of systems with uncertain parameters, Fuzzy Sets and Systems 130 (3) (2002) 277–289.

[10] M. Hanss, The extended transformation method for the simulation and analysis of fuzzy-parameterized models, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 11 (6) (2003) 711–727.

[11] Institute of Applied Analysis and Numerical Simulation, University of Stuttgart, Scientific/Educational Matlab Database (2002).
URL `http://matlabdb.mathematik.uni-stuttgart.de`

[12] L. Jaulin, M. Kieffer, O. Didrit, É. Walter, Applied Interval Analysis, Springer, London, Great Britain, 2001.

[13] A. Kaufmann, M. M. Gupta, Introduction to Fuzzy Arithmetic, Van Nostrand Reinhold Co., New York, 1991.

[14] A. Klimke, An efficient implementation of the transformation method of fuzzy arithmetic, in: E. Walker (Ed.), Proceedings of NAFIPS 2003, Chicago, IL, 2003, pp. 468–473.

[15] A. Klimke, An efficient implementation of the transformation method of fuzzy arithmetic (extended preprint version), IANS preprint 2003/009, Tech. rep., University of Stuttgart (2003).
URL `http://preprints.ians.uni-stuttgart.de`

[16] A. Klimke, Piecewise multilinear sparse grid interpolation in MATLAB, IANS report 2003/019, Tech. rep., University of Stuttgart (2003).
URL `http://preprints.ians.uni-stuttgart.de`

[17] A. Klimke, B. Wohlmuth, K. Willner, Uncertainty modeling using efficient fuzzy arithmetic based on sparse grids: applications to dynamic systems, IANS preprint 2004/003, Tech. rep., University of Stuttgart (2004).
URL `http://preprints.ians.uni-stuttgart.de`

[18] G. J. Klir, Fuzzy arithmetic with requisite constraints, Fuzzy Sets and Systems 91 (1997) 165–175.

[19] T. G. Kolda, R. M. Lewis, V. Torczon, Optimization by direct search: new perspectives on classical and modern methods, SIAM Review 45 (3) (2003) 385–482.

[20] R. E. Moore, Interval Analysis, Prentice Hall, Englewood Cliffs, NJ, 1966.

[21] A. Neumaier, Acta Numerica 2004, Cambridge University Press, 2004 (to appear), Ch. "Complete Search in Continuous Global Optimization and Constraint Satisfaction".

[22] M. Navara, Z. Žabokrtský, How to make constrained fuzzy arithmetic efficient, Soft Computing 6 (2001) 412–417.

[23] A. Neumaier, Taylor forms – use and limits, Reliab. Comput. 9 (1) (2003) 43–79.

[24] E. Novak, K. Ritter, High-dimensional integration of smooth functions over cubes, Numer. Math. 75 (1) (1996) 79–97.

[25] A. Schreiber, Smolyak's method for multivariate interpolation, Ph.D. thesis, Georg-August-Universität Göttingen, Germany (2000).

[26] S. A. Smolyak, Quadrature and interpolation formulas for tensor products of certain classes of functions, Soviet Math. Dokl. 4 (1963) 240–243.

[27] F. Sprengel, Interpolation and wavelets on sparse Gauss-Chebyshev grids, in: W. Haussmann et al. (Ed.), Multivariate Approximation, Vol. 101 of Mathematical Research, Akademie Verlag, Berlin, 1997, pp. 269–286.

[28] F. Sprengel, Periodic interpolation and wavelets on sparse grids, Numerical Algorithms 17 (1998) 147–169.

[29] G. W. Wasilkowski, H. Woźniakowski, Explicit cost bounds of algorithms for multivariate tensor product problems, J. Complexity 11 (1) (1995) 1–56.

[30] K. L. Wood, K. N. Otto, E. K. Antonsson, Engineering design calculations with fuzzy parameters, Fuzzy Sets and Systems 52 (1992) 1–20.

[31] H. Q. Yang, H. Yao, J. D. Jones, Calculating functions of fuzzy numbers, Fuzzy Sets and Systems 55 (1993) 273–283.

[32] L. A. Zadeh, Fuzzy sets, Information and Control 8 (1965) 338–353.

[33] C. Zenger, Sparse grids, in: Parallel Algorithms for Partial Differential Equations, Vol. 31 of Notes Numer. Fluid Mech., Vieweg, Braunschweig, 1991, pp. 241–251.

Andreas Klimke

Institute of Applied Analysis and Numerical Simulation
University of Stuttgart
Pfaffenwaldring 57
70569 Stuttgart, Germany

E-Mail: klimke@ians.uni-stuttgart.de

Barbara Wohlmuth

Institute of Applied Analysis and Numerical Simulation
University of Stuttgart
Pfaffenwaldring 57
70569 Stuttgart, Germany

E-Mail: wohlmuth@ians.uni-stuttgart.de

Erschienene Preprints ab Nummer 2004/001

Komplette Liste: `http://preprints.ians.uni-stuttgart.de`

2004/001 *Geis, W., Mishuris, G., Sändig, A.-M.*: 3D and 2D asymptotic models for piezoelectric stack actuators with thin metal inclusions

2004/002 *Klimke, A., Wohlmuth, B.*: Computing expensive multivariate functions of fuzzy numbers using sparse grids