Universität Stuttgart

Constructing dimension-adaptive sparse grid interpolants using parallel function evaluations

Andreas Klimke, Barbara Wohlmuth

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Preprint 2006/002

Universität Stuttgart

Constructing dimension-adaptive sparse grid interpolants using parallel function evaluations

Andreas Klimke, Barbara Wohlmuth

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Preprint 2006/002

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**

Constructing dimension-adaptive sparse grid interpolants using parallel function evaluations

Andreas Klimke

18th January 2006

Abstract

Dimension-adaptive sparse grid interpolation is a powerful tool to obtain surrogate functions of smooth, medium to high-dimensional objective models. In case of expensive models, the efficiency of the sparse grid algorithm is governed entirely by the time required for the function evaluations. In this paper, we first briefly analyze the inherent parallelism of the standard dimension-adaptive algorithm. Then, we present an enhanced version of the standard algorithm that permits, in each step of the algorithm, a specified number (equal to the number of desired processes) of function evaluations to be executed in parallel, thereby increasing the parallel efficiency.

1 Introduction

We consider the problem of obtaining a surrogate function of an expensive objective model in order to efficiently apply other procedures, such as an optimization, a Monte Carlo analysis, a fuzzy uncertainty analysis, or an active control algorithm. Sparse grid interpolation[1] offers a suitable tool to obtain such surrogate functions, especially in case of higher-dimensional problems.

Standard sparse grid interpolants are constructed from model evaluations at grid nodes that are selected a priori. The most important property of the method is the fact that the asymptotic error decay of full grid interpolation with increasing grid resolution is preserved up to a logarithmic factor in case of sufficiently smooth objective functions. This standard approach can be improved by so-called dimension-adaptive grids that construct the interpolants adaptively by using error estimators obtained from previous steps of the algorithm. The dimension-adaptive approach performs best if some of the input variables of the objective model do not contribute significantly to the result, or if the problem at hand exhibits some separable or additive structure.

In case of expensive objective models, the overall computing time of constructing the interpolant is usually almost completely governed by the time taken to evaluate the model. Unfortunately, the dimension-adaptive algorithm does by default not permit an arbitrary number of new support nodes to be added to the model. In general, the support nodes that

[1]The sparse grid concept is not limited to the problem of interpolation and approximation. A recent survey article [4] provides a detailed review of the fundamentals and application areas.

can always be processed in parallel are the ones belonging to the same sub-grid. However, especially in higher-dimensional problems, these sub-grids often contain only a few points in the order of four to eight, which is a too small number if powerful high performance computing systems are available that would allow the evaluation of tens or hundreds of permutations in parallel. In this paper, we therefore propose a modified version of the dimension-adaptive algorithm that will always produce a specified number of support nodes to add to the model per step of the algorithm.

The parallelization of sparse grid algorithms has been studied previously, to the most part in the context of solving partial differential equations [8, 9, 5] and data mining/fitting algorithms [6]; however, in these papers, the main focus is placed on the efficient parallel execution of the sparse grid algorithms rather than the parallel construction of the grid, since the underlying task of evaluating an expensive objective function is not governing the computing time in these cases.

The rest of the paper is organized as follows. In Section 2, we briefly review the conventional approach of dimension-adaptive sparse grids with an emphasis on issues concerning parallel function evaluations. In Section 3, we discuss our novel algorithm that generates an arbitrary specified number of function evaluations in each step. Section 4 presents an application example, the approximation of a large-scale finite element model. We provide some concluding remarks in Section 5.

2 Dimension-Adaptive Sparse Grids

The interpolation problem considered with sparse grid interpolation is an optimal recovery problem, i.e. the goal is to approximate an expensive objective function with a low approximation error while requiring only a low number of support nodes. Sparse grid interpolants are sums of tensor product formulas that require significantly fewer (several orders of magnitude with increasing number of variables) support nodes than conventional tensor-product-based interpolation on a full grid. At the same time, the asymptotic convergence rate is preserved up to a logarithmic factor compared to interpolation on the full tensor product grid. The approximation error depends on the type of basis functions that are used, with common possibilities being piecewise linear [2], piecewise polynomial [3], and global Lagrangian polynomial [1] bases, as well as interpolets and wavelets [4, ch. 4]. Error bounds have been derived in the mentioned articles or the references provided therein.

The basic dimension-adaptive algorithm proposed in [7], developed for numerical quadrature, was adopted to the interpolation problem in [10, ch. 3]. In the following, we recall this sequential algorithm, adhering to the notation therein. For the subroutine `spvalstep` (used to compute the hierarchical surpluses) and the algorithm `spinterp` for the evaluation of the interpolants, please see [10].

The dimension-adaptive algorithm constructs a sparse grid interpolant $A_{S_k}(f)$ of the form

$$A_{S_k}(f) = \sum_{\mathbf{i} \in S_k} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f), \tag{1}$$

where S_k are the sets of multi-indices uniquely identifying the (full) sub-grids that comprise the sparse grid. The lower index $k \in \mathbb{N}$ denotes the cardinality (#) of the set of indices S_k.

$\Delta^i = U^i - U^{i-1}$, $U^0 = 0$, are univariate difference formulas of interpolation formulas

$$U^i(f) = \sum_{x^i \in X^i} a_{x^i} \cdot f(x^i), \tag{2}$$

where the support nodes are given by $X^i = \{x^i \mid x^i \in [0,1]\}$, $\#X^i = m_i$, assuming, without loss of generality, the unit interval as the domain of interpolation in each dimension. The support nodes should be nested, i.e. $X^{i-1} \in X^i$, $X^0 = \emptyset$, such that the difference formulas Δ^i have the support nodes $X^i_\Delta = X^i \setminus X^{i-1}$. The basis functions must satisfy $a_{x^i} \in C([0,1])$, $a_{x^i}(x^i) = 1$, $a_{x^i}(y^i) = 0 \ \forall\ y^i \in X^i, x^i \neq y^i$, $i \in \mathbb{N}$.

The sets of indices S_k must satisfy the following *admissibility condition.* S is called admissible if for all $\mathbf{i} \in S$, $\mathbf{i} - \mathbf{e}_j \in S \ \text{ for } \ 1 \leq j \leq d, \ i_j > 1$ holds, where $\mathbf{e}_j$ is the j-th unit vector, and $\mathbf{i} \in \mathbb{N}^d$.

In addition to the admissibility condition, the construction of the dimension-adaptive interpolant is governed by the chosen *error indicator* that steers the refinement of the grid. In [10, ch. 3], it was suggested to use

$$g(\mathbf{i}) = \frac{1}{n(\mathbf{i})} \sum_{\mathbf{j}} |w^{\mathbf{i}}_{\mathbf{j}}|, \tag{3}$$

where $w^{\mathbf{i}}_{\mathbf{j}}$ are the *hierarchical surpluses* [4, 10] of the sub-grid $\mathbf{X_i} = X^{i_1}_\Delta \times \cdots \times X^{i_d}_\Delta$, and $n(\mathbf{i}) = \#\mathbf{X_i}$. The multi-index $\mathbf{j}$ runs over the points contained in the sub-grid. This indicator is also adopted for the numerical examples provided here. A second parameter that affects the grid construction is the degree of dimensional adaptivity ω, chosen to $\omega = 1$ here, i.e. greedy refinement. This is the case that is most difficult to handle from a parallelization point of view, since the optimal refinement is not partly governed by any rules known a priori that would help to predict the grid structure.

The final concept to be mentioned is the idea of *active* and *passive* indices. The set of all active indices $\mathcal{A}$ contains recently added indices where the error indicators are available, but which have not been processed yet for refinement. Processed indices are put into the set of old indices $\mathcal{O}$; data associated with old indices are required in computing the hierarchical surpluses, while active indices are not.

Example 1 To illustrate the sequential dimension-adaptive algorithm and its inherent parallelism, we consider the Branin test function

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

and its approximation in the domain $[-5, 10] \times [0, 15]$ using polynomial basis functions at the Chebyshev-Gauss-Lobatto nodes. The excellent performance of the dimension-adaptive approach using this interpolation scheme is depicted in Fig. 1. The error plot shows the actual error (maximum over the absolute interpolation error at the equidistant full grid node set with $N = (2^7 + 1)^2$ nodes) and the estimated error e_{est} as computed by line 22 of Algorithm 1.

For instance, when computing the interpolant with up to 29 nodes, five steps of the algorithm are performed (indicated in Fig. 2), i.e., the outer while-loop starting at line 5 is iterated 5 times. In the first step, 5 function evaluations are required (the objective function must be evaluated at each grid point to construct the interpolant, these are the grid points of the set

Algorithm 1 `spadaptvals`

```
In:  $d \in \mathbb{N}$. $f : [0,1]^d \to \mathbb{R}$: Objective function.
     $\delta_{\text{rel}}$, $\delta_{\text{abs}}$: Relative and absolute error tolerance.
     $N_{\max} \in \mathbb{N}$: Max. # of nodes. $\omega \in [0,1]$: Degree of dimensional adaptivity.
Out: $S_k$: Set of multi-indices.
     $Z_k$: Ordered set of hierarchical surpluses of corresp. sub-grids.
1:  Let $\mathcal{E}_\oplus$ be the set of unit vectors $\{\mathbf{e}_i \mid i = 1, \dots, d\}$ in $\mathbb{R}^d$.
2:  Let $e_{\text{est}} = \infty$, $k = 1$, $\mathcal{A} = \{(\mathbf{1})\}$, $\mathcal{O} = \emptyset$.
3:  Let $\mathbf{X_1} = X_\Delta^1 \times \cdots \times X_\Delta^1$, $N = \#\mathbf{X_1}$.
4:  Let $Z_\mathbf{1} = f(\mathbf{X_1})$.          {Evaluate $y = f(\mathbf{x})$ at all points $\mathbf{x} \in \mathbf{X_1}$}
5:  Let $y_{\max} = \max\{y \; \forall \; y \in Z_\mathbf{1}\}$, $y_{\min} = \min\{y \; \forall \; y \in Z_\mathbf{1}\}$.
6:  While $e_{\text{est}} \geq \max\{\delta_{\text{rel}} \cdot (y_{\max} - y_{\min}),\ \delta_{\text{abs}}\}$ And $N \leq N_{\max}$ Do
7:    Let $\mathbf{i}^{\text{act}} = \operatorname{argmin}\{|\mathbf{i}| \; \forall \; \mathbf{i} \in \mathcal{A}\}$.
8:    If $|\mathbf{i}^{\text{act}}| > (1-\omega) \cdot \max\{|\mathbf{i}| \; \forall \; \mathbf{i} \in \mathcal{A} \cup \mathcal{O}\}$ Then
9:      Let $\mathbf{i}^{\text{act}} = \operatorname{argmax}\{g(\mathbf{i}) \; \forall \; \mathbf{i} \in \mathcal{A}\}$.
10:   End If
11:   Let $\mathcal{A} = \mathcal{A} \setminus \{\mathbf{i}^{\text{act}}\}$, $\mathcal{O} = \mathcal{O} \cup \{\mathbf{i}^{\text{act}}\}$, $Z_\mathcal{O} = Z_\mathcal{O} \cup Z_{\mathbf{i}^{\text{act}}}$.
12:   For $j = 1$ To $d$ Do
13:     Let $\mathbf{i}^{\text{new}} = \mathbf{i}^{\text{act}} + \mathbf{e}_j$.
14:     If $\mathbf{i}^{new} - \mathbf{e}_l \in \mathcal{O}$ for all $\mathbf{e}_l \in \{\mathbf{e}_l \in \mathcal{E}_\oplus \mid i_l > 1,\ l = 1, \dots, d\}$ Then
15:       Let $\mathcal{A} = \mathcal{A} \cup \{\mathbf{i}^{\text{new}}\}$
16:       Let $\mathbf{X}_{\mathbf{i}^{\text{new}}} = X_\Delta^{i_1^{\text{new}}} \times \cdots \times X_\Delta^{i_d^{\text{new}}}$. Let $Y_{\mathbf{i}^{\text{new}}} = f(\mathbf{X}_{\mathbf{i}^{\text{new}}})$.
17:       Let $Z_{\mathbf{i}^{\text{new}}} = Y_{\mathbf{i}^{\text{new}}} - \texttt{spvalstep}(d, Z_\mathcal{O}, \{\mathbf{X}_{\mathbf{i}^{\text{new}}}\}, \{\mathbf{i}^{\text{new}}\}, \mathcal{O})$.
18:       Let $y_{\max} = \max\{y_{\max}, y \; \forall \; y \in Y_{\mathbf{i}^{\text{new}}}\}$, $y_{\min} = \min\{y_{\min}, y \; \forall \; y \in Y_{\mathbf{i}^{\text{new}}}\}$.
19:       Let $N = N + \#\mathbf{X}_{\mathbf{i}^{\text{new}}}$, $k = k + 1$.
20:     End If
21:   End For
22:   Let $e_{\text{est}} = \max\{w \; \forall \; w \in Z_\mathbf{i} \; \forall \; \mathbf{i} \in \mathcal{A}\}$.
23: End While
24: Let $S_k = \mathcal{O} \cup \mathcal{A}$,  $Z_k = Z_\mathcal{O} \cup \bigcup_{\mathbf{i} \in \mathcal{A}} Z_\mathbf{i}$.
```

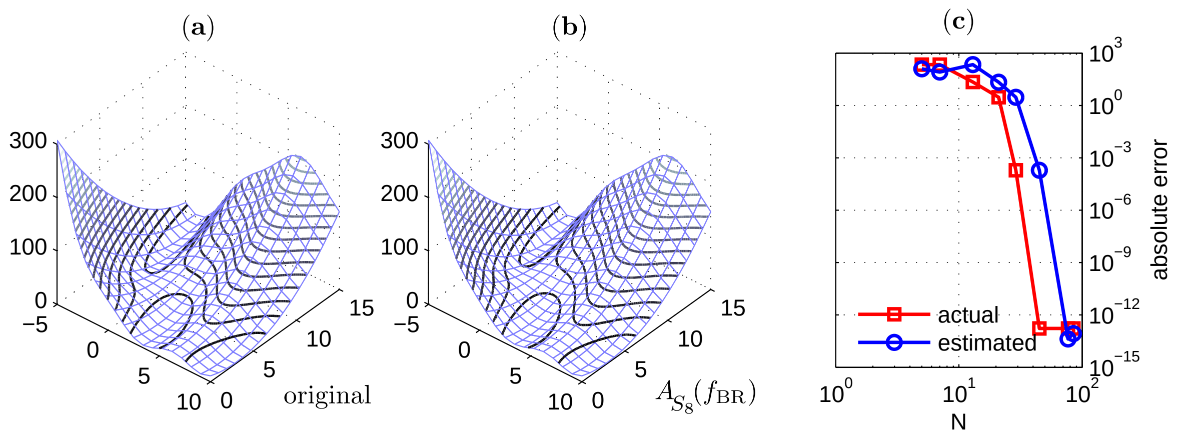


Figure 1: (**a**): f_{BR}, (**b**): $A_{S_8}^{\text{CGL}}(f_{\text{BR}})$ w/ 21 support nodes, (**c**): error plot.

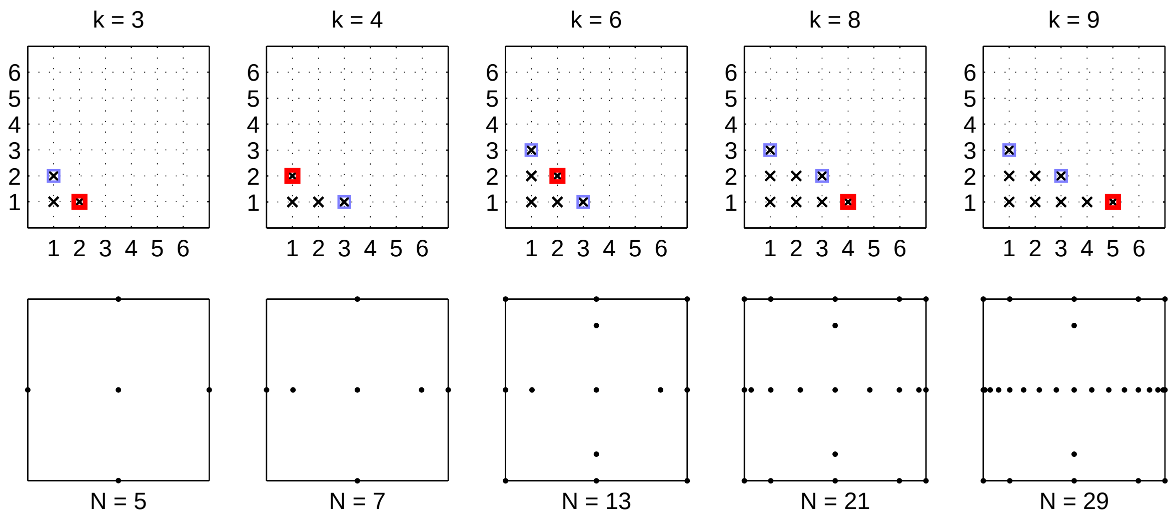


Figure 2: Steps of the algorithm (bold □: index set with largest error indicator).

of sub-grids $\mathbf{X}_\mathbf{i}$ with multi-indices $\mathbf{i} \in S_3 = \{(1,1),(2,1),(1,2)\}$). These evaluations can be performed in parallel. In the next step (the second column in Fig. 2), the index with largest error indicator $g(\mathbf{i})$ from the previous step determines the next index set to be added, $(3,1)$ in this case, requiring two function evaluations that can be executed in parallel. The third step (the third column in Fig. 2) adds the indices $(1,3)$ and $(2,2)$, requiring six function evaluations, and so on. From this example, the following properties of the algorithm become evident:

1. The number of function evaluations in each step can be rather small, and it usually changes in each step. Thus, when a fixed number of nodes or processes are allotted to execute the parallel function evaluations, load balancing becomes a problem, since the number of required evaluations will usually not match the number of processes available. This limits the achievable parallel efficiency of the algorithm. For instance, using $n_p = 5$ processes to compute $A_{S_9}(f_{\mathrm{BR}})$ will only achieve a speedup of just $s_{p_1} = 29/(1+2+2+2+2) = 2.9$ in this case.

2. Without modification of the algorithm, the number of performed steps cannot be reduced, since the error indicators that determine the refinement direction are not available a priori. The speedup in constructing $A_{S_9}(f_{\mathrm{BR}})$ is thus limited to $29/5 = 5.8$, for any $n_p \geq 8$.

To conclude this section, it can be stated that while the sequential Algorithm 1 allows to execute some function evaluations in parallel, its parallel efficiency is far from being optimal. The theoretically achievable parallel speedup s_{p_1} of constructing $A_{S_k}(f)$, depending on the number of processes used, can be determined (a posteriori) by the formula

$$s_{p_1}(A_{S_k}(f), n_p) = \frac{\#A_{S_k}(f)}{\mathrm{ceil}\left(\frac{n(\mathbf{1}) + \sum_{\mathbf{i}\in\mathcal{A}_1} n(\mathbf{i})}{n_p}\right) + \sum_{m=2}^{m_{\max}} \mathrm{ceil}\left(\frac{\sum_{\mathbf{i}\in\mathcal{A}_m\setminus\mathcal{A}_{m-1}} n(\mathbf{i})}{n_p}\right)} \leq \frac{\#A_{S_k}(f)}{m_{\max}}, \qquad (4)$$

where $\#A_{S_k}(f)$ denotes the number of grid point of $A_k(f)$, $m_{\max}$ denotes the number of iterations (outer while loops) performed by the algorithm, and $\mathcal{A}_m$ denotes the contents of the set $\mathcal{A}$ at line 22 at iteration step m. In the following section, we will discuss an approach that overcomes these limitations to a large extent.

3 The Proposed Enhanced Algorithm

The main idea is to introduce an additional condition that the enhanced Algorithm 2 must fulfill: in each step of the algorithm, a fixed specified number of permutations equal to the number of processes n_p should be generated. This number can be chosen freely by the user.

3.1 Description of the algorithm

We achieve the desired behavior by modifying Algorithm 1 in a suitable way. First, we introduce another set of indices $\mathcal{B}$ in addition to the sets $\mathcal{A}, \mathcal{O}$ that contains the active indices that have been added to the interpolant, but the error indicators and hierarchical surpluses of which are not yet available. Secondly, we separate each step of the algorithm into two main parts: The process of grid point generation (lines 6–25 of Algorithm 2) and the process of hierarchical surplus computation and error estimation (lines 26–33).

In order to achieve that at least n_p evaluations become available in each step, we pick not just the one active index with largest error from the set of active indices, but several ones (while N_{new}, the number of new function evaluations of the current step, is smaller than n_p). By selecting indices from $\mathcal{A} \setminus \mathcal{B}$ (line 9), we make sure that $g(\mathbf{i})$ is indeed available. If no more active index with error estimator is available, then $\mathcal{A} \setminus \mathcal{B} = \emptyset$, and an index according to the non-adaptive sparse grid construction scheme is selected, i.e., an index with smallest $|\mathbf{i}|$, $\mathbf{i} \in \mathcal{A}$ (line 7). This criterion is still also used to gradually shift from greedy to conservative refinement using the parameter ω (the degree of dimensional adaptivity).

An additional difficulty that has to be handled is the fact that the number of support nodes added to the interpolant by the new active indices (lines 15–18) do usually not match the desired number of processes n_p. The solution is to delay super-numerous permutations to the next step of the algorithm. This is achieved by line 22: $\mathbf{X}_{\text{next}}$ must contain new points selected from the sub-grids with active indices in $\mathcal{B}$, but without the points that may have been processed already in the previous step, stored in $\mathbf{X}_{\text{act}}$. Then, exactly n_p permutations can be evaluated in parallel (line 23).

The remaining part of the algorithm takes care of the hierarchical surpluses computation. All active indices where the function values of all corresponding grid points have been computed are put into the index set $\mathcal{C}$ (line 26), and are subsequently removed from $\mathcal{B}$ (line 27), since these indices will have their hierarchical surpluses computed (line 29), and are thus ready to be used in line 9, where the computation of the error indicator becomes necessary.

3.2 Properties of the algorithm

The new algorithm has the following properties:

Algorithm 2 `spadaptvalspar`

```
In:  $d \in \mathbb{N}$. $f : [0,1]^d \to \mathbb{R}$: Objective function.
     $\delta_{\text{rel}}$, $\delta_{\text{abs}}$: Relative and absolute error tolerance.
     $N_{\max} \in \mathbb{N}$: Maximum number of nodes.
     $\omega \in [0,1]$: Degree of dimensional adaptivity.
     $n_p \in \mathbb{N}$: Number of processes available for function evaluations.
Out: $S_k$: Set of multi-indices.
     $Z_k$: Ordered set of hierarchical surpluses of corresp. sub-grids.
1: $\mathcal{E}_\oplus$ be the set of unit vectors $\{\mathbf{e}_i \mid i = 1, \ldots, d\}$ in $\mathbb{R}^d$.
2: Let $e_{\text{est}} = \infty$, $k = 0$.
3: Let $\mathcal{A} = \mathcal{B} = \{(\mathbf{1})\}$, $\mathcal{O} = \emptyset$, $Z_0 = \emptyset$, $\mathbf{X}_{\text{act}} = \emptyset$.
4: Let $N = 0$, $N_{\text{new}} = 0$.
5: While $e_{\text{est}} \geq \max\{\delta_{\text{rel}} \cdot (y_{\max} - y_{\min}),\ \delta_{\text{abs}}\}$ And $N \leq N_{\max}$ Do
6:   While $N_{\text{new}} < n_p$ Do
7:     Let $\mathbf{i}^{\text{act}} = \operatorname{argmin}\{|\mathbf{i}| \ \forall\ \mathbf{i} \in \mathcal{A}\}$.
8:     If $|\mathbf{i}^{\text{act}}| > (1-\omega) \cdot \max\{|\mathbf{i}| \ \forall\ \mathbf{i} \in \mathcal{A}\}$ And $\mathcal{A} \setminus \mathcal{B} \neq \emptyset$ Then
9:       Let $\mathbf{i}^{\text{act}} = \operatorname{argmax}\{g(\mathbf{i}) \ \forall\ \mathbf{i} \in \{\mathcal{A} \setminus \mathcal{B}\}\}$.
10:    End If
11:    Let $\mathcal{A} = \mathcal{A} \setminus \{\mathbf{i}^{\text{act}}\}$, $\mathcal{O} = \mathcal{O} \cup \{\mathbf{i}^{\text{act}}\}$.
12:    For $j = 1$ To $d$ Do
13:      Let $\mathbf{i}^{\text{new}} = \mathbf{i}^{\text{act}} + \mathbf{e}_j$.
14:      If $\mathbf{i}^{new} - \mathbf{e}_l \in \mathcal{O}$ for all $\mathbf{e}_l \in \{\mathbf{e}_l \in \mathcal{E}_\oplus \mid i_l > 1,\ l = 1, \ldots, d\}$ Then
15:        Let $\mathcal{A} = \mathcal{A} \cup \{\mathbf{i}^{\text{new}}\}$.
16:        Let $\mathcal{B} = \mathcal{B} \cup \{\mathbf{i}^{\text{new}}\}$.
17:        Let $\mathbf{X}_{\mathbf{i}^{\text{new}}} = X_\Delta^{i_1^{\text{new}}} \times \cdots \times X_\Delta^{i_d^{\text{new}}}$.
18:        Let $N_{\text{new}} = N_{\text{new}} + \#\mathbf{X}_{\mathbf{i}^{\text{new}}}$, $k = k + 1$.
19:      End If
20:    End For
21:  End While
22:  Let $\mathbf{X}_{\text{next}} \subset (\{\mathbf{X}_\mathbf{i} \mid \mathbf{i} \in \mathcal{B}\} \setminus \mathbf{X}_{\text{act}})$ such that $\#\mathbf{X}_{\text{next}} = n_p$.
23:  Let fvals $= f(\mathbf{X}_{\text{next}})$.        {Evaluate in parallel.}
24:  Let $N_{\text{new}} = N_{\text{new}} - n_p$, $N_{\text{fevals}} = N_{\text{fevals}} + n_p$.
25:  Update the array entries $Y_\mathbf{i}$, $\mathbf{i} \in \mathcal{B}$, at the correct positions from fvals.
26:  Let $\mathcal{C} = \{\mathbf{i} \in \mathcal{B} \mid \mathbf{X}_\mathbf{i} \subseteq \mathbf{X}_{\text{act}}\}$.
27:  Let $\mathcal{B} = \mathcal{B} \setminus \mathcal{C}$, $\mathbf{X}_{\text{act}} = \mathbf{X}_{\text{act}} \setminus \{\mathbf{X}_\mathbf{i} \mid \mathbf{i} \in \mathcal{C}\}$.
28:  for all $\mathbf{i} \in \mathcal{C}$ Do
29:    Let $Z_\mathbf{i} = Y_\mathbf{i} - \texttt{spvalstep}(d, Z_k, \mathbf{X}_\mathbf{i}, \mathbf{i}, \mathcal{O})$, $Z_{k+1} = Z_k \cup Z_\mathbf{i}$.
30:    Let $y_{\max} = \max\{y_{\max}, y \ \forall\ y \in Y_\mathbf{i}\}$, $y_{\min} = \min\{y_{\min}, y \ \forall\ y \in Y_\mathbf{i}\}$.
31:    Let $N = N + \#\mathbf{X}_\mathbf{i}$, $k = k + 1$.
32:  End For
33:  Let $e_{\text{est}} = \max\{w \ \forall\ w \in Z_\mathbf{i} \ \forall\ \mathbf{i} \in \mathcal{A} \setminus \mathcal{B}\}$.
34: End While
35: Let $S_k = \mathcal{O} \cup (\mathcal{A} \setminus \mathcal{B})$.
```

1. By definition, the algorithm produces exactly n_p permutations to be evaluated in each step.

2. In the case of conservative grid refinement, i.e. $\omega = 0$, the results of the proposed enhanced dimension-adaptive algorithm will produce exactly the same grid as the sequential Algorithm 1, since in this case, the condition in line 8 is never true and the selection process is completely governed by the conventional, non-adaptive construction scheme with its selection criterion of line 7.

3. In the case of greedy ($\omega = 1$) or partly conservative ($0 < \omega < 1$) grid refinement, the grid may be refined with respect to several indices at once with each step. The implication of this is that the generated grid may be different from the one obtained with the sequential algorithm. This difference increases slightly for increasing parameter value n_p if the refinement of the dimension-adaptive grid differs significantly from the non-adaptive one. In this case, increasingly many indices may be selected from the criterion of line 7 rather than from the adaptive one of line 9 due to the fact that the set $A \setminus B$ becomes empty eventually. On the other hand, this is compensated if the value of d increases, since more new active index sets are produced in the loop over d (line 12). This last point can be seen as the limitation of the parallel efficiency that is natural in the parallelization of adaptive algorithms.

We now analyze the parallel efficiency of the new algorithm. Let $A_{S_l,\mathrm{par}}(f)$ be an interpolant obtained with Algorithm 2. Furthermore, let $A_{S_{k_1}}(f)$ and $A_{S_{k_2}}(f)$ be two subsequent interpolants of the sequential Algorithm 1 such that $S_{k_1} \subset S_{k_2}$ and the number of grid points satisfy

$$\#A_{S_{k_1}}(f) \leq \#A_{S_l,\mathrm{par}}(f) < \#A_{S_{k_2}}(f). \tag{5}$$

Then, the number of identical grid points in $A_{S_{k_1}}(f)$ and $A_{S_l,\mathrm{par}}(f)$ can be computed by adding up the number of grid points of the index sets belonging to both S_{k_1} and S_l. Relating this value to the total number of function evaluations N_{fevals} used to construct $A_{S_l,\mathrm{par}}(f)$ (computed in line 24 of Algorithm 2), we define the efficiency ratio

$$r_{\mathrm{eff}}(A_{S_l,\mathrm{par}}) = N_{\mathrm{fevals}}^{-1} \cdot \sum_{\mathbf{i} \in S_{k_1} \cap S_l} n(\mathbf{i}) \quad \in [0,1] \tag{6}$$

indicating how well the interpolant constructed in parallel corresponds to the sequential one of equal or slightly lower complexity. With this ratio, we obtain the effective speedup of Algorithm 2 to

$$s_{p_2} = r_{\mathrm{eff}}(A_{S_l,\mathrm{par}}) \cdot n_p \leq \frac{\#A_{S_{k_1}}(f)}{\mathrm{ceil}(\#A_{S_{k_1}}(f)/n_p)}. \tag{7}$$

We will study the described properties in the next section that deals with an application of the proposed algorithm.

4 An Application in Vibration Engineering

In [10, ch. 5], a dimension-adaptive interpolant of a finite element model was computed to approximate displacement response spectrum envelopes (see Fig. 3). The model characteristics are summarized in Table 1. Here, we focus on the sparse grid approximation of the model output for nine uncertain parameters, i.e. the model output had to be approximated for the nine-dimensional parameter domain $p_1 \times p_2 \times \cdots \times p_9$, where the interval bounds of the parameters were chosen according to [10, Table 5.6].

First, we computed a sparse grid interpolant with up to $N = 1013$ nodes with Algorithm 1, using a greedy refinement strategy ($\omega = 1$) and the piecewise multilinear basis functions of the Clenshaw-Curtis grid [10, 11]. The final interpolant was reached after $n_{\max} = 65$ iterations of the algorithm. Thus, in this case, the theoretically possible speedup of Algorithm 1 was limited to $1013/65 = 15.6$ for the interpolant with 1013 nodes. Fig. 4 (**a**) shows the theoretically achievable speedup according to Eq. (4) over the number of nodes of the interpolant, depending on the number of processes. While Algorithm 1 performed well for a low number of up to 8 processes, its parallel efficiency breaks down completely for 32 and more processes. Thereafter, we applied Algorithm 2. Fig. 4 (**b**) shows the effective speedup according to Eq. (7) depending on the number of processes. As the graph demonstrates, the speedup is nearly optimal for up to 16 processes. Furthermore, for a larger number of processes, Algorithm 2 significantly outperforms Algorithm 1, thereby achieving very good efficiency rates, especially with an increasing number of nodes.

Remark. In the presented speedup analysis of both algorithms, we have neglected all contributions other than the number of model evaluations (e.g. distributing the work to the processes, receiving the results, etc). This is a valid assumption in the case of expensive models where the construction of the sparse grid interpolant as well as the other tasks is small compared to one model evaluation. Furthermore, it was assumed that each model evaluation takes the same amount of computing time for each process, which can of course only hold for homogenous parallel computing architectures.

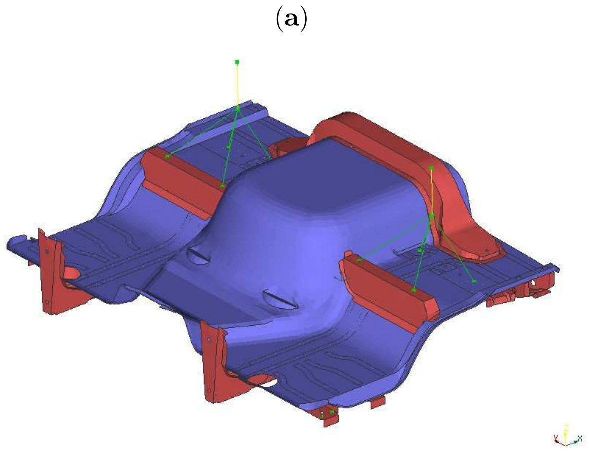
5 Concluding Remarks

In this paper, an enhanced algorithm was proposed to compute dimension-adaptive sparse grid interpolants using parallel function evaluations. We studied the properties with respect to parallelization of the original as well as the enhanced algorithm, and provided formulas analyzing their parallel performance. The effectiveness of the new algorithm was demonstrated by the approximation of a medium-scale finite element model. The algorithms have been implemented in MATLAB[2] and they will be made available within the Sparse Grid Interpolation Toolbox, based on [11] currently under development at our institute. Finally, let us remark that the parallel scheme is not limited to the construction of interpolants; they can be easily adapted to quadrature algorithms, such as described in [7].

Acknowledgments

This work was partially carried out under the HPC-EUROPA project (RII3-CT-2003-506079), with the support of the EU Research Infrastructure Action of the FP6. The primary author gratefully acknowledges the support of the hosting institute, the Laboratoire des Signaux et Systèmes at the École Supérieure d'Électricité, France, which provided excellent working

[2]MATLAB is a registered trademark of The Mathworks, Inc.



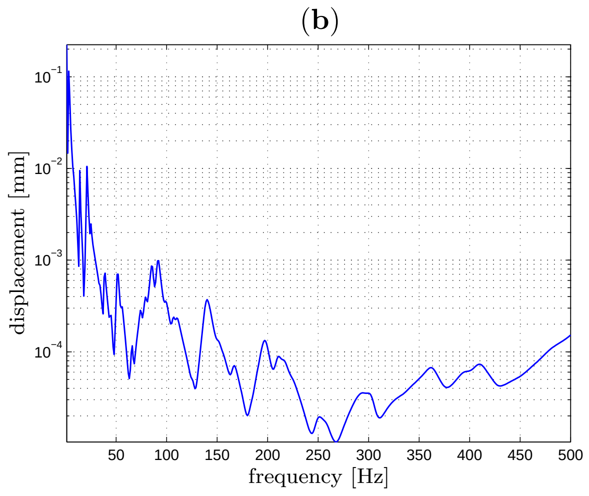
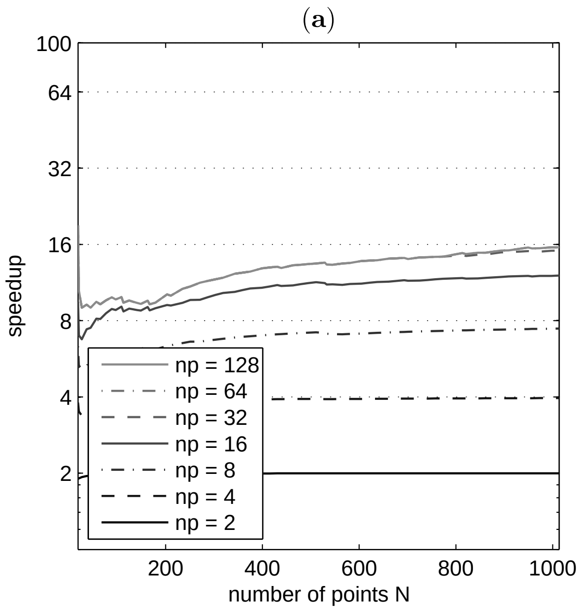


Figure 3: (**a**): Model, (**b**): Resulting response spectrum.

Number of nodes	22 395
Number of elements	21 254
Computing time (sequential)	approx. 15 min.
Frequency domain	1 – 500 Hz (modal basis up to 1kHz)

Table 1: Floor model characteristics



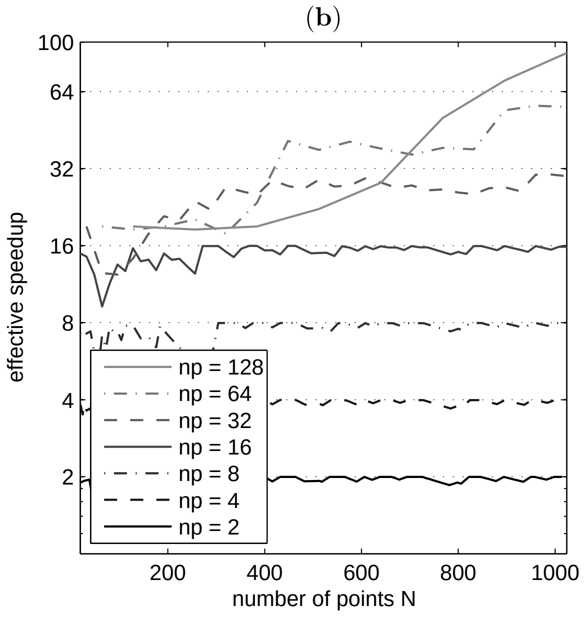


Figure 4: Comparison of speedup, (**a**): Algorithm 1, (**b**): Algorithm 2.

conditions, and in particular, Michel Kieffer and Éric Walter for many stimulating discussions. We record our distinct appreciation to DaimlerChrysler AG for providing the model used in Section 4, as well as INTES GmbH for providing licenses of the FE software PERMAS.

References

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

[2] H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poissongleichung.* PhD thesis, Technische Universität München, Germany, 1992.

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

[4] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[5] K. Everaars and B. Koren. Using coordination to parallelize sparse-grid methods for 3-D CFD problems. *Parallel Comput.*, 24(7):1081–1106, 1998.

[6] J. Garcke, M. Hegland, and O. Nielsen. Parallelisation of sparse grids for large scale data analysis. In *Lecture Notes in Computer Science*, volume 2659, pages 683–692. Springer, 2003.

[7] T. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71(1):65–87, 2003.

[8] M. Griebel. Parallel multigrid methods on sparse grids. In *Multigrid Methods III*, volume 98 of *Internat. Ser. Numer. Math.*, pages 211–221. Birkhäuser, Basel, 1991.

[9] M. Griebel. The combination technique for the sparse grid solution of PDEs on multiprocessor machines. *Parallel Processing Letters*, 2:61–70, 1992.

[10] A. Klimke. *Uncertainty modeling using fuzzy arithmetic and sparse grids.* PhD thesis, Universität Stuttgart, Shaker Verlag, Aachen, 2006.

[11] A. Klimke and B. Wohlmuth. Piecewise multilinear hierarchical sparse grid interpolation in Matlab. *ACM Transactions on Mathematical Software*, 31(4), 2005.