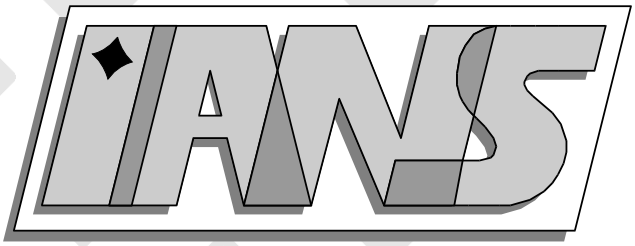
Universität Stuttgart



Piecewise Multilinear Sparse Grid Interpolation in MATLAB

Andreas Klimke

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Technical Report 2003/019

Universität Stuttgart

Piecewise Multilinear Sparse Grid Interpolation in MATLAB

Andreas Klimke

Berichte aus dem Institut für Angewandte Analysis und Numerische Simulation

Technical Report 2003/019

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**

Piecewise Multilinear Sparse Grid Interpolation in MATLAB

Andreas Klimke

Revised: March 1, 2004

Abstract

To recover/approximate smooth multivariate functions, sparse grids are superior to full grids due to a significant reduction of the required support nodes. The order of the convergence rate in the maximum norm is hereby preserved up to a logarithmic factor. We describe three possible piecewise multilinear interpolation schemes in detail and conduct a numerical comparison. Furthermore, we summarize the features of our efficient and easy-to-use sparse grid interpolation software package `spinterp` for MATLAB, which is available for free.

Key words: sparse grids, multivariate interpolation, Smolyak algorithm

---

1 Introduction

Sparse grids have been successfully applied to the solution of partial differential equations, initiated by Zenger in 1991 [14]. This is still the most active research area involving sparse grids, while other important application areas are being studied as well (e.g. numerical quadrature, image processing, and data compression – see [4] for references). This paper deals with interpolation on sparse grids.

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. [2, 9, 12, 13]). All of them employ Smolyak's construction [11], which forms the basis of all sparse grid methods. With Smolyak's famous 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 accuracy is reached.

In the following, we present an implementation of piecewise multilinear sparse grid interpolation. We compare three different sparse grid types with respect to their grid structure and matching piecewise multilinear basis functions. Thereafter, we perform numerical tests with respect to the error decay for a set of test functions. The last section gives a brief discussion of the user interface of our sparse grid interpolation software package `spinterp` for MATLAB. We conclude with some performance test results.

2 Multilinear sparse grid interpolation

2.1 Smolyak's algorithm

This brief description of Smolyak's construction for multivariate interpolation adheres to the notation given in [2]. We would like to approximate smooth functions $f : [0,1]^d \to \mathbb{R}$ using a finite number of support nodes. In the one-dimensional case, an interpolation formula is given by

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

with the set of support nodes $X^i = \{x_1^i, \ldots, x_{m_i}^i\}$, $x_k \in [0,1], 1 \leq k \leq m_i$, and the basis functions $a_{x^i} \in C([0,1])$, $a_{x^i}(x^i) = 1$, $a_{x^i}(y^i) = 0 \ \forall \ y^i \in X^i, x^i \neq y^i$, $i \in \mathbb{N}$. To obtain an interpolation formula for the multivariate case, one can use the tensor product formula

$$(U^{i_1} \otimes \cdots \otimes U^{i_d})(f) = \sum_{x^{i_1} \in X^{i_1}} \cdots \sum_{x^{i_d} \in X^{i_d}} (a_{x^{i_1}} \otimes \cdots \otimes a_{x^{i_d}}) \cdot f(x^{i_1}, \ldots, x^{i_d}). \tag{1}$$

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

$$A_{q,d}(f) = \sum_{|\mathbf{i}| \leq q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f) = A_{q-1,d}(f) + \underbrace{\sum_{|\mathbf{i}|=q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f)}_{\Delta A_{q,d}(f)}, \tag{2}$$

with $A_{d-1,d} = 0$. 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

$$H_{q,d} = \bigcup_{q-d+1 \leq |\mathbf{i}| \leq q} (X^{i_1} \times \cdots \times X^{i_d}) \tag{3}$$

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 (3)

to

$$H_{q,d} = \bigcup_{|\mathbf{i}|\leq q} (X_\Delta^{i_1} \times \cdots \times X_\Delta^{i_d}) = H_{q-1,d} \cup \Delta H_{q,d}, \text{ with} \tag{4}$$

$$\Delta H_{q,d} = \bigcup_{|\mathbf{i}|=q} (X_\Delta^{i_1} \times \cdots \times X_\Delta^{i_d}), \tag{5}$$

$H_{d-1,d} = \emptyset$, which is more convenient for a successive refinement of the grid with increasing parameter q.

2.2 Selecting a suitable sparse grid

For multilinear interpolation, it is natural to select sparse grids with sets X^i, $i \in \mathbb{N}$ of equidistant nodes. We have examined three possibilities to construct the sparse grid, namely

1. the "classical" maximum- or L_2-norm-based sparse grid H^{M}, including the boundary, as thoroughly discussed in [1, 3]. The points x_j^i comprising the set of support nodes X^i are defined as

$$\begin{aligned} m_i &= 2^i + 1, \\ x_j^i &= (j-1)/(m_i - 1) \quad \text{for } j = 1, \ldots, m_i, \text{ and } i \geq 1. \end{aligned}$$

2. the maximum-norm-based sparse grid, but excluding the points on the boundary, denoted by H^{NB}. Now, the x_j^i are defined as

$$\begin{aligned} m_i &= 2^i - 1, \\ x_j^i &= j/(m_i + 1) \quad \text{for } j = 1, \ldots, m_i, \text{ and} \end{aligned}$$

3. the Clenshaw-Curtis-type sparse grid H^{CC} with equidistant nodes, as described in [8, 9]. Although the original paper [5] uses Chebyshev-distributed nodes, we adhere to this grid name due to its association in the literature with the grid structure (resulting from the sequence $m_i = 2^{i-1} + 1$). Here, the x_j^i are defined as

$$m_i = \begin{cases} 1 & \text{if } i = 1, \\ 2^{i-1} + 1 & \text{if } i > 1. \end{cases}$$

$$x_j^i = \begin{cases} (j-1)/(m_i - 1) & \text{for } j = 1, \ldots, m_i \text{ if } m_i > 1, \\ 0.5 & \text{for } j = 1 \text{ if } m_i = 1. \end{cases}$$

All three resulting sets of points fulfill the important property $X^i \subset X^{i+1}$, and therefore also $H_{q,d} \subset H_{q+1,d}$. Figure 1 illustrates the grids $H_{6,2}^{\mathrm{M}}$, $H_{6,2}^{\mathrm{NB}}$ and $H_{6,2}^{\mathrm{CC}}$ for $d = 2$. Figure 2 illustrates the grids $H_{7,3}^{\mathrm{M}}$, $H_{7,3}^{\mathrm{NB}}$ and $H_{7,3}^{\mathrm{CC}}$ for $d = 3$. The number of grid points grows much faster with increasing q, d for H^{M}. The number of points of the Clenshaw-Curtis grid H^{CC} increases the slowest. To further illustrate the growth of the number of nodes with q, d depending on the chosen grid type, we have included Table 1.

In the following, we focus mainly on the Clenshaw-Curtis grid due to its most favorable behavior with respect to the grid size. Our numerical tests in Section 3 will show that it also outperforms the other grid types in most cases (although the asymptotic error decay remains the same).

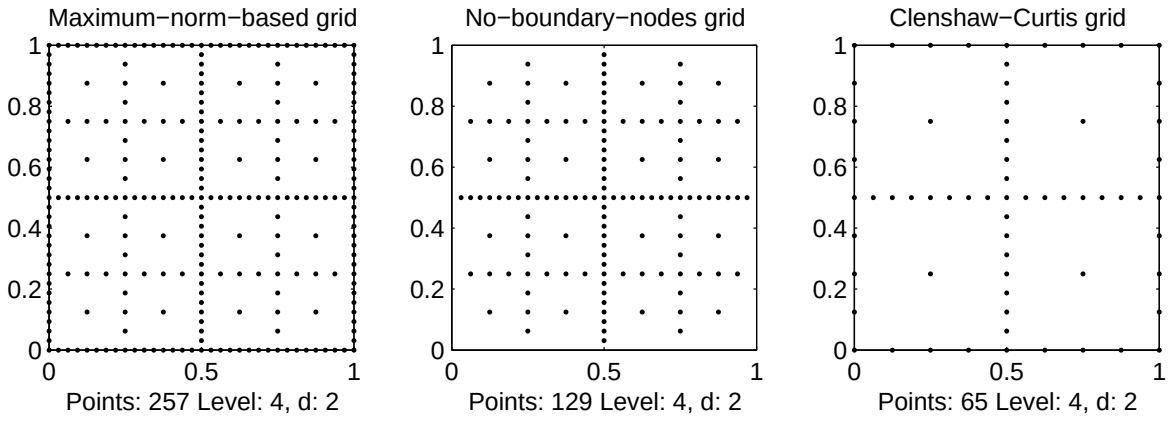


Figure 1: Sparse grids $H_{6,2}^{\mathrm{M}}$ (left), $H_{6,2}^{\mathrm{NB}}$ (middle) and $H_{6,2}^{\mathrm{CC}}$ (right).

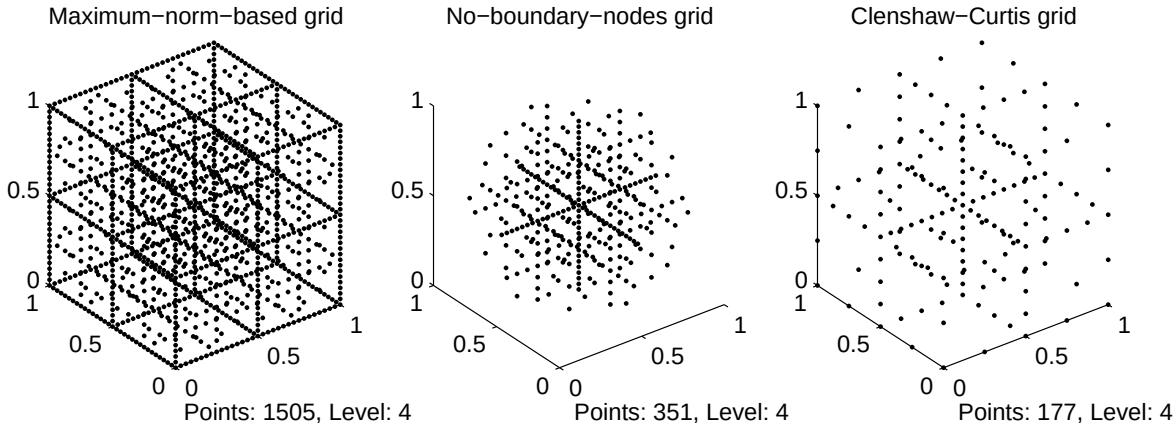


Figure 2: Sparse grids $H_{7,3}^{\mathrm{M}}$ (left), $H_{7,3}^{\mathrm{NB}}$ (middle) and $H_{7,3}^{\mathrm{CC}}$ (right).

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

	$d=2$			$d=4$			$d=8$		
n	M	NB	CC	M	NB	CC	M	NB	CC
0	9	1	1	81	1	1	6561	1	1
1	21	5	5	297	9	9	41553	17	17
2	49	17	13	945	49	41	1.9e5	161	145
3	113	49	29	2769	209	137	7.7e5	1121	849
4	257	129	65	7681	769	401	2.8e6	6401	3937
5	577	321	145	20481	2561	1105	9.3e6	31745	15713
6	1281	769	321	52993	7937	2929	3.0e7	141569	56737
7	2817	1793	705	1.3e5	23297	7537	9.1e7	5.8e5	1.9e5

2.3 The univariate nodal basis functions

To compute $A_{q,d}(f)$, we need to specify the basis functions a of the interpolation formulas $U^i(f)$. We provide the piecewise linear basis functions for each of the three grid types below.

1. For the maximum-norm-based grid, we define

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

   for $j = 1, \ldots, m_i$.

2. For the maximum-norm-based grid without boundary nodes, we define

$$a_{x_1^1}(x) = 1 \quad \text{for } i = 1, \text{ and}$$

$$\text{if } j = \begin{cases} 1, & a_{x_1^i}(x) = \begin{cases} 2 - (m_i + 1)x, & \text{if } x < \frac{2}{m_i+1} \\ 0, & \text{otherwise,} \end{cases} \\ m_i, & a_{x_{m_i}^i}(x) = \begin{cases} (m_i + 1)x - m_i + 1, & \text{if } x > \frac{m_i-1}{m_i+1} \\ 0, & \text{otherwise,} \end{cases} \\ \text{otherwise,} & a_{x_j^i}(x) = \begin{cases} 1 - (m_i + 1)\left|x - x_j^i\right|, & \text{if } \left|x - x_j^i\right| < \frac{1}{m_i+1}, \\ 0, & \text{otherwise,} \end{cases} \end{cases}$$

   for $i > 1$ and $j = 1, \ldots, m_i$.

3. For the Clenshaw-Curtis grid, we get

$$a_{x_1^1}(x) = 1 \quad \text{for } i = 1, \text{ and}$$

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

   for $i > 1$ and $j = 1, \ldots, m_i$.

The obtained sets of nodal basis functions are shown in Fig. 3. Note that since the grid H^{NB} does not have any nodes on the boundary, we can interpret the part of the basis functions with a function value greater than one as a domain of extrapolation. Instead of extrapolating towards the boundary, one could also extend the basis functions nearest to the boundary at the constant value one [15].

2.4 From univariate nodal to multivariate hierarchical

In this section, we will show how one can obtain the multivariate hierarchical formulation of Smolyak's algorithm of Section 2.1 in explicit form. The univariate nodal basis functions are hereby transformed into multivariate hierarchical bases. To do this, we utilize that $U^i(f)$ can

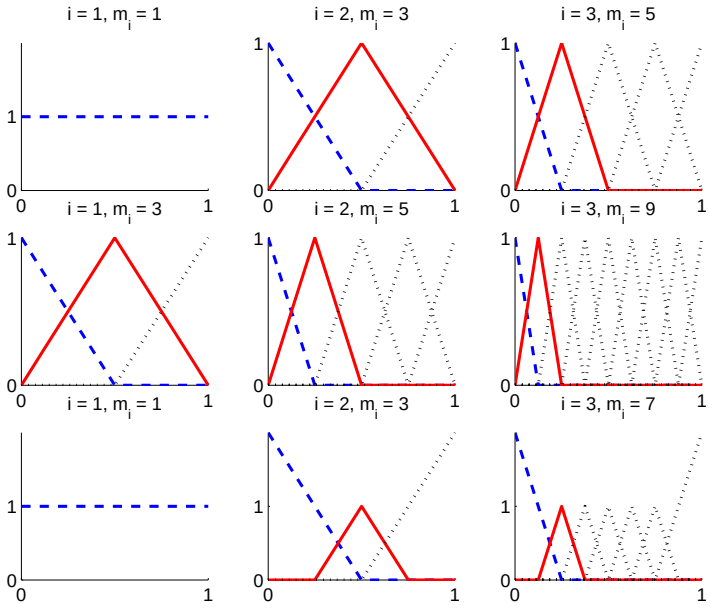


Figure 3: Nodal basis functions $a_{x_j^i}$ for H^{CC} (top), H^{M} (middle), and H^{NB} (bottom).

exactly represent $U^{i-1}(f)$. This is obviously the case for the piecewise linear basis functions chosen for the Clenshaw-Curtis grid and for the maximum-norm-based grid including the boundary. The nodal basis functions of the no-boundary-nodes grid, however, violate this assumption, and cannot be treated in this framework.

By taking advantage of the subset property $X^i \subset X^{i+1}$, we start with the transformation of the univariate nodal basis into the hierarchical one. By definition, we have

$$\Delta^i(f) = U^i(f) - U^{i-1}(f).$$

With

$$\begin{aligned} U^i(f) &= \sum_{x^i \in X^i} a_{x^i} \cdot f(x^i), \\ U^{i-1}(f) &= U^i(U^{i-1}(f)) \end{aligned}$$

we obtain

$$\begin{aligned} \Delta^i(f) &= \sum_{x^i \in X^i} a_{x^i} \cdot f(x^i) - \sum_{x^i \in X^i} a_{x^i} \cdot U^{i-1}(f)(x^i) \\ &= \sum_{x^i \in X^i} a_{x^i} \cdot \left(f(x^i) - U^{i-1}(f)(x^i)\right), \end{aligned}$$

and, since $f(x^i) - U^{i-1}(f)(x^i) = 0 \;\; \forall \; x^i \in X^{i-1}$,

$$\Delta^i(f) = \sum_{x^i \in X^i_\Delta} a_{x^i} \cdot \left(f(x^i) - U^{i-1}(f)(x^i)\right), \tag{6}$$

recalling $X^i_\Delta = X^i \setminus X^{i-1}$. Clearly, X^i_Δ has $m^\Delta_i = m_i - m_{i-1}$ elements, since $X^{i-1} \subset X^i$. By consecutively numbering the elements in X^i_Δ, and denoting the jth element of X^i_Δ as x^i_j, we can rewrite (6) as

$$\Delta^i(f) = \sum_{j=1}^{m^\Delta_i} a^i_j \cdot \underbrace{\big(f(x^i_j) - U^{i-1}(f)(x^i_j)\big)}_{w^i_j}, \tag{7}$$

To simplify the notation, we have replaced $a_{x^i_j}$ by a^i_j. Note that for all $\Delta^i(f)$, $i > 1$, we only have to consider the contributions of the basis functions belonging to the grid points that have not jet occurred in a previous set X^{i-k}, $1 \le k \le i-1$.

The tensor product formula (1) and the Smolyak algorithm (2) can now be applied to the Δ^i from equation (7) to obtain the sparse grid interpolation formula for the multivariate case in hierarchical form. Recalling Eq. (2), we have

$$A_{q,d}(f) = A_{q-1,d}(f) + \Delta A_{q,d}(f), \quad \text{with} \tag{8}$$

$$\Delta A_{q,d}(f) = \sum_{|\mathbf{i}|=q} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f), \tag{9}$$

with $A_{d-1,d} = 0$. We can express U^{i-1} in Eq. (7) by the following sum by using the telescopic property $\Delta^i = U^i - U^{i-1}$:

$$U^{i-1}(f) = \sum_{i_1=1}^{i-1} \Delta^{i_1}(f).$$

In case of the full grid, we would thus obtain a multivariate extension of an interpolant U^{i-1} with the tensor product formula (1) to

$$(U^{i-1} \otimes \cdots \otimes U^{i-1})(f) = \sum_{i_1=1}^{i-1} \cdots \sum_{i_d=1}^{i-1} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f).$$

However, the Smolyak formula $A_{q-1,d}$ only uses the support nodes and basis functions of the sparse grid with $|\mathbf{i}| \le q-1$, (between the index i of the univariate interpolation formula $U^i(f)$ and the parameter q of the Smolyak formula $A_{q,d}(f)$ constructed from $U^1(f), \ldots, U^i(f)$, we have the relation $q = i + d - 1$). Therefore, we use the reduced sum

$$A_{q-1,d}(f) = \sum_{|\mathbf{i}|\le q-1} (\Delta^{i_1} \otimes \cdots \otimes \Delta^{i_d})(f).$$

instead of $(U^{i-1} \otimes \cdots \otimes U^{i-1})(f)$ when constructing $\Delta A_{q,d}(f)$ in (9) from (7). Finally, we obtain the Smolyak algorithm in entirely hierarchical form with (8) and

$$\Delta A_{q,d}(f) = \sum_{|\mathbf{i}|=q} \sum_{\mathbf{j}} (a^{i_1}_{j_1} \otimes \cdots \otimes a^{i_d}_{j_d}) \cdot \underbrace{\Big(f(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d}) - A_{q-1,d}(f)(x^{i_1}_{j_1}, \ldots, x^{i_d}_{j_d})\Big)}_{w^{\mathbf{i}}_{\mathbf{j}}}, \tag{10}$$

with $\mathbf{j}$ denoting the multi-index $(j_1, \ldots, j_d)$, $j_l = 1, \ldots, m_{i_l}^{\Delta}$, and $l = 1, \ldots, d$. Note that all the required support nodes $\mathbf{x}_{\mathbf{j}}^{\mathbf{i}} = (x_{j_1}^{i_1}, \ldots, x_{j_d}^{i_d})$ of a single level k are included in the set $\Delta H_{k+d,d}$, as defined in Eq. (4). Furthermore, it is useful to define

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

as the *hierarchical surpluses* [4] at level k with $k = |\mathbf{i}| - d$, since $A_{k+d,d}$ 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.

Algorithm 1 summarizes the procedure of obtaining an increasingly accurate sparse grid interpolant via hierarchical construction. The hierarchical surpluses w^k are used to estimate the current interpolation error, and to formulate the break condition.

Algorithm 1 Compute $A_{q,d}(f)$ satisfying (estimated) error bounds $\delta_{\text{rel}}, \delta_{\text{abs}}$.

Initialization.

Let $f : [0,1]^d \to \mathbb{R}$, be the objective function.

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.

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 $\Delta H_{k+d,d}$ using Eq. (5).

Compute $Z_k = \{w^k \mid w^k = f(\mathbf{x}) - A_{k+d-1,d}(f)(\mathbf{x}),\ \mathbf{x} \in \Delta H_{k+d,d}\}$.

Construct $\Delta A_{k+d,d}(f)$ using Eq. (10) and Z_k.

Construct $A_{k+d,d}(f) = A_{k+d-1,d}(f) + \Delta A_{k+d,d}(f)$.

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

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

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

$k = k + 1$.

End

Let $q = k + d - 1$. Return $A_{q,d}(f)$.

As an example, let us consider the interpolation of a two-variate function ($d = 2$), and $q = 5$. We denote the two variables with x and y. We obtain the Smolyak algorithm summands to

$$\begin{aligned}
\Delta A_{2,2}(f) &= \sum_{|\mathbf{i}|=2} (\Delta^{i_1} \otimes \Delta^{i_2})(f) = (\Delta^1 \otimes \Delta^1)(f), \\
\Delta A_{3,2}(f) &= \sum_{|\mathbf{i}|=3} (\Delta^{i_1} \otimes \Delta^{i_2})(f) = (\Delta^2 \otimes \Delta^1)(f) + (\Delta^1 \otimes \Delta^2)(f), \\
\Delta A_{4,2}(f) &= (\Delta^3 \otimes \Delta^1)(f) + (\Delta^2 \otimes \Delta^2)(f) + (\Delta^1 \otimes \Delta^3)(f),
\end{aligned}$$

and thus

$$A_{5,2}(f) = \sum_{k=0}^{2} \Delta A_{k+2,2}(f) + \sum_{|\mathbf{i}|=5} (\Delta^{i_1} \otimes \Delta^{i_2})(f),$$

with, e.g.,

$$\begin{aligned}
(\Delta^1 \otimes \Delta^1)(f) &= a_1^1 \otimes a_1^1 \cdot f(x_1^1, y_1^1) = 1 \cdot f(x_1^1, y_1^1) \\
(\Delta^2 \otimes \Delta^1)(f) &= \sum_{j_1=1}^{m_2^\Delta} \sum_{j_2=1}^{m_1^\Delta} a_{j_1}^2 \otimes a_{j_2}^1 \cdot \left[f(x_{j_1}^2, y_{j_2}^1) - A_{2,2}(f)(x_{j_1}^2, y_{j_2}^1) \right] \\
(\Delta^3 \otimes \Delta^1)(f) &= \sum_{j_1=1}^{m_3^\Delta} \sum_{j_2=1}^{m_1^\Delta} a_{j_1}^3 \otimes a_{j_2}^1 \cdot \left[f(x_{j_1}^3, y_{j_2}^1) - A_{3,2}(f)(x_{j_1}^3, y_{j_2}^1) \right].
\end{aligned}$$

Fig. 4 shows all the resulting piecewise bilinear basis functions $a_{j_1}^{i_1} \otimes a_{j_2}^{i_2}$ needed to construct $A_{5,2}(f)$ for the Clenshaw-Curtis grid.

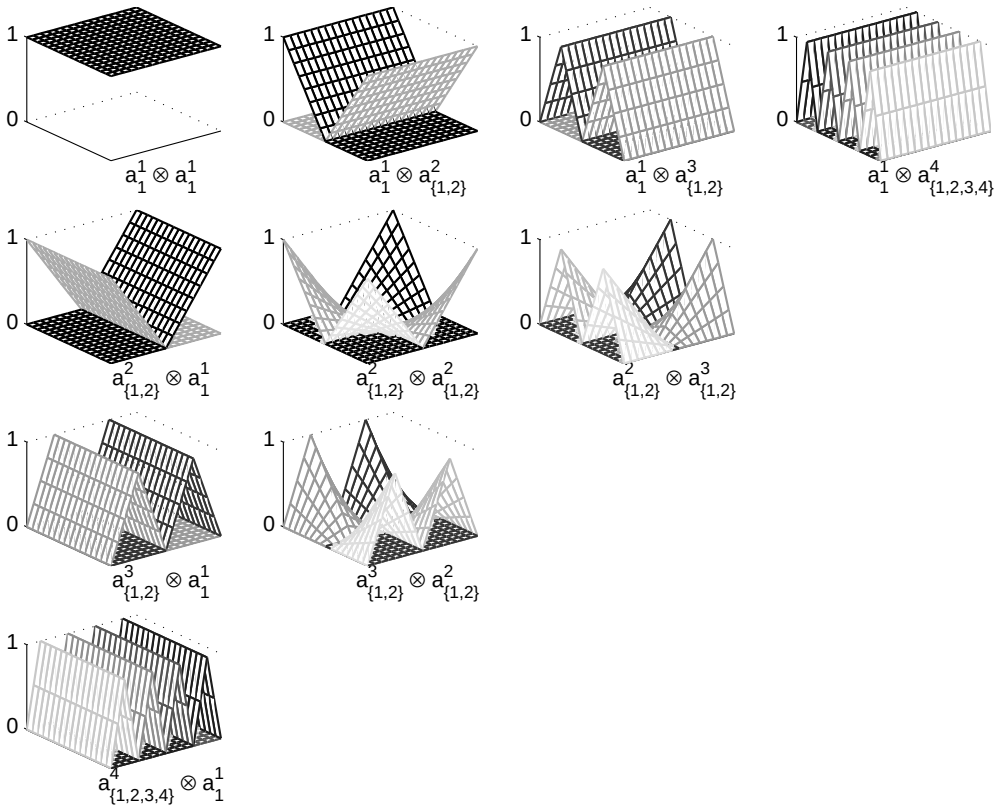


Figure 4: Bivariate hierarchical basis functions, Clenshaw-Curtis grid.

2.5 Accuracy of piecewise multilinear interpolation

We now take a brief look at the approximation quality. An a priori error estimate can be obtained for a d-variate function f if bounded mixed derivatives

$$D^{\alpha} f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots x_d^{\alpha_d}},$$

with $\alpha \in \mathbb{N}_0^d$, $|\alpha| = \sum_{i=1}^{d} \alpha_i$, and $\alpha_1, \ldots, \alpha_d \leq 2$, exist, i.e. $f \in F$ with

$$F := \left\{ f : [0,1]^d \to \mathbb{R},\ D^{\alpha} f \in C^0\big([0,1]^d\big),\ \alpha_1, \ldots, \alpha_d \leq 2) \right\}.$$

According to [1] or [2], the order of the interpolation error in the maximum norm is then given by

$$\|f - A_{q,d}(f)\|_{\infty} = \mathcal{O}(N^{-2} \cdot |\log_2 N|^{3\cdot(d-1)}), \tag{11}$$

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

3 Numerical results

We have conducted a comparison of the piecewise multilinear interpolation using the L_{∞}-based grids, with (H^{M}) and without (H^{NB}) boundary nodes and the Clenshaw-Curtis grid H^{CC}, using the testing package of Genz [6], designed for evaluating the performance of numerical integration algorithms. In [2], it was used to test the performance of sparse grid interpolation. The testing package consists of six families of functions defined on $[0,1]^d$ with the following characteristics:

$$\begin{aligned}
\text{oscillatory:}\quad & f_1(x) = \cos\left(2\pi w_1 + \sum_{i=1}^{d} c_i x_i\right), \\
\text{product peak:}\quad & f_2(x) = \prod_{i=1}^{d} \left(c_i^{-2} + (x_i - w_i)^2\right)^{-1}, \\
\text{corner peak:}\quad & f_3(x) = \left(1 + \sum_{i=1}^{d} c_i x_i\right)^{-(d+1)}, \\
\text{Gaussian:}\quad & f_4(x) = \exp\left(-\sum_{i=1}^{d} c_i^2 \cdot (x_i - w_i)^2\right), \\
\text{continuous:}\quad & f_5(x) = \exp\left(-\sum_{i=1}^{d} c_i \cdot |x_i - w_i|\right), \\
\text{discontinuous:}\quad & f_6(x) = \begin{cases} 0, & \text{if } x_1 > w_1 \text{ or } x_2 > w_2\ , \\ \exp\left(\sum_{i=1}^{d} c_i x_i\right), & \text{otherwise.} \end{cases}
\end{aligned}$$

Varying test functions can be obtained by altering the parameters $c = (c_1, \ldots, c_n)$ and $w = (w_1, \ldots, w_n)$. We chose these parameters randomly from $[0, 1]$. Similar to [2], we normalized the c_i such that $\sum_{i=1}^{d} c_i = b_j$, with b_j depending on d, f_j according to

j	1	2	3	4	5	6
b_j	1.5	d	1.85	7.03	20.4	4.3

Furthermore, we normalized the w_i such that $\sum_{i=1}^{d} w_i = 1$.

Fig. 5 shows the absolute error

$$e = \max_{i=1,\ldots,100} |f(y_i) - A_{q,d}(f)(y_i)| \tag{12}$$

over the total number of sparse grid points N for $d = 3$ and $d = 5$, respectively, for the six test functions. In (12), we have replaced the error norm (11) by a discrete approximation. The points $y_1, \ldots, y_{100} \in [0, 1]^d$ were generated randomly. As expected, all schemes show the same asymptotic behavior according to (11) in case of $f \in F$. However, for the oscillatory, the product peak, and the Gaussian-shaped test functions, interpolation with H^{CC} and H^{NB} produced better results than H^{M}, i.e. less support nodes were needed to achieve similar accuracies. In case of the corner peak test function, H^{M} sometimes performed slightly better. We assume that this difference results from the different distribution of the support nodes (see Figs. 1,2), which is more dense for H^{M} at the boundary. In case of the continuous and the discontinuous test functions, all three grid types could not provide the asymptotic error decay of (11), since $f \notin F$.

4 Implementation in Matlab

In this section, we describe our Matlab implementation for piecewise multilinear sparse grid interpolation. While we suggest to use the Clenshaw-Curtis grid due to the test results of the previous section, the user may easily select any of the two other grid types described in this paper. To make the tool as easy as possible to use, we decided to follow the approach of [10]. All sparse grid interpolation routines can be called in the same way, and providing additional options is optional.

4.1 Description of the user interface

The following functions comprise our implementation. For further details on the syntax of each function, please use `help <function_name>` within MATLAB.

- `init.m`: Calling this file will add the relevant directories containing the sparse grid algorithms to your `Matlab` path. Furthermore, it displays a list of available demonstration m-files.
- `spvals.m`: Determines the hierarchical surpluses of the sparse grid for a `d`-variate function `fun` over a specified interval box. By default, the Clenshaw-Curtis grid is used. The function `fun` may be an inline function, a function handle, or a function file, and

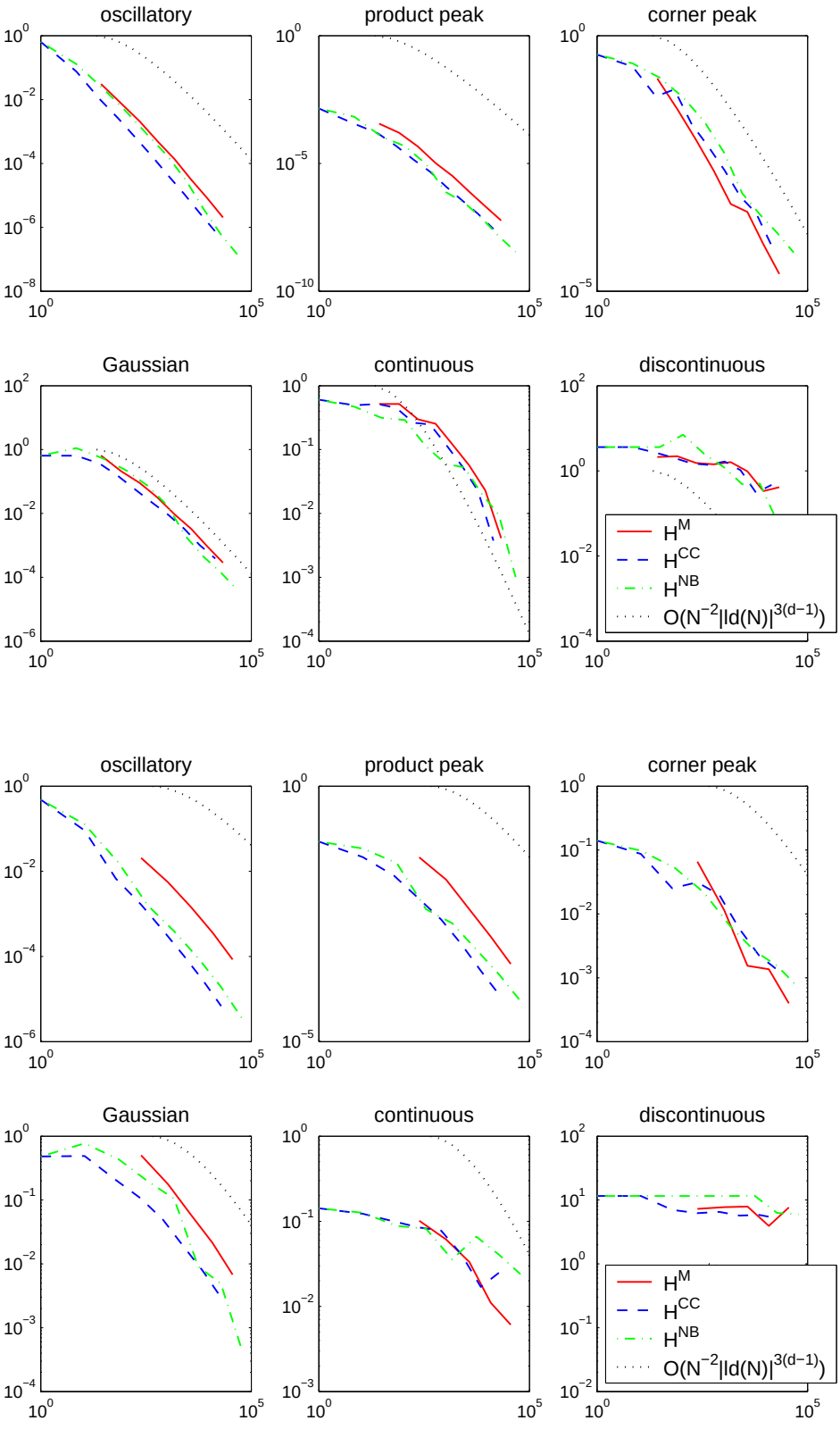


Figure 5: Error plots for $A_{q,d}(f_{1,\ldots,6})$ with H^{M}, H^{NB} and H^{CC}, $d = 3$ (top) and $d = 5$ (bottom). The absolute error according to (12) is plotted over the number of grid points N.

must accept `d` parameters (unless other parameters `p1`, `p2`, ... (see below) are passed to the function). The following syntax options are available:

`z = spvals(fun, d)`: Computes the sparse grid representation `z` of a function `fun`. The grid is computed over the `d`-dimensional unit cube $[0,1]^d$.

`z = spvals(fun, d, range)`: If the range of values is not the unit cube, you may specify the `range` of the input parameters as a `d`×2 matrix. E.g. if the range of parameter 1 is $[0,2]$, and the range of parameter 2 is $[1,4]$, you would set `range` to `[0 2; 1 4]`.

`z = spvals(fun, d, range, options)`: In addition to the simple syntax, you may pass an `options` structure to `spvals`, which can be used to change the default sparse grid interpolation parameters. Create the options argument with the `spset` function. For an explanation of the available options, please refer to Table 2.

`z = spvals(fun, d, range, options, p1, p2, ...)`: Any input parameters after the options structure are passed as additional parameters to the function `fun` during the function evaluation process within `spvals`.

As result, `spvals` returns a structure containing the sparse grid representation including some statistical information (see Table 3 for more information). You may access each property with `z.<propertyName>`.

- `spinterp.m`: Once the sparse representation of a function has been determined with `spvals`, one can compute interpolated values with `spinterp` for any point `y` $= (y_1, \ldots, y_d)$. The syntax is

  ```
  ip = spinterp(z, y1, ..., yd)
  ```

  Here is a simple example for computing the sparse grid data for a function f with $d = 3$. Then, the interpolated value at the point $(0.5, 0.2, 0.2)$ and the absolute interpolation error is determined.

  ```
  >> f = inline('x^2 + y^2 - 2*z');
  >> z = spvals(f,3);
  >> f_interp = spinterp(z, 0.5, 0.2, 0.2)
  f_interp =
     -0.1063
  >> error = abs(f(0.5, 0.2, 0.2) - f_interp)
  error =
      0.0037
  ```

  You may also call `spinterp` for multiple points at once, with `y1`, ..., `yn` containing vectors or matrices of equal dimension, i.e. `size(y1)` = ... = `size(yd)`.

- `spset`: The `spset` function creates an options structure that you can supply to the `spvals` function. `spset` accepts property name/property value pairs using the syntax

  ```
  >> options = spset('name1', value1, 'name2', value2, ...)
  ```

  The function is case-insensitive. Called with no input parameters, `spset` displays the possible values and the defaults:

Table 2: Available options configurable with `spset`.

Property	*Value*	*Description*
`GridType`	String	The grid type. The default type is `'Clenshaw-Curtis'`, other possibly values are `'Maximum'` and `'NoBoundary'`.
`RelTol`	Scalar	A relative error tolerance that applies to all hierarchical surpluses w^k of the current deepest level k of the sparse grid interpolation formula. The default value is 10^{-2} (1 % accuracy). The interpolation depth level k is increased until all w^k are less than $\max\left(\texttt{RelTol} \cdot (\max(\texttt{fvals}) - \min(\texttt{fvals})), \texttt{AbsTol}\right)$, with `fvals` containing all results evaluating `fun` up to that point.
`AbsTol`	Scalar	Absolute error tolerance. The default value is 10^{-6}.
`Vectorized`	on | off	Indicates if `fun` is available for vectorized evaluation. The default value is `'off'`. Vectorized coding of `fun` can significantly reduce the computation time used by `spvals`.
`MinDepth`	Integer	Minimum number of levels $n = q - d$ to compute (default is 2).
`MaxDepth`	Integer	Maximum number of levels to compute (default is 8).
`VariablePositions`	Vector	Sometimes it is useful to change the order of inputs to `fun`. Please refer to `help spset` for additional information.
`NumberOfOutputs`	Integer	If `fun` produces multiple outputs (all must be scalar), indicate this here to perform the sparse grid computation for many output variables at once. Also see the example `spdemovarout.m`.
`PrevResults`	Structure	An existing result structure obtained from `spvals` may be provided to further refine an existing sparse grid.

Table 3: `spvals` output properties.

Property	*Value*	*Description*
`vals`	Cell array	Contains `maxLevel`+1 matrices of the hierarchical surpluses of each interpolation depth level.
`gridType`	String	The grid type.
`d`	Integer	The problem dimension.
`range`	Matrix	The range of the input parameters. An empty matrix indicates the interval box $[0, 1]^d$.
`maxLevel`	Integer	The actual interpolation depth of the sparse grid representation. It depends on the requested minimum/maximum interpolation depth `MinDepth`/`MaxDepth` and the requested error tolerances `RelTol`/`AbsTol`.
`estRelError`	Scalar	The estimated relative error e with respect to `fevalRange`: $e = \frac{\max(\text{surpluses\_of\_deepest\_level } w^k)}{\max(\text{fevalRange}) - \min(\text{fevalRange})}$
`fevalRange`	Matrix	The minimum and maximum of all function values encountered during the evaluation of `fun` at the sparse grid support points. Each row of the matrix contains the results of one output parameter.
`nPoints`	Integer	The number of grid points.
`fevalTime`	Scalar	The amount of time spent evaluating `fun` at the sparse grid points.
`surplusCompTime`	Scalar	Time spent computing the hierarchical surpluses.

```
>> spset
         GridType: [ {Clenshaw-Curtis} | Maximum | NoBoundary ]
           RelTol: [ positive scalar {1e-2} ]
           AbsTol: [ positive scalar {1e-6} ]
       Vectorized: [ on | {off} ]
         MinDepth: [ positive integer {2} ]
         MaxDepth: [ positive integer {8} ]
VariablePositions: [ double matrix {[]} ]
  NumberOfOutputs: [ positive integer {1} ]
      PrevResults: [ struct {[]} ]
```

To modify an existing options structure, us the syntax

```
>> options = spset(oldoptions, 'name1', value1, ...)
```

This sets `options` equal to the existing options structure `oldoptions` and adds or overwrites the specified name/value pairs.

- `spget.m`: You may query an existing options structure with `spget`. The syntax is

  ```
  >> value = spget(options, 'name')
  ```

  If the property `name` exists, the current value is returned, otherwise, an empty matrix `[]`.

- `spdim.m`: Computes the number of points of the sparse grid $H_{q,d}$ (by default the Clenshaw-Curtis grid H^{CC}). You may use an options structure to select other grid types. The syntax options are

  ```
  >> spdim(n,d)
  >> spdim(n,d,options)
  ```

  where `n` denotes the interpolation depth $n = q - d$ of the corresponding interpolation formula $A_{q,d}(f)$. For example, to compute the number of points of $H^{\mathrm{CC}}_{17,10}$, use the syntax

  ```
  >> spdim(7,10)
  ans =
        652065
  ```

  with $n = q - d = 7$.

- `spgrid.m`: Explicitly computes the coordinates of the points of a single level of a sparse grid (the Clenshaw-Curtis grid by default). This function is internally used by `spvals` to compute the grid points that f needs to be evaluated for, and is usually not required to be called by the user.

- `plotgrid.m`: Plots the sparse grid for the dimensions `d` $= 2$ and `d` $= 3$ (the Clenshaw-Curtis grid is used by default). You may use an options structure to select other grid types. The syntax options are

  ```
  >> plotgrid(n,d)
  >> plotgrid(n,d,options)
  ```

where `n` denotes the interpolation depth $n = q - d$ of the corresponding interpolation formula $A_{q,d}(f)$.

- `spdemo.m`: A script file demonstrating a simple example in two dimensions, including multiple evaluations of the interpolating function $A_{q,d}(f)$ through a vectorized call of `spinterp`. Other demos are available (see `init.m`).

4.2 Performance test results

The source code has been optimized for Matlab Version 6.5 and JIT compilation. We have measured the performance of our implementation under Linux on a i686 PC with 1.4 GHz.

Fig. 6 (**a**) shows the time to compute the sparse grid representation (i.e. the hierarchical surpluses) of a d-variate function, depending on d and the number of sparse grid points N. The legend shows the time taken to determine $A_{q,d}(f)$ for the maximum level $n = q - d$. E.g., for $d = 4$, $n = 10$, $q = n + d = 14$, it took $t(10) = 39.8$ sec. to compute $A_{14,4}(f)$. Fig. 6 (**b**) shows the time to compute 1000 interpolated values depending on the number of sparse grid points and the problem dimension d. The legend shows the time required to evaluate $A_{q,d}(f)$ for the maximum level $n = q - d$.

4.3 Obtaining the source code

The complete source code is available for download [7].

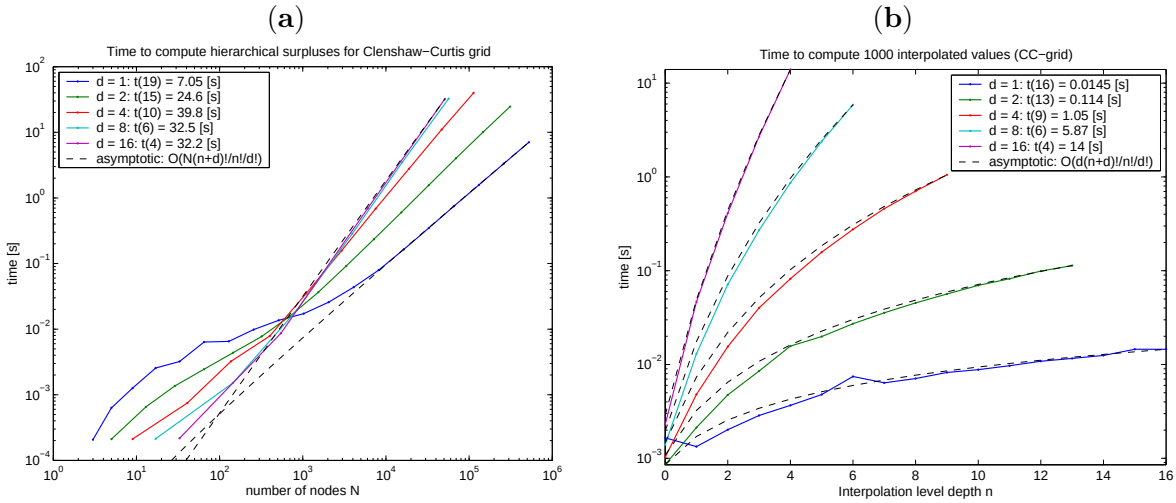


Figure 6: (**a**): Performance determining $A_{q,d}(f)$. N denotes the number of sparse grid points. (**b**): Performance evaluating $A_{q,d}(f)$ 1000 times. The computation time is plotted over the interpolation level depth n.

References

[1] S. Achatz. *Adaptive finite Dünngitter-Elemente höherer Ordnung für elliptische partielle Differentialgleichungen mit variablen Koeffizienten*. PhD thesis, Technische Universität München, 2003.

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

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

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

[5] C. W. Clenshaw and A. R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2:197–205, 1960.

[6] A. C. Genz. A package for testing multiple integration subroutines. In P. Keast and G. Fairweather, editors, *Numerical Integration*, pages 337–340. Kluwer, Dordrecht, 1987.

[7] Institute of Applied Analysis and Numerical Simulation, University of Stuttgart. Scientific/Educational Matlab Database [online]. 2002 [cited August 15, 2003]. Available online: `http://matlabdb.mathematik.uni-stuttgart.de`.

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

[9] A. Schreiber. *Smolyak's method for multivariate interpolation*. PhD thesis, Georg-August-Universität Göttingen, Germany, 2000.

[10] L. F. Shampine and M. W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, 18(1):1–22, 1997.

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

[12] F. Sprengel. Interpolation and wavelets on sparse Gauss-Chebyshev grids. In W. Haussmann et al., editor, *Multivariate Approximation*, volume 101 of *Mathematical Research*, pages 269–286. Akademie Verlag, Berlin, 1997.

[13] F. Sprengel. Periodic interpolation and wavelets on sparse grids. *Numerical Algorithms*, 17:147–169, 1998.

[14] C. Zenger. Sparse grids. In *Parallel Algorithms for Partial Differential Equations*, volume 31 of *Notes Numer. Fluid Mech.*, pages 241–251. Vieweg, Braunschweig, 1991.

[15] S. Zimmer. Private communication, 2003.

Andreas Klimke

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

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

Erschienene Preprints ab Nummer 2003/001

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

2003/001 *Lamichhane, B. P., Wohlmuth, B. I.*: Mortar Finite Elements for Interface Problems.

2003/002 *Dryja, M., Gantner, A., Widlund, O. B., Wohlmuth, B. I.*: Multilevel Additive Schwarz Preconditioner For Nonconforming Mortar Finite Element Methods.

2003/003 *Klimke, A., Hanss, M.*: On the Reliability of the Influence Measure in the Transformation Method of Fuzzy Arithmetic.

2003/004 *Klimke, A.*: RANDEXPR: A Random Symbolic Expression Generator.

2003/005 *Klimke, A.*: How to Access Matlab from Java.

2003/006 *Merkle, T.*: Phase separation in solid mixtures under elastic loadings with application to solder materials.

2003/007 *Lamichhane, B. P., Wohlmuth, B. I.*: Second Order Lagrange Multiplier Spaces for Mortar Finite Elements in 3D.

2003/008 *Fritz, A., Hüeber, S., Wohlmuth, B. I.*: A comparison of mortar and Nitsche techniques for linear elasticity.

2003/009 *Klimke, A.*: An Efficient Implementation of the Transformation Method of Fuzzy Arithmetic

2003/010 *Steinbach, O.*: A Note on the Ellipticity of the Single Layer Potential in two-dimensional Linear Elastostatics

2003/011 *Steinbach, O.*: Artificial Multilevel Boundary Element Preconditioners

2003/012 *Bürger, R., Karlsen, K.H.*: On a diffusively corrected kinematic-wave traffic model with changing road surface conditions

2003/013 *Wohlmuth, B. I.*: A short note on: An optimal a priori estimate for non linear multibody contact problems.

2003/014 *Sändig, A.-M.*: Distributionentheorie mit Anwendungen auf partielle Differentialgleichungen. Vorlesung im Wintersemester 2002/03.

2003/015 *Sändig, A.-M.*: Proseminar Funktionenrume, Wintersemester 2002/2003.

2003/016 *Merkle, T.*: An energy method for the strong nonlinear Cahn-Larché equation system.

2003/017 *Bürger, R.*: Mathematische Modelle für Mehrkomponentenströmungen.

2003/018 *Langer, U., Steinbach, O., Wendland, W.L. (eds.)*: Workshop on Fast Boundary Element Methods in Industrial Applications Söllerhaus, 15.-18.10.2003

2003/019 *Klimke, A.*: Piecewise Multilinear Sparse Grid Interpolation in Matlab.