Sparse Grids in a Nutshell

Jochen Garcke

Abstract The technique of sparse grids allows to overcome the curse of dimensionality, which prevents the use of classical numerical discretization schemes in more than three or four dimensions, under suitable regularity assumptions. The approach is obtained from a multi-scale basis by a tensor product construction and subsequent truncation of the resulting multiresolution series expansion. This entry level article gives an introduction to sparse grids and the sparse grid combination technique.

1 Introduction

The sparse grid method is a special discretization technique, which allows to cope with the curse of dimensionality of grid based approaches to some extent. It is based on a hierarchical basis [13, 42, 43], a representation of a discrete function space which is equivalent to the conventional nodal basis, and a sparse tensor product construction.

The sparse grid method was originally developed for the solution of partial differential equations [45, 22, 5]. Besides working directly in the hierarchical basis a sparse grid representation of a function can also be computed using the combination technique [25], here a certain sequence of partial functions represented in the conventional nodal basis is linearly combined. The sparse grid method in both its formulations is nowadays successfully used in many applications.

The underlying idea of sparse grids can be traced back to the Russian mathematician Smolyak [37], who used it for numerical integration. The concept is also closely

Jochen Garcke
Institut für Numerische Simulation
Universität Bonn
Wegelerstr. 6
D-53115 Bonn
e-mail: garcke@ins.uni-bonn.de

related to hyperbolic crosses [2, 38, 39, 40], boolean methods [11, 12], discrete blending methods [4] and splitting extrapolation methods [31].

For the representation of a function f defined over a d-dimensional domain the sparse grid approach employs $\mathcal{O}(h_n^{-1} \cdot \log(h_n^{-1})^{d-1})$ grid points in the discretization process, where $h_n := 2^{-n}$ denotes the mesh size and n is the discretization level. It can be shown that the order of approximation to describe a function f, under certain smoothness conditions, is $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1})^{d-1})$. This is in contrast to conventional grid methods, which need $\mathcal{O}(h_n^{-d})$ for an accuracy of $\mathcal{O}(h_n^2)$. Therefore, to achieve a similar approximation quality sparse grids need much less points in higher dimensions than regular full grids. The curse of dimensionality of full grid method arises for sparse grids to a much smaller extent and they can be used for higher dimensional problems.

For ease of presentation we will consider the domain $\Omega = [0,1]^d$ in the following. This situation can be achieved for bounded rectangular domains by a proper rescaling.

2 Sparse grids

We introduce some notation while describing the conventional case of a piecewise linear finite element basis. Let $\underline{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$ denote a multi-index. We define the anisotropic grid $\Omega_{\underline{l}}$ on $\bar{\Omega}$ with mesh size $h_{\underline{l}} := (h_{l_1}, \ldots, h_{l_d}) = 2^{-\underline{l}} := (2^{-l_1}, \ldots, 2^{-l_d})$; $\Omega_{\underline{l}}$ has different, but equidistant mesh sizes h_{l_t} in each coordinate direction t, $t = 1, \ldots, d$. This way the grid $\Omega_{\underline{l}}$ consists of the points

$$x_{\underline{l},\underline{j}} := (x_{l_1,j_1}, \ldots, x_{l_d,j_d}), \tag{1}$$

with $x_{l_t,j_t} := j_t \cdot h_{l_t} = j_t \cdot 2^{-l_t}$ and $j_t = 0, \ldots, 2^{l_t}$. For a grid $\Omega_{\underline{l}}$ we define an associated space $V_{\underline{l}}$ of piecewise d-linear functions[1]

$$V_{\underline{l}} := \operatorname{span}\{\phi_{\underline{l},\underline{j}} \mid j_t = 0, \ldots, 2^{l_t},\ t = 1, \ldots, d\} = \operatorname{span}\{\phi_{\underline{l},\underline{j}} \mid 0 \le \underline{j} \le 2^{\underline{l}}\}, \tag{2}$$

which is spanned by the usual basis of d-dimensional piecewise d-linear hat functions

$$\phi_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \phi_{l_t,j_t}(x_t). \tag{3}$$

The one-dimensional functions $\phi_{l,j}(x)$ with support $[x_{l,j} - h_l, x_{l,j} + h_l] \cap [0,1] = [(j-1)h_l, (j+1)h_l] \cap [0,1]$ are defined by:

$$\phi_{l,j}(x) = \begin{cases} 1 - |x/h_l - j|, & x \in [(j-1)h_l, (j+1)h_l] \cap [0,1], \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

[1] "$\le$" refers to the element-wise relation for multi-indices: $\underline{k} \le \underline{l} :\Leftrightarrow \forall_t k_t \le l_t$. Furthermore, $a \le \underline{l}$ implies $\forall_t a \le l_t$.

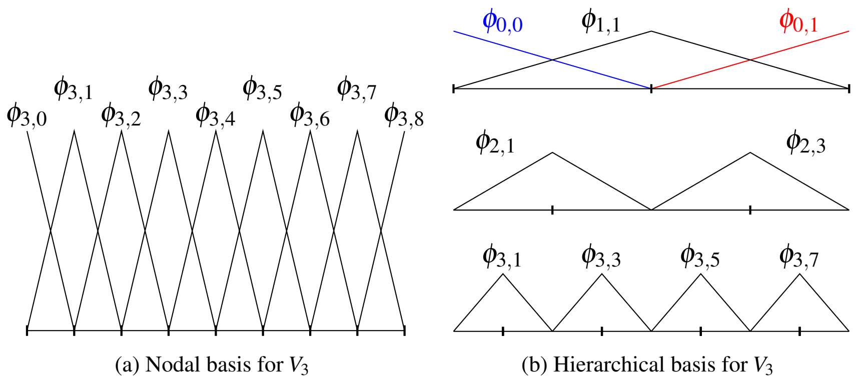

Fig. 1: Nodal and hierarchical basis of level $n = 3$.

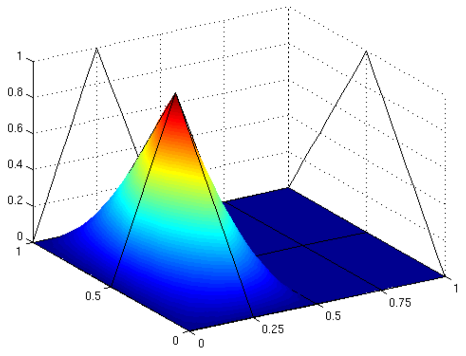

Fig. 2: Basis function $\phi_{1,1}$ on grid $\Omega_{2,1}$.

In Figure 1a we give an example for the one-dimensional case and show all $\phi_{l,j} \in V_3$. Figure 2 shows a two-dimensional basis function.

2.1 Hierarchical subspace-splitting

Till now and in the following the multi-index $\underline{l} \in \mathbb{N}^d$ denotes the level, i.e. the discretization resolution, be it of a grid $\Omega_{\underline{l}}$, a space $V_{\underline{l}}$, or a function $f_{\underline{l}}$, whereas the multi-index $\underline{j} \in \mathbb{N}^d$ gives the spatial position of a grid point $x_{\underline{l},\underline{j}}$ or the corresponding basis function $\phi_{\underline{l},\underline{j}}(\cdot)$.

We now define a hierarchical difference space $W_{\underline{l}}$ via

$$W_{\underline{l}} := V_{\underline{l}} \setminus \bigoplus_{t=1}^{d} V_{\underline{l}-\underline{e}_t}, \tag{5}$$

where $\underline{e}_t$ is the t-th unit vector. In other words, $W_{\underline{l}}$ consists of all $\phi_{\underline{k},\underline{j}} \in V_{\underline{l}}$ (using the hierarchical basis) which are not included in any of the spaces $V_{\underline{k}}$ smaller[2] than $V_{\underline{l}}$. To complete the definition, we formally set $V_{\underline{l}} := 0$, if $l_t = -1$ for at least one $t \in \{1,\dots,d\}$. As can easily be seen from (2) and (5), the definition of the index set

$$\mathsf{B}_{\underline{l}} := \left\{ \underline{j} \in \mathbb{N}^d \,\middle|\, \begin{array}{ll} j_t = 1,\dots,2^{l_t}-1, \quad j_t \text{ odd}, t = 1,\dots,d, \text{ if } l_t > 0, \\ j_t = 0,1, \qquad\qquad\qquad\quad t = 1,\dots,d, \text{ if } l_t = 0 \end{array} \right\} \tag{6}$$

leads to

$$W_{\underline{l}} = \operatorname{span}\{\phi_{\underline{l},\underline{j}} | \underline{j} \in \mathsf{B}_{\underline{l}}\}. \tag{7}$$

These hierarchical difference spaces now allow us the definition of a multilevel subspace decomposition. We can write $V_n := V_{\underline{n}}$ as a direct sum of subspaces

$$V_n := \bigoplus_{l_1=0}^{n} \cdots \bigoplus_{l_d=0}^{n} W_{\underline{l}} = \bigoplus_{|\underline{l}|_\infty \le n} W_{\underline{l}}. \tag{8}$$

Here, $|\underline{l}|_\infty := \max_{1 \le t \le d} l_t$ and $|\underline{l}|_1 := \sum_{t=1}^{d} l_t$ are the discrete ℓ_∞- and the discrete ℓ_1-norm of $\underline{l}$, respectively.

The family of functions

$$\{\phi_{\underline{l},\underline{j}} | \underline{j} \in \mathsf{B}_{\underline{l}}\}_{\underline{l}=\underline{0}}^{\underline{n}} \tag{9}$$

is just the hierarchical basis [13, 42, 43] of V_n, which generalizes the one-dimensional hierarchical basis [13], see Figure 1b, to the d-dimensional case with a tensor product ansatz. Observe that the supports of the basis functions $\phi_{\underline{l},\underline{j}}(\underline{x})$, which span $W_{\underline{l}}$, are disjunct for $\underline{l} > 0$. See Figure 3 for a representation of the supports of the basis functions of the difference spaces W_{l_1,l_2} forming V_3.

Now, each function $f \in V_n$ can be represented as

$$f(\underline{x}) = \sum_{|\underline{l}|_\infty \le n} \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \cdot \phi_{\underline{l},\underline{j}}(\underline{x}) = \sum_{|\underline{l}|_\infty \le n} f_{\underline{l}}(\underline{x}), \quad \text{with } f_{\underline{l}} \in W_{\underline{l}}, \tag{10}$$

where $\alpha_{\underline{l},\underline{j}} \in \mathbb{R}$ are the coefficients of the representation in the hierarchical tensor product basis and $f_{\underline{l}}$ denotes the hierarchical component functions. The number of basis functions which describe a $f \in V_n$ in nodal or hierarchical basis is $(2^n+1)^d$. For example a resolution of 17 points in each dimensions, i.e. $n = 4$, for a ten-dimensional problem therefore needs $2 \cdot 10^{12}$ coefficients, we encounter the curse of dimensionality.

Furthermore, we can define

[2] We call a discrete space $V_{\underline{k}}$ smaller than a space $V_{\underline{l}}$ if $\forall_t k_t \le l_t$ and $\exists t : k_t < l_t$. In the same way a grid $\Omega_{\underline{k}}$ is smaller than a grid $\Omega_{\underline{l}}$.

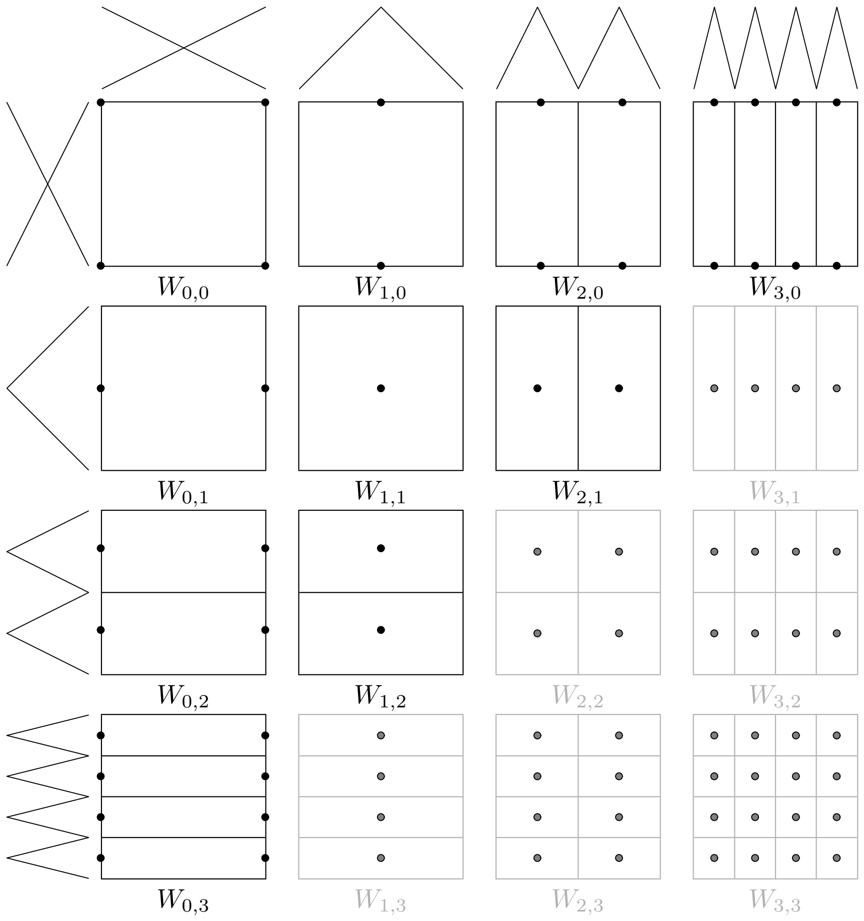

Fig. 3: Supports of the basis functions of the hierarchical subspaces $W_{\underline{l}}$ of the space V_3. The sparse grid space V_3^s contains the upper triangle of spaces shown in black.

$$V := \lim_{n\to\infty} \bigoplus_{\underline{k}\le n} W_{\underline{k}},$$

which by a completion with respect to the H^1-norm, is simply the underlying Sobolev space H^1, i.e. $\overline{V}^{H^1} = H^1$. Any function $f \in V$ can be uniquely decomposed as [8]

$$f(\underline{x}) = \sum_{\underline{l}\in\mathbb{N}^d} f_{\underline{l}}(\underline{x}), \qquad \text{with } f_{\underline{l}} \in W_{\underline{l}}.$$

Note also, that for the spaces $V_{\underline{l}}$ the following decomposition holds

$$V_{\underline{l}} := \bigoplus_{k_1=0}^{l_1} \cdots \bigoplus_{k_d=0}^{l_d} W_{\underline{k}} = \bigoplus_{\underline{k} \leq \underline{l}} W_{\underline{k}}.$$

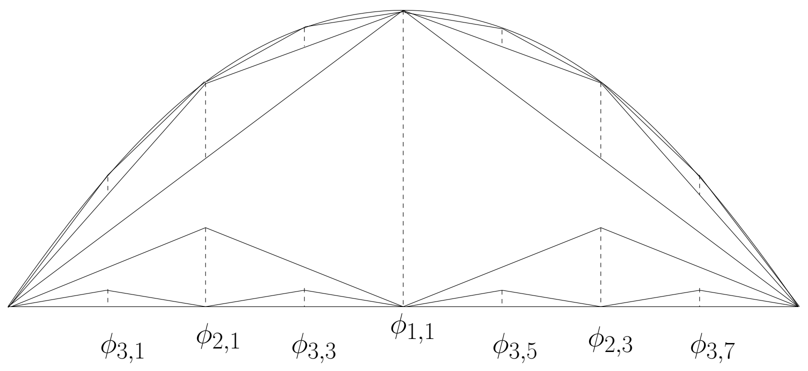

Fig. 4: Interpolation of a parabola with the hierarchical basis of level $n = 3$.

2.2 *Properties of the hierarchical subspaces*

Now consider the d-linear interpolation of a function $f \in V$ by a $f_n \in V_n$, i.e. a representation as in (10). First we look at the linear interpolation in one dimension, for the hierarchical coefficients $\alpha_{l,j}$, $l \geq 1$, j odd, holds

$$\begin{aligned}\alpha_{l,j} &= f(x_{l,j}) - \frac{f(x_{l,j} - h_l) + f(x_{l,j} + h_l)}{2} = f(x_{l,j}) - \frac{f(x_{l,j-1}) + f(x_{l,j+1})}{2} \\ &= f(x_{l,j}) - \frac{f(x_{l-1,(j-1)/2}) + f(x_{l-1,(j+1)/2})}{2}.\end{aligned}$$

This and Figure 4 illustrate why the $\alpha_{l,j}$ are also called *hierarchical surplus*, they specify what has to be added to the hierarchical representation from level $l-1$ to obtain the one of level l. We can rewrite this in the following operator form

$$\alpha_{l,j} = \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}_{l,j} f$$

and with that we generalize to the d-dimensional hierarchization operator as follows

$$\alpha_{\underline{l},\underline{j}} = \left(\prod_{t=1}^{d} \begin{bmatrix} -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix}_{l_t,j_t} \right) f. \tag{11}$$

Note that the coefficients for the basis functions associated to the boundary are just $\alpha_{0,j} = f(x_{0,j}), j = 0,1.$

Now let us define the so-called Sobolev-space with dominating mixed derivative H^2_{mix} in which we then will show approximation properties of the hierarchical basis. First we consider mixed derivatives and define

$$D^{\underline{k}} f := \frac{\partial^{|\underline{k}|_1} f}{\partial \underline{x}^{\underline{k}}} = \frac{\partial^{|\underline{k}|_1} f}{\partial x_1^{k_1} \cdots \partial x_d^{k_d}}$$

With that we define the norm as

$$\|f\|^2_{H^s_{mix}} = \sum_{0 \le \underline{k} \le s} \left| \frac{\partial^{|\underline{k}|_1}}{\partial \underline{x}^{\underline{k}}} f \right|^2_2 = \sum_{0 \le \underline{k} \le s} \left| D^{\underline{k}} f \right|^2_2,$$

and the space H^s_{mix} in the usual way:

$$H^s_{mix} := \left\{ f : \Omega \to \mathbb{R} \,\middle|\, \|f\|^2_{H^s_{mix}} < \infty \right\}.$$

Obviously it holds $H^s_{mix} \subset H^s$. Furthermore we define the semi-norm $|f|_{H^2_{mix}} := |f|_{H^{\underline{2}}_{mix}}$ by

$$|f|_{H^{\underline{k}}_{mix}} := \left| \frac{\partial^{|\underline{k}|_1}}{\partial \underline{x}^{\underline{k}}} f \right|_2 = \left| D^{\underline{k}} f \right|_2.$$

Note that the continuous function spaces H^s_{mix}, like the discrete spaces $V_{\underline{l}}$, have a tensor product structure [41, 28, 24, 29] and can be represented as a tensor product of one dimensional spaces:

$$H^s_{mix} = H^s \otimes \cdots \otimes H^s.$$

We now look at the properties of the hierarchical representation of a function f, especially at the size of the hierarchical surpluses. We recite the following proofs from [5, 6, 7, 8], see these references for more details on the following and results in other norms like $\|\cdot\|_\infty$ or $\|\cdot\|_E$. For ease of presentation we assume $f \in H^2_{0,mix}(\bar{\Omega})$, i.e. zero boundary values, and $\underline{l} > 0$ to avoid the special treatment of level 0, i.e. the boundary functions in the hierarchical representation.

Lemma 1 *For any piecewise d-linear basis function $\phi_{\underline{l},\underline{j}}$ holds*

$$\|\phi_{\underline{l},\underline{j}}\|_2 = \left(\frac{2}{3}\right)^{d/2} \cdot 2^{-|\underline{l}|_1/2}.$$

Proof. Follows by straightforward calculation.

Lemma 2 *For any hierarchical coefficient $\alpha_{\underline{l},\underline{j}}$ of $f \in H^2_{0,mix}(\bar{\Omega})$ in (10) it holds*

$$\alpha_{\underline{l},\underline{j}} = \prod_{t=1}^{d} -\frac{h_{l_t}}{2} \int_\Omega \phi_{\underline{l},\underline{j}} \cdot D^{\underline{2}} f(\underline{x}) d\underline{x}. \tag{12}$$

Proof. In one dimension partial integration provides

$$\begin{aligned}
\int_\Omega \phi_{l,j}(x) \cdot \frac{\partial^2 f(x)}{\partial x^2} dx &= \int_{x_{l,j}-h_l}^{x_{l,j}+h_l} \phi_{l,j}(x) \cdot \frac{\partial^2 f(x)}{\partial x^2} dx \\
&= \left[\phi_{l,j}(x) \cdot \frac{\partial f(x)}{\partial x} \right]_{x_{l,j}-h_l}^{x_{l,j}+h_l} - \int_{x_{l,j}-h_l}^{x_{l,j}+h_l} \frac{\partial \phi_{l,j}(x)}{\partial x} \cdot \frac{\partial f(x)}{\partial x} dx \\
&= - \int_{x_{l,j}-h_l}^{x_{l,j}} \frac{1}{h_l} \cdot \frac{\partial f(x)}{\partial x} dx + \int_{x_{l,j}}^{x_{l,j}+h_l} \frac{1}{h_l} \cdot \frac{\partial f(x)}{\partial x} dx \\
&= \frac{1}{h_l} \cdot \big(f(x_{l,j} - h_l) - 2 f(x_{l,j}) + f(x_{l,j} + h_l) \big) \\
&= -\frac{2}{h_l} \cdot \alpha_{l,j}.
\end{aligned}$$

The d-dimensional result is achieved via the tensor product formulation (11).

Lemma 3 *Let $f \in H^2_{0,mix}(\bar{\Omega})$ be in hierarchical representation as above, it holds*

$$|\alpha_{\underline{l},\underline{j}}| \leq \frac{1}{6^{d/2}} \cdot 2^{-(3/2)\cdot|\underline{l}|_1} \cdot \left| f|_{supp(\phi_{\underline{l},\underline{j}})} \right|_{H^2_{mix}}.$$

Proof.

$$\begin{aligned}
|\alpha_{\underline{l},\underline{j}}| &= \left| \prod_{t=1}^{d} -\frac{h_{l_t}}{2} \int_\Omega \phi_{\underline{l},\underline{j}} \cdot D^{\underline{2}} f(\underline{x}) d\underline{x} \right| \leq \prod_{t=1}^{d} \frac{2^{-l_t}}{2} \cdot \|\phi_{\underline{l},\underline{j}}\|_2 \cdot \left\| D^{\underline{2}} f|_{\text{supp}(\phi_{\underline{l},\underline{j}})} \right\|_2 \\
&\leq 2^{-d} \cdot \left(\frac{2}{3}\right)^{d/2} \cdot 2^{-(3/2)\cdot|\underline{l}|_1} \cdot \left| f|_{\text{supp}(\phi_{\underline{l},\underline{j}})} \right|_{H^2_{mix}}
\end{aligned}$$

Lemma 4 *For the components $f_{\underline{l}} \in W_{\underline{l}}$ of $f \in H^2_{0,mix}(\bar{\Omega})$ from (10) holds*

$$\|f_{\underline{l}}\|_2 \leq 3^{-d} \cdot 2^{-2\cdot|\underline{l}|_1} \cdot |f|_{H^2_{mix}}. \tag{13}$$

Proof. Since the supports of all $\phi_{\underline{l},\underline{j}}$ of $f_{\underline{l}}$ are mutually disjoint we can write

$$\|f_{\underline{l}}\|_2^2 = \left\| \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \cdot \phi_{\underline{l},\underline{j}} \right\|_2^2 = \sum_{\underline{j} \in \mathsf{B}_{\underline{l}}} |\alpha_{\underline{l},\underline{j}}|^2 \cdot \|\phi_{\underline{l},\underline{j}}\|_2^2.$$

With Lemma 3 and 1 it now follows

$$\begin{aligned}||f_{\underline{l}}||_2^2 &\le \sum_{\underline{j}\in \mathsf{B}_{\underline{l}}} \frac{1}{6^d} \cdot 2^{-3\cdot|\underline{l}|_1} \cdot \left| f|_{\text{supp}(\phi_{\underline{l},\underline{j}})} \right|^2_{H^2_{mix}} \cdot \left(\frac{2}{3}\right)^d \cdot 2^{-|\underline{l}|_1} \\ &\le \frac{1}{3^{2d}} \cdot 2^{-4\cdot|\underline{l}|_1} \cdot |f|^2_{H^2_{mix}}\end{aligned}$$

which completes the proof.

2.3 Sparse grids

Motivated by the relation (13) of the "importance" of the hierarchical components $f_{\underline{l}}$ Zenger [45] introduced the so-called *sparse grids*, where hierarchical basis functions with a small support, and therefore a small contribution to the function representation, are not included in the discrete space of level n anymore.

Formally we define the sparse grid function space $V_n^s \subset V_n$ as

$$V_n^s := \bigoplus_{|\underline{l}|_1 \le n} W_{\underline{l}}. \tag{14}$$

We replace in the definition (8) of V_n in terms of hierarchical subspaces the condition $|\underline{l}|_\infty \le n$ with $|\underline{l}|_1 \le n$. In Figure 3 the employed subspaces $W_{\underline{l}}$ are given in black, whereas in grey are given the difference spaces $W_{\underline{l}}$ which are omitted in comparison to (8). Every $f \in V_n^s$ can now be represented, analogue to (10), as

$$f_n^s(\underline{x}) = \sum_{|\underline{l}|_1 \le n} \sum_{\underline{j}\in \mathsf{B}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}}(\underline{x}) = \sum_{|\underline{l}|_1 \le n} f_{\underline{l}}(\underline{x}), \quad \text{with } f_{\underline{l}} \in W_{\underline{l}}. \tag{15}$$

The resulting grid which corresponds to the approximation space V_n^s is called sparse grid. Examples in two and three dimensions are given in Figure 5.

Note that sparse grids were introduced in [45, 22], and are often used in this form, with a slightly different selection of hierarchical spaces using the definition

$$V_{0,n}^s := \bigoplus_{|\underline{l}|_1 \le n+d-1} W_{\underline{l}}, \qquad l_t > 0. \tag{16}$$

This definition is especially useful when no degrees of freedom exist on the boundary, e.g. for the numerical treatment of partial differential equations with Dirichlet boundary conditions. Using $V_{0,n}^s$ the finest mesh size which comes from the level of refinement n in the sparse grid corresponds to the full grid case again when only interior points are considered.

The following results hold for both definitions V_n^s and $V_{0,n}^s$. The proofs are somewhat easier without basis functions on the boundary, therefore we only consider this case here, full results can be found in the given literature, e.g. [5, 6, 7, 30, 8]. Furthermore, in the following we use the sparse grid space $V_{0,n}^s$, this allows us to have the same smallest mesh size h^{-n} inside the sparse grid of level n as in the

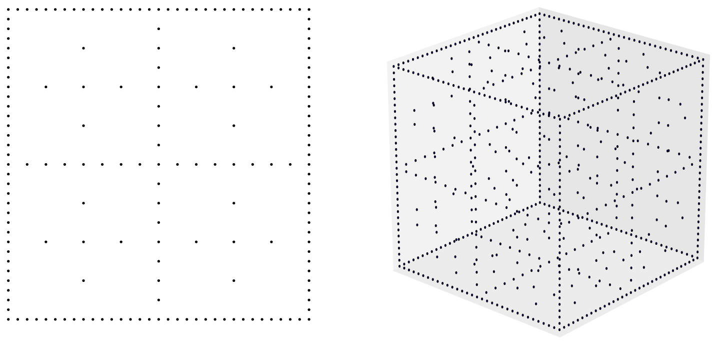

Fig. 5: Two-dimensional sparse grid (left) and three-dimensional sparse grid (right) of level $n = 5$.

corresponding full grid of level n and more closely follows the referenced original publications.

First we look at approximation properties of sparse grids. For the proof we follow [8] and first look at the error for the approximation of a function $f \in H^2_{0,mix}$, which can be represented as $\sum_{\underline{l}} f_{\underline{l}}$, i.e. an infinite sum of partial functions from the hierarchical subspaces, by $f^s_{0,n} \in V^s_{0,n}$ which can be written as a corresponding finite sum. The difference therefore is

$$f - f^s_{0,n} = \sum_{\underline{l}} f_{\underline{l}} - \sum_{|\underline{l}|_1 \le n+d-1} f_{\underline{l}} = \sum_{|\underline{l}|_1 > n+d-1} f_{\underline{l}}.$$

For any norm now holds

$$\|f - f^s_{0,n}\| \le \sum_{|\underline{l}|_1 > n+d-1} \|f_{\underline{l}}\|. \tag{17}$$

We need the following technical lemma to estimate the interpolation error

Lemma 5 *For $s \in \mathbb{N}$ it holds*

$$\begin{aligned}\sum_{|\underline{l}|_1 > n+d-1} 2^{-s|\underline{l}|_1} &= 2^{-s\cdot n} \cdot 2^{-s\cdot d} \sum_{i=0}^{\infty} 2^{-s\cdot i} \cdot \binom{i+n+d-1}{d-1} \\ &\le 2^{-s\cdot n} \cdot 2^{-s\cdot d} \cdot 2 \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}\left(n^{d-2}\right)\right),\end{aligned}$$

Proof. First we use that there are $\binom{i-1}{d-1}$ possibilities to represent i as a sum of d natural numbers

$$\begin{aligned}\sum_{|\underline{l}|_1>n+d-1} 2^{-s|\underline{l}|_1} &= \sum_{i=n+d}^{\infty} 2^{-s\cdot i}\cdot \sum_{|\underline{l}|_1=i} 1\\ &= \sum_{i=n+d}^{\infty} 2^{-s\cdot i}\cdot \binom{i-1}{d-1}\\ &= 2^{-s\cdot n}\cdot 2^{-s\cdot d}\cdot \sum_{i=0}^{\infty} 2^{-s\cdot i}\cdot \binom{i+n+d-1}{d-1}.\end{aligned}$$

We now represent the sum as the $(d-1)$-derivative of a function and get

$$\begin{aligned}&\sum_{i=0}^{\infty} x^i\cdot \binom{i+n+d-1}{d-1}\\ &= \frac{x^{-n}}{(d-1)!}\left(\sum_{i=0}^{\infty} x^{i+n+d-1}\right)^{(d-1)} = \frac{x^{-n}}{(d-1)!}\cdot\left(x^{n+d-1}\cdot\frac{1}{1-x}\right)^{(d-1)}\\ &= \frac{x^{-n}}{(d-1)!}\cdot\sum_{k=0}^{d-1}\binom{d-1}{k}\cdot\left(x^{n+d-1}\right)^{(k)}\cdot\left(\frac{1}{1-x}\right)^{(d-1-k)}\\ &= \sum_{k=0}^{d-1}\binom{d-1}{k}\cdot\frac{(n+d-1)!}{(n+d-1-k)!}\cdot x^{d-1-k}\cdot\frac{(d-1-k)!}{(d-1)!}\cdot\left(\frac{1}{1-x}\right)^{d-1-k+1}\\ &= \sum_{k=0}^{d-1}\binom{n+d-1}{k}\cdot\left(\frac{x}{1-x}\right)^{d-1-k}\cdot\frac{1}{1-x}.\end{aligned}$$

With $x=2^{-s}$ it follows

$$\sum_{i=0}^{\infty} 2^{-s\cdot i}\cdot\binom{i+n+d-1}{d-1} \le 2\cdot\sum_{k=0}^{d-1}\binom{n+d-1}{k}.$$

The summand for $k=d-1$ is the largest one and it holds

$$2\cdot\frac{(n+d-1)!}{(d-1)!n!} = 2\cdot\left(\frac{n^{d-1}}{(d-1)!}+\mathcal{O}\left(n^{d-2}\right)\right)$$

which finishes the proof.

Theorem 1 *For the interpolation error of a function $f\in H^2_{0,mix}$ in the sparse grid space $V^s_{0,n}$ holds*

$$||f-f^s_n||_2 = \mathcal{O}(h_n^2\log(h_n^{-1})^{d-1}). \tag{18}$$

Proof. Using Lemma 4 and 5 we get

$$\begin{aligned}||f-f^s_n||_2 &\le \sum_{|\underline{l}|_1>n+d-1} \|f_{\underline{l}}\|_2 \le 3^{-d}\cdot 2^{-2|\underline{l}|_1}\cdot |f|_{H^2_{mix}}\\ &\le 3^{-d}\cdot 2^{-2\cdot n}\cdot |f|_{H^2_{mix}}\cdot\left(\frac{n^{d-1}}{(d-1)!}+\mathcal{O}(n^{d-2})\right),\end{aligned}$$

which gives the desired relation.

Note that corresponding results hold in the maximum-norm as well:

$$||f - f_n^s||_\infty = \mathcal{O}\left(h_n^2 \log(h_n^{-1})^{d-1}\right)$$

for $f \in H^2_{0,mix}$ and that for the energy norm one achieves $\mathcal{O}(h_n)$, here the order is the same as in the full grid case [8].

We see that the approximation properties in the L_2-norm for functions from $H^2_{0,mix}$ when using sparse grids are somewhat worse in comparison to full grids, which achieve $\mathcal{O}(h_n^2)$. But this is offset by the much smaller number of grid points needed, as we will see when we now look at the size of the sparse grid space.

Lemma 6 *The dimension of the sparse grid space $\mathring{V}^s_{0,n}$, i.e. the number of inner grid points, is given by*

$$\left|\mathring{V}^s_{0,n}\right| = \mathcal{O}\left(h_n^{-1} \cdot \log(h_n^{-1})^{d-1}\right) \tag{19}$$

Proof. We again follow [8] and use in the first part the definition (16) and the size of a hierarchical subspace $|W_{\underline{l}}| = 2^{|\underline{l}-\underline{1}|_1}$. The following steps use similar arguments as in the preceding Lemma 5.

$$\begin{aligned}\left|\mathring{V}^s_{0,n}\right| = \left|\bigoplus_{|\underline{l}|_1 \le n+d-1} W_{\underline{l}}\right| &= \sum_{|\underline{l}|_1 \le n+d-1} 2^{|\underline{l}-\underline{1}|_1} = \sum_{i=d}^{n+d-1} 2^{i-d} \cdot \sum_{|\underline{l}|_1 = i} 1 \\ &= \sum_{i=d}^{n+d-1} 2^{i-d} \cdot \binom{i-1}{d-1} = \sum_{i=0}^{n-1} 2^i \cdot \binom{i+d-1}{d-1}.\end{aligned}$$

We now represent the summand as the $(d-1)$-derivative of a function evaluated at $x = 2$

$$\begin{aligned}
&\sum_{i=0}^{n-1} x^i \cdot \binom{i+d-1}{d-1} \\
&= \frac{1}{(d-1)!} \sum_{i=0}^{n-1} \left(x^{i+d-1}\right)^{(d-1)} = \frac{1}{(d-1)!} \left(x^{d-1} \cdot \frac{1-x^n}{1-x}\right)^{(d-1)} \\
&= \frac{1}{(d-1)!} \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \left(x^{d-1} - x^{n+d-1}\right)^{(k)} \cdot \left(\frac{1}{1-x}\right)^{(d-1-k)} \\
&= \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \frac{(d-1)!}{(d-1-k)!} \cdot x^{d-1-k} \cdot \frac{(d-1-k)!}{(d-1)!} \cdot \left(\frac{1}{1-x}\right)^{d-1-k+1} \\
&- \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \frac{(n+d-1)!}{(n+d-1-k)!} \cdot x^{n+d-1-k} \cdot \frac{(d-1-k)!}{(d-1)!} \cdot \left(\frac{1}{1-x}\right)^{d-1-k+1} \\
&= \sum_{k=0}^{d-1} \binom{d-1}{k} \cdot \left(\frac{x}{1-x}\right)^{d-1-k} \cdot \frac{1}{1-x} \\
&- x^n \cdot \sum_{k=0}^{d-1} \binom{n+d-1}{k} \cdot \left(\frac{x}{1-x}\right)^{d-1-k} \cdot \frac{1}{1-x}.
\end{aligned}$$

We observe that the first sum is constant in n and therefore not relevant for the order, but note that for $x = 2$ that sum falls down to $(-1)^d$ anyway and get

$$(-1)^d + 2^n \cdot \sum_{k=0}^{d-1} \binom{n+d-1}{k} \cdot (-2)^{d-1-k}.$$

The summand for $k = d-1$ is again the largest one and it holds

$$2^n \cdot \frac{(n+d-1)!}{(d-1)!n!} = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2})\right)$$

which gives a total order of $\mathcal{O}(2^n \cdot n^{d-1})$ or in other notation, with $h_n = 2^{-n}$, of $\mathcal{O}(h_n^{-1} \cdot \log(h_n^{-1})^{d-1})$.

This is far less than the size of the corresponding full grid space $|\mathring{V}_n| = \mathcal{O}(h_n^{-d}) = \mathcal{O}(2^{d \cdot n})$ and allows the treatment of higher dimensional problems while still achieving good accuracy.

Note that a practical realisation of sparse grids involves suitable data structures and special algorithms, e.g. for efficient matrix-vector multiplications in Galerkin methods for the numerical solution of partial differential equations. Further details and references can be found for example in [14, 34, 44][3]. Also note that sparse grid functions do not possess some properties which full grid functions have, e.g. a sparse

[3] Note that for the purpose of interpolation a sparse grid toolbox for Matlab is available at `http://www.ians.uni-stuttgart.de/spinterp/`.

grid function need not be monotone [32, 34].

The sparse grid structure introduced so far defines an a priori selection of grid points that is optimal if certain smoothness conditions are met, i.e. if the function has bounded second mixed derivatives, and no further knowledge of the function is known or used. If the aim is to approximate functions which do not fulfil this smoothness condition, or to represent functions that show significantly differing characteristics, e.g. very steep regions beyond flat ones, spatially adaptive refinement may be used as well. Depending on the characteristics of the problem and function at hand adaptive refinement strategies decide which points, and corresponding basis functions, should be incrementally added to the sparse grid representation to increase the accuracy.

In the sparse grid setting, usually an error indicator coming directly from the hierarchical basis is employed [23, 14, 35, 34]: depending on the size of the hierarchical surplus $\alpha_{\underline{l},\underline{j}}$ it is decided whether a basis function should be marked for further improvement or not. This is based on two observations: First, the hierarchical surplus gives the absolute change in the discrete representation at point $x_{\underline{l},\underline{j}}$ due to the addition of the corresponding basis function $\phi_{\underline{l},\underline{j}}$, it measures its contribution in a given sparse grid representation (15) in the maximum-norm. And second, a hierarchical surplus represents discrete second derivatives according to (12) and hence can be interpreted as a measure of the smoothness of the considered function at point $x_{\underline{l},\underline{j}}$. Further details on spatially adaptive sparse grids, their realisation and the state of the art can be found in [14, 34, 35].

2.4 Hierarchy using constant functions

An alternative hierarchical representation of a function in V_n is based on a slightly different hierarchy, which starts at level -1 with the constant. To be precise, we define the one-dimensional basis functions $\widetilde{\phi}_{l,j}(x)$ by

$$\begin{aligned} \widetilde{\phi}_{-1,0} &:= 1, \\ \widetilde{\phi}_{0,0} &:= \phi_{0,1}, \\ \widetilde{\phi}_{l,j} &:= \phi_{l,j} \quad \text{for } l \geq 1, \end{aligned}$$

with $\phi_{l,j}$ defined as in (4). Obviously it holds $\phi_{0,0} = \widetilde{\phi}_{-1,0} - \widetilde{\phi}_{0,0}$. The d-dimensional basis functions are constructed as a tensor product as before

$$\widetilde{\phi}_{\underline{l},\underline{j}}(\underline{x}) := \prod_{t=1}^{d} \widetilde{\phi}_{l_t,j_t}(x_t). \tag{20}$$

We introduce index sets $\widetilde{\mathsf{B}}_{\underline{l}}$ analogue to (6)

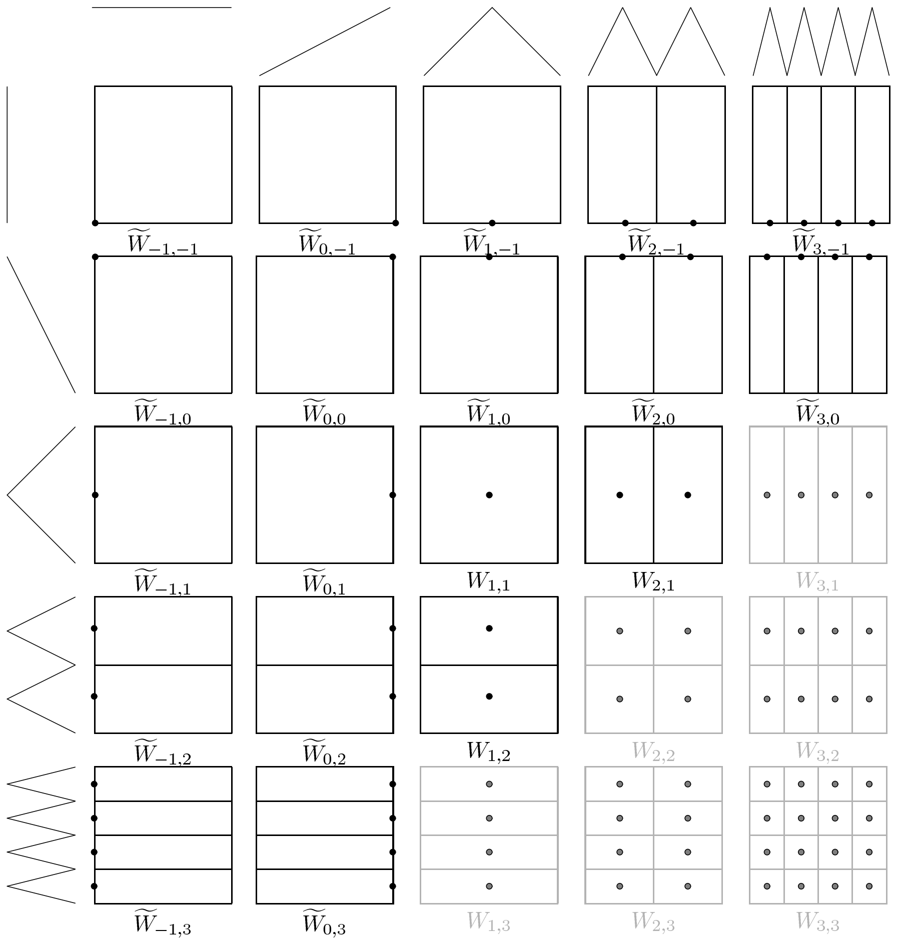

Fig. 6: Supports of the basis functions of the hierarchical subspaces $W_{\underline{l}}$ and $\widetilde{W}_{\underline{l}}$ of the space V_3. The sparse grid space V_3^s contains the upper triangle of spaces shown in black, the space $\widetilde{V}_3^s$ includes also $\widetilde{W}_{4,-1}$ and $\widetilde{W}_{-1,4}$, which are not shown.

$$\widetilde{\mathsf{B}}_{\underline{l}} := \left\{ \underline{j} \in \mathbb{N}^d \,\middle|\, \begin{array}{ll} j_t = 1,\ldots,2^{l_t}-1, & j_t \text{ odd}, t = 1,\ldots,d, \text{ if } l_t > 0, \\ j_t = 0, & \qquad\quad t = 1,\ldots,d, \text{ if } l_t \in \{0,-1\} \end{array} \right\}.$$

Now we can define slightly modified hierarchical difference spaces $\widetilde{W}_{\underline{l}}$ analogue to (7) by

$$\widetilde{W}_{\underline{l}} = \operatorname{span}\{\widetilde{\phi}_{\underline{l},\underline{j}}, \underline{j} \in \widetilde{\mathsf{B}}_{\underline{l}}\}, \tag{21}$$

see Figure 6.

It is easy to see that $\widetilde{W}_{\underline{l}} = W_{\underline{l}}$ holds for $\underline{l} \geq \underline{0}$. We now can define a full grid space $\widetilde{V}_n$ by using the newly defined modified hierarchical subspaces

$$\widetilde{V}_n := \bigoplus_{l_1=-1}^{n} \cdots \bigoplus_{l_d=-1}^{n} \widetilde{W}_{\underline{l}} = \bigoplus_{|\underline{l}|_\infty \leq n} \widetilde{W}_{\underline{l}}. \tag{22}$$

Again it holds $\widetilde{V}_n = V_n$ for $n \geq 0$.

A corresponding definition of a sparse grid space $\widetilde{V}_n^s$ using

$$\widetilde{V}_n^s := \bigoplus_{|\underline{l}|_1 \leq n} \widetilde{W}_{\underline{l}} \tag{23}$$

on the other hand does not give the original sparse grid space V_n^s. But if we exclude a few spaces it holds

$$V_n^s = \widetilde{V}_n^s \setminus \bigoplus_{\substack{|\underline{l}|_1 = n \text{ and} \\ \exists l_t = -1}} \widetilde{W}_{\underline{l}} \quad \text{for } n \geq 0, \tag{24}$$

see Figure 6.

As before, every $f \in \widetilde{V}_n^s$ can now be represented, analogue to (15), as

$$f(\underline{x}) = \sum_{|\underline{l}|_1 \leq n} \sum_{\underline{j} \in \widetilde{\mathsf{B}}_{\underline{l}}} \alpha_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}}(\underline{x}) = \sum_{|\underline{l}|_1 \leq n} f_{\underline{l}}(\underline{x}) \quad \text{with } f_{\underline{l}} \in \widetilde{W}_{\underline{l}}. \tag{25}$$

The key observation is now that the partial functions $f_{\underline{l}}$ with $\exists l_t = -1$ are lower-dimensional functions: they are constant in those dimensions t where $l_t = -1$; $f_{\underline{l}}$ possesses no degree of freedom in these dimensions. Such a function representation for $f(\underline{x})$ can therefore be formally written in the *ANalysis Of VAriance* (ANOVA) form, which is well known from statistics,

$$f(\underline{x}) = f_{\underline{-1}} + \sum_{\substack{|\underline{l}|_1 \leq n \text{ and} \\ |\{l_t | l_t = -1\}| = d-1}} f_{\underline{l}} + \cdots + \sum_{\substack{|\underline{l}|_1 \leq n \text{ and} \\ |\{l_t | l_t = -1\}| = 1}} f_{\underline{l}} + \sum_{\substack{|\underline{l}|_1 \leq n \text{ and} \\ |\{l_t | l_t = -1\}| = 0}} f_{\underline{l}}, \tag{26}$$

with $f_{\underline{l}} \in \widetilde{W}_{\underline{l}}$. The ANOVA order, the number of relevant non-constant dimensions, of the component functions $f_{\underline{l}}$ grows from 0 on the left to d on the right.

At this stage this is just a formal play with the representation, but it becomes quite relevant when one can build such a representation for a given function in an adaptive fashion, i.e. one chooses which component functions up to which ANOVA order are used for a reasonable approximation of some f. If the ANOVA order can be limited to q with $q \ll d$, the complexity estimates do not depend on the dimension d but on the ANOVA order q, allowing the treatment of even higher dimensional problems. An ANOVA-based dimension adaptive refinement algorithm in the hierarchical sparse grid basis is presented and evaluated in [14].

3 Sparse grid combination technique

The so-called *combination technique* [25], which is based on multi-variate extrapolation [10], is another method to achieve a function representation on a sparse grid. The function is discretized on a certain sequence of grids using a nodal discretization. A linear combination of these partial functions then gives the sparse grid representation. This approach can have numerical advantages over working directly in the hierarchical basis, where e.g. the stiffness matrix is not sparse and efficient computations of the matrix-vector-product are challenging in the implementation [1, 3, 6, 14, 34, 44]. There are close connections of the combination technique to boolean [11, 12] and discrete blending methods [4], as well as the splitting extrapolation-method [31].

In particular, we discretize a function f on a certain sequence of anisotropic grids $\Omega_{\underline{l}} = \Omega_{l_1,\dots,l_d}$ with uniform mesh sizes $h_t = 2^{-l_t}$ in the t-th coordinate direction. These grids possess in general different mesh sizes for the different coordinate directions. To be precise, we consider all grids $\Omega_{\underline{l}}$ with

$$|\underline{l}|_1 := l_1 + \dots + l_d = n - q, \quad q = 0,..,d-1, \quad l_t \geq 0. \tag{27}$$

The grids employed by the combination technique of level 4 in two dimensions are shown in Figure 7.

Note that in the original [25] and other papers, a slightly different definition was used:

$$|\underline{l}|_1 := l_1 + \dots + l_d = n + (d-1) - q, \quad q = 0,..,d-1, \quad l_t > 0.$$

This is again in view of situations where no degrees of freedom are needed on the boundary, e.g. for Dirichlet boundary conditions, see (16) and the remarks afterwards.

A finite element approach with piecewise d-linear functions $\phi_{\underline{l},\underline{j}}(\underline{x})$ on each grid $\Omega_{\underline{l}}$ now gives the representation in the nodal basis

$$f_{\underline{l}}(\underline{x}) = \sum_{j_1=0}^{2^{l_1}} \dots \sum_{j_d=0}^{2^{l_d}} \alpha_{\underline{l},\underline{j}} \phi_{\underline{l},\underline{j}}(\underline{x}).$$

Finally, we linearly combine the discrete partial functions $f_{\underline{l}}(\underline{x})$ from the different grids $\Omega_{\underline{l}}$ according to the combination formula

$$f_n^c(\underline{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\underline{l}|_1 = n-q} f_{\underline{l}}(\underline{x}). \tag{28}$$

The resulting function f_n^c lives in the sparse grid space V_n^s, where the combined interpolant is identical with the hierarchical sparse grid interpolant f_n^s [25]. This can be seen by rewriting each $f_{\underline{l}}$ in their hierarchical representation (10) and some straightforward calculation using the telescope sum property, i.e. the hierarchical functions get added and subtracted.

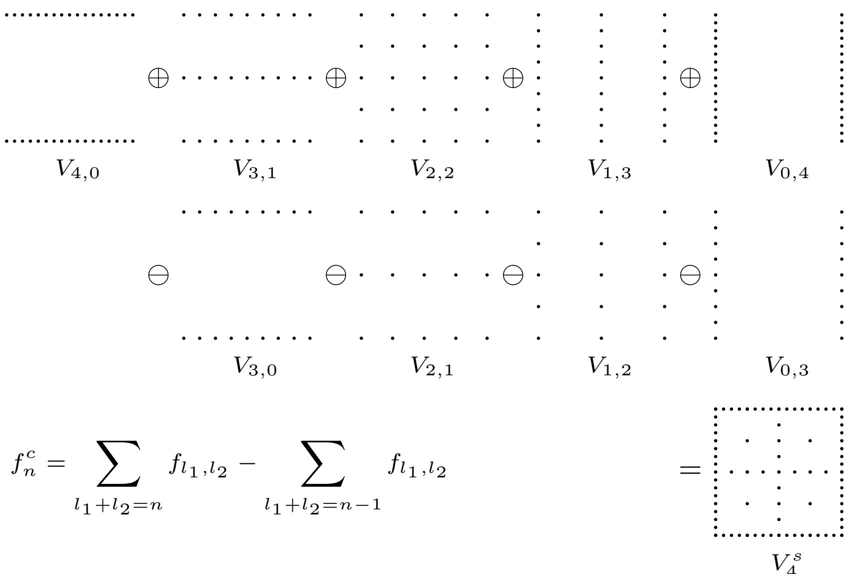

Fig. 7: Combination technique with level $n = 4$ in two dimensions

Lemma 7 *For a given function f the interpolant f_n^c using the combination technique (28) is the hierarchical sparse grid interpolant f_n^s from (15).*

Proof. We write it exemplary in the two dimensional case, using $\hat{f}_{l_1+l_2} \in W_{l_1,l_2}$ instead of all the basis functions of W_{l_1,l_2} for ease of presentation:

$$\begin{aligned}
f_n^c &= \sum_{l_1+l_2=n} f_{l_1,l_2} - \sum_{l_1+l_2=n-1} f_{l_1,l_2} \\
&= \sum_{l_1\le n} \sum_{k_1\le l_1} \sum_{k_2\le n-l_1} \hat{f}_{k_1,k_2} - \sum_{l_1\le n-1} \sum_{k_1\le l_1} \sum_{k_2\le n-l_1-1} \hat{f}_{k_1,k_2} \\
&= \sum_{k_1\le l_1=n} \sum_{k_2=0} \hat{f}_{k_1,k_2} + \sum_{l_1\le n-1} \sum_{k_1\le l_1} \left(\sum_{k_2\le n-l_1} \hat{f}_{k_1,k_2} - \sum_{k_2\le n-l_1-1} \hat{f}_{k_1,k_2} \right) \\
&= \sum_{k_1\le l_1=n} \sum_{k_2=n-l_1} \hat{f}_{k_1,k_2} + \sum_{l_1\le n-1} \sum_{k_1\le l_1} \sum_{k_2=n-l_1} \hat{f}_{k_1,k_2} \\
&= \sum_{l_1\le n} \sum_{k_2=n-l_1} \sum_{k_1\le n-k_2} \hat{f}_{k_1,k_2} \\
&= \sum_{k_2\le n} \sum_{k_1\le n-k_2} \hat{f}_{k_1,k_2} = \sum_{k_1+k_2\le n} \hat{f}_{k_1,k_2}
\end{aligned}$$

This last expression is exactly (15).

Alternatively, one can view the combination technique as an approximation of a projection into the underlying sparse grid space. The combination technique is then an exact projection into the sparse grid space if and only if the partial projections commute, i.e. the commutator $[P_{V_1}, P_{V_2}] := P_{V_1}P_{V_2} - P_{V_2}P_{V_1}$ is zero for all pairs of involved grids [27].

Note that the solution obtained with the combination technique f_n^c for the numerical treatment of partial differential equations, i.e. when the solutions on the partial grids are combined according to the combination formula (28), is in general not the sparse grid solution f_n^s. However, the approximation property is of the same order as long as a certain series expansion of the error exists [25]. Its existence was shown for model-problems in [9].

Lemma 8 *Assume that the exact solution f is sufficiently smooth and that the pointwise error expansion*

$$f - f_{\underline{l}} = \sum_{i=1}^{d} \sum_{j_1,\dots,j_m \subset 1,\dots,d} c_{j_1,\dots,j_m}(h_{j_1},\dots,h_{j_m}) \cdot h_{j_1}^p \cdot \dots \cdot h_{j_m}^p, \tag{29}$$

with bounded $c_{j_1,\dots,j_m}(h_{j_1},\dots,h_{j_m}) \le \kappa$, holds for $\underline{l} \le n$. Then

$$|f - f_n^c| = \mathcal{O}\left(h_n^2 \cdot \log(h_n^{d-1})\right). \tag{30}$$

Proof. Let us again consider the two dimensional case and consider the pointwise error of the combined solution $f - f_n^c$ following [25]. We have

$$\begin{aligned} f - f_n^c &= f - \sum_{l_1+l_2=n} f_{l_1,l_2} + \sum_{l_1+l_2=n-1} f_{l_1,l_2} \\ &= \sum_{l_1+l_2=n} \left(f - f_{l_1,l_2}\right) - \sum_{l_1+l_2=n-1} \left(f - f_{l_1,l_2}\right). \end{aligned}$$

Plugging in the error expansion (29) leads to

$$\begin{aligned} f - f_n^c = &\sum_{l_1+l_2=n} \left(c_1(h_{l_1}) \cdot h_{l_1}^2 + c_2(h_{l_2}) \cdot h_{l_2}^2 + c_{1,2}(h_{l_1},h_{l_2}) \cdot h_{l_1}^2 h_{l_2}^2\right) \\ &- \sum_{l_1+l_2=n-1} \left(c_1(h_{l_1}) \cdot h_{l_1}^2 + c_2(h_{l_2}) \cdot h_{l_2}^2 + c_{1,2}(h_{l_1},h_{l_2}) \cdot h_{l_1}^2 h_{l_2}^2\right) \\ = &\left(c_1(h_n) + c_2(h_n) + \sum_{l_1+l_2=n} c_{1,2}(h_{l_1},h_{l_2}) \right. \\ &\left. -4 \sum_{l_1+l_2=n-1} c_{1,2}(h_{l_1},h_{l_2}) \right) \cdot h_n^2. \end{aligned}$$

And using $c_i \le \kappa$ we get the estimate (30)

$$\begin{aligned}
|f-f_n^c| &\leq 2\kappa \cdot h_n^2 + \left| \sum_{l_1+l_2=n} c_{1,2}(h_{l_1},h_{l_2}) - 4 \sum_{l_1+l_2=n-1} c_{1,2}(h_{l_1},h_{l_2}) \right| \cdot h_n^2 \\
&\leq 2\kappa \cdot h_n^2 + \sum_{l_1+l_2=n} \left|c_{1,2}(h_{l_1},h_{l_2})\right| \cdot h_n^2 + 4 \sum_{l_1+l_2=n-1} \left|c_{1,2}(h_{l_1},h_{l_2})\right| \cdot h_n^2 \\
&\leq 2\kappa \cdot h_n^2 + \kappa \cdot n h_n^2 + 4\kappa(n-1)h_n^2 \\
&= \kappa \cdot h_n^2(5n-2) = \kappa \cdot h_n^2(5\log(h_n^{-1})-2) \\
&= \mathcal{O}\left(h_n^2 \cdot \log(h_n^{-1})\right).
\end{aligned}$$

Observe that cancellation occurs for h_{l_i} with $l_i \neq n$ and the accumulated $h_{l_1}^2 h_{l_2}^2$-terms result in the $\log(h_n^{-1})$-term. The approximation order $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1}))$ is just as in Theorem 1. See [25, 33, 36] for results in higher dimensions.

Similar to (26) one can consider an ANOVA representation in the form of a combination technique, which in general terms is a function representation for $f(\underline{x})$ of the type

$$f(\underline{x}) = \sum_{\{j_1,\ldots,j_q\}\subset\{1,\ldots,d\}} c_{j_1,\ldots,j_q} f_{j_1,\ldots,j_q}(x_{j_1},\ldots,x_{j_q}), \tag{31}$$

where each $f_{j_1,\ldots,j_q}(x_{j_1},\ldots,x_{j_q})$ depends only on a subset of size q of the dimensions and may have different refinement levels for each dimension. Again, one especially assumes here that $q \ll d$, so that the computational complexity depends on the so-called *superposition* (or *effective*) dimension q. The hierarchy here again starts with a level -1 of constant functions and we note again that if one builds the tensor product between a constant in one dimension and a $(d-1)$-linear function the resulting d-dimensional function is still $(d-1)$-linear, one gains no additional degrees of freedom. But formally introducing a level -1, and using this as coarsest level, will allow us to write a combined function in the ANOVA-style (31), in other words each partial function might only depend on a subset of all dimensions. The size of each grid $\Omega_{\underline{l}}$ is now of order $\mathcal{O}(2^q(|\underline{l}|_1+(d-q))$, where $q = \#\{l_i | l_i \geq 0\}$.

An advantage of such a viewpoint arises if one can select which grids to employ and does not use the grid sequence (27). In such a so-called dimension adaptive procedure one considers an index set I which only needs to fulfil the following *admissibility condition* [21, 26]

$$\underline{k} \in \mathsf{I} \text{ and } \underline{j} \leq \underline{k} \quad \Rightarrow \quad \underline{j} \in \mathsf{I}, \tag{32}$$

in other words an index $\underline{k}$ can only belong to the index set I if all smaller grids $\underline{j}$ belong to it. The combination coefficients for a dimension adaptive combination technique, which are related to the "inclusion/exclusion" principle from combinatorics, depend only on the index set [20, 26, 27]:

$$f_{\mathsf{I}}^c(\underline{x}) := \sum_{\underline{k}\in\mathsf{I}} \left(\sum_{\underline{z}=\underline{0}}^{\underline{1}} (-1)^{|\underline{z}|_1} \cdot \chi^{\mathsf{I}}(\underline{k}+\underline{z}) \right) f_{\underline{k}}(\underline{x}) \tag{33}$$

where χ^{I} is the characteristic function of I defined by

$$\chi^{\mathsf{I}}(\underline{k}) := \begin{cases} 1 & \text{if } \underline{k} \in \mathsf{I}, \\ 0 & \text{otherwise.} \end{cases}$$

Further details on dimension adaptive algorithms and suitable refinement strategies for the sparse combination technique can be found in [21, 17, 19].

3.1 Optimised combination technique

As mentioned, the combination technique only gives the same order if the above error expansion exists. In some cases even divergence of the combination technique can be observed [15, 16, 27]. But an optimised combination technique [27] can be used instead to achieve good approximations with a combination technique and especially to avoid the potential divergence. Here the combination coefficients are not fixed, but depend on the underlying problem and the function to be represented. Optimised combination coefficients are in particular relevant for dimension adaptive approaches [17, 19].

For ease of presentation we assume a suitable numbering of the involved spaces from (28) for now. To compute the optimal combination coefficients c_i one minimises the functional

$$J(c_1,\ldots,c_m) = \left\| P_n^s f - \sum_{i=1}^{m} c_i P_i f \right\|^2,$$

where one uses a suitable scalar product and a corresponding orthogonal projection P stemming from the problem under consideration. By $P_n^s f$ we denote the projection into the sparse grid space V_n^s, by $P_i f$ the projection into one of the spaces from (28).

By simple expansion and using

$$\langle P_n^s f, P_j f\rangle = \langle P_n^s f, P_j^* P_j f\rangle = \langle P_j P_n^s f, P_j f\rangle = \langle P_j f, P_j f\rangle$$

one gets

$$J(c_1,\ldots,c_m) = \sum_{i,j=1}^{m} c_i c_j \langle P_i f, P_j f\rangle - 2\sum_{i=1}^{m} c_i \|P_i f\|^2 + \|P_n^s f\|^2.$$

While this functional depends on the unknown quantity $P_n^s f$, the location of the minimum of J does not. By differentiating with respect to the combination coefficients c_i and setting each of these derivatives to zero we see that minimising this expression corresponds to finding c_i which have to satisfy

$$\begin{bmatrix} \|P_1 f\|^2 & \cdots & \langle P_1 f, P_m f\rangle \\ \langle P_2 f, P_1 f\rangle & \cdots & \langle P_2 f, P_m f\rangle \\ \vdots & \ddots & \vdots \\ \langle P_m f, P_1 f\rangle & \cdots & \|P_m f\|^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix} = \begin{bmatrix} \|P_1 f\|^2 \\ \|P_2 f\|^2 \\ \vdots \\ \|P_m f\|^2 \end{bmatrix}.$$

The solution of this small system creates little overhead. However, in general to compute the scalar product $\langle P_{\underline{i}} f, P_{\underline{j}} f\rangle$ of the two projections into the discrete spaces $V_{\underline{i}}$ and $V_{\underline{j}}$ one needs to embed both spaces into the joint space $V_{\underline{k}}$, with $k_t = \max(i_t, j_t)$, into which the partial solutions $P_{\underline{l}} f = f_{\underline{l}}, \underline{l} = \underline{i}, \underline{j}$ have to be interpolated. One easily observes that $V_{\underline{k}}$ is of size $\mathcal{O}(h_n^{-2})$ in the worst case, as opposed to $\mathcal{O}(h_n^{-1})$ for the $V_{\underline{l}}, \underline{l} = \underline{i}, \underline{j}$; an increase in computational complexity thus results, but does not depend on d. In specific situations the computational complexity can be smaller though [16].

Using these optimal coefficients c_i the combination formula for a sparse grid of level n is now just

$$f_n^c(\underline{x}) := \sum_{q=0}^{d-1} \sum_{|\underline{l}|_1 = n-q} c_{\underline{l}} f_{\underline{l}}(\underline{x}). \tag{34}$$

Finally note that one also can interpret the optimised combination technique as a Galerkin formulation which uses the partial solutions as ansatz functions. That way one can formulate an optimised combination technique for problems where the projection arguments do not hold and are replaced by Galerkin conditions, which for example is the case for eigenvalue problems [18].

4 Sparse grids in python

We give the listing of some python code for a sparse grid without functions on the boundary, i.e. according to formula (16). In this code the action is done in the hierarchical subspaces, so everything is done while going through all subspaces $W_{\underline{l}}$. A different way, especially needed for adaptive sparse grids, is to work directly in the hierarchical basis structure and go to the left and right neighbours in each dimension and that way run over all grid points.

If you are interested in the source files, they can be found at `https://github.com/jgarcke/sparse-grid-py`.

A MATLAB implementation of sparse grids can be found here
`http://www.ians.uni-stuttgart.de/spinterp/` and here
`http://sparse-grids.de/`.

A C++ code is available here
`http://www5.in.tum.de/SGpp`.

Listing 1: function representation on a regular sparse grid

```
# This sparse grid code works for a _regular_ sparse grid all operations
# work over the hierarchical subspaces.
# It was written as an exercise to see what operations can be done this way
# (and to learn python).
#
# The other, and maybe more natural, way to do sparse grids is using a
# hierarchical structure with left and right sons.
# If one needs adaptive sparse grid one probably needs to do it that way

import math, copy

class gridPoint:
  """ position of a grid point,
      also stores function value """
  def __init__(self, index=None, domain=None):
    self.hv = [] # hierarchical value
    self.fv = [] # function value
    if index is None:
      self.pos = [] # position of grid point
    else:
      self.pos = self.pointPosition(index, domain)

  def pointPosition(self, index, domain=None):
    coord = list()
    if domain is None:
      for i in range(len(index)/2):
        coord.append(index[2*i+1]/2.**index[2*i])
    else:
      for i in range(len(index)/2):
        coord.append((domain[i][1] - domain[i][0]) \
                    *index[2*i+1] / 2.**index[2*i] + domain[i][0])
    return coord

  def printPoint(self):
    if self.pos is []:
      pass
    else:
      out = ""
      for i in range(len(self.pos)):
        out += str(self.pos[i]) + "\t"
      print out

class sparseGrid:
  """ A sparse grid of a certain level consists of a set of indices and
      associated grid points gP on a given domain of dimension dim.
      Action is what happens when one traverses the sparse grid.
  """
  def __init__(self,dim=1,level=1):
    self.dim = dim
    self.level = level
    self.gP = {} # hash, indexed by tuple(l_1,p_1,l_2,p_2,...,l_d,p_d)
    self.indices = [] # entries: [l_1,p_1,...,l_d,p_d], level,position
    self.domain = ((0.0,1.0),)*dim
    self.action = ()

  def printGrid(self):
    print self.hSpace

  def evalAction(self):
    basis = copy.deepcopy(self.evalPerDim[0][self.hSpace[0]-1][0])
    value = self.evalPerDim[0][self.hSpace[0]-1][1]
    # compute index and its value on x of the one non-zero basis function
    # in this hierarchical sup-space
    for i in range(1,self.dim):
      value *= self.evalPerDim[i][self.hSpace[i]-1][1]
      basis += self.evalPerDim[i][self.hSpace[i]-1][0]
    # add contribution of this hierarchical space
    self.value += self.gP[tuple(basis)].hv*value

  def evalFunct(self,x):
    """ evaluate a sparse grid function, hierarchival values have to be set """
    self.value = 0.0
    self.evalPerDim = []
    # precompute values of one dim basis functions at x for the evaluation
    for i in range(self.dim):
      self.evalPerDim.append([])
      for j in range(1,self.level+1):
        # which basis is unzero on x for dim i and level j
        pos = (x[i]-self.domain[i][0])/(self.domain[i][1] \
                                       -self.domain[i][0])
        basis = int(math.ceil(pos*2**(j-1))*2-1)
        # test needed for x on left boundary
        if basis == -1:
          basis = 1
          self.evalPerDim[i].append([[j,basis]])
        else:
          self.evalPerDim[i].append([[j,basis]])
        # value of this basis function on x[i]
        self.evalPerDim[i][j-1].append(evalBasis1D(x[i],\
                self.evalPerDim[i][j-1][0],self.domain[i]))
    self.action = self.evalAction
    self.loopHierSpaces()
    return self.value

  def loopHierSpaces(self):
    """ go through the hierarchical subspaces of the sparse grid """
    for i in range(1,self.level+1):
      self.hSpace = [i]
      self.loopHierSpacesRec(self.dim-1,self.level-(i-1))

  def loopHierSpacesRec(self,dim, level):
    """ d-dimensional recursion through all hierarchical subspaces """
    if dim > 1:
      for i in range(1,level+1):
        self.hSpace.append(i)
        self.loopHierSpacesRec(dim-1,level-(i-1))
        self.hSpace.pop()
    else:
      for i in range(1,level+1):
        self.hSpace.append(i)
        self.action()
        self.hSpace.pop()

  def generatePoints(self):
    """ fill self.gP with the points for the indices generated beforehand """
    # generate indices of grid points for the given level and dim
    self.indices = self.generatePointsRec(self.dim,self.level)
    # add positions of sparse grid points
    for i in range(len(self.indices)):
      self.gP[tuple(self.indices[i])] = gridPoint(self.indices[i],self.domain)

  def generatePointsRec(self,dim, level, cur_level=None):
    """ run over all hierarchical subspaces and add all their indices """
    basis_cur = list()
    if cur_level == None:
      cur_level = 1
    # generate all 1-D basis indices of current level (i.e. step 2)
    for i in range (1,2**(cur_level)+1,2):
      basis_cur.append([cur_level,i])
    if dim == 1 and cur_level == level:
      return basis_cur # we have all
    elif dim == 1: # generate some in this dim for higher level
      basis_cur += self.generatePointsRec(dim,level,cur_level+1)
      return basis_cur
    elif cur_level == level:
      #crossproduct of this dim indices and other (dim-1) ones
      return cross(basis_cur,\
                   self.generatePointsRec(dim-1,level-cur_level+1))
    else:
      #crossproduct of this dim indices and other (dim-1) ones
      #since levels left, generate points for higher levels
      return cross(basis_cur, self.generatePointsRec(dim-1,\
                   level-cur_level+1)) \
                   + self.generatePointsRec(dim, level, cur_level+1)

  def nodal2Hier1D(self,node,i,j,dim):
    """ conversion from nodal to hierarchical basis in one dimension
        (i,j) gives index in this dim current node
        node is the (d-1) index to treat """
    # get left/right neighbours of node
    left = [i-1,j/2]
    right = [i-1,j/2+1]
    # left, right can be points of upper level (if index is even)
    while left[1]%2 == 0 and left[0] > 0:
      left = [left[0]-1,left[1]/2]
    while right[1]%2 == 0 and right[0] > 0:
      right = [right[0]-1,right[1]/2]
    # index of node is multi-dimensional
    if len(node) > 2:
      # build d-dim index for current node and its neighbours
      preCurDim  = node[0:2*dim]
      postCurDim = node[2*dim:len(node)+1]
      index = preCurDim + [i,j] + postCurDim
      left  = preCurDim + left  + postCurDim
      right = preCurDim + right + postCurDim
    else:#this case can only happen in 2D
      if dim == 0:
        index = [i,j] + node
        left  = left  + node
        right = right + node
      else:
        index = node + [i,j]
        left  = node + left
        right = node + right
    #in case we are on the left boundary
    if left[2*dim] == 0:
      if right[2*dim] != 0:
        self.gP[tuple(index)].hv -= 0.5*self.gP[tuple(right)].hv
    elif right[2*dim] == 0: #or the right boundary
      self.gP[tuple(index)].hv -= 0.5*self.gP[tuple(left)].hv
    else: #normal inner node
      self.gP[tuple(index)].hv -= 0.5*(self.gP[tuple(left)].hv + self.gP[tuple(right)\
           ].hv)

  # conversion from nodal to hierarchical basis
  def nodal2Hier(self):
    for i in range(len(self.indices)):
      self.gP[tuple(self.indices[i])].hv = self.gP[tuple(self.indices[i])].fv
    # conversion is done by succesive one-dim conversions
    for d in range(0,self.dim):
      for i in range(self.level,0,-1):
        # generate all indices to process
        indices = self.generatePointsRec(self.dim-1,self.level-i+1)
        for j in range(1,2**i+1,2):
          for k in range(len(indices)):
            self.nodal2Hier1D(indices[k],i,j,d)

# compute cross-product of args
def cross(*args):
  ans = []
  for arg in args[0]:
    for arg2 in args[1]:
      ans.append(arg+arg2)
  return ans
  #alternatively:
  #ans = [[]]
  #for arg in args:
    #ans = [x+y for x in ans for y in arg]

  #return ans

# evaluation of the basis functions in one dimension
def evalBasis1D(x, basis, interval=None):
  if interval is None:
    return 1. - abs(x*2**basis[0]-basis[1])
  else:
    pos = (x-interval[0])/(interval[1]-interval[0])
    return 1. - abs(pos*2**basis[0]-basis[1])
```

Listing 2: unit test for listing 1

```
import pysg
import unittest
import math
class testFunctest(unittest.TestCase):
  """ simple test if sparse grid for sparse grid in 3d of level 3 """
  def testSGNoBound3D(self):
    sg = pysg.sparseGrid(3,3)
    sg.generatePoints()
    # right number of grid points
    self.assertEqual(len(sg.indices),31)
    for i in range(len(sg.indices)):
      sum = 1.0
      pos = sg.gP[tuple(sg.indices[i])].pos
      for j in range(len(pos)):
        sum *= 4.*pos[j]*(1.0-pos[j])
      sg.gP[tuple(sg.indices[i])].fv = sum
    # convert to hierarchical values
    sg.nodal2Hier()
    # does the evaluation of sparse grid function in
    # hierarchical values give the correct value gv
    for i in range(len(sg.indices)):
      self.assertEqual(sg.gP[tuple(sg.indices[i])].fv,\
        sg.evalFunct(sg.gP[tuple(sg.indices[i])].pos))

  def testSGNoBound2D(self):
    sg = pysg.sparseGrid(2,3)
    sg.generatePoints()
    # right number of grid points
    self.assertEqual(len(sg.indices),17)
    for i in range(len(sg.indices)):
      sum = 1.0
      pos = sg.gP[tuple(sg.indices[i])].pos
      for j in range(len(pos)):
        sum *= 4.*pos[j]*(1.0-pos[j])
      sg.gP[tuple(sg.indices[i])].fv = sum
    # convert to hierarchical values
    sg.nodal2Hier()
    # does the evaluation of sparse grid function in
    # hierarchical values give the correct value gv
    for i in range(len(sg.indices)):
      self.assertEqual(sg.gP[tuple(sg.indices[i])].fv,\
        sg.evalFunct(sg.gP[tuple(sg.indices[i])].pos))

if __name__=="__main__":
  unittest.main()
```

References

1. S. Achatz. Higher order sparse grid methods for elliptic partial differential equations with variable coefficients. *Computing*, 71(1):1–15, 2003.
2. K. I. Babenko. Approximation of periodic functions of many variables by trigonometric polynomials. *Dokl. Akad. Nauk SSSR*, 132:247–250, 1960. Russian, Engl. Transl.: Soviet Math. Dokl. 1:513–516, 1960.
3. R. Balder. *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern*. Dissertation, Technische Universität München, 1994.
4. G. Baszenski, F.-J. Delvos, and S. Jester. Blending approximations with sine functions. In D. Braess, editor, *Numerical Methods in Approximation Theory IX*, ISNM 105, pages 1–19. Birkhäuser, Basel, 1992.
5. H.-J. Bungartz. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. Dissertation, Institut für Informatik, Technische Universität München, 1992.
6. H.-J. Bungartz. *Finite Elements of Higher Order on Sparse Grids*. Habilitation, Institut für Informatik, Technische Universität München and Shaker Verlag, Aachen, 1998.
7. H.-J. Bungartz and M. Griebel. A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives. *J. Complexity*, 15:167–199, 1999. also as Report No 524, SFB 256, Univ. Bonn, 1997.
8. H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.
9. H.-J. Bungartz, M. Griebel, D. Röschke, and C. Zenger. Pointwise convergence of the combination technique for the Laplace equation. *East-West J. Numer. Math.*, 2:21–45, 1994.
10. H.-J. Bungartz, M. Griebel, and U. Rüde. Extrapolation, combination, and sparse grid techniques for elliptic boundary value problems. *Comput. Methods Appl. Mech. Eng.*, 116:243–252, 1994. Also in C. Bernardi and Y. Maday, Editoren, International conference on spectral and high order methods, ICOSAHOM 92. Elsevier, 1992.
11. F.-J. Delvos. d-variate Boolean interpolation. *J. Approx. Theory*, 34:99–114, 1982.
12. F.-J. Delvos and W. Schempp. *Boolean Methods in Interpolation and Approximation*. Pitman Research Notes in Mathematics Series 230. Longman Scientific & Technical, Harlow, 1989.
13. G. Faber. Über stetige Funktionen. *Mathematische Annalen*, 66:81–94, 1909.
14. C. Feuersänger. *Sparse Grid Methods for Higher Dimensional Approximation*. Dissertation, Institut für Numerische Simulation, Universität Bonn, Sept. 2010.
15. J. Garcke. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern*. Doktorarbeit, Institut für Numerische Simulation, Universität Bonn, 2004.
16. J. Garcke. Regression with the optimised combination technique. In W. Cohen and A. Moore, editors, *Proceedings of the 23rd ICML '06*, pages 321–328, New York, NY, USA, 2006. ACM Press.
17. J. Garcke. A dimension adaptive sparse grid combination technique for machine learning. In W. Read, J. W. Larson, and A. J. Roberts, editors, *Proceedings of the 13th Biennial Computational Techniques and Applications Conference, CTAC-2006*, volume 48 of *ANZIAM J.*, pages C725–C740, 2007.
18. J. Garcke. An optimised sparse grid combination technique for eigenproblems. *PAMM*, 7(1):1022301–1022302, 2007.
19. J. Garcke. A dimension adaptive combination technique using localised adaptation criteria. In H. G. Bock, X. P. Hoang, R. Rannacher, and J. P. Schlöder, editors, *Modeling, Simulation and Optimization of Complex Processes*, pages 115–125. Springer Berlin Heidelberg, 2012.
20. T. Gerstner and M. Griebel. Numerical Integration using Sparse Grids. *Numer. Algorithms*, 18:209–232, 1998.
21. T. Gerstner and M. Griebel. Dimension–Adaptive Tensor–Product Quadrature. *Computing*, 71(1):65–87, 2003.
22. M. Griebel. A parallelizable and vectorizable multi-level algorithm on sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for partial differential equations, Notes on Numerical Fluid Mechanics*, volume 31, pages 94–100. Vieweg Verlag, Braunschweig, 1991. also as SFB Bericht, 342/20/90 A, Institut für Informatik, TU München, 1990.

23. M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998.
24. M. Griebel and S. Knapek. Optimized tensor-product approximation spaces. *Constructive Approximation*, 16(4):525–540, 2000.
25. M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.
26. M. Hegland. Adaptive sparse grids. In K. Burrage and R. B. Sidje, editors, *Proc. of 10th CTAC-2001*, volume 44 of *ANZIAM J.*, pages C335–C353, 2003.
27. M. Hegland, J. Garcke, and V. Challis. The combination technique and some generalisations. *Linear Algebra and its Applications*, 420(2–3):249–275, 2007.
28. R. Hochmuth. *Wavelet Bases in numerical Analysis and Restrictal Nonlinear Approximation*. Habilitation, Freie Universität Berlin, 1999.
29. R. Hochmuth, S. Knapek, and G. Zumbusch. Tensor products of Sobolev spaces and applications. Technical Report 685, SFB 256, Univ. Bonn, 2000.
30. S. Knapek. *Approximation und Kompression mit Tensorprodukt-Multiskalenräumen*. Doktorarbeit, Universität Bonn, April 2000.
31. C. B. Liem, T. Lü, and T. M. Shih. *The Splitting Extrapolation Method*. World Scientific, Singapore, 1995.
32. J. Noordmans and P. Hemker. Application of an adaptive sparse grid technique to a model singular perturbation problem. *Computing*, 65:357–378, 2000.
33. C. Pflaum and A. Zhou. Error analysis of the combination technique. *Numer. Math.*, 84(2):327–350, 1999.
34. D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, Aug. 2010. Dissertation.
35. D. Pflüger, B. Peherstorfer, and H.-J. Bungartz. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity*, 26(5):508–522, 2010.
36. C. Reisinger. *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2004. in Vorbereitung.
37. S. A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Dokl. Akad. Nauk SSSR*, 148:1042–1043, 1963. Russian, Engl. Transl.: Soviet Math. Dokl. 4:240–243, 1963.
38. V. N. Temlyakov. Approximation of functions with bounded mixed derivative. *Proc. Steklov Inst. Math.*, 1, 1989.
39. V. N. Temlyakov. *Approximation of Periodic Functions*. Nova Science, New York, 1993.
40. V. N. Temlyakov. On approximate recovery of functions with bounded mixed derivative. *J. Complexity*, 9:41–59, 1993.
41. G. Wahba. *Spline models for observational data*, volume 59 of *Series in Applied Mathematics*. SIAM, Philadelphia, 1990.
42. H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49:379–412, 1986.
43. H. Yserentant. Hierarchical bases. In J. R. E. O'Malley et al., editors, *Proc. ICIAM'91*, Philadelphia, 1992. SIAM.
44. A. Zeiser. Fast Matrix-Vector Multiplication in the Sparse-Grid Galerkin Method. *Journal of Scientific Computing*, 47(3):328–346, Nov. 2010.
45. C. Zenger. Sparse grids. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, 1990*, volume 31 of *Notes on Num. Fluid Mech.*, pages 241–251. Vieweg-Verlag, 1991.