Sparse Grid Tutorial

Jochen Garcke
Technische Universität Berlin
Institut für Mathematik
Sekretariat MA 3-3
Straße des 17. Juni 136
D-10623 Berlin
garcke@math.tu-berlin.de

Tuesday 11th January, 2011

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 [Fab09, Yse86, Yse92], a representation of a discrete function space which is equivalent to the conventional nodal basis, and a sparse tensor product construction.

The method was originally developed for the solution of partial differential equations [Zen91, Gri91, Bun92, Bal94, Ach03] and is now also successfully used for integral equations [FHP96, GOS99], interpolation and approximation [Bas85, Tem89, SS99, GK00, Kna00, KW05b]. Furthermore there is work on stochastic differential equations [ST03a, ST03b], differential forms in the context of the Maxwell-equation [GH03] and with a wavelet-based sparse grid discretization parabolic problems are treated in [vPS04]. Besides Galerkin finite element approaches there are also finite differences on sparse grids [Gri98, Sch99, Kos02, GK03] and finite volume approaches [Hem95].

Besides working directly in the hierarchical basis a sparse grid representation of a function can also be computed using the combination technique [GSZ92], where a certain sequence of partial functions is linearly combined. Applications include eigenvalue problems [GG00], numerical integration [GG98, BD03, GG03], parabolic equations for options pricing [Rei04], machine learning [GGT01, GG02, Gar04, GG05, Gar06] and data analysis [LNSH05] and it is used for solving the stochastic master equation applied to gene regulatory networks [HBS+06].

The underlying idea of sparse grids can be traced back to the Russian mathematician Smolyak [Smo63], who used it for numerical integration. The concept is also closely related to hyperbolic crosses [Bab60, Tem89, Tem93a, Tem93b], boolean methods [Del82, DS89], discrete blending methods [BDJ92] and splitting extrapolation methods [LLS95].

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. 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 $\mathcal{O}(h_n^2)$. Since the curse of dimensionality of full grid method arises for sparse grids at this much smaller extent they can be used for higher dimensional problems.

For ease of presentation we will consider the domain $\Omega = [0,1]^d$ here and 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 $h_{\underline{l}} := (h_{l_1}, \ldots, h_{l_d}) := (2^{-l_1}, \ldots, 2^{-l_d})$, it has different, but equidistant mesh sizes in each coordinate direction t. 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

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

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:

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

See Figure 1(a) for a one-dimensional example and Figure 2 for 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, 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 position of a grid point $x_{\underline{l},\underline{j}}$ or the corresponding basis function $\phi_{\underline{l},\underline{j}}(\cdot)$.

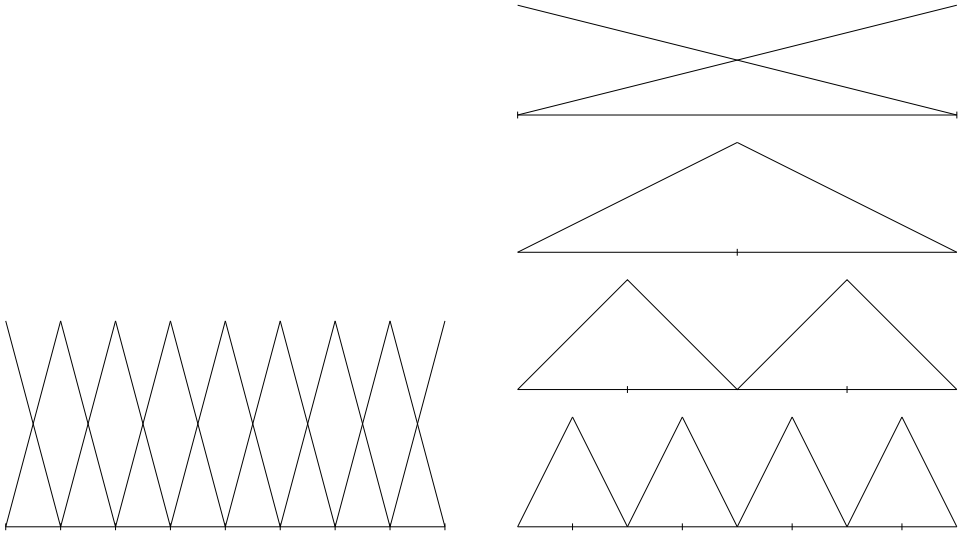
We now define a hierarchical difference space $W_{\underline{l}}$ via

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

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[1] than $V_{\underline{l}}$. To complete the definition, we formally set $V_{\underline{l}} := 0$, if $l_t = -1$ for at least one $t \in \{1, \ldots, d\}$. As can easily be seen from (2) and (5), the definition of the index set

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

[1] 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 $\Omega_{\underline{l}}$.



(a) Nodal basis for V_3 (b) Hierarchical basis for V_3

Figure 1: Nodal and hierarchical basis of level 3

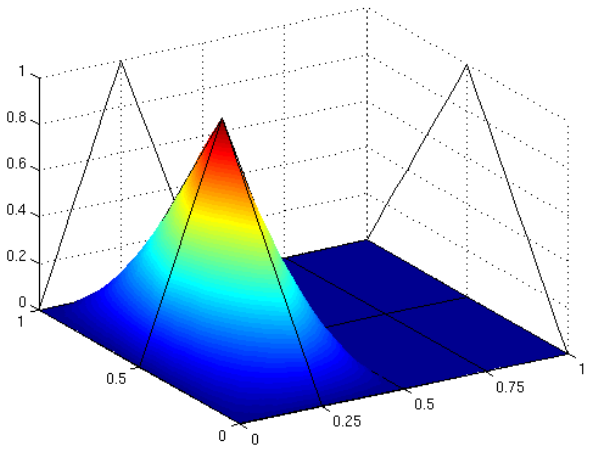


Figure 2: Basis function $\phi_{1,1}$ on grid $\Omega_{2,1}$.

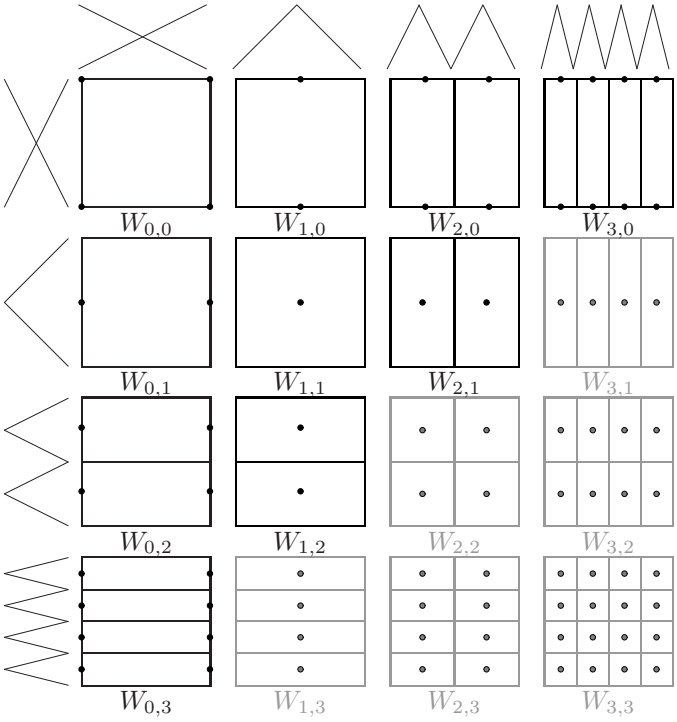


Figure 3: Supports of the basisfunctions of the hierarchical subspaces $W_{\underline{l}}$ of the space V_3

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 and in the following " $\le$ " refers to the element-wise relation for multi-indices. $|\underline{l}|_\infty := \max_{1\le t\le d} l_t$ 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 [Fab09, Yse86, Yse92] of V_n, which generalizes the one-dimensional hierarchical basis [Fab09], see Figure 1(b), 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 2.1 for a representation of the supports of the basis functions of the difference spaces W_{l_1,l_2} forming $V_{3,3}$.

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

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

where $\alpha_{\underline{l},\underline{j}} \in \mathbb{R}$ are the coefficients of the representation in the hierarchical tensor product basis. The number of basis functions, which describe a $f \in V_n$ in nodal or hierarchical

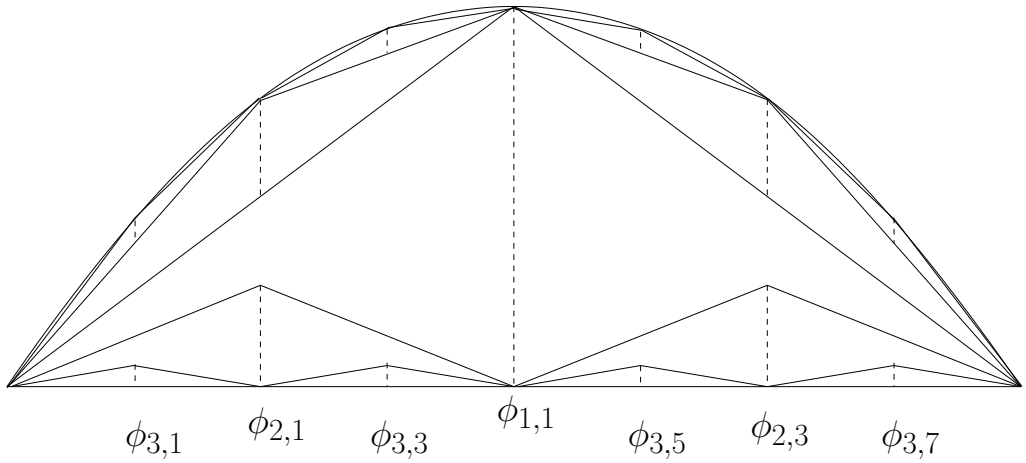


Figure 4: Interpolation with the hierarchical basis

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.

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

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

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 $\alpha_{l,j}$, $l \geq 1$, holds

$$\alpha_{l,j} = f(x_{l,j}) - \frac{f(x_{l,j} - h) + f(x_{l,j} + h)}{2} = f(x_{l,j}) - \frac{f(x_{l,j-1}) + f(x_{l,j+1})}{2}.$$

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

$$\alpha_{l,j} = \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l,j} f$$

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

$$\alpha_{\underline{l},\underline{j}} = \left(\prod_{t=1}^{d} \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l_t,j_t} \right) f \tag{11}$$

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. We define

the norm as

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

and the space H^2_{mix} in the usual way:

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

Furthermore we define the semi-norm $|f|_{H^2_{mix}} := |f|_{H^{\underline{2}}_{mix}}$

$$|f|_{H^{\underline{k}}_{mix}} := \left| \frac{\partial^{|\underline{k}|_1}}{\partial \underline{x}^{\underline{k}}} f \right|_2^2,$$

Note that the continuous function spaces H^s_{mix}, like the discrete spaces $V_{\underline{l}}$, have a tensor product structure [Wah90, Hoc99, GK00, HKZ00] 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 surplusses. We recite the following proofs from [Bun92, Bun98, BG99, BG04], 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.

Straightforward calculation shows

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

$$\|\phi_{\underline{l},\underline{j}}\|_2 \le C(d) \cdot 2^{-|\underline{l}|_1/2}.$$

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

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

Proof. In one dimension partial integration provides

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

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

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

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

Proof.

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

□

Lemma 4. *$f \in H^2_{0,mix}(\bar{\Omega})$ is as above in hierarchical representation. For its components $f_{\underline{l}} \in W_{\underline{l}}$ holds*

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

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

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

With Lemma 3 and 1 it now follows

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

which completes the proof. □

2.3 Sparse grids

Motivated by the relation (13) of the "importance" of the hierachical components $f_{\underline{l}}$ Zenger [Zen91] and Griebel [Gri91] introduce the so-called *sparse grids*, where hierarchical basis functions with a small support, and therefore a small part in 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

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

We replace in the definition (8) of V_n in terms of hierarchical subspaces the condition $|\underline{l}|_\infty \leq n$ with $|\underline{l}|_1 \leq n$. In Figure 2.1 the used subspaces are given in black, the difference spaces $W_{\underline{l}}$ which are not used anymore, in comparison to (8), are given in grey. Every $f \in V_n^s$ can now be represented, analogue to (10), as

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

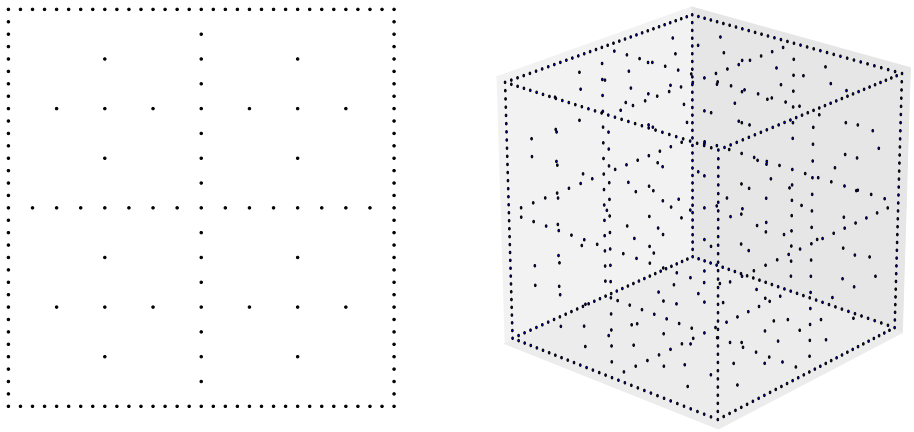


Figure 5: Two-dimensional sparse grid (left) and three-dimensional sparse grid (right) of level $n = 5$ each.

The resulting grids corresponding to the approximation space V_n^s are called sparse grids. Note that sparse grids were introduced in [Zen91, Gri91], and are used in this form, using the definition

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

This definition is especially useful for the numerical treatment of partial differential equations with Dirichlet boundary conditions where no degrees of freedom exist on the boundary. Using $V_{0,n}^s$ the level of refinement n in the sparse grid corresponds to the full grid case again. Move Examples in two and three dimensions are given in Figure 5. Sparse grids are studied in detail in [Bun92, Bun98, BG99, Kna00, BG04] besides others.

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, we only consider this case here, full results can be found in the given literature. 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 corresponding full grid of level n and more closely follows the referenced original publications.

First we look at the size of the sparse grid space.

Lemma 5. *The dimension of the sparse grid space $\mathring{V}_{0,n}^s$, i.e. the number of inner grid points, is given by*

$$|\mathring{V}_{0,n}^s| = \mathcal{O}(h_n^{-1} \cdot \log(h_n^{-1})^{d-1}) \tag{17}$$

Proof. We follow [BG04] and use in the first part the definition (16), the size of a hierarchical subspace $|W_{\underline{l}}| = 2^{|\underline{l}-\underline{1}|_1}$ and that there are $\binom{i-1}{d-1}$ possibilities to represent i as a sum

of d natural numbers.

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

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

$$\begin{aligned}
&\sum_{i=0}^{n-1} 2^i \cdot \binom{i+d-1}{d-1} \\
= \;& \frac{1}{(d-1)!} \sum_{i=0}^{n-1} \left(x^{i+d-1} \right)^{(d-1)} \Big|_{x=2} \\
= \;& \frac{1}{(d-1)!} \left(x^{d-1} \cdot \frac{1-x^n}{1-x} \right)^{(d-1)} \Big|_{x=2} \\
= \;& \frac{1}{(d-1)!} \sum_{i=0}^{d-1} \binom{d-1}{i} \cdot \left(x^{d-1} - x^{n+d-1} \right)^{(i)} \cdot \left(\frac{1}{1-x} \right)^{(d-1-i)} \Big|_{x=2} \\
= \;& (-1)^d + 2^n \cdot \sum_{i=0}^{d-1} \binom{n+d-1}{i} \cdot (-2)^{d-1-i}.
\end{aligned}$$

The summand for $i = d-1$ is 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.

We now look at approximation properties of sparse grids. For the proof we again follow [BG04] 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 $f^s_{0,n} \in \mathring{V}^s_{0,n}$ which can be written as a corresponding finite sum. The difference therefore is

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

For any norm now holds

$$\|f - f^s_{0,n}\| \leq \sum_{|\underline{l}|_1 > n+d-1} \|f_{\underline{l}}\|. \tag{18}$$

We need the following lemma to estimate the interpolation error

Lemma 6. *For $s \in \mathbb{N}$ it holds*

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

Proof. The steps of the proof are similar to the ones in the previous lemma, first we get

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

Since

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

it follows with $x = 2^{-s}$ the relation

$$\sum_{i=0}^{\infty} 2^{-s\cdot i} \cdot \binom{i+n+d-1}{d-1} \leq 2 \cdot \sum_{k=0}^{d-1} \binom{n+d-1}{k} = 2 \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}) \right)$$

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*

$$||f - f^s_n||_2 = \mathcal{O}(h_n^2 \log(h_n^{-1})^{d-1}). \tag{19}$$

Proof.

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

which gives the wanted relation. □

Note that same results holds in the maximum-norm as well:

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

for $f \in H^2_{0,mix}$.

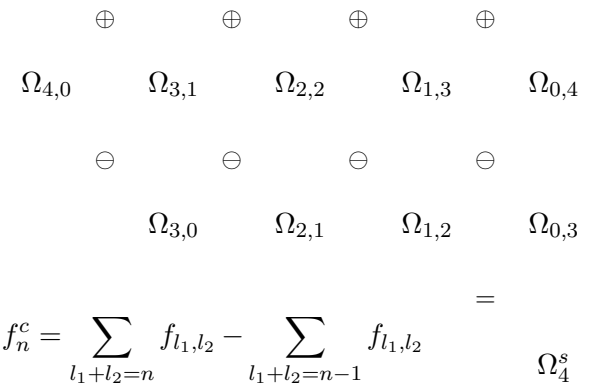


Figure 6: Combination technique with level $n = 4$ in two dimensions

2.4 Sparse grid combination technique

The so-called *combination technique* [GSZ92], which is based on multi-variate extrapolation [BGR94], 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 the on-the-fly computation of the matrix-vector-product, which complexity scales better, is challenging in the implementation [Ach03, Bal94, Bun98].

In particular, we discretize the function f on a certain sequence of anisotropic grids $\Omega_{\underline{l}} = \Omega_{l_1,...,l_d}$ 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

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

Note that in the original [GSZ92] and other papers as well, a slightly different definition was used, again due to Dirichlet boundary conditions,

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

The grids employed by the combination technique of level 4 in two dimensions are shown in Figure 6.

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

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

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

$$f_n^c(\underline{x}) := \sum_{q=0}^{d-1} (-1)^q \binom{d-1}{q} \sum_{|\underline{l}|_1 = n-q} f_{\underline{l}}(\underline{x}). \tag{20}$$

The resulting function f_n^c lives in the sparse grid space V_n^s, the combined interpolant is identically with the hierarchical sparse grid interpolant f_n^s [GSZ92]. 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. 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

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

with $t := n + 1 - l_1$. This is exactly (15).

Note that the solution obtained with the combination technique f_n^c for the numerical treatment of partial differential equations is in general not the sparse grid solution f_n^s. However, the approximation property is of the same order as long a series expansion of the error

$$f - f_{\underline{l}} = \sum_{i=1}^{d} \sum_{j_1,\ldots,j_m \subset 1,\ldots,d} c_{j_1,\ldots,j_m}(h_{j_1},\ldots,h_{j_m}) \cdot h_{j_1}^p \cdot \ldots \cdot h_{j_m}^p, \tag{21}$$

with bounded $c_{j_1,\ldots,j_m}(h_{j_1},\ldots,h_{j_m}) \le \kappa$ exists, see [GSZ92]. Its existence was shown for model-problems in [BGRZ94].

Let us again consider the two dimensional case and consider the error of the combined solution $f - f_n^c$ following [GSZ92]. We have

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

Plugging in the error expansion (21) leads to

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

And we get the estimation, using $c_i \leq \kappa$,

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

Observe that terms are canceled for h_{l_i} with $l_i \neq n$ and the accumulated $h_{l_1}^2 h_{l_2}^2$ result in $\log(h_n^{-1})$-term. The approximation order $\mathcal{O}(h_n^2 \cdot \log(h_n^{-1}))$ is just as in Theorem 1. See [GSZ92, PZ99, Rei04] for results in higher dimension.

As mentioned, the combination technique only gives the same order if the above error expansion exists. If this is not the case an optimised combination technique [HGC07] can be used to achieve good approximations. Here the combination coefficients are not fixed, but depend on the underlying problem and the function to be represented.

The combination technique was used in fluid dynamics [GHSZ93, GH95, GT95, Hub96, Kra02], for the Lamé-equation [Heu97], for eigenvalue-problems [Gar98, GG00], singular perturbation problems [NH00], machine learning and data analysis [GGT01, GG02, Gar04, GG05, LNSH05] and parabolic equations for option pricing [Rei04].

There are close connections to so-called boolean [Del82, DS89] and discrete blending methods [BDJ92], as well as the splitting extrapolation-method [LLS95]. Furthermore, there are relations between the sparse grid combination technique and additive models like e.g. ANOVA [Wah90] or MARS [Fri91].

A 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, write to `garcke@math.tu-berlin.de`.

A MATLAB implementation of sparse grids [KW05a, Kli06] can be found here `http://www.ians.uni-stuttgart.de/spinterp/`.

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

# 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
class sparseGrid:
```

```
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

# evaluate a sparse grid function, hierarchival values have to \
   be set
def evalFunct(self,x):
    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

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

    # d-dimensional recursion through all hierarchical subspaces
    def loopHierSpacesRec(self,dim, level):
        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()

    # fill self.gP with the points for the indices generated \
       beforehand
    def generatePoints(self):
        # 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)

    # we run over all hierarchical subspaces and add all their \
       indices
    def generatePointsRec(self,dim, level, cur_level=None):
        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)

    # 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
    def nodal2Hier1D(self,node,i,j,dim):
        # 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

```
# define the unit tests
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))

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

References

[Ach03] Achatz, S. *Higher Order Sparse Grid Methods for Elliptic Partial Differential Equations with Variable Coefficients.* Computing, 71(1):1–15, 2003.

[Bab60] Babenko, KI. *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.

[Bal94] Balder, R. *Adaptive Verfahren für elliptische und parabolische Differentialgleichungen auf dünnen Gittern.* Dissertation, Technische Universität München, 1994.

[Bas85] Baszenski, G. *N–th order polynomial spline blending.* In Schempp, W and Zeller, K, editors, *Multivariate Approximation Theory III*, ISNM 75, pages 35–46. Birkhäuser, Basel, 1985.

[BD03] Bungartz, HJ and Dirnstorfer, S. *Multivariate Quadrature on Adaptive Sparse Grids.* Computing, 71:89–114, August 2003.

[BDJ92] Baszenski, G, Delvos, FJ, and Jester, S. *Blending approximations with sine functions.* In Braess, D, editor, *Numerical Methods in Approximation Theory IX*, ISNM 105, pages 1–19. Birkhäuser, Basel, 1992.

[BG99] Bungartz, HJ and Griebel, M. *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.

[BG04] Bungartz, HJ and Griebel, M. *Sparse Grids.* Acta Numerica, 13:147–269, 2004.

[BGR94] Bungartz, HJ, Griebel, M, and Rüde, U. *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.

[BGRZ94] Bungartz, HJ, Griebel, M, Röschke, D, and Zenger, C. *Pointwise convergence of the combination technique for the Laplace equation.* East-West J. Numer. Math., 2:21–45, 1994.

[Bun92] Bungartz, HJ. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung.* Dissertation, Institut für Informatik, Technische Universität München, 1992.

[Bun98] Bungartz, HJ. *Finite Elements of Higher Order on Sparse Grids.* Habilitation, Institut für Informatik, Technische Universität München and Shaker Verlag, Aachen, 1998.

[Del82] Delvos, FJ. *d-Variate Boolean Interpolation.* J. Approx. Theory, 34:99–114, 1982.

[DS89] Delvos, FJ and Schempp, W. *Boolean Methods in Interpolation and Approximation.* Pitman Research Notes in Mathematics Series 230. Longman Scientific & Technical, Harlow, 1989.

[Fab09] Faber, G. *Über stetige Funktionen.* Mathematische Annalen, 66:81–94, 1909.

[FHP96] Frank, K, Heinrich, S, and Pereverzev, S. *Information Complexity of Multivariate Fredholm Integral Equations in Sobolev Classes.* J. of Complexity, 12:17–34, 1996.

[Fri91] Friedman, JH. *Multivariate adaptive regression splines.* Ann. Statist., 19(1):1–141, 1991.

[Gar98] Garcke, J. *Berechnung von Eigenwerten der stationären Schrödingergleichung mit der Kombinationstechnik.* Diplomarbeit, Institut für Angewandte Mathematik, Universität Bonn, 1998.

[Gar04] Garcke, J. *Maschinelles Lernen durch Funktionsrekonstruktion mit verallgemeinerten dünnen Gittern.* Doktorarbeit, Institut für Numerische Simulation, Universität Bonn, 2004.

[Gar06] Garcke, J. *Regression with the optimised combination technique.* In Cohen, W and Moore, A, editors, *Proceedings of the 23rd ICML '06*, pages 321–328. ACM Press, New York, NY, USA, 2006.

[GG98] Gerstner, T and Griebel, M. *Numerical Integration using Sparse Grids.* Numer. Algorithms, 18:209–232, 1998.

[GG00] Garcke, J and Griebel, M. *On the computation of the eigenproblems of hydrogen and helium in strong magnetic and electric fields with the sparse grid combination technique.* Journal of Computational Physics, 165(2):694–716, 2000.

[GG02] Garcke, J and Griebel, M. *Classification with sparse grids using simplicial basis functions.* Intelligent Data Analysis, 6(6):483–502, 2002. (shortened version appeared in KDD 2001, Proc. of the Seventh ACM SIGKDD, F. Provost and R. Srikant (eds.), pages 87-96, ACM, 2001).

[GG03] Gerstner, T and Griebel, M. *Dimension–Adaptive Tensor–Product Quadrature.* Computing, 71(1):65–87, 2003.

[GG05] Garcke, J and Griebel, M. *Semi-supervised learning with sparse grids.* In Amini, MR, Chapelle, O, and Ghani, R, editors, *Proceedings of ICML, Workshop on Learning with Partially Classified Training Data*, pages 19–28. 2005.

[GGT01] Garcke, J, Griebel, M, and Thess, M. *Data mining with sparse grids.* Computing, 67(3):225–253, 2001.

[GH95] Griebel, M and Huber, W. *Turbulence simulation on sparse grids using the combination method.* In Satofuka, N, Periaux, J, and Ecer, A, editors, *Parallel Computational Fluid Dynamics, New Algorithms and Applications*, pages 75–84. North-Holland, Elsevier, 1995.

[GH03] Gradinaru, V and Hiptmair, R. *Multigrid for discrete differential forms on sparse grids.* Computing, 71(1):17–42, 2003.

[GHSZ93] Griebel, M, Huber, W, Störtkuhl, T, and Zenger, C. *On the parallel solution of 3D PDEs on a network of workstations and on vector computers.* In Bode, A and Cin, MD, editors, *Parallel Computer Architectures: Theory, Hardware, Software, Applications*, volume 732 of *Lecture Notes in Computer Science*, pages 276–291. Springer Verlag, 1993.

[GK00] Griebel, M and Knapek, S. *Optimized tensor-product approximation spaces.* Constructive Approximation, 16(4):525–540, 2000.

[GK03] Griebel, M and Koster, F. *Multiscale Methods for the Simulation of Turbulent Flows.* In Hirschel, E, editor, *Numerical Flow Simulation III*, volume 82 of *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, pages 203–214. Springer-Verlag, 2003.

[GOS99] Griebel, M, Oswald, P, and Schiekofer, T. *Sparse Grids for Boundary Integral Equations.* Numer. Mathematik, 83(2):279–312, 1999.

[Gri91] Griebel, M. *A parallelizable and vectorizable multi-level algorithm on sparse grids.* In Hackbusch, W, 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.

[Gri98] Griebel, M. *Adaptive Sparse Grid Multilevel Methods for elliptic PDEs based on finite differences.* Computing, 61(2):151–179, 1998.

[GSZ92] Griebel, M, Schneider, M, and Zenger, C. *A combination technique for the solution of sparse grid problems.* In de Groen, P and Beauwens, R, editors, *Iterative Methods in Linear Algebra*, pages 263–281. IMACS, Elsevier, North Holland, 1992.

[GT95] Griebel, M and Thurner, V. *The efficient solution of fluid dynamics problems by the combination technique.* Int. J. Num. Meth. for Heat and Fluid Flow, 5(3):251–269, 1995. Also as SFB Bericht 342/1/93 A, Institut für Informatik. TU München, 1993.

[HBS+06] Hegland, M, Burden, C, Santoso, L, MacNamara, S, and Booth, H. *A solver for the stochastic master equation applied to gene regulatory networks.* Journal of Computational and Applied Mathematics, In Press, Corrected Proof, 2006.

[Hem95] Hemker, P. *Sparse-Grid finite-volume multigrid for 3D-problems.* Advances in Computational Mathematics, 4:83–110, 1995.

[Heu97] Heusser, N. *Berechnung einer Dünngitterlösung der dreidimensionalen Lamé-Gleichung mit der Kombinationsmethode.* Diplomarbeit, Institut für Angewandte Mathematik, Rheinische Friedrich-Wilhelms-Universität Bonn, 1997.

[HGC07] Hegland, M, Garcke, J, and Challis, V. *The combination technique and some generalisations.* Linear Algebra and its Applications, 420(2–3):249–275, 2007.

[HKZ00] Hochmuth, R, Knapek, S, and Zumbusch, G. *Tensor products of Sobolev spaces and applications.* submitted, 2000. Also as Technical Report 685, SFB 256, Univ. Bonn.

[Hoc99] Hochmuth, R. *Wavelet Bases in numerical Analysis and Restrictal Nonlinear Approximation.* Habilitation, Freie Universität Berlin, 1999.

[Hub96] Huber, W. *Turbelenzsimulation mit der Kombinationsmethode auf Workstation-Netzen und Parallelrechnern.* Dissertation, Institut für Informatik, Technische Universität München, 1996.

[Kli06] Klimke, A. *Sparse Grid Interpolation Toolbox – User's Guide.* Technical Report IANS report 2006/001, University of Stuttgart, 2006.

[Kna00] Knapek, S. *Approximation und Kompression mit Tensorprodukt-Multiskalenräumen.* Doktorarbeit, Universität Bonn, April 2000.

[Kos02] Koster, F. *Multiskalen-basierte Finite Differenzen Verfahren auf adaptiven dnnen Gittern.* Doktorarbeit, Universität Bonn, January 2002.

[Kra02] Kranz, CJ. *Untersuchungen zur Kombinationstechnik bei der numerischen Strömungssimulation auf versetzten Gittern.* Dissertation, Institut für Informatik, Technische Universität München, 2002.

[KW05a] Klimke, A and Wohlmuth, B. *Algorithm 847: Spinterp: piecewise multilinear hierarchical sparse grid interpolation in MATLAB.* ACM Trans. Math. Softw., 31(4):561–579, 2005.

[KW05b] Klimke, A and Wohlmuth, B. *Computing expensive multivariate functions of fuzzy numbers using sparse grids.* Fuzzy Sets and Systems, 154(3):432–453, 2005.

[LLS95] Liem, CB, Lü, T, and Shih, TM. *The Splitting Extrapolation Method.* World Scientific, Singapore, 1995.

[LNSH05] Laffan, SW, Nielsen, OM, Silcock, H, and Hegland, M. *Sparse Grids: a new predictive modelling method for the analysis of geographic data.* International Journal of Geographical Information Science, 19(3):267–292, 2005.

[NH00] Noordmans, J and Hemker, P. *Application of an Adaptive Sparse Grid Technique to a Model Singular Perturbation Problem.* Computing, 65:357–378, 2000.

[PZ99] Pflaum, C and Zhou, A. *Error analysis of the combination technique.* Numer. Math., 84(2):327–350, 1999.

[Rei04] Reisinger, C. *Numerische Methoden für hochdimensionale parabolische Gleichungen am Beispiel von Optionspreisaufgaben.* Ph.D. thesis, Ruprecht-Karls-Universität Heidelberg, 2004. In Vorbereitung.

[Sch99] Schiekofer, T. *Die Methode der Finiten Differenzen auf dünnen Gittern zur Lösung elliptischer und parabolischer partieller Differentialgleichungen.* Doktorarbeit, Universität Bonn, 1999.

[Smo63] Smolyak, SA. *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.

[SS99] Sickel, W and Sprengel, F. *Interpolation on Sparse Grids and Nikol'skij–Besov spaces of dominating mixed smoothness.* J. Comput. Anal. Appl., 1:263–288, 1999.

[ST03a] Schwab, C and Todor, R. *Sparse finite elements for stochastic elliptic problems — higher order moments.* Computing, 71(1):43–63, 2003.

[ST03b] Schwab, C and Todor, RA. *Sparse finite elements for elliptic problems with stochastic loading.* Numer. Math., 95(4):707–734, 2003.

[Tem89] Temlyakov, VN. *Approximation of functions with bounded mixed derivative.* Proc. Steklov Inst. Math., 1, 1989.

[Tem93a] Temlyakov, VN. *Approximation of Periodic Functions.* Nova Science, New York, 1993.

[Tem93b] Temlyakov, VN. *On approximate recovery of functions with bounded mixed derivative.* J. Complexity, 9:41–59, 1993.

[vPS04] von Petersdorff, T and Schwab, C. *Numerical Solution of Parabolic Equations in High Dimensions.* Mathematical Modeling and Numerical Analysis, 2004. To appear.

[Wah90] Wahba, G. *Spline models for observational data*, volume 59 of *Series in Applied Mathematics.* SIAM, Philadelphia, 1990.

[Yse86] Yserentant, H. *On the Multi-Level Splitting of Finite Element Spaces.* Numerische Mathematik, 49:379–412, 1986.

[Yse92] Yserentant, H. *Hierarchical bases.* In R. E. O'Malley, J et al., editors, *Proc. ICIAM'91.* SIAM, Philadelphia, 1992.

[Zen91] Zenger, C. *Sparse Grids.* In Hackbusch, W, 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.