Spatially Adaptive Refinement

Dirk Pflüger
Department of Informatics, Technische Universität München,
Boltzmannstr. 3, 85748 Garching, dirk.pflueger@in.tum.de

March 17, 2014

Preprint. To appear in J. Garcke and M. Griebel (ed.), Sparse Grids and Applications of Lecture Notes in Computational Science and Engineering, p. 243262. Springer, Berlin Heidelberg, October 2012

Abstract

While sparse grids allow one to tackle problems in higher dimensionalities than possible for standard grid-based discretizations, real-world applications often come along with requirements or restrictions which enforce problem-dependent adaptations of the standard sparse grid technique. Consider, for example, interpolations where the function values at grid points are obtained via time-consuming numerical simulations. Then, only very few grid points can be spent; classical convergence might be out of reach. Another hurdle is that real-world problems often do not meet the smoothness requirements of the sparse grid method. Thus, the standard approach has to be fine-tuned to the problem at hand, especially in higher-dimensional settings.

Therefore, a suitable choice of basis functions can be required, as well as criteria for problem-adapted refinement. Fortunately, and in contrast to full grids, the hierarchical basis formulation of the direct sparse grid approach conveniently provides a reasonable criterion for spatially adaptive refinement practically for free. This can serve as a starting point to develop suitable modifications.

We show several problems stemming from different fields of application and demonstrate modifications of the standard sparse grid approach. They enable one to cope with the properties and requirements of the corresponding problem and can serve as examples for similar challenges.

1 Introduction

Sparse grids allow us to cope with the curse of dimensionality, at least to some extent. The term "sparse grids" has been coined for the solution of partial differential equations [28], and sparse grids have meanwhile been applied to various problems, see, for example, the survey in [4]. More recent work includes stochastic and non-stochastic partial differential equations in various settings [25, 11, 27, 3], as well as applications in economics [22, 18], regression [15, 12, 21], classification [14, 6, 21], fuzzy modeling [19], and more.

For sufficiently smooth functions, sparse grids enable one to reduce the number of grid points by orders of magnitude from $O((2^n)^d)$ for full grids to only $O(2^n n^{d-1})$, while keeping a similar accuracy as in the full grid case. To obtain these bounds, only a certain degree of smoothness is required (the mixed second derivatives have to be bounded), but no other knowledge about the problem at hand. Note, that the constants in the Landau-notation depend on the application.

To be able to deal with problems that do not meet the smoothness requirement, or to further reduce the number of grid points for functions that exhibit a low effective dimensionality or that contain regions of small and large variation, adaptivity can be employed. To this end, the hierarchical basis directly provides a straightforward indicator where to refine in general. Whereas this is typically very good to start with, the success for real-world applications often depends on a suitable choice of the basis, the criterion for adaptive refinement, and the selection of refinement parameters. Therefore, available knowledge about the problem should be used wherever possible.

To illustrate this, we picked a few applications which nicely demonstrate several approaches for adaptive refinement, and we give hints what to look for and consider when thinking about adapting to a problem at hand.

2 Sparse Grids

To briefly recall the most important properties and to clarify our notation, we describe the basic principles of sparse grids in the following; see, e.g., [4, 21] for further details. Sparse grids are based on a hierarchical (and thus inherently incremental and adaptive) formulation of the one-dimensional basis which is then extended to the d-dimensional setting via a tensor product approach.

We consider high-dimensional piecewise d-linear functions $f_N : [0,1]^d \to \mathbb{R}$ (which are defined on an equidistant mesh and scaled to the unit-hypercube) as a weighted sum of N basis functions,

$$f_N(\vec{x}) = \sum_{i=1}^{N} \alpha_i \varphi_i(\vec{x}) \,. \tag{1}$$

We therefore derive one-dimensional basis functions $\varphi_{l,i}$, depending on a level l and an index i, out of the reference hat function $\varphi(x) := \max(1 - |x|, 0)$ via translation and scaling as $\varphi_{l,i}(x) := \varphi(2^l x - i)$. The hierarchical basis for a certain level n with

mesh-width $h_n = 2^{-n}$ is then

$$\Phi_n := \left\{ \varphi_{l,i}(x) : i = 1, \ldots, 2^l,\ i \text{ odd},\ 1 \leq l \leq n \right\}, \tag{2}$$

omitting even-indexed basis functions on each level, see Fig. 1 (left) for $n = 3$. If we denote $\vec{l}$ and $\vec{i}$ as multi-indices of levels and indices for a certain basis function, we can then write d-dimensional basis functions $\varphi_{\vec{l},\vec{i}}$ as a product of the respective one-dimensional ones,

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{k=1}^{d} \varphi_{l_k,i_k}(x_k),$$

obtaining piecewise d-linear basis functions. They are centered at grid points $\vec{x}_{\vec{l},\vec{i}} = (i_1 2^{l_1}, \ldots, i_d 2^{l_d})$, and $|\vec{l}|_1$ denotes the classical l^1 norm for vectors, i.e. the sum of the one-dimensional levels.

In higher-dimensional settings, we obtain hierarchical increments (function spaces) $W_{\vec{l}}$ for which the grid points are the Cartesian product of the one-dimensional ones on the respective one-dimensional levels. We denote the corresponding basis $\Phi_{\vec{l}}$, i.e. $\text{span}(\Phi_{\vec{l}}) = W_{\vec{l}}$. Figure 1 (middle) shows the grids of the two-dimensional hierarchical increments $W_{\vec{l}}$ up to level 3 in each dimension. Note that in each $W_{\vec{l}}$, all basis functions have supports with piecewise disjoint interiors.

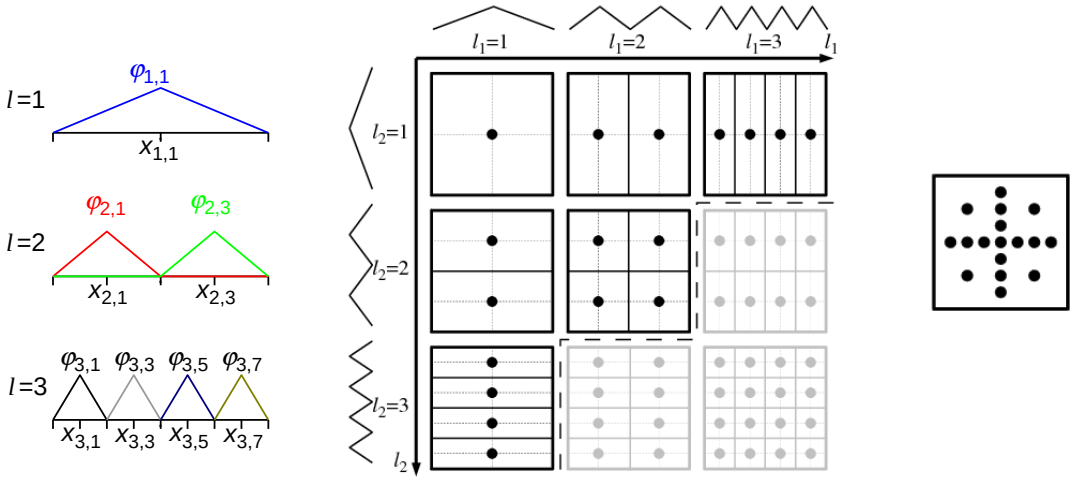

Figure 1: One-dimensional basis functions up to level 3 (left), and tableau of hierarchical increments $W_{\vec{l}}$ up to level 3 in both dimensions (middle). Leaving out the grayed-out $W_{\vec{l}}$, we obtain the sparse grid of level 3 (right).

The hierarchical representation now allows one to select only those subspaces that contribute most to the overall solution. This can be done by an a priori selection (see

[4] for details). We then obtain a sparse grid space such as

$$V_n^{(1)} := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}},$$

which in this case is optimized with respect to both the L^2-norm and the maximum-norm. In the example in Fig. 1, we can neglect the gray $W_{\vec{l}}$ for $n = 3$, which leads to the regular (non adaptive) sparse grid in Fig. 1 (right). To this end, the function f under consideration has to be sufficiently smooth, i.e., the mixed second derivatives $\left|D^{\vec{2}} f\right| := \left|\frac{\partial^{2d}}{\partial x_1^2 \cdots \partial x_d^2} f\right|$ have to be bounded.

3 Adaptivity

A straightforward possibility to adapt to a problem at hand is to refine by adding new subspaces, weakening the diagonal cut-off in the subspace scheme. Adding a new subspace $W_{\vec{l}}$ in an incremental way requires that all backward neighbors, i.e., all subspaces $W_{\vec{l}'}$ with $\vec{l}' \leq \vec{l}$ (componentwise comparison of multi-indices), have already been included in the current set of subspaces. As this refinement treads all grid points with respect to a single dimension in a uniform way, this is referred to as dimensionally adaptive refinement. Note that this corresponds to the adaptive refinement in the context of the combination technique [16].

To determine where to refine, one can consider all subspaces that can be added next (for which all the backward neighbors exist) and add the one which reduces the error most (based on a suitable error measure). Unfortunately, this is infeasible in many applications, such as settings where function values are costly to obtain or where PDEs have to be solved: too many grid points which will quite likely never be used have to be examined. A frequently used alternative is to consider all current subspaces that can still be refined. The one which contributes most to the error is identified and refined by creating all the missing direct forward neighbors (incrementing the level in one of the dimensions each). This approach typically reduces the computational effort significantly.

When working in the direct, hierarchical sparse grid basis, the same approaches can be applied. But, in contrast to the combination technique, single grid points can be easily refined, and spatial (local) adaptivity can be employed. New basis functions just extend the current basis, and no side-effects such as hanging nodes have to be considered. In the d-dimensional hierarchical structure, a grid point has $2d$ children and up to d parents. The $2d$ children of a grid point $\vec{x}_{\vec{l},\vec{i}}$ are $\{\vec{x}_{\tilde{\vec{l}},\tilde{\vec{i}}} : \tilde{l}_k = l_k + 1, \tilde{i}_k = i_k \pm 1, \tilde{l}_t = l_t, \tilde{i}_t = i_t, t \neq k, k = 1, \ldots, d\}$. Refinable grid points (or leaves) in the hierarchical structure are all those grid points for which at least one child does not exist yet. New direct candidates are all grid points for which all the parents are contained in the current grid, similar to the subspaces in the dimensionally adaptive refinement.

Considering the refinement of single grid points, it is even more obvious that it is infeasible to look at all new candidates: already in the one-dimensional case there are as many grid points on the next level as there are in the current grid—and they all have to

be considered. Solving a PDE, e.g., this would require to solve the PDE N times for N new candidates to select the one with the lowest error. Therefore, the standard strategy is to consider all refinable grid points of the current grid (for which all information has already been obtained), and refine the most promising one(s) by adding all missing children.

Note that for conventional algorithms working on sparse grids, all hierarchical ancestors of each grid point have to exist. To keep a grid consistent, all missing parents of new grid points have to be created recursively. This can result in significantly more than $2d$ grid points to be created per refinement, see Fig. 2 for an example.

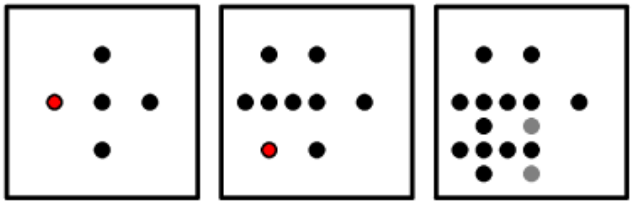

Figure 2: Starting with a regular grid of level 2 (left), we refine one grid point (emphasized) by creating all children in the hierarchical tree of basis functions (middle), and we repeat this once more. To keep the grid consistent, two missing parents have to be created (right).

The hierarchical basis, in contrast to the nodal basis, inherently provides a simple, though effective criterion of where to refine to minimize the L^2-norm of the error. Consider a sparse grid interpolant $f_N \in V_n^{(1)}$, $f_N(\vec{x}) = \sum_{|\vec{l}|_1 \leq n+d-1} \sum_{\varphi_{\vec{l},\vec{i}} \in \Phi_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x})$, with surplusses (hierarchical coefficients) $\alpha_{\vec{l},\vec{i}}$. Recall, that the smoothness requirement which we impose at f is that $\left|D^{\vec{2}} f\right|$ has to be bounded. The surplusses can then be expressed via their integral representation as

$$\alpha_{\vec{l},\vec{i}} = \left(\prod_{j=1}^{d} \frac{-h_j}{2} \right) \int_{\Omega^d} \varphi_{\vec{l},\vec{i}}(\vec{x})\, D^{\vec{2}} f \, d\vec{x}\,,$$

see [4] for details.

Two important lessons can be learned: First, both the product of the mesh-widths h_j and the size and shape of the basis function $\varphi_{\vec{l},\vec{i}}$ depend only on $\vec{l}$ and not on the index of the grid point. For each grid point on the same level (with constant $|\vec{l}|_1$), the absolute value of the surplus depends mainly on $|D^{\vec{2}} f|$ within the support of the corresponding basis function. The more f varies on the support, the higher in general the absolute value of the surplus. Thus, the absolute value of the hierarchical coefficient can directly be used as a criterion for adaptive refinement, assuming a similar level sum for all refinement candidates. The fact that the hierarchical basis provides a cheap, simple criterion for

refinement and does not necessitate to derive error estimators depending on the problem at hand, is one of the main advantages of the hierarchical approach. Note that, of course, there is no guarantee that this works in every setting, and that each and every criterion of where to refine can be fooled.

Second, $D^{\vec{2}}f$ can be assumed to be locally constant in the case of convergence. Then the contribution of a basis function decreases by $1/4$ when increasing the level by one in one dimension. The surplus can thus be used to check for convergence with respect to the discretization level or to detect unwanted noise in the function values [21]. For the interpolation of the normalized product of one-dimensional parabolas $f(\vec{x}) := 4^d \prod_{k=1}^{d}(1 - x_k)x_k$, this decay of the surplusses can be observed right from the second level. This is the reason why this function is frequently used to test and illustrate sparse grid interpolation.

4 Problem-Awareness

Refining grid points in the hierarchical basis, an incremental method is obtained which is equipped with a simple criterion and adapts locally to the problem at hand. But experience shows that in real-world settings a mere surplus-based refinement might not be sufficient to solve a problem with sufficient accuracy, and convergence might even be out of reach. The reasons for this are manifold and typically caused by the fact that the number of grid points is limited. For example, a problem with low effective dimensionality is posed in a too high-dimensional setting and the number of grid points grows too fast. Or the number of grid points that can be spent is restricted because it is very expensive to obtain or store function evaluations: each grid point might require an expensive simulation, or the memory requirements for storing a whole $3d$ simulation result at each grid point might be too high. There, the number of grid points that can be spent is severely limited. Furthermore, the problem itself could impose additional requirements to the sparse grid function.

In all these cases, the standard adaptive sparse grid approach has to be adapted to the problem at hand. In the following, we show some problems and strategies to give a start even if no additional knowledge about the problem is available. Ingredients are a suitable choice of the one-dimensional basis functions, threshold-based refinement, coarsening, weighted adaptivity, and trading off broad against steep refinement. But first, let us comment on the treatment of the boundary of the sparse grid domain Ω.

So far, we have only considered functions that are zero on the domain's boundary $\delta\Omega$. To allow for non-zero values on the boundary, usually additional grid points located directly on $\delta\Omega$ are introduced. The most common approach is to spend two more degrees of freedom in the one-dimensional hierarchical scheme for the grid on level 1 (which up to now only used $\varphi_{1,1}$), namely the basis functions with level 0 and indices 0 and 1. This leads in the d-dimensional case to 3^d unknowns for the initial grid and introduces an exponential dependency on the dimensionality that is significant for practical computations. For a problem in 100 dimensions, we could not even start computing the solution even if there was only one relevant dimension.

Therefore, the grid points on the boundary should be omitted wherever possible. Instead, the basis functions adjacent to the boundary have to be modified. A good choice is

$$\varphi_{l,i}(x) := \begin{cases} 1 & \text{if } l = 1 \wedge i = 1\,, \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x & \text{if } x \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 1\,, \\ \left\{ \begin{array}{ll} 2^l \cdot x + 1 - i & \text{if } x \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1\,, \\ \varphi\left(x \cdot 2^l - i\right) & \text{else} \end{cases}$$

for the one-dimensional basis functions, extrapolating linearly towards the boundary [21, 20, 6], see Fig. 3 (left). This allows to start with only $O(d)$ basis functions (one center point and one pair of points for probing in each coordinate direction); a suitable refinement will then create grid points where it is really necessary. This approach can be similarly applied to other types of basis functions, such as piecewise polynomial ones, B-splines or (pre-)wavelets, see [21] for details.

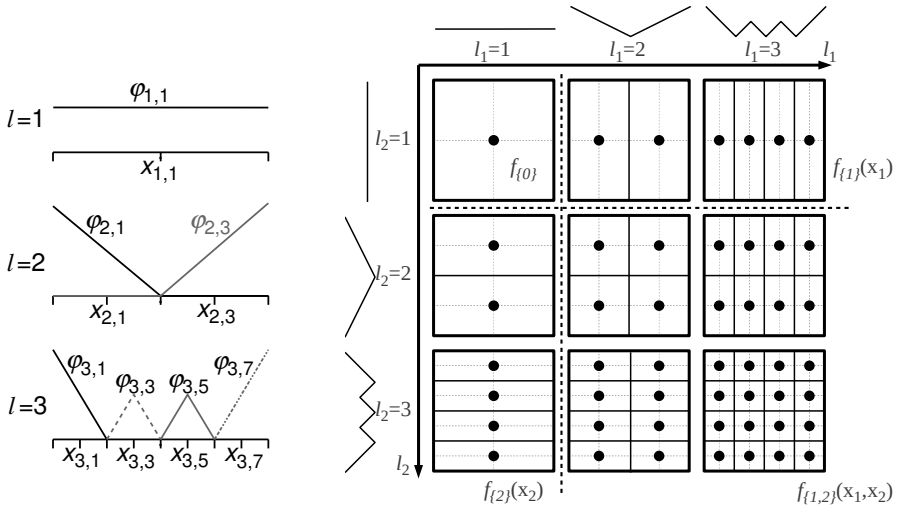

Figure 3: The classical one-dimensional hierarchical hat basis functions with boundary basis functions on level 0, dashed (left) and the modified basis functions (right)

There is an additional advantage of starting with a constant basis function on the first level. This allows for anchored ANOVA-style decompositions, representing a d-dimensional function f as

$$f(x_1 \ldots x_d) = f_{\{0\}} + \sum_i f_{\{i\}}(x_i) + \sum_{i<j} f_{\{i,j\}}(x_i, x_j) + \ldots + f_{\{1,\ldots,d\}}(x_1 \ldots x_d)\,,$$

anchored at the grid point $\vec{x}_{\vec{1},\vec{1}}$ on the first level. Figure 3 (right) illustrates the decomposition in terms of subspaces in two dimensions. If it can be identified which ANOVA

components are important for a certain problem, then refinement can be restricted to create only grid points belonging to the respective subspaces. The identification could be analytically, or even empirically by sampling the function on a sparse grid with sufficiently high level, identifying the ANOVA components based on the sizes of their surplusses, and then coarsening the grid by omitting all grid points belonging to unimportant ANOVA components.

5 Examples and Strategies

As several of our examples focus on the approximation of high-dimensional functions based on noise-prone data sets, we start with a description of sparse-grid-based data mining. The other examples do not require an extra introduction. Please note, that our aim is not to cover the examples in detail and to full extent, but rather to describe approaches to and strategies for adaptive refinement which have shown to be of relevance in plenty of settings.

Two prominent and related tasks in data mining are classification and regression, both of which aim to generalize from known data to predict a target property for new, previously unknown data. Thus, we start with a set S of m data points $\vec{x}_j \in \mathbb{R}^d$ of which we know some target value $y_j \in K$ in the d-dimensional feature space,

$$S = \left\{(\vec{x}_j, y_j) \in \mathbb{R}^d \times K\right\}_{j=1,\ldots,m} .$$

To be able to learn and generalize, we assume that we obtained S by a random and noise-prone sampling of an unknown function f which we aim to reconstruct. This function will then allow us to predict a target value at new locations $\vec{x}$. As we are dealing with finite data, we can scale in the following to the domain $[0,1]^d$.

For the task of binary classification, think of a bank discriminating potential customers into creditworthy and non-creditworthy, or a production facility predicting the faultiness of products based on standard measurements. Thus, we use two distinct target values, $K = \{-1, +1\}$. Nevertheless, we learn a continuous function f and then check whether the function value is negative or not. This way, the absolute function value provides a measure of confidence in our prediction.

For the task of regression, arbitrary real values are allowed, $K = \mathbb{R}$. An example, which we will use later on, is the prediction of a certain physical property of galaxies which is costly and difficult to obtain. Thus, learning f from the observations that have already been obtained with a lot of effort allows us to predict this property for new galaxies.

We restrict ourselves to reconstructions f_N of f in some sparse grid space $V_n^{(1)}$. To obtain a unique f_N and to be able to deal with noise, we solve the regularized least squares problem

$$f_N \stackrel{!}{=} \operatorname*{arg\,min}_{f_N \in V_N} \left(\frac{1}{m} \sum_{j=1}^{m} \left(y_j - f_N(\vec{x}_j)\right)^2 + \lambda \sum_{k=1}^{N} \alpha_k^2 \right) , \tag{3}$$

see [21, 14] and the references cited therein for further details. On the one hand, we ensure closeness to our training data, minimizing the mean square error (MSE) on it. On the other hand, we incorporate the smoothness assumption in data mining, which states that close data points are very likely to have a similar function value, by enforcing some degree of smoothness of f_N and preventing oscillations. Note that working in the hierarchical basis allows us to use an unconventional regularization functional: in the piecewise linear hierarchical setting without grid points on the boundary, $\sum_{k=1}^{N} \alpha_k^2$ corresponds to the squared Sobolev norm $\|f\|_{H^1_{\text{mix}}}$ for normalized basis functions $\tilde{\varphi}_{\vec{l},\vec{i}}(\vec{x}) := \sqrt{2^{-|\vec{l}|_1-d}}\varphi_{\vec{l},\vec{i}}(\vec{x})$ and is thus well-suited for our choice of basis functions [21], i.e.,

$$\|f\|^2_{H^1_{\text{mix}}} := \|\frac{\partial^d}{\partial x_1, \ldots, \partial x_d} f\|^2_{L^2} = \sum_{k=1}^{N} \alpha_k^2 \,.$$

The trade-off between error and smoothness can be influenced by a good choice of the regularization operator λ; for a given data set this can be achieved via cross-validation [2], for example. Note that we follow a general approach: other classification methods can be formulated the same way choosing different error and smoothness terms [9].

Minimizing (3), we obtain a system of linear equations

$$\left(\frac{1}{m} B B^T + \lambda I\right) \vec{\alpha} = \frac{1}{m} B \vec{y}\,, \tag{4}$$

the solution of which is the coefficient vector $\vec{\alpha}$ of f_N. The identity matrix I stems from the smoothness term; the $N \times m$ and $m \times N$ matrices B and B^T, $b_{ij} = \varphi_i(\vec{x}_j)$, and the vector $\vec{y}$ of the target values y_i from the error term.

The main advantage of the mesh-based sparse grid approach is that the resulting algorithms scale only linearly in the number of training data points—in contrast to most classical, data-centered approaches which scale typically at least quadratically or even worse. Thus, almost arbitrary amounts of data can be dealt with.

5.0.1 Straightforward Adaptive Refinement

The first example demonstrates that mere dimensional adaptivity, refining and adding whole subspaces, can be insufficient: in contrast to what is known from the solution of PDEs, spending too many grid points in wrong regions can lead to a higher overall error, as this allows the function to adapt too well to the noise in the training data (overfitting). The example is a two-dimensional artificial classification task, taken from [23]. It consists of two data sets, one for training and one for testing, and has been constructed to comprise 8% of error. It shows typical characteristics of real-world data sets, as it is neither linearly separable nor very complicated. Figure 4 shows the two data sets as well as the best separation manifold obtained by an adaptive sparse grid. Already eight refinement steps are enough to obtain an excellent accuracy of 91.5% on the test data—out of a maximum of 92%.

It can be seen that the separation boundary is resolved better in the central, critical region than in regions where very little information (data points) about the underlying

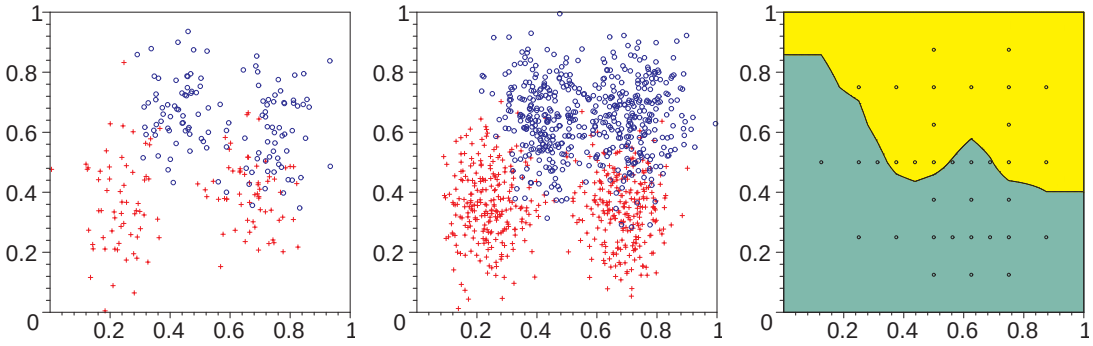

Figure 4: Ripley data set: 250 data points for training (left), 1000 to test on (middle), and the classification areas together with the corresponding sparse grid (right).

function is given. This is due to the adaptive refinement. Interestingly, after only eight refinements, overfitting starts to take over and the accuracy deteriorates. For this data set a mere dimension-adaptive refinement leads to a lower accuracy: whereas more grid points towards the center of the feature space are necessary to improve, spending the same discretization level on the whole horizontal main axis of the grid leads to local overfitting towards the boundary and to higher errors.

5.0.2 Criterion for Adaptive Refinement and Dimensional Adaptivity

The second example is a classical, artificial 10-dimensional data set, the Friedman1 data set, obtained by a random sampling of an analytical function, enriched by normally distributed noise [10]:

$$f(x_1, \ldots, x_{10}) = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \epsilon\,. \tag{5}$$

All ten variables $x_1, \ldots, x_{10}$ are in $[0, 1]$, thus we do not have to normalize this data set. The function value depends only on the first five variables, the other five variables serve as noise. The additional noise term ϵ is normally distributed, $\mathcal{N}(0, 1)$. We generate data sets uniformly distributed in Ω with 90,000 data points for training and 10,000 each for validation and testing.

As we are dealing with large data sets, the choice of the regularization parameter λ is uncritical, and the focus can be directly put on minimizing the MSE to reduce the number of grid points further. We therefore modify our criterion for adaptive refinement to target the contribution of a basis function to the squared error. At each point $\vec{x}_i$ of the training data, the local error can be computed as

$$e_i = y_i - f_N(\vec{x}_i) = y_i - \sum_{j=1}^{N} \varphi_j(\vec{x}_i)\alpha_j\,. \tag{6}$$

The contribution of a certain basis function φ_j to $f_N(\vec{x}_i)$ and thus to e_i depends both on its function value $\varphi_j(\vec{x}_i)$ and its coefficient α_j. To preserve a feasible computational

complexity, we presume

$$|\varphi_j(\vec{x}_i)\alpha_j e_i^2| \tag{7}$$

as a simplified measure for φ_j's share in the squared local error e_i^2. Accumulating them for all $\vec{x}_i$ results in the total contribution

$$c_j := \sum_{i=1}^{m} \left|\varphi_j(\vec{x}_i)\alpha_j e_i^2\right| \tag{8}$$

of basis function φ_j. Note that already the mere surplus-based criterion works very well.

Slight additional improvements can be made in terms of the number of grid points that are required, by selecting a suitable refinement strategy, identifying and restricting to the relevant ANOVA terms, and choosing the right basis functions (piecewise polynomials or B-splines); see [21]. Using a good choice, we obtain excellent results, outperforming other techniques [12] and just being matched by the dimensionally adaptive combination technique [13], see Tab. 1.

	sparse grids					
data set	regular	adaptive	opticom	opticom-dim-adapt	SVM	MARS
Friedman1	0.990	0.976	1.340	1.035	1.148	1.205

Table 1: MSE for the Friedman1 data set for different approaches.

Of course, we are dealing with an idealistic setting here: the points to train on are uniformly distributed on the whole domain, and we have plenty of information about the underlying function due to the number of training data. Regular and adaptive sparse grids match each other, apart from the number of grid points (and thus the computational effort that has to be spent).

Furthermore, this example nicely shows that spatially adaptive refinement provides dimensional adaptivity for free. Let $P_{i,j} : [0,1]^d \to [0,1]^2$, $(x_1, \ldots, x_d) \mapsto (x_i, x_j)$ denote the projections of grid points onto the coordinate planes. It can be clearly seen in Fig. 5 that most grid points are spent in the joint dimensions x_1 and x_2 as they correlate most. In contrast, no grid points but those on the second level are wasted in the irrelevant dimensions $x_5, \ldots, x_{10}$: the grid points on the second level are required to probe in these dimensions.

5.0.3 Strategies for Adaptive Refinement

Our third example demonstrates different strategies for refinement. It is a real-world data set from astrophysics, targeting the estimation of the redshift of galaxies. The cosmological redshift of a galaxy is related to its distance from earth, see [21] for details. Whereas spectroscopic measurements provide accurate data but are difficult to obtain, estimates based on photometric data are cheap but usually less accurate, leading to a

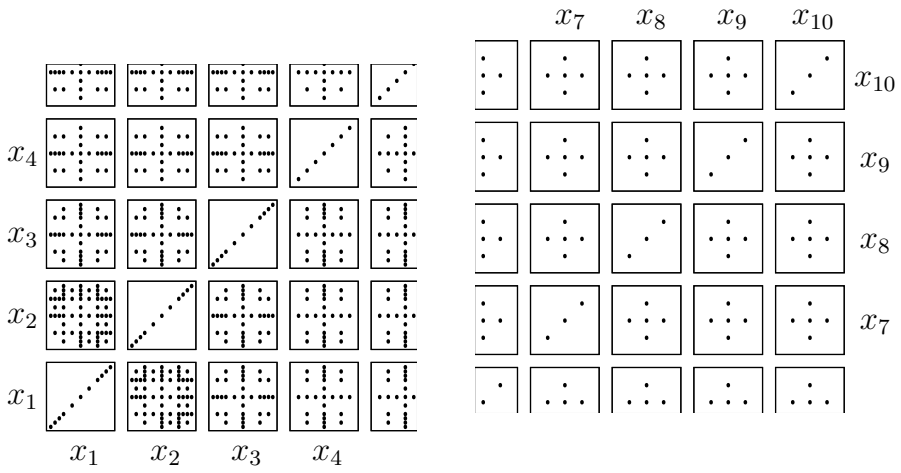

Figure 5: Grid projections for the first and last four dimensions each. Most correlation can be observed for x_1 and x_2, whereas no additional grid points are spent in the irrelevant dimensions.

higher amount of uncertainty in the data collected. This raises the question whether it is possible to estimate the redshift of galaxies using photometric measurements, and how well one can generalize from known data (both photometric and spectroscopic).

The Sloan Digital Sky Survey data base in its release 5 [1] provides data for more than 430,000 galaxies, out of which we take 60,000 for testing. This time, the data is not uniformly distributed, but rather on the joint diagonal of the (normalized) domain. Therefore, the use of a suitable strategy of how many grid points are to be refined per refinement step is of importance.

This results in different numbers of grid points per refinement step, see Fig. 6. Refining a certain ratio of (refinable) grid points leads to an exponential increase in the grid size. On the other hand, refining a low, fixed number of grid points per step results in almost a linear increase in the number of grid points. Note that we employ the same surplus criterion for adaptive refinement as before.

The different strategies effect the behavior of the error. To examine this, we train for different choices on a subset of 60,000 data points. Figure 7 shows the MSE on the test data. It can be seen that the choice of the amount of grid points to refine per step significantly impacts the performance. Refining too many grid points at once, the problem cannot be explored well enough before overfitting starts. The four strategies which are refining a certain percentage of grid points at once exhibit the highest minimal error rates. On the other hand, refining too few grid points per step results in an adaptivity which is too greedy. The best results are achieved refining constantly 100 or 200 grid points per refinement step.

The choice of a good trade-off between greedy and broad refinement depends, of course, on the data to train on. Working on huge amounts of data this can be determined, as

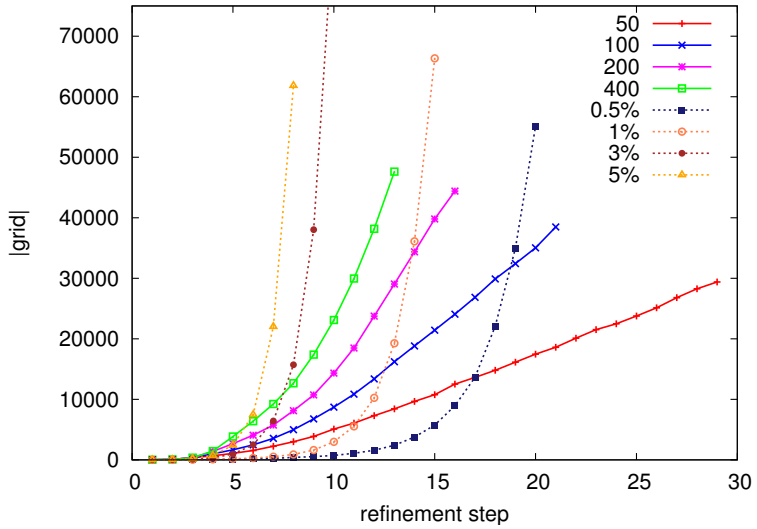

Figure 6: The growth of the number of grid points for different strategies of how many grid points are to be refined per refinement step.

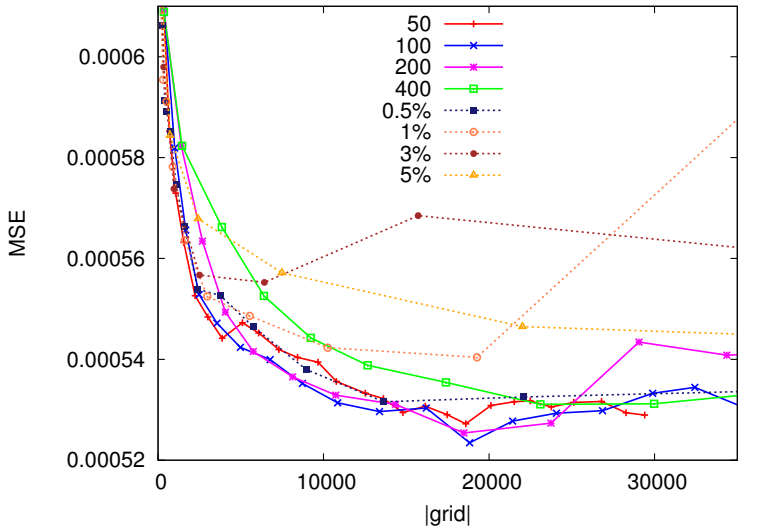

Figure 7: MSE on the test data for different strategies of how many grid points are to be refined per refinement step. It can be seen that obtaining a low error depends on a suitable choice.

Method	σ_{rms}
CWW [8]	0.0666
Bruzual-Charlot [8]	0.0552
Interpolated spectra [8]	0.0451
1^{rst}-nearest neighbors [8]	0.0365
ClassX [24]	0.0340
Polynomial fit [8]	0.0318
SVMs [26]	0.027
Kd-tree [8]	0.0254
Adaptive sparse grids	0.0220

Table 2: Comparison of the root mean square error for photometric redshift estimation, obtained from several studies on different versions of the SDSS data set.

in the example above, for a smaller validation subset. If we train on the whole data set, other methods can be outperformed, see Tab. 2 for the root mean square error (RMSE). This is mainly due to the fact that conventional, data-centered methods have to restrict themselves to smaller subsets of the data. Note that the whole data set can really be dealt with in reasonable time if the whole power of current commodity computers is exploited; see [17] for the efficient parallelization on a hybrid system with both multi cores and GPUs.

5.0.4 Choice of Basis Functions

The next example requires a special choice of basis functions. The task is to determine the price of an early exercise option on d stocks S_i. The holder has to determine at any early exercise time whether to exercise the option or not. He or she will exercise if the current payoff is higher than what can be expected if further holding onto the option, so if

$$P(\vec{S}(t_i), t_i) \geq \mathbb{E}\left[V(\vec{S}(t_{i+1}), t_{i+1}) \,|\, \vec{S}(t_i), t_i\right] ,.$$

Typically, nested Monte Carlo (MC) simulations are performed to simulate the option and to determine the expectation value for each simulation and each time step. Fortunately, the computationally expensive nesting can be avoided by a least squares Monte Carlo simulation [7], obtaining the expectation values for all simulations at each time-step by solving a regression problem; see [21] for details.

This setting is challenging for sparse grids as the data, obtained by MC simulations, is noisy, clustered, and only little information is available towards the domain's boundary, see Fig. 8 for a two-dimensional example. This does not fit well with the local support of the basis functions, leading to wiggles at the boundary. Furthermore, it is desirable in the context of option pricing to obtain functions which are continuously differentiable and which have zero curvature (second derivatives) at the boundary.

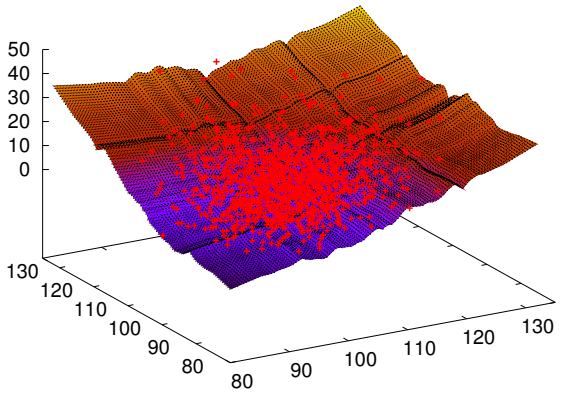

Figure 8: Data set obtained by MC simulations for a two-dimensional basket option.

This task can successfully be solved using modified basis functions which adapt to the problem very well. Here, B-splines of a certain degree p are employed, see Fig. 9 (left). They are then modified towards the boundary (right) to avoid to spend unnecessary degrees of freedom where little or no information is available. Additionally, the boundary modification ensures zero curvature at the boundary. Furthermore, using B-splines of degree p guarantees $p-1$ times continuously differentiable functions. The disadvantage of this choice is that basis functions within the same level overlap. Thus, sparse grid algorithms such as function evaluations become much more expensive.

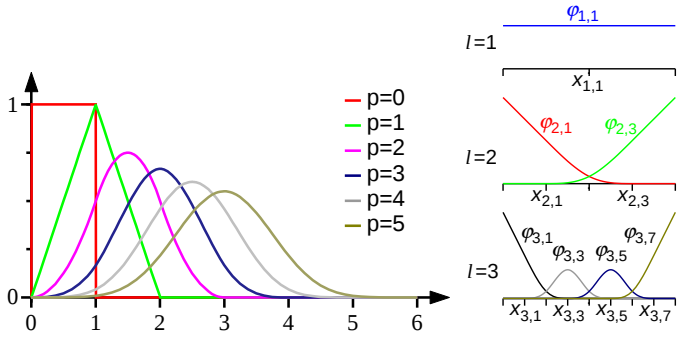

Figure 9: B-spline construction (left) and modified B-splines of degree $p = 3$ (right).

Figure 10 shows the regressed function for a basket of two stocks using a sparse grid of level 6 and B-spline basis functions of degree 3 and 7. Whereas for $p = 3$ still some artifacts can be observed, the function for $p = 7$ is nearly symmetric (the parameters for both stocks S_1 and S_2 are the same) and expresses the training data very well. Note that it shows a reasonable behavior even in regions where there are no training data points at all, due to the less local support of the basis functions. For high degrees, only few typical sparse grid artifacts can be observed. In contrast, often narrow bumps or

dips occur for small degrees, which are caused by long basis functions.

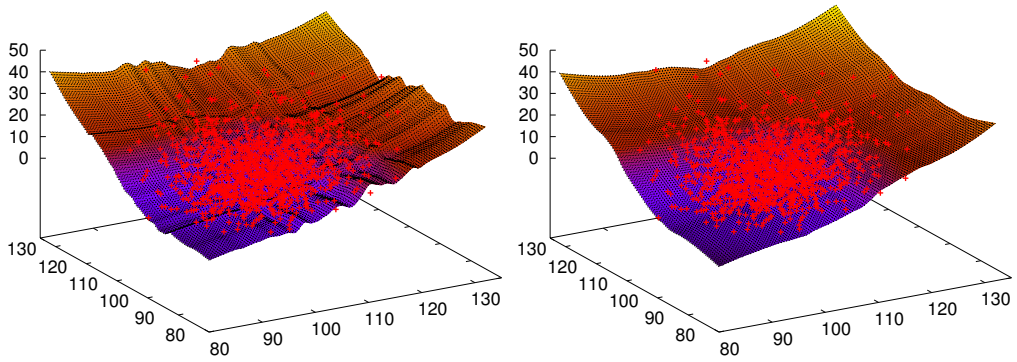

Figure 10: Regressed function for B-splines of degree 3 (left) and 7 (right).

If the dynamics of the underlying stocks do not exhibit too high variance, then excellent numerical results can be obtained for up to 8-dimensional options; see [21] for exact results and comparisons. Note that the choice of the basis fulfills additionally the application's requirements to have zero curvature at the boundary and several times continuously differential functions.

5.0.5 Weighted/Guided Refinement

Finally, we point out how one might guide refinement, depending on the requirements of the application. We take an example from plasma physics, where sparse grids come into play to speed up parameter scans. Several parameters of a gyrokinetic model are to be optimized; consider the minimization of the overall energy to run a fusion plant where the energy depends on the choice of parameters, as an example. As a side-constraint, we require the particle transport in the fusion plant to be zero,

$$\Gamma(\omega_n^{\text{ion}}, \omega_t^{\text{elec.}}, \omega_t^{\text{ion}}, \text{temp}^{\text{elec.}}, \ldots) = 0\,.$$

The optimization can be costly if an iterative method requires to evaluate the function Γ multiple times to approximate gradients, each function evaluation requiring a whole simulation run. The idea is thus to compute a surrogate of Γ, a multi-dimensional sparse grid interpolant, which can then replace the simulation code during optimization. The construction of the surrogate can be done well in advance in an offline stage and thus consume a higher computational effort than a single run of the optimization algorithm. For illustration purposes, we restrict ourselves in the following to the two-dimensional setting.

The function itself is rather smooth, with one continuous kink starting in the middle of the domain and extending to the boundary in one direction, see Fig. 11 (left). Measuring the MSE and maximum error on 100,000 uniformly distributed points in the scaled domain, we can observe that there is almost no difference between full grids and regular

sparse grids, Fig. 11 (right). Here, we have been employing sparse grids with grid points on the boundary.

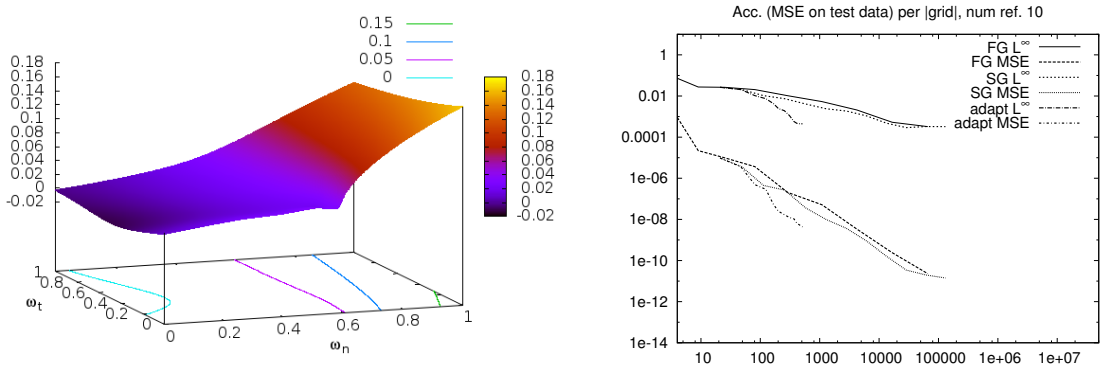

Figure 11: The underlying function (left), and both the MSE on the test data and the maximum error measured for full grids, regular sparse grids and the standard refinement (right).

If we additionally employ surplus-based adaptive refinement, we obtain a significant improvement, and the error is converging faster (same figure for the first 500 grid points). This is already better, but not good enough. The grid in Fig. 12 (left) shows that adaptivity spends most grid points towards the kink. Reminding that the optimization problem aims for $\Gamma = 0$, it is apparent that it is not the best strategy to aim for a low error in the whole domain. Restricting our data set to $|\Gamma| \leq \epsilon$, $\epsilon = 0.001$, it can be observed that the region of interest is in another part of the domain, see Fig. 12 (right). The refinement should thus be guided to express the region around $\Gamma = 0$.

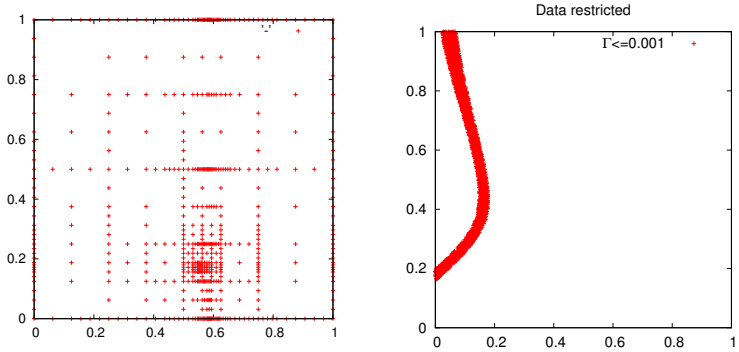

Figure 12: The sparse grid with standard refinement (left), and the region of interest around $\Gamma = 0$ (right).

This can be achieved by weighting the criterion for adaptive refinement. Rather than a mere surplus-based refinement, we take

$$|\alpha_{\vec{l},\vec{i}}| \exp\left(-cf(\vec{x}_{\vec{l},\vec{i}})\right)$$

into consideration. This puts less emphasis on grid points with a function value far away from $\Gamma = 0$. Figure 13 shows that the resulting grid is guided to the region of interest. Furthermore, measuring the error only in the region $|\Gamma| \leq \epsilon$, several orders of magnitude in convergence can be gained compared to the previous, straightforward approach. Needing only around 5000 grid points, the machine accuracy is reached. Note that the function is only coarsely interpolated far away from $\Gamma = 0$, but definitely well enough so that optimization algorithms can get quickly to the region of interest.

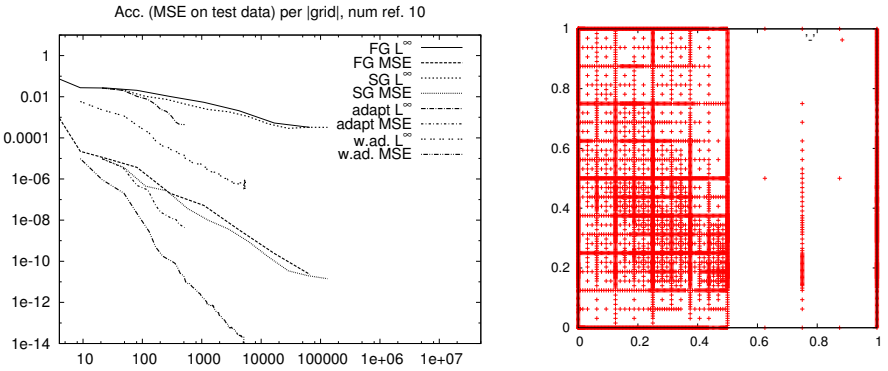

Figure 13: The MSE on the test data, extended by the weighted adaptivity (left), and the corresponding sparse grid (right).

We have successfully applied a similar guided refinement in computational finance to compute the PDE solution of the Black-Scholes equation in option pricing. There, the main focus lies on a a good solution at a certain point in the d-dimensional space. Thus, the surplusses are multiplied with a Gaussian weight depending on the distance to the point of interest. As a diffusive part leads to increased smoothness in time, additionally a coarsening step is employed after each time step, removing all grid points with a surplus lower than a certain threshold, see [5] for further details.

6 Conclusions

Whereas sparse grids already enable to deal with higher-dimensional settings than possible for classical mesh-based methods, adaptive refinement is often crucial. It allows to tackle problems that do not meet the smoothness requirements of the sparse grid approach and problems that require to spend as few grid points as possible. This requires to adapt the standard approach to the problem at hand.

We have shown criteria and strategies for adaptive refinement that have proved to be useful in several occasions. They have been illustrated for different problems, both real-world and artificial ones.

Whereas the hierarchical surplus of a grid point can be used as a cheap, free and effective criterion for adaptive refinement, often more considerations have to be spent to reduce the number of grid points even further, e.g.:

- The choice of basis functions has to be adapted in high dimensionalities or where the problem imposes requirements at the sparse grid function itself. Especially the boundary treatment plays an important role.
- The number or percentage of grid points that are to be refined at once determines whether a greedy or a broad refinement occurs. This is especially critical in settings where noise requires not to spend too many grid points in the wrong place.
- If knowledge about the problem is available, weighted (or guided) refinement can significantly improve the results. This allows to encourage refinement in regions of interest. Similarly, refinement can be restricted to important ANOVA components, and coarsening can be employed to get rid of superfluous grid points.

Finally it has to be noted that counter examples can be found for each and every criterion for adaptive refinement. While there is no one-size-fits-all strategy, the simple surplus-based approach frequently gives a good start and can then be adapted to the task at hand. In any case, knowledge about the problem should be incorporated into the criteria and strategies for adaptivity wherever possible.

References

[1] J. K. Adelman-McCarthy et al. The fifth data release of the Sloan Digital Sky Survey. *ApJS*, 172:634–644, 2007.

[2] D. M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1):125–127, 1974.

[3] J. Benk, H.-J. Bungartz, A.-E. Nagy, and S. Schraufstetter. An option pricing framework based on theta-calculus and sparse grids. In *Progress in Industrial Mathematics at ECMI 2010*, 2010.

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

[5] H.-J. Bungartz, A. Heinecke, D. Pflüger, and S. Schraufstetter. Option pricing with a direct adaptive sparse grid approach. *Journal of Computational and Applied Mathematics*, 2011.

[6] H.-J. Bungartz, D. Pflüger, and S. Zimmer. Adaptive sparse grid techniques for data mining. In H.G. Bock, E. Kostina, X.P. Hoang, and R. Rannacher, editors, *Modelling, Simulation and Optimization of Complex Processes, Proceedings of the High Performance Scientific Computing 2006, Hanoi, Vietnam*, pages 121–130. Springer, 2008.

[7] J. F. Carrière. Valuation of early-exercise price of options using simulation and nonparametric regression. *Insurance: Mathematics and Economics*, 19:19–30, 1996.

[8] I. Csabai et al. The application of photometric redshifts to the SDSS Early Data Release. *Astron. J.*, 125:580–592, 2003.

[9] T. Evgeniou, M. Pontil, and T. Poggio. Regularization networks and support vector machines. In *Advances in Computational Mathematics*, pages 1–50. MIT Press, 2000.

[10] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19, 1991.

[11] B. Ganapathysubramanian and N. Zabaras. Sparse grid collocation schemes for stochastic natural convection problems. *J. Comput. Phys.*, 225(1):652–685, 2007.

[12] J. Garcke. Regression with the optimised combination technique. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 321–328, New York, NY, USA, 2006. ACM Press.

[13] J. Garcke. A dimension adaptive sparse grid combination technique for machine learning. In Wayne Read, Jay 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.

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

[15] J. Garcke and M. Hegland. Fitting multidimensional data using gradient penalties and the sparse grid combination technique. *Computing*, 84(1-2):1–25, 2009.

[16] M. Hegland. Adaptive sparse grids. In K. Burrage and Roger B. Sidje, editors, *Proc. of 10th Computational Techniques and Applications Conference CTAC-2001*, volume 44, pages C335–C353, 2003.

[17] A. Heinecke and D. Pflüger. Multi- and many-core data mining with adaptive sparse grids. In *Proceedings of the 8th ACM International Conference on Computing Frontiers*, pages 29:1–29:10, New York, USA, May 2011. ACM Press.

[18] M. Holtz. *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance*. Dissertation, Institut für Numerische Simulation, Universität Bonn, 2008.

[19] A. Klimke, R. Nunes, and B. Wohlmuth. Fuzzy arithmetic based on dimension-adaptive sparse grids: a case study of a large-scale finite element model under uncertain parameters. *Internat. J. Uncertain. Fuzziness Knowledge-Based Systems*, 14:561–577, 2006.

[20] D. Pflüger. Data Mining mit Dünnen Gittern. Diplomarbeit, IPVS, Universität Stuttgart, 2005.

[21] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. Verlag Dr. Hut, München, 2010.

[22] C. Reisinger and G. Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. *SIAM J. Scientific Computing*, 29(1):440–458, 2007.

[23] B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1995.

[24] A. A. Suchkov, R. J. Hanisch, and B. Margon. A census of object types and redshift estimates in the SDSS photometric catalog from a trained decision-tree classifier. *Astron. J.*, 130:2439–2452, 2005.

[25] T. von Petersdorff and C. Schwab. Sparse finite element methods for operator equations with stochastic data. *Appl. Math.*, 51(2):145–180, 2006.

[26] Y. Wadadekar. Estimating photometric redshifts using support vector machines. *Publications of the Astronomical Society of the Pacific*, 117:79, 2005.

[27] G. Widmer, R. Hiptmair, and C. Schwab. Sparse adaptive finite elements for radiative transfer. *Journal of Computational Physics*, 227:6071–6105, 2008.

[28] C. Zenger. Sparse grids. In Wolfgang Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations*, volume 31 of *Notes on Numerical Fluid Mechanics*, pages 241–251. Vieweg, 1991.