Using Adaptive Sparse Grids to Solve High-Dimensional Dynamic Models*

Johannes Brumm
DBF, University of Zurich
johannes.brumm@uzh.ch

Simon Scheidegger
DBF, University of Zurich
simon.scheidegger@uzh.ch

November 3, 2013

Abstract

We present a flexible and scalable method to compute global solutions of high-dimensional non-smooth dynamic models. Within a time-iteration setup, we interpolate policy functions using an adaptive sparse grid algorithm with piecewise multi-linear (hierarchical) basis functions. As the dimensionality increases, sparse grids grow considerably slower than standard tensor product grids. In addition, the grid scheme we use is automatically refined locally and can thus capture steep gradients or even non-differentiabilities. To further increase the maximal problem size we can handle, our implementation is fully hybrid parallel, i.e. using a combination of MPI and OpenMP. This parallelization enables us to efficiently use modern high-performance computing architectures. Our time iteration algorithm scales up nicely to more than one thousand parallel processes. To demonstrate the performance of our method, we apply it to high-dimensional international real business cycle models with capital adjustment costs and irreversible investment.

Keywords: Adaptive Sparse Grids, High-Performance Computing, International Real Business Cycles, Non-Differentiabilities, Occasionally Binding Constraints, Parallel Computing

JEL Classification: C63, C68, F41

*We are very grateful to Felix Kübler for helpful discussions and support. We thank seminar participants at University of Chicago, Stanford University, University of Zürich, and the CEF 2013 in Vancouver. Moreover, we thank Xiang Ma for very instructive email discussions regarding the workings of adaptive sparse grids. We are grateful for the support of Olaf Schenk, Antonio Messina and Riccardo Murri concerning HPC related issues. We acknowledge CPU time granted on the University of Zürich's 'Schrödinger' HPC cluster. Johannes Brumm gratefully acknowledges financial support from the ERC.

1 Introduction

There are many important economic phenomena that cannot be captured by models if they do not account for interactions between different firms, sectors, and countries, or if they only consider local dynamics around steady states. At the latest, this has become obvious through the recent financial crisis with its tremendous spillover effects and large price fluctuations. Yet already in the nineties, more and more economists included various kinds of heterogeneity in their models and also started to use global solution techniques (see, e.g. [38] or [24]). However, solving for the global solution of a model that includes substantial heterogeneity is very costly: Using conventional solution methods, the computation time and storage requirements increase exponentially with the amount of heterogeneity, i.e. the dimensionality of the problem.

This paper makes an effort to shift the limits of how much heterogeneity we can assume in an economic model and still be able to compute an accurate global solution in a reasonable amount of time. We achieve this by employing a highly parallel implementation of a so-called adaptive sparse grid method [25] within a time iteration framework. This method can handle high-dimensional problems even if they exhibit non-smooth behavior like non-differentiable policy functions.

Standard algorithms that are used to compute global solutions of economic models rely on a grid-based numerical representation of a multivariate policy function (see, [19]). However, starting with a one-dimensional discretization scheme that employs N grid points, a straightforward extension to d dimensions leads to N^d grid points. Sparse grids are able to alleviate this so-called 'curse of dimensionality' by reducing the number of grid points from $\mathcal{O}\left(N^d\right)$ to $\mathcal{O}\left(N \cdot (\log N)^{d-1}\right)$ with only slightly deteriorated accuracy if the underlying function is sufficiently smooth (see, e.g. [5], with references therein).

The sparse grid construction we are using was introduced by Zenger [40] for the solution of partial differential equations (PDEs). However, the underlying principle, a sparse tensor product decomposition, goes back to the seminal work of Smolyak [37]. Sparse grids have been applied to a whole range of different research fields such as physics, visualization, and finance (see, e.g. [10, 5, 14, 27]). Using the original formulation of Smolyak [37], Krüger and Kübler [23] were the first to solve dynamic economic models using sparse grids. Recently, Judd et al. [18] propose an implementation that is more efficient and also allows for grids that are ex ante chosen to be finer in some dimensions than in others. However, these two papers rely on global polynomials as basis functions, which fail to capture the local behavior of policy functions that are not sufficiently smooth. In contrast, our algorithm can capture non-smooth behaviour. The reason is that we use hierarchical basis functions with an adaptive grid refinement strategy. The basis functions we use are hat functions, which are piecewise multi-linear with local support. The space of basis functions is hierarchically structured into interpolation levels. For a basis function at a given level L, there are several basis functions of the next finer level $L+1$ among which the support of the original function of level L is subdivided. Using the value of a level L function, an automatic grid adaptation strategy decides whether the interpolation is refined locally by adding the associated $L+1$ functions. Importantly, this refinement scheme scales just linearly with increasing dimension (see, e.g. [25, 31, 29]). Such an adaptive sparse grid with hierarchical local basis functions offers the

promise of an efficient and accurate solution of economic problems that are both high-dimensional and non-smooth.

However, the latter class of problems requires substantial computation time even if an efficient solution method is applied. Therefore, our implementation aims to access high-performance computing (HPC) facilities. Their mainstream hardware design nowadays consists of shared memory nodes with several multi-core CPUs that are connected via a network structure. Hence, efficient parallel programming must make use of multiple computational units by combining distributed memory parallelization on the node interconnect with shared memory parallelization inside each node (see, e.g. [32]). We address this challenge by an implementation that is 'hybrid' parallel, i.e. using MPI ('Message Passing Interface'; cf. [36]) between nodes and OpenMP (shared memory parallelism; cf. [17]) within the nodes. In the hybrid MPI/OpenMP mode, we are able to use efficiently at least 1,200 cores.

To show that our algorithm can solve standard high-dimensional economic problems, we solve the international real business cycle (IRBC) model with adjustment costs. For this application the performance of alternative algorithms is well documented in a study comparing various solution methods (see, [22]). Like many of these established methods, we use a time iteration procedure (see, e.g. [19]) to solve for an equilibrium that is Markov in the capital stock and the productivity levels of all countries. The innovation of our approach lies within each time iteration step, where we use an (adaptive) sparse grid to interpolate the policy functions. As the policy functions in this model are very smooth, our linear interpolation scheme has a disadvantage compared to smooth interpolation schemes. Nevertheless, we can compute quite accurate solutions for models that are of higher dimension than any that have been reported in the comparison study by [22]. However, the purpose and comparative advantage of our algorithm lies in solving models that exhibit non-smooth behavior. To demonstrate its performance with respect to such models, we augment the IRBC model with irreversible investment. In spite of the non-differentiabilities (also called 'kinks') induced by this assumption we are still able to compute accurate solutions for high-dimensional examples. The adaptivity of the grid now ensures that we can capture the kinks fairly well without increasing the number of gridpoints too much.

The main contribution of this paper is as follows: We apply for the first time an adaptive sparse grid algorithm to solve large-scale economic models. Using modern high-performance computing facilities, we are able to compute accurate global solutions for models with more than 20 dimensions, and also to high-dimensional models with kinks.

In addition to the above discussed literature on sparse grids, both in mathematics and economics, our paper is also closely related to two other stands of the literature. First, to papers that develop methods for solving dynamic economic models with occasionally binding constraints, e.g. [15, 2, 9]. While these methods are able to match the non-differentiabilities induced by such constraints very precisely, they are not as flexible and scalable as our method, which is therefore superior when it comes to problems with more than three or four continuous state variables. Second, our paper is part of the emergent literature on parallel computing applications in economics (see, e.g. [6]). To the best of our knowledge, we are the first to efficiently use current high-performance computing technology to solve dynamic economic models. We are able to do so

as our implementation is fully hybrid parallel.

The remainder of the paper is organized as follows. In Sec. 2, we explain the construction of adaptive sparse grids and also provide simple test cases for their use in interpolation. In Sec. 3, we embed adaptive sparse grid interpolation in a time iteration algorithm to solve high-dimensional IRBC models, also with irreversible investment. We discuss the performance of this algorithm and report how hybrid parallelization can speed up the computations. Sec. 4 concludes.

2 From Full Grids to Adaptive Sparse Grids

In this section, we first provide a brief introduction to 'classical', i.e. non-adaptive, sparse grid interpolation. In contrast to the sparse grid interpolation schemes used so far in economics (see, e.g. [23, 18]), we follow a different approach and use hierarchical basis functions (see [5, 10], with references therein). We then proceed to adaptive sparse grids (see, e.g. [29, 30]). The latter use the hierarchical structure of the grid and of the associated basis functions to refine the sparse grid such that it can capture the local behavior of the interpolant. Based on examples, we show why adaptive sparse grids are superior in interpolating functions that exhibit steep gradients or kinks.

2.1 Notation

We first introduce some notation and definitions that we will require later on [5, 10]. For all our considerations, we will focus on the domain $\Omega = [0,1]^d$, where d is the dimensionality of the problem. This situation can be achieved for other domains by a proper rescaling. Furthermore, let $\vec{l} = (l_1, ..., l_d) \in \mathbb{N}^d$ and $\vec{i} = (i_1, ..., i_d) \in \mathbb{N}^d$ denote multi-indices representing the grid refinement level as well as the spatial position of a $d-$dimensional grid point $\vec{x}_{\vec{l},\vec{i}}$. We then define the anisotropic, full grid $\Omega_{\vec{l}}$ on Ω with mesh size $h_{\vec{l}} := (h_{l_1}, ..., h_{l_d}) = 2^{-\vec{l}} := \left(2^{-l_1}, ..., 2^{-l_d}\right)$. $\Omega_{\vec{l}}$ has different but equidistant mesh sizes h_{l_t} in each coordinate direction $t = 1, ..., d$. In this way, the grid $\Omega_{\vec{l}}$ consists of the points

$$\vec{x}_{\vec{l},\vec{i}} := (x_{l_1,i_1}, ..., x_{l_d,i_d}), \tag{1}$$

where $x_{l_t,i_t} := i_t \cdot h_{l_t} = i_t \cdot 2^{-l_t}$, and $i_t \in \{0, 1, ..., 2^{l_t}\}$. In addition, when dealing with $d-$dimensional multi-indices such as $\vec{l}$, we use relational operators component-wise,

$$\vec{l} \leq \vec{k} \Leftrightarrow l_t \leq k_t, \forall t \in \{1, \ldots, d\}. \tag{2}$$

We use the l_1-norm $|\vec{l}|_1$,

$$|\vec{l}|_1 := \sum_{t=1}^{d} l_t \tag{3}$$

and the maximum norm $|\vec{l}|_\infty$,

$$|\vec{l}|_\infty := \max_{1 \leq t \leq d} l_t. \tag{4}$$

Finally, we define for $d-$dimensional functions $f : \Omega \to \mathbb{R}$ the L_2-norm

$$\|f\|_{L_2} := \left(\int_\Omega |f(\vec{x})|^2 d\vec{x} \right)^{1/2}. \tag{5}$$

2.2 Hierarchical Basis Functions in One Dimension

We use a sparse grid method that is based on a hierarchical decomposition of the underlying approximation space. Such a hierarchical structure is crucial both for local adaptivity (see Sec. 2.5) and for the use of parallel computing (see Sec. 3.4). We now explain this hierarchical structure starting with the one dimensional case, i.e. $\Omega = [0, 1]$. Afterwards, we will extend it to the multivariate case using tensor products.

The first ingredient of a sparse grid interpolation method is a one-dimensional multilevel basis. Let us assume that a function $f : \Omega \to \mathbb{R}$ of interest is sufficiently smooth [5]. For the time being we also assume for simplicity that the function f vanishes at the boundary, i.e. $f|_{\partial\Omega} = 0$. We delegate the treatment of non-zero boundaries to the Appendix A. An interpolation formula is then given by

$$f(x) \approx u(x) := \sum_i \alpha_i \phi_i(x) \tag{6}$$

with coefficients α_i and a set of appropriate piecewise linear basis functions $\phi_i(\vec{x})$. In the 'classical' sparse grid approach, a hierarchical basis based on standard hat functions,

$$\phi(x) = \begin{cases} 1 - |x| & \text{if } x \in [-1, 1] \\ 0 & \text{else} \end{cases} \tag{7}$$

is used. The standard hat function is then taken to generate a family of basis functions $\phi_{l,i}$ having support $[x_{l,i} - h_l, x_{l,i} + h_l]$ by dilation and translation, i.e.

$$\phi_{l,i}(x) := \phi\left(\frac{x - i \cdot h_l}{h_l}\right). \tag{8}$$

This basis is usually termed *nodal basis* [5]. The basis functions defined in Eq. 8 are used to define the function spaces V_l consisting of piecewise linear functions

$$V_l := \text{span}\{\phi_{l,i} : 1 \leq i \leq 2^l - 1\}. \tag{9}$$

The hierarchical increment spaces W_l are defined by

$$W_l := \text{span}\{\phi_{l,i} : i \in I_l\}, \tag{10}$$

using the index set

$$I_l = \{i \in \mathbb{N}, 1 \leq i \leq 2^l - 1, i \text{ odd}\}. \tag{11}$$

In this way, the hierarchical increment spaces W_l are related to the nodal spaces V_l by the direct sum

$$V_l = \bigoplus_{k \leq l} W_k. \tag{12}$$

Fig. 1 shows the first three levels of these hierarchical, piecewise linear basis functions. Going now back to Eq. 6, we can see that any function $f \in V_l$ can be uniquely represented as

$$f(x) \approx u(x) = \sum_{k=1}^{l} \sum_{i \in I_k} \alpha_{k,i} \phi_{k,i}(x), \tag{13}$$

with the coefficients $\alpha_{k,i} \in \mathbb{R}$. Note that the supports of all basis functions $\phi_{k,i}$ spanning W_k are mutually disjoint, as can be seen in Fig. 1.

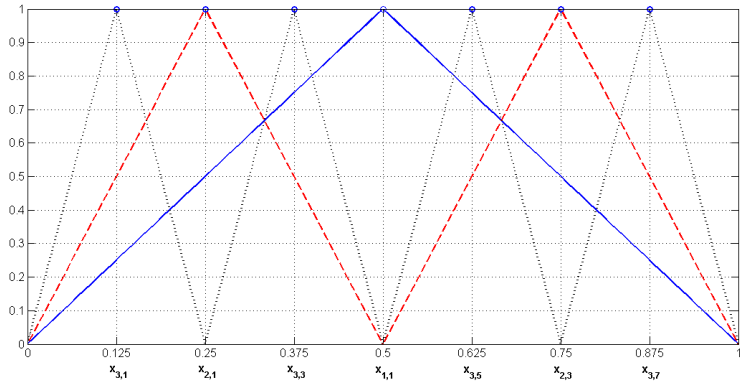

Figure 1: Hierarchical basis functions of level 1 (solid blue), 2 (dashed red) and 3 (dotted black) for V_3.

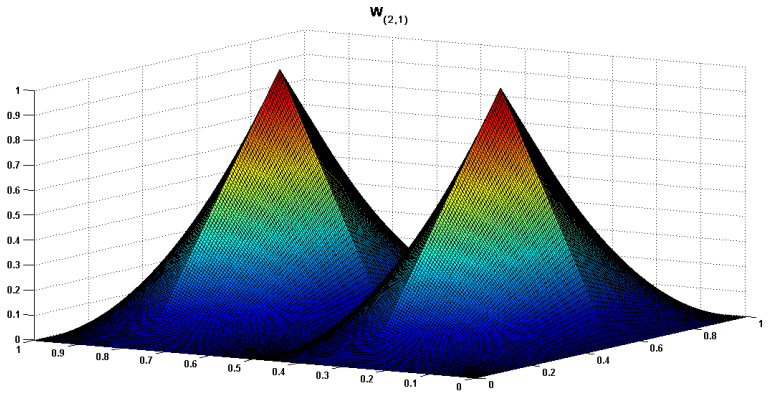

Figure 2: Basis functions on the subspace $W_{2,1}$.

2.3 Hierarchical Basis Functions in Multiple Dimensions

This one dimensional hierarchical basis can now be extended to a $d-$dimensional one on the unit cube $\Omega = [0,1]^d$ by a tensor construction. In line with the above argumentation related to the one-dimensional case, all notations can be transferred to the arbitrary-dimensional case as well.

For each grid point $\vec{x}_{\vec{l},\vec{i}}$, an associated piecewise $d-$linear basis function $\phi_{\vec{l},\vec{i}}(\vec{x})$ is defined as the product of the one-dimensional basis functions (see, Eq. 7)

$$\phi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{t=1}^{d} \phi_{l_t,i_t}(x_t)\,. \tag{14}$$

Note that each of the multidimensional (nodal) basis functions $\phi_{\vec{l},\vec{i}}(\vec{x})$ has support of size $2 \cdot h_{\vec{l}}$. The latter basis functions are again used to define function spaces $V_{\vec{l}}$ consisting of piecewise $d-$linear functions that are zero on the

boundary of Ω:

$$V_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{1} \leq \vec{i} \leq 2^{\vec{l}} - \vec{1}\}. \tag{15}$$

Again, the index set $I_{\vec{l}}$ is given by

$$I_{\vec{l}} := \{\vec{i} : 1 \leq i_t \leq 2^{l_t} - 1, i_t \text{ odd}, 1 \leq t \leq d\}. \tag{16}$$

Note that the hierarchical increments, formally defined as

$$W_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}, \tag{17}$$

can alternatively be written as

$$W_{\vec{l}} := V_{\vec{l}} \setminus \bigoplus_{t=1}^{d} V_{\vec{l}-\vec{e_t}}, \tag{18}$$

where $\vec{e_t}$ is the $t-$th unit vector.

In other words, $W_{\vec{l}}$ consist of all $\phi_{\vec{l},\vec{i}} \in V_{\vec{l}}$ (using the hierarchical basis functions) which are not included in any of the spaces $V_{\vec{k}}$ smaller than $V_{\vec{l}}$ [1]. An example of such a function space is given if Fig. 2. These hierarchical difference spaces now allow us the definition of a multilevel space decomposition. In line with the sparse grid literature (see, e.g. [29, 10, 5]), we define $V_n := V_{\vec{n}}$ as a direct sum of spaces. Consequently, the hierarchical increment spaces $W_{\vec{l}}$ are related to the nodal spaces $V_{\vec{l}}$ of piecewise $d-$linear functions with mesh width h_l in each dimension by

$$V_n := \bigoplus_{l_1=1}^{n} \cdots \bigoplus_{l_d=1}^{n} W_{\vec{l}} = \bigoplus_{|l|_\infty \leq n} W_{\vec{l}}, \tag{19}$$

leading to a full grid with $(2^n - 1)^d$ grid points. The interpolant of f, namely $u(\vec{x}) \in V_n$ can uniquely be represented by

$$f(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_\infty \leq n} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|l|_\infty \leq n} f_{\vec{l}}(\vec{x}), \tag{20}$$

with $f_l \in W_{\vec{l}}$ and $\alpha_{\vec{l},\vec{i}} \in \mathbb{R}$.

For sufficiently smooth f and its interpolant $u \in V_n$ [5], we obtain an asymptotic error decay of

$$\| f(\vec{x}) - u(\vec{x}) \|_{L_2} \in \mathcal{O}\left(h_n^2\right), \tag{21}$$

but at the cost of

$$\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right) \tag{22}$$

function evaluations, encountering the so-called *curse of dimensionality*. The latter, i.e. the exponential dependence of the overall computational effort on the number of dimensions is a prohibitive obstacle for the numerical treatment of high-dimensional problems. The number of dimensions typically does not allow to handle more than four to five dimensional problems with reasonable accuracy. For example a resolution of 15 points in each dimension, i.e. $n = 4$, for a ten-dimensional problem therefore needs $0.58 \cdot 10^{12}$ coefficients, which brings us already to the capacity limits of today's most advanced computer systems [8].

[1] Note that a function space $V_{\vec{k}}$ is called 'smaller' than a space $V_{\vec{l}}$ if $\forall k_t \leq l_t$ and $\exists t : k_t < l_t$. Similarly, a grid $\Omega_{\vec{k}}$ is smaller than $\Omega_{\vec{l}}$ [5, 10].

2.4 Classical Sparse Grids

As a consequence of Sec. 2.3, the question that needs to be answered is how we can construct discrete approximation spaces that are better than V_n in the sense that the same number of invested grid points leads to a higher order of accuracy [40, 5]. The 'classical' sparse grid construction arises from a cost to benefit analysis (see, e.g., [40, 10, 5], with references therein) in function approximation. Thereby, functions $f(\vec{x}) : \Omega \to \mathbb{R}$ which have bounded mixed derivatives,

$$D^{\vec{l}} f := \frac{\partial^{|\vec{l}|_1}}{\partial x_1^{l_1} \cdots \partial x_d^{l_d}} f \tag{23}$$

for $|l|_\infty \leq 2$ are considered. These functions belong to a so-called Sobolev space $H_2^{\text{mix}}(\Omega)$ with

$$H_2^{\text{mix}}(\Omega) := \{f : \Omega \to \mathbb{R} : D^{\vec{l}} f \in L_2(\Omega), |\vec{l}|_\infty \leq 2, f|_{\partial\Omega} = 0\}. \tag{24}$$

Under this prerequisite, the hierarchical coefficients $\alpha_{\vec{l},\vec{i}}$ (see, Eq. 20 and [5]) rapidly decay for functions $f \in H_2^{\text{mix}}$, namely as

$$|\alpha_{\vec{l},\vec{i}}| = \mathcal{O}\left(2^{-2|\vec{l}|_1}\right). \tag{25}$$

The strategy of constructing a sparse grid is now to leave out those subspaces from the full grid space V_n that only contribute little to the overall interpolant [5]. An optimization with respect to the number of degrees of freedom, i.e. the grid points, and the resulting approximation accuracy directly leads to the *sparse grid* space $V_{0,n}^S$ of level n, defined by

$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}, \tag{26}$$

where the index 0 in $V_{0,n}^S$ implies $f|_{\partial\Omega} = 0$. Note that the concrete choice of subspaces depends on the norm in which we measure the error. The result obtained in Eq. 26 is optimal for the L_2-norm and the $L_\infty-$norm [5].

The dimension of the space $V_{0,n}^S$, i.e. the number of grid points required is now given by [5, 10]

$$|V_{0,n}^S| = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}\left(n^{d-2}\right)\right) = \mathcal{O}\left(h_n^{-1} \cdot \left(\log(h_n^{-1})\right)^{d-1}\right). \tag{27}$$

This shows the order $\mathcal{O}\left(2^n \cdot n^{d-1}\right)$, which is a significant reduction of the number of grid points and, in consequence, of the computational and storage requirements compared to $\mathcal{O}\left(2^{nd}\right)$ given in Eq. 22 of the full grid space $|V_n|$ (see, Tabs. 1 and 7). In analogy to Eq. 20, a function $f \in V_{0,n}^S \subset V_n$ can now be expanded by

$$f_{0,n}^S(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|l|_1 \leq n+d-1} f_{\vec{l}}(\vec{x}), \tag{28}$$

where $f_l \in W_{\vec{l}}$. Note that $\alpha_{\vec{l},\vec{i}} \in \mathbb{R}$ are commonly termed the *hierarchical surpluses* [40, 5]. They are simply the difference between the function values

d	$\lvert V_n \rvert$	$\lvert V_{0,n}^S \rvert$
1	15	15
2	225	49
3	3375	111
4	50'625	209
5	759'375	351
10	$5.77 \cdot 10^{11}$	2'001
15	$4.37 \cdot 10^{17}$	5'951
20	$3.33 \cdot 10^{23}$	13'201
30	$1.92 \cdot 10^{35}$	41'601
40	$1.11 \cdot 10^{47}$	95'201
50	$6.38 \cdot 10^{58}$	182'001
100	>Googol	1'394'001

Table 1: Number of grid points for several different types of grids of level 4. Left column: dimension; middle column: full grid; right column: classical, L_2 optimal sparse grid with no points at the boundaries.

at the current and the previous interpolation levels (see, Fig. 3). The elegant point about Eq. 28 is that it allows to utilize the previous results generated to improve the interpolation. As we have chosen our set of grid points to be nested, i.e. in a way the set of points X^{l-1} at level $l-1$ with support nodes $\vec{x}_{\vec{l},\vec{i}}$ is contained in X^l, namely $X^{l-1} \subset X^l$, the extension of the interpolation level from level $l-1$ to l only requires to evaluate the function at grid points that are unique to X^l, that is, at $X_\Delta^l = X^l \backslash X^{l-1}$.

The asymptotic accuracy of the interpolant deteriorates only slightly from $\mathcal{O}\left(h_n^2\right)$ in case of the full grid (cf. Eq. 21) down to

$$\mathcal{O}\left(h_n^2 \cdot \log(h_n^{-1})^{d-1})\right), \tag{29}$$

as shown e.g. in [5, 10]. Eqs. 27 and 29 condense the essence why sparse grids are especially well-suited for high-dimensional problems. Note that Eq. 29 also holds for sparse grids with non-vanishing boundaries, i.e. $f|_{\partial\Omega} \neq 0$ [5]. For sparse grid constructions with non-zero boundaries, we point the reader to Appendix A.

The coefficients $\alpha_{\vec{l},\vec{i}}$ in the interpolant of the function f (see, Eq. 28) can quite easily be determined due a nice property of the hierarchical grid, namely its nested structure, in which the set of points X^{l-1} at level $l-1$ with support nodes $\vec{x}_{\vec{l},\vec{i}}$ is contained in X^l, i.e. $X^{l-1} \subset X^l$.

In one dimension, the following relation for the hierarchical coefficients $\alpha_{l,i}$, $l \geq 1$, i odd holds [5, 10]:

$$\begin{aligned} \alpha_{l,i} &= f\left(x_{l,i}\right) - \frac{f\left(x_{l,i} - h_l\right) + f\left(x_{l,i} + h_l\right)}{2} \\ &= f\left(x_{l,i}\right) - \frac{f\left(x_{l,i-1}\right) + f\left(x_{l,i+1}\right)}{2} \\ &= f\left(x_{l,i}\right) - \frac{f\left(x_{l-1,(i-1)/2}\right) + f\left(x_{l-1,(i+1)/2}\right)}{2}. \end{aligned} \tag{30}$$

Note that the coefficients for the basis functions associated to a non-zero boundary in case of the 'Clenshaw-Curtis' grid (see, Sec. A) are simply given by

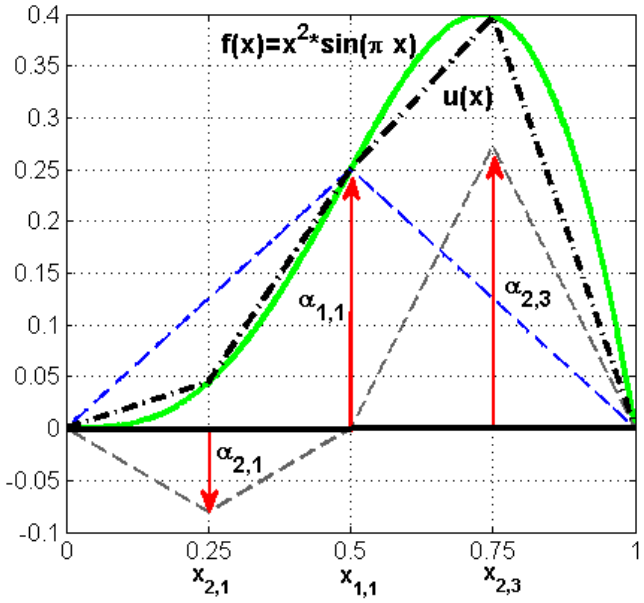

Figure 3: Construction of $u(x)$ interpolating $f(x) = x^2 \cdot \sin(\pi \cdot x)$ with hierarchical linear basis functions of levels 1 and 2. The *hierarchical surpluses* $\alpha_{l,i}$ that belong to the respective basis functions are indicated by arrows (cf, Eq. 30). They are simply the difference between the function values at the current and the previous interpolation levels.

$\alpha_{2,i} = f(x_{2,i}), i = 0, 1.$

In operator form, Eq. 30 can conveniently be rewritten as [10, 5]

$$\alpha_{l,i} = \left[-\frac{1}{2}\ 1 - \frac{1}{2}\right]_{l,i} f, \tag{31}$$

with a generalization to the $d-$dimensional case

$$\alpha_{\vec{l},\vec{i}} = \left(\prod_{t=1}^{d}\left[-\frac{1}{2}\ 1 - \frac{1}{2}\right]_{l_t,i_t}\right) f. \tag{32}$$

Note that coefficients are called *hierarchical surpluses* [5] since a coefficient $\alpha_{\vec{l},\vec{i}}$ corrects the interpolant of level $l-1$ at the points $\vec{x}_{l,i}$ to the actual value of $f(\vec{x}_{l,i})$, as displayed in Fig. 3.

2.5 Adaptive Sparse Grids

The sparse grid structure introduced in the previous sections defines an a priori selection of grid points that is optimal if certain smoothness conditions are met, i.e. if the function has bounded mixed derivatives (cf., Sec. 2.4 and Eq. 24). However, in many applications (see, e.g. [30], with references therein) including economic models with occasionally binding constraints, these prerequisites are not met: the functions of interest often show kinks or finite discontinuities.

Thus, the sparse grid methods outlined so far may fail to provide good approximations, as they can capture the local behaviour only to some limited extend. Therefore, the next task is to find a way of efficiently approximating functions which do not fulfill the necessary smoothness conditions given in Eq. 24.

A very effective strategy to achieve this is to adaptively refine the sparse grid at points around the discontinuity region and spend less points in the region of smooth variation (see, e.g. [30, 31, 29, 13, 4, 16, 25]). By doing so, resources are only invested where needed. While there are various ways to refine a sparse grid (see, e.g. [29], with references therein), we outline only briefly the basic ideas behind the algorithms that we are using in the course of solving our economic models below and omit the technical details. For these, we refer the reader to the original articles, namely the ones by [25] and [29].

When approximating a function as a sum of piecewise linear basis functions, we can hope that the main contributions to the interpolant stem from comparatively few terms with big surpluses (cf., Eq. 28 and Fig. 3). The key point of the refinement strategies of [25, 29] therefore is to monitor the size of the hierarchical surpluses. Recall from Sec. 2.3 and Sec. 2.4 that the interpolated function is represented by a linear combination of hierarchical, piecewise linear hat functions. The piecewise linear hat function has local support in contrast to approaches such as Chebyshev polynomials [26], so it can in principle be used to resolve discontinuities. The coefficients of the hat functions - the hierarchical surpluses - are just the hierarchical increments between two successive interpolation levels. The magnitude of the hierarchical surplus reflects the local regularity of the function. For smooth functions, its value tends to zero as the level l tends to infinity (cf., Eq. 25). On the other hand, for a non-smooth function, a singularity/discontinuity is indicated by the magnitude of the hierarchical surplus. The larger the magnitude is, the stronger the singularity. Therefore, the hierarchical surplus serves as a natural error indicator for the sparse grid interpolation.

Technically, the adaptive grid refinement can be built on top of the hierarchical grid structure. Let us first consider the one dimensional case. The equidistant grid points form a tree-like data structure [25], as displayed in Fig. 4. Going from one level to the next, we see that for each grid point there are two *sons*. For example, the point 0.5 from level $l = 1$ is the *father* of the points 0.25 and 0.75 from level $l = 2$. In the $d-$dimensional case, there are then consequently two *sons* in each dimension for each grid point, i.e. $2d$ *sons*, when going from one to the next hierarchical grid level. Moreover, note that the *sons* are also the neighbor points of the *father*. Recall now from Sec. 2.4 and Eq. 16 that the neighboring points are the support nodes of the hierarchical basis in the next interpolation level. Therefore, by adding neighboring points, we simply add the support nodes from the next interpolation level, i.e. we refine an interpolation from level $l - 1$ to l. In order to adaptively refine the grid, we use the hierarchical surpluses as an error indicator in order to detect the smoothness of the solution and refine the hierarchical basis functions $\phi_{\vec{l},\vec{i}}$ whose magnitude of the hierarchical surplus satisfies:[2]

$$|\alpha_{\vec{l},\vec{i}}| \geq \epsilon \tag{33}$$

Whenever this criterion is satisfied, $2d$ neighbor points of the current point are

[2] Note that there exist also alternative refinement strategies, as described e.g. in [30].

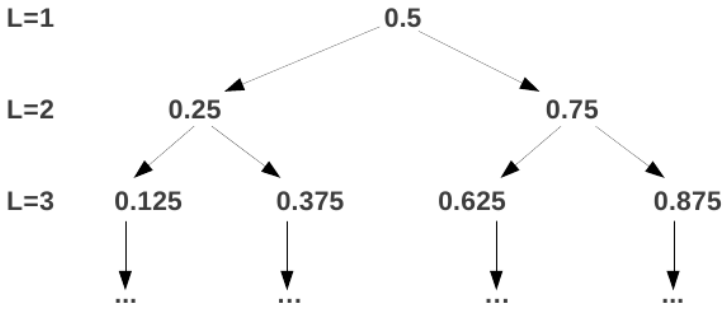

Figure 4: One-dimensional tree-like structure of a 'classical' sparse grid (cf., Sec. 2.4) for the hierarchical levels $l = 1, 2, 3$.

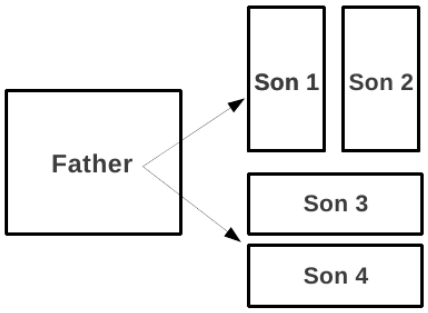

Figure 5: A schematic example of nodes (denoted either by *Father* or *Son*) and supports (size of the respective boxes) of a locally refined sparse grid in two dimensions according to [25, 4].

added to the sparse grid, as shown in Fig. 5. Note that this refinement method is linear in its scaling and thus does not suffer from the curse of dimensionality. For more technical information regarding the implementation of adaptive sparse grid methods, we refer to [25, 29].
Note that in our application in Sec. 3, we interpolate several policies on one grid, i.e. we interpolate a function

$$f : \Omega \to \mathbb{R}^m.$$

Therefore, we get m surpluses at each gridpoint and we thus have to replace the refinement criterion in Eq. 33 by

$$g\left(\alpha^1_{\vec{l},\vec{i}}, \ldots, \alpha^m_{\vec{l},\vec{i}}\right) \geq \epsilon, \tag{34}$$

where the refinement choice is governed by a function $g : \mathbb{R}^m \to \mathbb{R}$. A natural choice for g is the maximum function. However, the optimal choice of g depends on the application at hand, as we will discuss in Sec. 3.5.

Analytical Examples

We now demonstrate the ability of adaptive sparse grid algorithms to efficiently interpolate functions that exhibit steep gradients and kinks. This part contains analytical tests in one and two dimensions in order to foster the understanding of the adaptive sparse grid algorithms in use, namely the ones by [29] and [25]. Their behaviour will be investigated with respect to different settings of grid- and basis functions. Note, however, that these algorithms were extensively and carefully tested before, e.g. via the test problems by [11]. Therefore, we restrict ourselves below to a few examples.

For the testing, we proceed as follows [25, 21]: We pick a (non-smooth) analytical function $f(\vec{x})$, construct the interpolant $u(\vec{x})$ of $f(\vec{x})$ (cf, Eq. 28), then randomly generate 1000 test points from a uniform distribution in $[0,1]^d$, and finally compute the maximum error as follows:

$$e = \max_{i=1,\ldots,1000} |f(\vec{x_i}) - u(\vec{x_i})|. \tag{35}$$

Moreover, we also assess which choice of the sparse grid and its respective basis functions suits our purposes best, either the ones by [29] or the ones by [25]. More precisely, we compare the following two settings:

- *setting A* (algorithm by [25]):
  - grid: Curtis-Clenshaw grid (cf., Eq. 65)
  - basis functions: modified linear basis functions (cf., Eq. 66)
- *setting B* (algorithm by [29]):
  - grid: 'standard' sparse grid (cf., Eq. 11)
  - basis functions: modified linear basis functions (cf, Eq. 64)

As a first educational example, we apply *'setting B'* to the one dimensional test function

$$f(x) = \frac{1}{|0.5 - x^4| + 0.01}. \tag{36}$$

The refinement criterion for the adaptive sparse grid algorithm (cf., Eq. 33) is chosen to be $\epsilon = 10^{-2}$. With this setting, the maximum interpolation error reaches $\mathcal{O}\left(10^{-2}\right)$ (cf., Eq. 35), while 109 grid points have to be spent. In contrast, to attain the same level of convergence, 1023 equidistant grid points have to be spent,[3] as shown in Fig. 6. From Fig. 6, it is obvious that the adaptive sparse grid places points in regions where high resolution is needed, while putting only few points in areas where the function varies little. This fact makes adaptive sparse grid algorithms favourable over all other (sparse) grid interpolation methods if kinks or discontinuities have to be handled. As non-adaptive methods can only provide one resolution over the whole domain, they waste resources where not needed.

As a second example, we apply both *'setting A'* and *'setting B'* with a threshold of $\epsilon = 10^{-2}$ to the two dimensional test function

$$\frac{1}{|0.5 - x^4 - y^4| + 0.1}, \tag{37}$$

[3]Note that in the one dimensional case, an ordinary sparse grid of level l corresponds to the full grid with the same level of refinement.

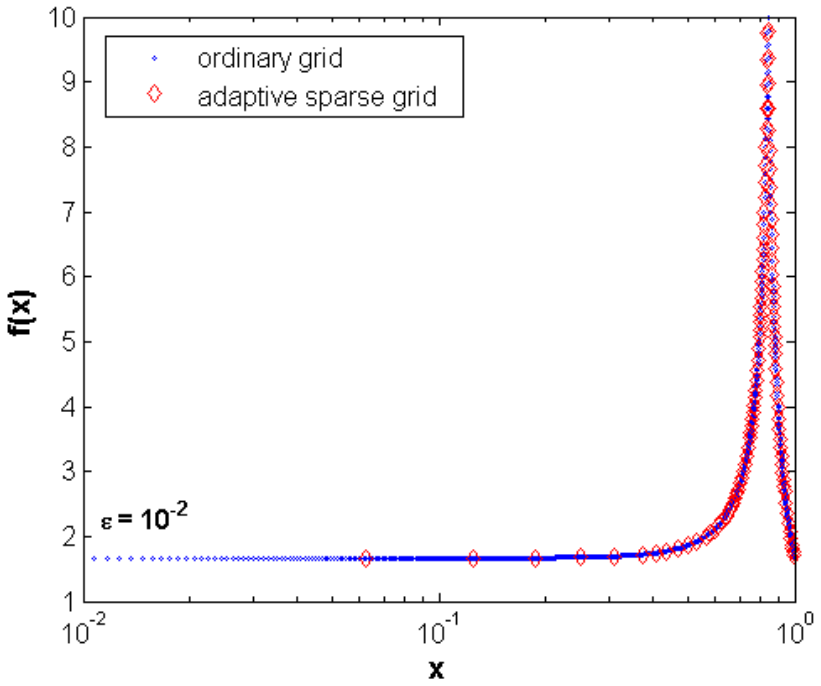

Figure 6: Evaluation of the function given by Eq. 36 at the adaptive sparse grid points (red diamond) and the 'full grid' (blue dots). Both grids attain an maximum error $e = \mathcal{O}\left(10^{-2}\right)$. The adaptive sparse grid reaches this level of convergence with 109 points, whereas the full grid needs 1023 points.

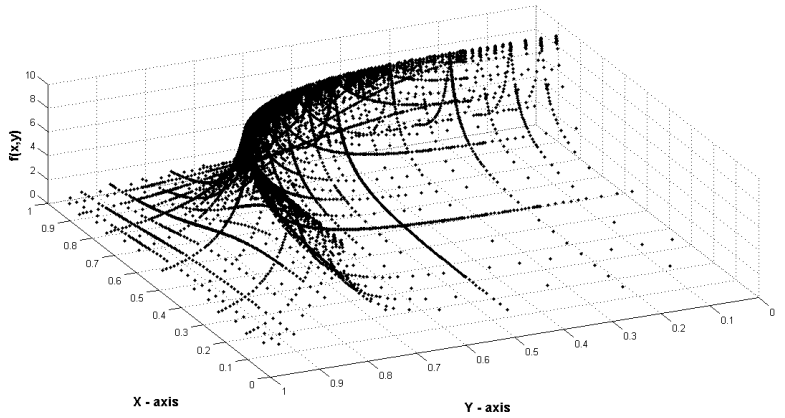

Figure 7: Evaluation of Eq. 37 at the grid points obtained by the adaptive sparse grid algorithm 'setting B' after 15 refinement steps.

a line-singularity, as shown in Fig. 7. In Fig. 8, we show the convergence rate of 'setting A' and 'setting B' with respect to the required grid points, and compare

it to their corresponding, conventional sparse grids. Strikingly, 'setting A' for

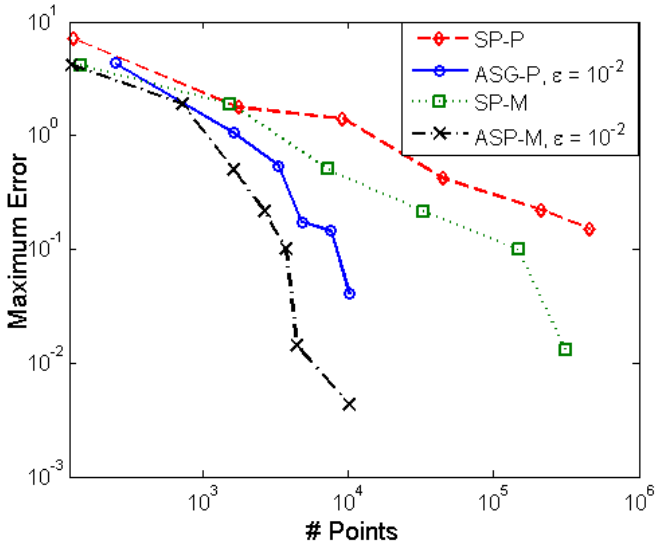

Figure 8: Comparison of the interpolation error (cf., Eq. 35) for conventional and adaptive sparse grid interpolation. Note that the adaptive sparse grid algorithm 'setting A' is labelled by 'ASP-M' (the 'classical' sparse grid by 'SP-M') whereas 'setting B' is denoted by 'ASG-P' (and 'SG-P'). Both adaptive grids were obtained by applying a threshold $\epsilon = 10^{-2}$.

example reaches an accuracy of $e \approx 1.4 \cdot 10^{-2}$, where the interpolation refinement level is 15 and the number of points is $4,411$ as opposed to $311,297$ points using the same level of refinement in the conventional sparse grid. An exemplary evolution of the adaptive sparse grid is shown in Fig. 9. Note that the line of discontinuity is automatically detected by the adaptive sparse grid algorithm (cf. Figs. 9 and 7). From Fig. 8, it also gets apparent that 'setting A' is generally converging even faster than 'setting B'. Thus, we will use 'setting A' to solve the international real business cycle model in Section 3.

3 Application to Dynamic Models

In this section, we apply the adaptive sparse grid method to an economic example, namely a international real business cycle model. We first introduce the model in Sec. 3.1, then present an extension with irreversible investment in Sec. 3.2, and subsequently outline the time iteration algorithm in Sec. 3.3. Finally, we present the parallelization of the code in Sec. 3.4 and discuss its performance in Sec. 3.5.

3.1 International Real Business Cycle Model

To demonstrate the capabilities of adaptive sparse grids in solving dynamic economic models we apply this method to a real business cycle model with

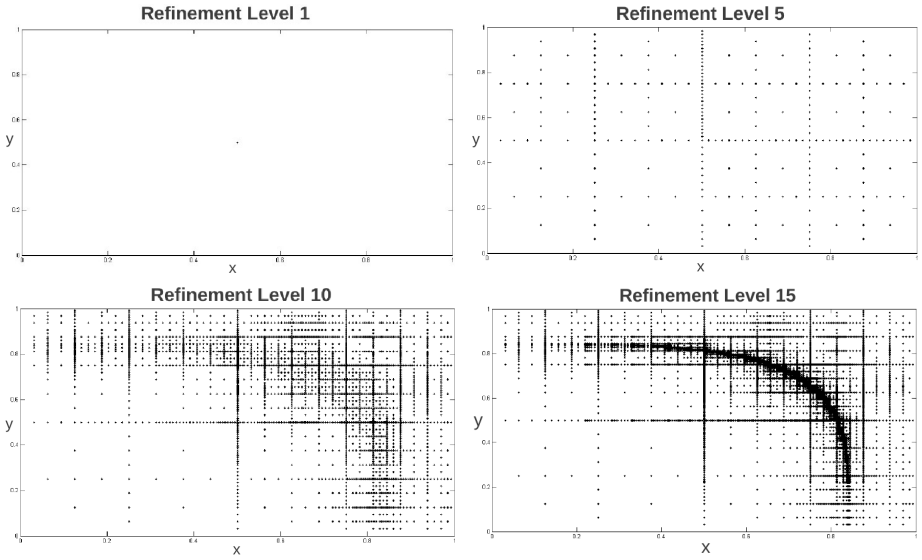

Figure 9: The evolution of an adaptive sparse grid on $[0,1]^2$ with a threshold $\epsilon = 10^{-2}$. The refinement levels $1, 5, 10$ and 15 are shown.

multiple countries, i.e. an international real business cycle model. This model has become a standard for testing computational methods for solving high-dimensional dynamic models (see, [7], with references therein).

Model Description

There are N countries that differ with respect to their exogenous productivity (and possibly preferences) as well as their endogenous capital stock. They all produce, trade and consume a single homogeneous good. Production of country j at time t is given by

$$y_t^j = a_t^j \cdot f^j(k_t^j) \tag{38}$$

where a_t^j, f^j, and k_t^j are productivity, a neoclassical production function, and the capital stock of country j respectively. The law of motion of productivity is given by

$$\ln a_t^j = \ln a_{t-1}^j + \sigma \left(e_t^j + e_t \right), \tag{39}$$

where the shock e_t^j is specific to country j, while e_t is a global shock. These shocks are all i.i.d. standard normal.

The law of motion of capital is given by

$$k_{t+1}^j = k_t^j \cdot (1 - \delta) + i_t^j, \tag{40}$$

where δ is the rate of capital depreciation, and i_t^j is investment. There is a convex adjustment cost on capital, given by

$$\Gamma_t^j(k_t^j, k_{t+1}^j) = \frac{\phi}{2} \cdot k_t^j \cdot \left(\frac{k_{t+1}^j}{k_t^j} - 1 \right)^2. \tag{41}$$

The aggregate (i.e. global) resource constraint is thus given by

$$\sum_{j=1}^{N} y_t^j \geq \sum_{j=1}^{N} \left(i_t^j + \Gamma_t^j(k_t^j, k_{t+1}^j) + c_t^j \right), \tag{42}$$

where c_t^j denotes consumption of country j at time t. Substituting and rearranging, we get

$$\sum_{j=1}^{N} \left(a_t^j \cdot f^j(k_t^j) + k_t^j \cdot (1-\delta) - k_{t+1}^j - \Gamma_t^j(k_t^j, k_{t+1}^j) - c_t^j \right) \geq 0. \tag{43}$$

We assume that the preferences of each country are represented by a time separable utility function with discount factor β and per-period utility function u^j. By further assuming complete markets, the decentralized competitive equilibrium allocation can be obtained as the solution to a social planner's problem, where the welfare weights, τ^j, of the various countries depend on their initial endowments. More precisely, the social planner solves

$$\max_{\{c_t^j, k_t^j\}} \mathbb{E}_0 \sum_{j=1}^{N} \tau^j \cdot \left(\sum_{t=1}^{\infty} \beta^t \cdot u^j(c_t^j) \right), \tag{44}$$

subject to the aggregate resource constraint (43).

First Order Conditions

To get the first order conditions (FOCs) of problem (44), we differentiate the Lagrangian with respect to c_t^j :

$$\tau^j \cdot u_c^j(c_t^j) - \lambda_t = 0, \tag{45}$$

and with respect to k_{t+1}^j:

$$\begin{aligned} &\lambda_t \left[-1 - \frac{\partial \Gamma_t^j(k_t^j, k_{t+1}^j)}{\partial k_{t+1}^j} \right] + \\ &\beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \cdot \left[a_{t+1}^j \cdot f_k^j(k_{t+1}^j) + (1-\delta) - \frac{\partial \Gamma_{t+1}^j(k_{t+1}^j, k_{t+2}^j)}{\partial k_{t+1}^j} \right] \right\} = 0, \end{aligned} \tag{46}$$

where λ_t denotes the multiplier on the time t resource constraint. Differentiating the adjustment cost function (41), simplifying, and defining the growth rate of capital by $g_t^j = k_t^j / k_{t-1}^j - 1$, Eq. 46 reads:

$$\begin{aligned} &-\lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j \right] + \\ &\beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \cdot \left[a_{t+1}^j \cdot f_k^j(k_{t+1}^j) + 1 - \delta + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2 \right) \right] \right\} = 0. \end{aligned} \tag{47}$$

Concerning the production function $f^j(k_t^j)$ and the marginal utility function $u_c^j(c_t^j)$, we assume the following standard functional forms:

$$f^j(k_t^j) = A \cdot (k_t^j)^{\alpha} \tag{48}$$

and

$$u_c^j(c_t^j) = (c_t^j)^{-\frac{1}{\gamma_j}}, \tag{49}$$

which imply

$$f_k^j(k_t^j) = A \cdot \alpha \cdot (k_t^j)^{\alpha-1}, \tag{50}$$

$$c_t^j = \left(\frac{\lambda_t}{\tau_j}\right)^{-\gamma^j} \tag{51}$$

Using these expressions, we finally get the system of $N+1$ equilibrium conditions that we solve in our computations, namely for all countries $j \in \{1, ..., N\}$:

$$\begin{aligned} &\lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j\right] - \\ &\beta \cdot \mathbb{E}_t \left\{\lambda_{t+1} \left[a_{t+1}^j \cdot A \cdot \alpha \cdot (k_{t+1}^j)^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2\right)\right]\right\} = 0, \end{aligned} \tag{52}$$

and the aggregate resource constraint

$$\sum_{j=1}^{N} \left(a_t^j \cdot A \cdot (k_t^j)^{\alpha} + k_t^j \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g_{t+1}^j)^2\right) - k_{t+1}^j - \left(\frac{\lambda_t}{\tau_j}\right)^{-\gamma^j}\right) = 0. \tag{53}$$

We solve the IRBC model by iterating on Eqs. 52 and 53. The implementation details are provided in Sec. 3.3.

Parameterization

With respect to the parameter choices, we follow Juillard and Villemot [20], who provide the model specifications for the comparison study that uses the IRBC model to compare several solution methods (see, [7]). We choose an asymmetric specification where preferences are heterogeneous across countries. In particular, the intertemporal elasticity of substitution (IES) of the N countries is evenly spread over the interval $[0.25, 1]$. This corresponds to model A5 in Juillard and Villemot [20]. The only difference between our parameterization and theirs is that we use a (quarterly) depreciation rate of $\delta = 1\%$, while they write down the model such that they have effectively no depreciation. The parameters we use are reported in Tab. 2. The welfare weights τ^j need not to be specified as they do not matter for the capital allocation, but only for the consumption allocation which we do not consider. The parameter A is chosen such that capital of each country is equal to 1 in the deterministic steady state.

3.2 IRBC Model With Irreversible Investment

To demonstrate that our algorithm can handle non-smooth behavior, we include irreversible investment in the IRBC model of Sec. 3.1. More precisely, we assume that investment cannot be negative, thus for each country $j \in \{1, ..., N\}$ the following constraint has to be satisfied:

$$k_{t+1}^j \geq k_t^j \cdot (1-\delta). \tag{54}$$

Parameter	Symbol	Value
discount factor	β	0.99
IES of country j	γ^j	a+(j-1)(b-a)/(N-1) with a=0.25, b=1
capital share	α	0.36
depreciation	δ	0.01
std. of log-productivity shocks	σ	0.01
autocorrelation of log-productivity	ρ	0.95
intensity of capital adjustment costs	ϕ	0.50
number of countries	N	2, 4, 6, 8, 10, or 11

Table 2: Choice of parameters for the IRBC model

As a consequence, we have to solve a system of $2N + 1$ equilibrium conditions. Namely, for all countries $j \in \{1, ..., N\}$ the Euler equations and the irreversibility constraints:

$$\begin{aligned} &\lambda_t \cdot \left[1 + \phi \cdot g^j_{t+1}\right] - \mu^j_t \\ &- \beta \cdot \mathbb{E}_t \left\{ \lambda_{t+1} \left[a^j_{t+1} \cdot A \cdot \alpha \cdot (k^j_{t+1})^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g^j_{t+2} \cdot \left(g^j_{t+2} + 2 \right) \right] \right\} = 0, \\ &k^j_{t+1} - k^j_t(1-\delta) \geq 0, \mu^j_t \geq 0, \left(k^j_{t+1} - k^j_t(1-\delta) \right) \cdot \mu^j_t = 0 \end{aligned} \tag{55}$$

and also the aggregate resource constraint:

$$\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^\alpha + k^j_t \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g^j_{t+1})^2 \right) - k^j_{t+1} - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0. \tag{56}$$

In Eq. 55 the variable μ^j_t denotes the Kuhn-Tucker multiplier for the irreversibility constraint $k^j_{t+1} - k^j_t \cdot (1-\delta) \geq 0$.
The parameters we use for the IRBC model with irreversible investment are the same as the ones we use for the IRBC model, except that we set $\phi = 0$. Thus, the convex adjustment costs on capital are now replaced by the irreversibility constraint.

3.3 Time Iteration Algorithm

To compute an equilibrium of the IRBC model, we embed adaptive sparse grid interpolation in a time iteration algorithm[4] (see, e.g. [19]). We solve for an equilibrium that is Markov in the the physical state of the economy, which is given by the capital stock and the productivity levels of all N countries:

$$\left(a^1_t, \ldots, a^N_t, k^1_t, \ldots, k^N_t\right). \tag{57}$$

Denoting the state space by $S \subset \mathbb{R}^{2N}_+$, we thus solve for policies

$$p = \left(k^1_{t+1}, \ldots, k^N_{t+1}, \lambda_t\right) : S \to \mathbb{R}^{N+1}_+. \tag{58}$$

[4]Note that the focus in this section lies on the description of the time iteration algorithm for the smooth IRBC model – however, the time-iteration procedure for the non-smooth IRBC works analogously.

Such policies represent an equilibrium of the IRBC model, only if they satisfy Eqs. 52 and 53 at all points in the state space. To compute policies that approximately satisfy this condition, we use time iteration and employ adaptive sparse grid interpolation in each of its iteration steps.

The structure of the time iteration algorithm is as follows:

1. Make an initial guess for next period's policy function:

$$p_{init} = \left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1}\right).$$

   Set $p_{next} = p_{init}$.

2. Make one time iteration step:

   (a) Using an adaptive sparse grid procedure, construct a set G of grid points

$$g = \left(a_t^1, \ldots, a_t^N, k_t^1, \ldots, k_t^N\right) \in G \subset S$$

   and optimal policies at these points

$$p(g) = \left(k_{t+1}^1(g), \ldots, k_{t+1}^N(g), \lambda_t(g)\right)$$

   by solving at each grid point g the system of equilibrium conditions (cf. Eqs. 52 and 53, or 55 and 56) given next period's policy

$$p_{next} = \left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1}\right).$$

   (b) Define the policy function p by interpolating between $\{p(g)\}_{g \in G}$.

   (c) Calculate (an approximation for) the error, e.g.

$$\eta = \|p - p_{next}\|_\infty.$$

   If $\eta > \epsilon$, set $p_{next} = p$ and go to step 2, else go to step 3.

3. The (approximate) equilibrium policy function is given by p.

The essential step in this algorithm is 2(a). To understand the details of this step, it is crucial to know which variables in Eqs. 52 and 53 (or 55 and 56) are grid points, known policy functions, or unknown policies respectively.

3.4 Parallelization and Scaling

In order to enable the solution of 'large' problems in a reasonable amount of time, we aim to access modern high-performance computing architectures. The time iteration algorithm used to solve the IRBC model is therefore parallelized by a hybrid parallelization scheme, i.e. with MPI [36] (distributed memory parallelization) between the compute nodes and OpenMP [17] within the nodes. We achieve this as follows. We construct adaptive sparse grids according to the algorithm of [25] (cf. 'setting A', Sec. 2.5), which allows for an MPI parallel evaluation of the hierarchical surpluses, i.e. it distributes the newly generated points within a refinement step among different multicores (see, Fig. 10). On top of this, we add an additional level of parallelism. Locally, we solve the nonlinear system of equations (see, Sec. 3.1 and Eqs. 52 and 53 or Eqs. 55 and 56,

respectively) in a shared memory fashion (OpenMP) in order to evaluate the hierarchical surpluses at these particular gridpoints. A schematic illustration of an arbitrary time step is displayed in in Fig. 10. Note that in our implementation, we solve the set of nonlinear equations with IPOPT [39] in combination with PARDISO [35, 34].

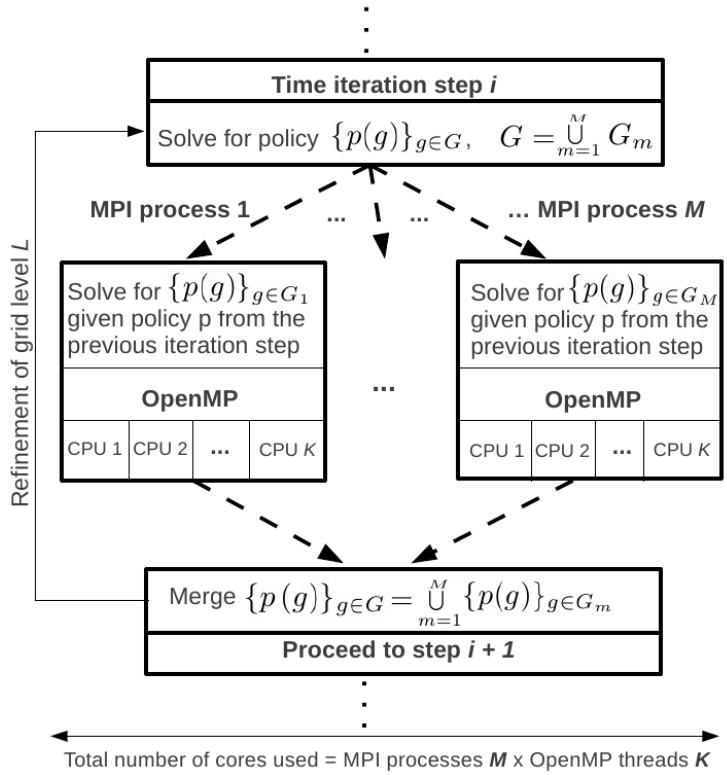

Figure 10: Schematic representation of the hybrid parallelization of a time iteration step (see, Sec. 3.3). The total number of cores used is given by the number of MPI processes times the numbers OpenMP threads.

The advantage of this parallelization scheme is that it drastically reduces the amount of MPI communication required between different multicores. This fact is important, as the MPI communication overhead between different processes can turn out to be a roadblock for the efficiency of the code when going to 'large' process numbers [36].

Indicative performance numbers are shown in Fig. 11, where we display the speedup S_p and the efficiency E_p of the code. These two quantities are defined by [33]

$$S_p = \frac{T_0}{T_p}, \qquad E_p = \frac{S_p}{p} = \frac{T_0}{pT_p}, \tag{59}$$

where T_0 is the execution time of the test benchmark, T_p is the execution time of the algorithm running with a multiple of $p-$times the processes of the baseline. We see that the code scales nicely to at least about $1,200$ parallel processes, as shown in Fig. 11.

In Fig. 12, we show how the number of gridpoints grows with the number of continuous state variables in models that are solved with an adaptive sparse grid. As expected, the number of points grows only moderately with the dimension, i.e. $\sim \mathcal{O}(d)$. The running times on the other hand grow faster, because the number of gridpoints is not the only thing that is increasing with the dimension of the problem. The size of the equation systems (cf., Eqs. 52 and 53, or Eqs. 55 and 56, respectively) that have to be solved at each gridpoint also grows linearly in the dimension, implying that the size of the Jacobians that have to be computed grows quadratically. Therefore, the time spent solving the set of nonlinear equations represents the true roadblock of the time iteration scheme presented here. However, we can - at least to some limited extend - control the increasing running times by simply using a larger number of CPUs. Thus, while 14 to 16 dimensional models are still comfortably tractable on a single desktop or workstation with acceptable accuracy, models with 20 or more dimensions need to be run on a high performance computer in order to get results in a reasonable amount of time.

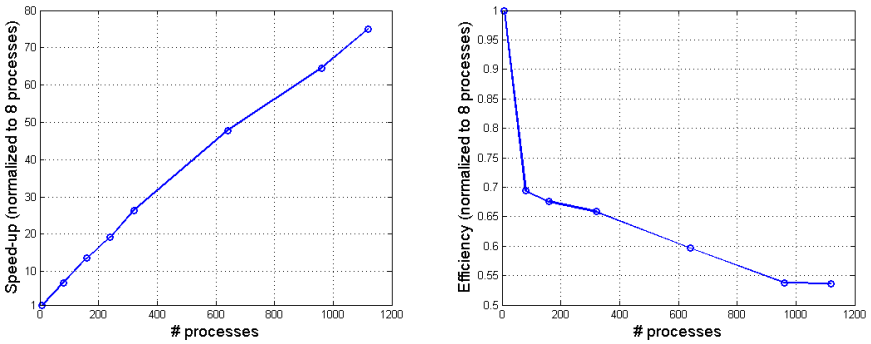

Figure 11: The figure shows 'strong scaling' of the code under a hybrid parallelization with MPI between nodes and Open MP within nodes. The test was performed on the Schrödinger system (nodes with 8 core Intel Xeon X5560 2.8 GHz processors) at the University of Zürich and consisted of one representative time step. The test problem was a 10-dimensional classical sparse grid of refinement level 3 on which we solve Eqs. 52 and 53. The speedup, normalized to 8 processes, is shown on the left hand side and refers to how much the parallel algorithm is faster compared to its baseline (see, Eq. 59). The efficiency is displayed on the right hand side.

3.5 Performance and Accuracy

So far, we have described the models we solve, the algorithm we use to solve them, and the way in which we parallelize this algorithm. In this section, we show how this algorithm performs in solving the IRBC model. Yet first of all, some implementation details and the measures we use to assess accuracy have to be described.

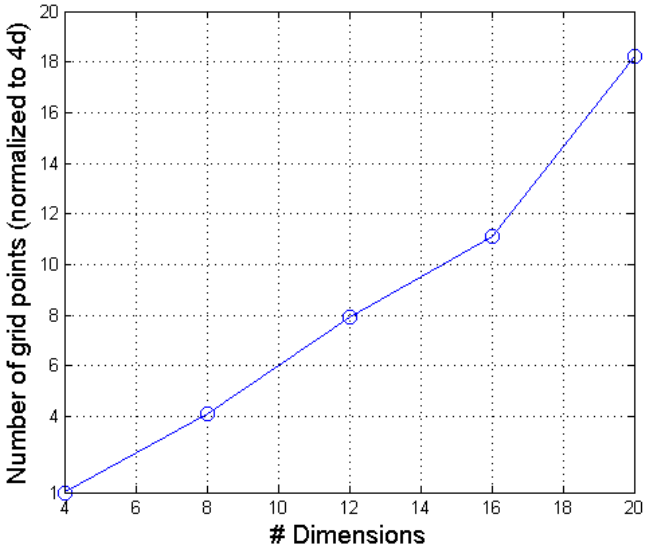

Figure 12: The figure shows how the number of gridpoints grows with increasing dimensionality of the problem, normalized to 4 dimensions. The test problem was the IRBC model that was run with an adaptive sparse grid and the refinement criterion set to $|\alpha_{\vec{l},\vec{i}}| \geq \epsilon = 2.5 \cdot 10^{-3}$ (cf., Eq. 33). Note that this choice of ϵ leads to model accuracies comparable to the ones reported in Tab. 4.

Implementation Details

One important detail is the integration procedure used to evaluate the expectations operator, e.g. in Eq. 52. As we want to focus on the grid structure, we chose an integration rule that is simple and fast, yet not very accurate.[5] In particular, we use a simple monomial rule that uses just two evaluation points per shock, i.e. $2(N+1)$ points in total (see, [19], with references therein). As we apply the same rule along the time-iteration algorithm as well as for the error evaluation, this choice factors out the question of finding integration procedures that are both accurate and efficient. In principle integration could also be carried out using an (adaptive) sparse grid (see Appendix B), yet not over the same space that the policy functions are interpolated on. Therefore, we view integration as a problem that is orthogonal to the choice of the grid structure, and thus do not focus on it.

Another implementation detail concerns the possibility of accelerating the time iteration procedure by starting with coarse grids and later changing to finer grids. For instance, using classical sparse grids, the overall computation time can be reduced by one order of magnitude when using a level 1 grid for 200 iterations, followed by 80 iterations on a level 2 grid, before finally using a level 3 grid for 20 periods instead of running all 300 iterations with a grid of level 3. Our tests indicate that this approach yields the same accuracy of results as if the

[5]For the IRBC, however, the integration rule we use delivers the same accuracy as Monte Carlo integration with 1000 random evaluation points.

entire simulation would have been carried out at level 3. As with integration, it is not the focus of this paper to discuss the most efficient acceleration strategy. Of course, if results are compared below, they are achieved with comparable acceleration strategies.

Error Measure

To evaluate the accuracy of our solutions we compute (unit-free) errors in the $N+1$ equilibrium conditions. Namely, for all countries $j \in \{1, ..., N\}$ we get one Euler equation error:

$$\left[\beta \cdot \mathbb{E}_t \left\{\lambda_{t+1} \cdot \left[a^j_{t+1} \cdot A \cdot \alpha \cdot (k^j_{t+1})^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g^j_{t+2} \cdot \left(g^j_{t+2} + 2\right)\right]\right\}\right] \cdot \left[\lambda_t \cdot \left(1 + \phi \cdot g^j_{t+1}\right)\right]^{-1} - 1, \quad (60)$$

and we get one additional error from the aggregate resource constraint:

$$\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^{\alpha} + k^j_t \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g^j_{t+1})^2\right) - k^j_{t+1} - \left(\frac{\lambda_t}{\tau_j}\right)^{-\gamma^j}\right) \cdot \left(\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^{\alpha} + k^j_t \cdot \left(-\frac{\phi}{2} \cdot (g^j_{t+1})^2\right)\right)\right)^{-1}. \quad (61)$$

These expressions are evaluated by using the computed equilibrium policy function to calculate both today's policy and next period's policy. We compute these errors for all points in the state space that are visited along a (ten thousand period) long simulation path. For each of these points we get $N+1$ errors. We then take the maximum over the absolute value of these errors, which results in one error for each point. Over all these errors, we compute both the maximum (Max. Error) and the average (Avg. Error), which we all report in $\log_{10}$-scale. In case of the IRBC model with irreversible investment there is one additional complication. Denoting the error defined in Eq. 60 by EE^j and defining the percentage violation of the irreversibility constraint by

$$IC^j \equiv -\frac{k^j_{t+1}}{k^j_t \cdot (1-\delta)} \quad (62)$$

the error is now given by

$$\max\left(EE^j, IC^j, \min\left(-EE^j, -IC^j\right)\right) \quad (63)$$

The first term within the max operator, EE^j, is positive when the marginal cost of investing in country j today is lower then the discounted marginal benefit of this investment tomorrow. Thus, investment in country j is sub-optimally low. Independent of irreversibility, this is always an error, as the irreversibility constraint does not prohibit investing more. The second term, IC^j, is positive if the irreversibility constraint is violated; in this case, it measures the relative size of the violation. Finally, if $-EE^j$ is positive, then the marginal cost of investing in country j today is higher then the discounted marginal benefit

Dimension	Level	Points	Max. Error	Avg. Error
4	3	137	-2.82	-3.68
4	4	401	-2.98	-4.19
4	5	1105	-3.19	-4.24
4	6	2929	-3.28	-4.55

Table 3: Errors for a 'classical' sparse grid solution of the smooth IRBC model with fixed dimension and increasing approximation level.

Dimension	Level	Points	Max. Error	Avg. Error
4	3	137	-2.97	-3.55
8	3	849	-3.04	-3.83
12	3	2649	-2.71	-3.78
16	3	6049	-2.72	-3.80
20*	2	841	-2.58	-3.29
22*	2	1013	-2.60	-3.29

Table 4: Errors for a 'classical' sparse grid solution of the smooth IRBC model with increasing dimension and fixed approximation level 3, except for dimensions 20 and 22 where we use only level 2.

of this investment tomorrow. Thus, investment in country j is sub-optimally high. Thus, lower investment would be optimal. Yet, if the constraint is almost binding investment can only be lowered slightly; in this case, the error is given by the slack in the irreversibility constraint, which is $-IC^j$. Therefore, in the case that $-EE^j$ is positive, the error is not simply given by $-EE^j$ but by $\min\left(-EE^j, -IC^j\right)$.

Results for the smooth IRBC model

We first consider sparse grids that are not adaptive. Tab. 3 shows how the approximation errors decrease as we increase the resolution level of the grid keeping the dimensionality of the problem fixed at $2N = 4$. We can see that the maximal and average errors fall as the level of the grid and thereby the number of gridpoints increases. This is exactly what is to be expected. Note that the errors are reasonably low even for a relatively small number of gridpoints. This is simply because the policy functions of the IRBC model are very smooth. Therefore, an adaptive grid cannot improve much on classical sparse grids. For the same reason, the accuracy of our solutions falls short of the accuracy obtained by some smooth approximation methods used in the comparison study summarized by [22].

Let us now turn to higher dimensions. In Tab. 4, we vary the dimensionality of the problem, keeping the approximation level fix at 3, except for dimensions 20 and 22 where we use only level 2 in order to safe CPU resources. We find that the accuracy fluctuates a little as we change the dimensionality of the problem, which is to be expected especially because the preference heterogeneity depends on the dimension of the problem (see Tab. 2). Importantly, however, there seems to be no clear downward trend in accuracy. Thus, for a given resolution level the accuracy of the solution does not deteriorate with the dimensional-

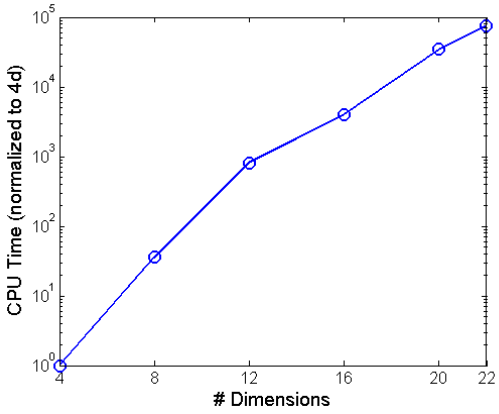

Figure 13: The figure shows how CPU time (in log-scale) increases with the dimensionality of the problem. CPU time is normalized to the computation time of the four dimensional case. The test problem was one representative time-iteration step in a 'classical' sparse grid of level 2.

ity of the problem. Clearly, the number of gridpoints increases, but far from exponentially, which is the defining feature of sparse grids. Of course, what matters in the end is not the number of gridpoints needed, but the computation time and memory required. Yet, both time and memory consumption of the algorithm highly depend on the number of gridpoints. Concerning CPU time, Fig. 13 shows that it substantially increases as we consider higher dimensions. While for example an average time step for $2N = 4$ consumes ~ 0.003 CPUh in 'classical' sparse grid of level 3, the same model with $2N = 16$ needs ~ 93 CPUh. This is largely driven by the increasing size of the non-linear equation systems that have to be solved, and only to a lesser extend due to the increasing number of gridpoints. All in all, the increase in CPU time is less than exponential (cf, Fig. 13). Thus, we can - at least to some limited extend - control the increasing running times by simply using a larger number of cores, as explained in Sec. 3.4.

Results for the non-smooth IRBC model

We now turn to the IRBC model with irreversible investment, as described in Sec. 3.2. The assumption that investment is not reversible induces kinks in the policy functions that we interpolate. Two such kinks can be observed in Fig. 14. For the two country case, Fig. 14 plots the capital choice for country 2 as a function of the capital holding of country 1 keeping all other (three) state variables fix.
When looking at Fig. 14 one has to keep in mind that a kink that appears as a single point in that slice through the $2N$-dimensional space is indeed a $(2N-1)$-dimensional hypersurface. Clearly, such high-dimensional kinks pose a substantial challenge for interpolation.
In contrast to the IRBC model, the 'classical' sparse grid is now much less successful in reducing the approximation errors, as Tab. 5 indicates. In particular,

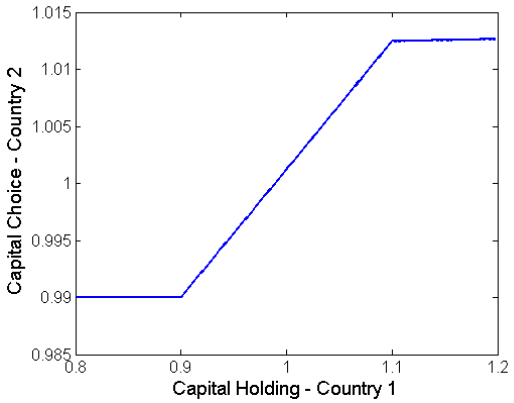

Figure 14: Capital choice of country 2 as a function of capital holding of country 1, where all other 3 state variables of the two country model are kept fixed at their deterministic steady state levels. The four-dimensional policy function was interpolated on an adaptive sparse grid with threshold $\epsilon_{k_i} = 10^{-2}$ and maximum refinement level of $L = 6$.

Dimension	Level	Points	Max. Error	Avg. Error
4	2	41	-1.63	-2.70
4	4	401	-1.71	-2.87
4	6	2929	-1.71	-3.08

Table 5: Errors for a 'classical' sparse grid solution of the non-smooth IRBC model with fixed dimension and increasing approximation level.

the maximum error hardly decreases as the number of gridpoints increases. The reason is that even with high resolution, it is hard to at least roughly match the kinks.

Tab. 6 shows that adaptive sparse grids are more effective in reducing the approximation error, as they put additional resolution where needed. For instance, an adaptive grid with 435 points already achieves a maximum error that is better than for a fixed grid with 2929 points (see Tab. 5), while the average error is not much worse. From Tab. 6, one can see that there are two control parameters in the adaptive grid refinement scheme, namely the maximum refinement level, as well as the refinement threshold. The latter determines around which *father*-gridpoints *son*-gridpoints are added (cf., Eq. 33 and Fig. 5). In addition to the size of the threshold, we have to choose which variables determine the refinement, as we are interpolating several policies on one single grid. We consider two different choices (corresponding to different choices of the g-function in Eq. 34): First, we choose the refinement to be governed by the maximum absolute surpluses of the capital policies k_i; second, we use the maximum surpluses of the Kuhn-Tucker multipliers μ_i. From Tab. 6 it is obvious that refinement based on the capital levels is more effective in reducing the average error, while a refinement based on multipliers is more effective in reducing the maximum error. This makes perfect sense, as the maximum errors occur near the kinks, which the Kuhn-Tucker multipliers capture best.

Max. Level	Statistics	Refinement Threshold			
		$\epsilon_k = 0.015$	$\epsilon_k = 0.010$	$\epsilon_\mu = 0.015$	$\epsilon_\mu = 0.010$
6	Points	200	677	213	435
	Max. Error	-1.60	-1.95	-1.99	-2.41
	Avg. Error	-2.70	-2.92	-2.41	-2.75
10	Points	384	2558	334	1996
	Max. Error	-1.60	-2.43	-1.99	-2.46
	Avg. Error	-2.70	-2.91	-2.70	-2.80

Table 6: Errors for an adaptive sparse grid solution of the **non-smooth** IRBC model with fixed dimension, for different refinement thresholds, and for various maximal approximation levels (first row). The columns with ϵ_k and ϵ_μ represent models in which we refined the grid either with respect to capital (k_i) or the Kuhn-Tucker multipliers (μ_i).

By cleverly choosing the maximal refinement level, the refinement rule, as well as the refinement threshold, we can (already in low dimensions) speed-up the computations by at least one order of magnitude since considerably fewer points have to be computed. Note that choosing $|\epsilon| \to 0$ leads to a 'classical' sparse grid of the corresponding refinement level, which is not desirable. On the other hand, setting $|\epsilon|$ too big may lead to the case that the regions of interest will not be detected, as the automatic refinement stops already after only a few refinement levels (see results for $\epsilon_k = 0.015$ in Tab. 6). Reducing $|\epsilon|$ can therefore be seen as the major control parameter for reducing the average errors. The maximum refinement level on the other hand gives control over how much the adaptive sparse grid can be refined at a local instance. This allows for a very high local resolution in areas of the computation where the function values vary considerably, while regions of smooth behavior obtain less grid points. We display this behaviour in Fig. 15, where a two dimensional projection of two adaptive grids is shown which only differ in their maximum refinement level. The maximum refinement level therefore can be considered as the major control parameter for reducing the maximum error, as the case $\epsilon_k = 0.01$ shows best. However, if the threshold is to high, then higher refinement levels cannot help much in reducing the maximum error, as the examples $\epsilon_k = 0.015$ and $\epsilon_\mu = 0.015$ in Tab. 6 show. In any case, one has to experiment with the maximum refinement level as well as with the refinement rule and threshold in order to find an efficient update scheme for a particular problem at hand. In the case of the non-smooth IRBC model for example, we find $\epsilon_\mu = 0.015$ with maximal refinement level 6 provides the best balance between a low number of gridpoints and a relatively high accuracy.

4 Conclusion

We are the first to use an adaptive sparse grid algorithm to solve dynamic economic models. We achieve this by embedding this algorithm in a time-iteration procedure. In addition, we provide a fully hybrid parallel implementation of the resulting time-iteration adaptive sparse grid algorithm. With this implementation, we can efficiently use current high-performance computing technology and

Figure 15: A 2 dimensional slice of an adaptive sparse grid that was generated in the course of running a $2N = 4$ dimensional simulation. The x-axis shows capital holding of country 1, the y-axis shows capital holding of country 2, while the productivities of the two countries are kept fixed at their mean levels. The 'blue-dotted' grid was generated by the refinement threshold $\epsilon = 0.01$ and the maximum refinement level fixed at $L = 6$. The 'red-crossed' grid was created by the same $\epsilon = 0.01$, but with the maximum refinement level set to $L = 8$.

are, to the best of our knowledge, the first paper on dynamic economic models to do so. We thus provide an important contribution to the emergent literature on parallel computing applications in economics.

The time-iteration adaptive sparse grid algorithm we use is highly flexible and scalable. First, by choosing the resolution level we can tightly control accuracy, thus being able to strike the right balance between running times and the desired accuracy of the solution. Second, keeping the resolution level fixed, we can increase the dimensionality of the problem without worsening accuracy. Third, due to the highly parallelized implementation, we can speed up the computations tremendously by simply using a larger number of CPUs. This allows us to solve hard problems in a relatively short amount of time, and to tackle problems that were so far non-tractable.

We apply this algorithm to an IRBC model with adjustment costs and up to more than 20 continuous state variables. We are also able to solve a high-dimensional IRBC model with irreversible investment. In that application the comparative advantage of the adaptive sparse grid comes into full play, as it can efficiently capture the kinks induced by irreversibility without wasting additional gridpoints in regions of the state space where they are not needed.

Note that to solve the IRBC model we embed the adaptive sparse grid in a time iteration procedure that operates in the space of policy function. However, adaptive sparse grids are also very promising for interpolating value functions in classical dynamic programming applications. For such applications, one only needs to approximate a one-dimensional value function on the adaptive sparse grid, whereas we have to approximate several policy functions on a single grid. We hope that our paper inspires many economic applications of adaptive sparse grids. Being scalable and flexible, adaptive sparse grids can make use of modern high-performance computing infrastructure, and they can be applied to a broad variety of setups where high dimensional functions have to be interpolated efficiently. This tool thus offers the promise to economic modellers of being able to solve models that include much more heterogeneity than was previously possible.

A Sparse Grids with Non-Zero Boundaries

In Sec. 2, we have assumed that the functions under consideration vanish at the boundary of the domain, i.e. $f|_{\partial\Omega} = 0$. To allow for non-zero values on the boundaries, the procedure one usually follows is to add additional gridpoints that are directly located on $\partial\Omega$, and additional, overlapping basis functions are added [3, 29, 21]. However, this method is not suited for higher-dimensional problems, since at least 3^d support nodes are needed [21], with almost all gridpoints located on the boundary and only a few in the inner part.

A natural way to mitigate this effect while still being able to handle $f|_{\partial\Omega} \neq 0$ is, among others, to omit gridpoints on the boundary and modify the interior basis functions instead to extrapolate towards the boundary of the domain. This approach is especially well suited in settings where for example high accuracy close to the boundary is not required and where the underlying function does not change too much towards the boundary [29].

An appropriate choice for the modified one-dimensional basis functions reads [29]:

$$\phi_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2^l \cdot x_t + 1 - i & \text{if } x_t \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1 \\ \phi\left(x_t \cdot 2^l - i\right) & \text{else.} \end{cases} \tag{64}$$

where the support nodes have the same coordinates as in the ordinary sparse grid (cf., Eq. 26). The $d-$dimensional basis functions are obtained in analogy to Eq. 14 by a tensor product construction of the one-dimensional ones. We denote this function space V_n^S, as it also handles boundary points.

Another popular choice to handle non-zero boundaries is the so-called 'Clenshaw-Curtis' sparse grid $V_n^{S,CC}$ with equidistant support nodes [25, 21, 1, 28]. The only difference compared to the sparse grid $V_{n,0}^S$ is that the index set of the

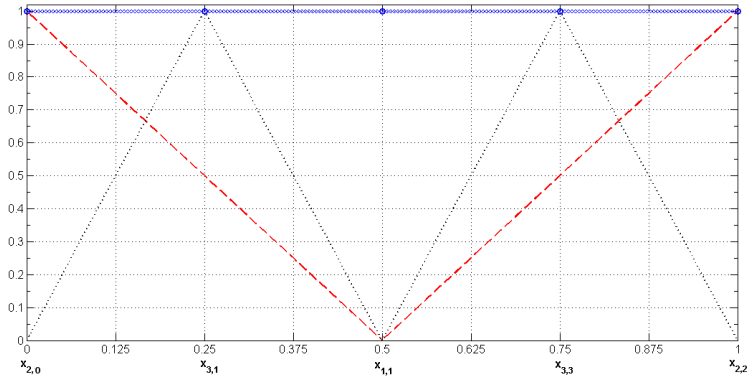

Figure 16: Hierarchical basis functions of the 'Clenshaw-Curtis' grid. Level 1 (solid blue), level 2 (dashed red), and level 3 (dotted black).

support nodes are not given by Eq. 16, but rather by

$$I_{\vec{l}} := \begin{cases} \{\vec{i} : i_t = 1, 1 \leq t \leq d\} & \text{if } l = 1 \\ \{\vec{i} : 0 \leq i_t \leq 2,\ i_t \text{ even}, 1 \leq t \leq d\} & \text{if } l = 2 \\ \{\vec{i} : 1 \leq i_t \leq 2^{l_t - 1} - 1,\ i_t \text{ odd}, 1 \leq t \leq d\} & \text{else} \end{cases} \tag{65}$$

and the one-dimensional basis hat functions:

$$\phi_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 1 - 2 \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 0 \\ \left\{ \begin{array}{ll} 2 \cdot x_t - 1 & \text{if } x_t \in \left[\frac{1}{2}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 2 \\ \phi_{l,i}(x_t) & \text{else.} \end{cases} \tag{66}$$

The modified basis functions of the 'Clenshaw-Curtis' grid are displayed in Fig. 16, whereas the support nodes are shown in Figs. 17 and 18. Note that $\phi_{l,i}(x)$ is given by Eq. 8.

Again, the $d-$dimensional basis functions are obtained by a tensor product construction of the one-dimensional ones. Note moreover that this is in fact a *'Newton-Cotes'* grid using equidistant support nodes [25]; however, the name *'Clenshaw-Curtis'* adhered in the literature due to its association with the grid structure, using the same number of gridpoints as its original name donor.

It is also worth mentioning that the number of gridpoints of the Clenshaw-Curtis grid $|V_n^{S,CC}|$ grows even slower than $|V_n^S|$ (see, Tab. 7).

Note at this point sparse grid methods are not only restricted to piecewise $d-$linear basis functions and their modified counterparts; there are more types of basis functions possible such as piecewise $d-$polynomial ones (see, [5, 29], with references therein). However, we focus in this work on the simple linear hat functions as they have 'good' properties for resolving discontinuities as well as for the adaptive refinement of the sparse grid (see, Sec. 2.5), as they have non-overlapping support.

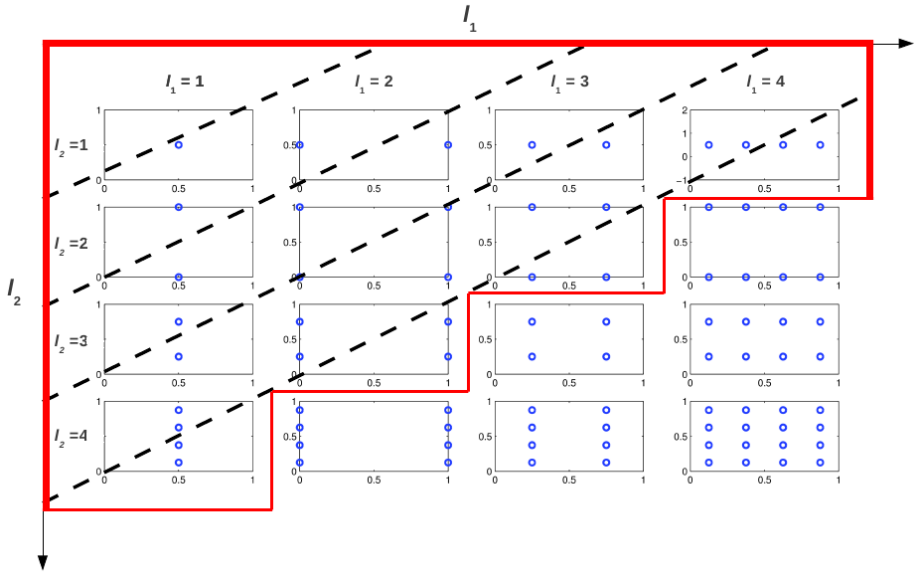

Figure 17: Schematic construction of a 'Clenshaw-Curtis' sparse grid in two dimensions (cf., Eq. 65), i.e. $d = 2$. Hierarchical increment spaces W_{l_1,l_2} for $1 \leq l_1, l_2 \leq n = 4$. The area enclosed by the red bold lines marks the region where $|\vec{l}| \leq n + d - 1$, fulfilling Eq. 26. The blue dots represent the gridpoints of the respective subspaces. Finally, the dashed black lines indicate the hierarchical increment spaces for constant $|\vec{l}|$.

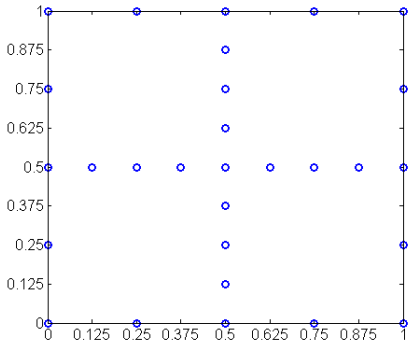

Figure 18: Sparse grid space $V_4^{S,CC}$ in 2 dimensions, constructed according to Eq. 26 from the increments displayed in Fig. 17. Note that it contains only 29 support nodes, whereas a full grid would consist of 225 points.

B Hierarchical Integration

The sparse grid approach can also be used for high-dimensional numerical integration, e.g. for the computation of expectations [12, 25, 28].

Starting from Eq. 28, the mean from the interpolant can be evaluated as

d	$\lvert V_n \rvert$	$\lvert V_{0,n}^S \rvert$	$\lvert V_n^{S,CC} \rvert$
1	15	15	9
2	225	49	29
3	3375	111	69
4	50'625	209	137
5	759'375	351	241
10	$5.77 \cdot 10^{11}$	2'001	1'581
15	$4.37 \cdot 10^{17}$	5'951	5'021
20	$3.33 \cdot 10^{23}$	13'201	11'561
30	$1.92 \cdot 10^{35}$	41'601	37'941
40	$1.11 \cdot 10^{47}$	95'201	88'721
50	$6.38 \cdot 10^{58}$	182'001	171'901
100	>Googol	1'394'001	1'353'801

Table 7: Number of gridpoints for several different types of grids of level 4. First column: dimension; second column: full grid; third column: 'classical' L_2 optimal sparse grid with no points at the boundaries; last column: 'Clenshaw-Curtis' sparse grid.

follows:

$$\mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \int_\Omega \phi_{\vec{l},\vec{i}}(\vec{x}) d\vec{x}, \tag{67}$$

where we assume for simplicity that the probability density is 1 on $\Omega = [0,1]^d$. Using in the example here the basis functions described in Eqs. 65 and 66, the one dimensional integral can now be computed analytically [25]:

$$\int_0^1 \phi_{l,i}\left(x\right) dx = \begin{cases} 1, & \text{if } l = 1 \\ \frac{1}{4} & \text{if } l = 2 \\ \\ 2^{1-l} & \text{else} \end{cases} \tag{68}$$

The multi-dimensional integrals are therefore again the product of one dimensional integrals. Following [25], we denote $\int_\Omega \phi_{l,i}\left(\vec{x}\right) d\vec{x} = J_{\vec{l},\vec{i}}$. Now, we can rewrite Eq. 67 by

$$\mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot J_{\vec{l},\vec{i}}. \tag{69}$$

Expression 69 states that the mean is an arithmetic sum of the product of the hierarchical surpluses and the integral weights at each point of the interpolant [25]. All other sorts of integrals can be computed in a similar fashion.

References

[1] Volker Barthelmann, Erich Novak, and Klaus Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12:273–288, 2000.

[2] J. Brumm and M. Grill. Computing equilibria in dynamic models with occasionally binding constraints. *Journal of Economic Dynamics and Control, accepted for publication.*

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

[4] H.-J. Bungartz and S. Dirnstorfer. Multivariate quadrature on adaptive sparse grids. *Computing*, 71:89–114, 2003.

[5] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.

[6] Yongyang Cai, Kenneth L. Judd, Greg Thain, and Stephen J. Wright. Solving dynamic programming problems on a computational grid. NBER Working Papers 18714, National Bureau of Economic Research, Inc, January 2013.

[7] Wouter J. Den Haan, Kenneth L. Judd, and Michel Juillard. Computational suite of models with heterogeneous agents ii: Multi-country real business cycle models. *Journal of Economic Dynamics and Control*, 35(2):175–177, February 2011.

[8] J. J. Dongarra and A. J. van der Steen. High-performance computing systems: Status and outlook. *Acta Numerica*, 21:379–474, 4 2012.

[9] Giulio Fella. A generalized endogenous grid method for non-smooth and non-concave problems. *Review of Economic Dynamics, In Press*, 2013.

[10] J. Garcke and M. Griebel. *Sparse Grids and Applications*. Lecture Notes in Computational Science and Engineering Series. Springer-Verlag GmbH, 2012.

[11] Alan Genz. Testing multidimensional integration routines. In *Proc. of international conference on Tools, methods and languages for scientific and engineering computation*, pages 81–94, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[12] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numerical Algorithms*, 18:209–232, 1998.

[13] M. Griebel. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179, 1998. also as Proceedings Large-Scale Scientific Computations of Engineering and Environmental Problems, 7. June - 11. June, 1997, Varna, Bulgaria, Notes on Numerical Fluid Mechanics 62, Vieweg-Verlag, Braunschweig, M. Griebel, O. Iliev, S. Margenov and P. Vassilevski (editors).

[14] C. Hager, S. Hüeber, and B. Wohlmuth. Numerical techniques for the valuation of basket options and its greeks. *J. Comput. Fin.*, 13(4):1–31, 2010.

[15] Thomas Hintermaier and Winfried Koeniger. The method of endogenous gridpoints with occasionally binding constraints among endogenous variables. *Journal of Economic Dynamics and Control*, 34(10):2074–2088, October 2010.

[16] John D Jakeman and Stephen G Roberts. Local and dimension adaptive sparse grid interpolation and quadrature. *arXiv preprint arXiv:1110.0010*, 2011.

[17] Gabriele Jost, Barbara Chapman, and Ruud van der Pas. *Using OpenMP - Portable Shared Memory Parallel Programming.* MIT Press, 2007.

[18] Kenneth Judd, Lilia Maliar, Rafael Valero, and Serguei Maliar. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. Working Papers. Serie AD 2013-06, Instituto Valenciano de Investigaciones Económicas, S.A. (Ivie), September 2013.

[19] Kenneth L Judd. *Numerical methods in economics.* The MIT press, 1998.

[20] Michel Juillard and Sébastien Villemot. Multi-country real business cycle models: Accuracy tests and test bench. *Journal of Economic Dynamics and Control*, 35(2):178–185, February 2011.

[21] Andreas Klimke and Barbara Wohlmuth. Algorithm 847: Spinterp: piecewise multilinear hierarchical sparse grid interpolation in matlab. *ACM Trans. Math. Softw.*, 31(4):561–579, December 2005.

[22] Robert Kollmann, Serguei Maliar, Benjamin A. Malin, and Paul Pichler. Comparison of solutions to the multi-country real business cycle model. *Journal of Economic Dynamics and Control*, 35(2):186 – 202, 2011. Computational Suite of Models with Heterogeneous Agents II: Multi-Country Real Business Cycle Models.

[23] Dirk Krueger and Felix Kubler. Computing equilibrium in OLG models with stochastic production. *Journal of Economic Dynamics and Control*, 28(7):1411 – 1436, 2004.

[24] Deborah J. Lucas. Asset pricing with undiversifiable income risk and short sales constraints: Deepening the equity premium puzzle. *Journal of Monetary Economics*, 34(3):325 – 341, 1994.

[25] Xiang Ma and Nicholas Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *J. Comput. Phys.*, 228(8):3084–3113, May 2009.

[26] Benjamin A Malin, Dirk Krüger, and Felix Kübler. Solving the multi-country real business cycle model using a smolyak-collocation method. *Journal of Economic Dynamics and Control*, 35(2):229–239, October 2010.

[27] Alin Murarasu, Josef Weidendorfer, Gerrit Buse, Daniel Butnaru, and Dirk Pflüeger. Compact data structure and parallel alogrithms for the sparse grid technique. In *16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2011.

[28] Erich Novak and Klaus Ritter. High dimensional integration of smooth functions over cubes. *Numerische Mathematik*, 75:79–97, 1996.

[29] Dirk Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems.* PhD thesis, München, August 2010.

[30] Dirk Pflüger. Spatially adaptive refinement. In Jochen Garcke and Michael Griebel, editors, *Sparse Grids and Applications*, Lecture Notes in Computational Science and Engineering, pages 243–262, Berlin Heidelberg, October 2012. Springer.

[31] Dirk Pflüger, Benjamin Peherstorfer, and Hans-Joachim Bungartz. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity*, 26(5):508—-522, October 2010. published online April 2010.

[32] Rolf Rabenseifner, Georg Hager, and Gabriele Jost. Hybrid mpi/openmp parallel programming on clusters of multi-core smp nodes. In *Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, PDP '09, pages 427–436, Washington, DC, USA, 2009. IEEE Computer Society.

[33] Thomas Rauber and Gudula Rünger. *Parallel Programming: for Multicore and Cluster Systems*. Springer, 2010 edition, March 2010.

[34] Olaf Schenk, Matthias Bollhöfer, and Rudolf A. Römer. On large-scale diagonalization techniques for the anderson model of localization. *SIAM Rev.*, 50(1):91–112, February 2008.

[35] Olaf Schenk, Andreas Wächter, and Michael Hagemann. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Comput. Optim. Appl.*, 36(2-3):321–341, April 2007.

[36] Anthony Skjellum, William Gropp, and Ewing Lusk. *Using MPI*. MIT Press, 1999.

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

[38] Chris I. Telmer. Asset-pricing puzzles and incomplete markets. *The Journal of Finance*, 48(5):1803–1832, 1993.

[39] Andreas Waechter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, May 2006.

[40] Christoph 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.