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

Johannes Brumm
Karlsruhe Institute of Technology
johannes.brumm@gmail.com

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

May 22, 2017

Abstract

We present a flexible and scalable method for computing global solutions of high-dimensional stochastic dynamic models. Within a time iteration or value function iteration setup, we interpolate functions using an adaptive sparse grid algorithm. With increasing dimensions, sparse grids grow much more slowly than standard tensor product grids. Moreover, adaptivity adds a second layer of sparsity, as grid points are added only where they are most needed, for instance in regions with steep gradients or at non-differentiabilities. To further speed up the solution process, our implementation is fully hybrid parallel, combining distributed and shared memory parallelization paradigms, and thus permits an efficient use of high-performance computing architectures. To demonstrate the broad applicability of our method, we solve two very different types of dynamic models: first, high-dimensional international real business cycle models with capital adjustment costs and irreversible investment; second, multiproduct menu-cost models with temporary sales and economies of scope in price setting.

Keywords: Adaptive Sparse Grids, High-Performance Computing, International Real Business Cycles, Menu Costs, Occasionally Binding Constraints.

JEL Classification: C63, E30, F44.

*We thank the co-editor and four anonymous referees for their extremely valuable comments and suggestions. We are very grateful to Felix Kubler for helpful discussions and support. We thank Fernando Alvarez, Daniel Harenberg, Ken Judd, Francesco Lippi, Karl Schmedders, Olaf Schenk, Mirko Wiederholt, and seminar participants at the University of Zurich, Stanford University, the University of Chicago, Argonne National Laboratory, Cornell University, USI Lugano, CEF 2013 in Vancouver, the BYU Computational Public Economics Conference 2014, the Dynare Conference 2015 in Brussels, St. Louis FED, the Consumer Financial Protection Bureau, the IMF, the University of Geneva, the University of Copenhagen, Goethe University Frankfurt, and the University of Bonn for valuable comments. Moreover, we thank Xiang Ma for very instructive e-mail discussions regarding the workings of adaptive sparse grids. This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID s555. Johannes Brumm gratefully acknowledges financial support from the ERC. Simon Scheidegger gratefully acknowledges support from the Hoover Institution and PASC.

1 Introduction

This paper demonstrates that global solutions of economic models with substantial heterogeneity and frictions can be computed accurately and fast. We achieve this by building a highly parallel iterative algorithm based on adaptive sparse grids (ASGs). This method can handle high-dimensional problems even if they exhibit non-smooth behavior such as non-differentiable policy or value functions. Using modern high-performance computing facilities, we are able to compute global solutions for dynamic stochastic models with up to 100 dimensions, and also for 20-dimensional models with non-differentiabilities.

Such a powerful global solution method is required because many economic phenomena can only be captured if one goes beyond considering the local dynamics around steady states and at the same time takes into account the interdependence between different firms, sectors, or countries. In the 1990s, more and more economists started to take these concerns seriously by using global solution methods[1] and by including various kinds of heterogeneity in their models (see, e.g., [64, 50, 1, 48]). However, solving for global solutions of general-equilibrium rational-expectations models with substantial heterogeneity is very costly in terms of computation time and storage requirements. Standard grid-based algorithms suffer from the curse of dimensionality: starting with a 1-dimensional discretization scheme that employs M grid points, a straightforward extension to d dimensions leads to M^d points; thus the total number of points grows exponentially in the dimension.

Sparse grids (SGs) are able to alleviate this curse by reducing the number of points from the order $\mathcal{O}\left(M^d\right)$ to $\mathcal{O}\left(M \cdot (\log M)^{d-1}\right)$ with only slightly deteriorated accuracy if the underlying function is sufficiently smooth (see, e.g., [16], and references therein). In contrast to previous applications of sparse grid methods in economics, which all rely on global polynomials (see, e.g., [63, 66, 41]), we use hierarchical basis functions with an adaptive grid refinement strategy (as in [51]). Based on the interpolation coefficient of a given function within the hierarchy of basis functions, an automatic grid adaptation algorithm decides whether the grid is to be refined locally. If it is, then $2d$ basis functions of the next finer level are added, their coefficients are computed, and depending on their values the grid might be refined even further. This type of local adaptivity has obvious advantages when applied to models where policy or value functions exhibit non-differentiabilities. In addition, we show that for smooth models too, in particular if they are very-high-dimensional, ASGs can substantially improve upon standard SGs. Adaptive sparse grids with hierarchical local basis functions thus offer the promise of efficient and accurate solutions to high-dimensional non-smooth economic problems and to very-high-dimensional smooth problems.

In order to speed up the solution process, we make efficient use of state-of-the-art high-performance computing (HPC) facilities (see [24]). A typical contemporary HPC hardware design features thousands of compute nodes each consisting of multiple central processing units (CPUs) with attached graphic processing units (GPUs). Our hybrid parallelization scheme therefore combines distributed memory parallelization among different nodes with shared memory parallelization inside each node, and partially offloads the function evaluations to GPUs. In this way, we are able to use at least $2,000$ compute nodes efficiently. Note that our parallelization strategy, being generic, can also be applied to other (sparse) grid methods and is therefore a contribution in itself. It is, however, one of the highlights of our paper that this parallelization scheme even works for the non-regular grid structure of our locally adaptive

[1]We use the term "global solution" for a solution that is computed using equilibrium conditions at many points in the state space of a dynamic model—in contrast to a "local solution", which rests on a local approximation around a steady state of the model. For a method that computes such a global solution, we use the term "global solution method". This use of the word "global" is not to be confused with its use in the term "global optimization method", which refers to a method that aims to find a global optimum.

method.

To demonstrate the performance of our method in solving high-dimensional economic problems, we first solve an international real business cycle (IRBC) model in which the stocks of country specific capital are subject to adjustment costs and investment is irreversible. Without the occasionally binding constraints arising from irreversible investment, this model was used in a comprehensive coordinated study that documented the performance of alternative solution methods (see [23, 42, 46]). Like some of these established methods, we use a time iteration[2] procedure to solve for an equilibrium that is recursive in the capital stocks and the productivity levels of all countries. Within each time iteration step we use an adaptive sparse grid algorithm to interpolate the policy functions. Using this method, we solve up to 100-dimensional versions of the IRBC model without irreversibility, compared to 20 dimensions reported in the comparison study cited above. Furthermore, we show that ASGs are much more flexible than SGs when aiming for a certain accuracy in a reasonable time. The strongest comparative advantage of our algorithm, however, lies in solving models that exhibit non-smooth behavior. For the IRBC model with irreversibility constraints we are able to compute accurate solutions for up to 20-dimensional cases—despite the non-differentiabilities in policy functions induced by the occasionally binding constraints. The adaptivity of the grid now ensures that grid points are placed close to the kinks while the grid remains coarse where policy functions are smoother. Other important potential applications where kinks in policy functions naturally occur are models with collateral constraints on borrowing (see, e.g., [44, 13]) or with the zero lower bound on the nominal interest rate (see, e.g., [27, 52]).

To emphasize the broad applicability of ASGs in dynamic economics, we additionally consider an application that differs in important ways from the IRBC model: a menu-cost model where multiproduct firms face economies of scope in price setting and have the option to set temporary prices (as in Midrigan [54]). Due to the fixed costs of adjusting prices, firms make discrete choices. As a consequence, we have to solve the model by value function iteration and have to keep track of the values associated with each of the choices possible. In this context, ASGs are particularly suitable for interpolation, as discrete choices in discrete-time models typically induce kinks in value functions. The menu-cost model thus illustrates that our method can also be combined with value function iteration and can handle discrete choices, which feature in many important economic applications—for instance models of consumer default or sovereign default (see, e.g., [22, 49, 6]). When it comes to the ongoing debate about monetary (non-)neutrality in menu-cost models (see, e.g., [31, 43, 3]), our application shows that raising the number of products relative to Midrigan [54] results in a better match of the high kurtosis of empirical price changes and at the same time further increases the real effects of monetary policy. These results are in line with findings from menu-cost models without temporary price changes. In such models, Bhattarai and Schoenle [10] obtain numerical results for up to three goods, while Alvarez and Lippi [4] and Alvarez et al. [3] provide quasi-analytical solutions for an arbitrary number of goods. Our paper thus confirms their results in the presence of temporary price changes and offers a method that might, in future research, be fruitfully applied to substantial extensions of currently used menu-cost models.

The remainder of this paper is organized as follows: Section 2 provides a brief overview of the related literature. In Section 3, we explain the construction of ASGs and provide a simple test case. In Section 4, we embed ASG interpolation in a time iteration algorithm to solve high-dimensional IRBC models, we discuss how hybrid parallelization can speed up the computations, and we report the performance of this algorithm. Section 5 presents the application of our method to the menu-cost model. Section 6 concludes.

[2]We use the term "time iteration" for an algorithm that iteratively updates policy functions by solving the period-to-period first-order equilibrium conditions (see, e.g., [39], Sec. 17.8).

2 Related Literature

Our paper is closely related to five strands of the literature. First, sparse grid methods in general; second, their application to dynamic economic models; third, solution methods for models with occasionally binding constraints; fourth, simulation-based methods for high-dimensional problems; finally, parallel computing in economics.[3]

The sparse grid construction we are using was introduced by Zenger [67] for the solution of partial differential equations. However, the underlying principle, a sparse tensor product decomposition, goes back to the seminal work of Smolyak [63]. Sparse grids have been applied to a wide range of different research fields, including physics, visualization, and data mining (see, e.g., [11, 16, 28, 35, 38, 55]). In mathematical finance, Hager et al. [33] and Bungartz et al. [17] use sparse grids for the valuation of basket options; in econometrics, Heiss and Winschel [36] integrate likelihood functions on sparse grids.

Using the original formulation of Smolyak [63], Krueger and Kubler [47] were the first to solve dynamic economic models using sparse grids, while Winschel and Kraetzig [66] embed this method into a Bayesian estimation framework. Judd et al. [41] propose an implementation of the Smolyak algorithm 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 three papers all rely on global polynomials as basis functions, which generally fail to capture the local behavior of policy functions that are not sufficiently smooth. Furthermore, these papers do not provide parallel implementations of their time iteration algorithms.

There are several papers that develop methods for solving dynamic economic models with occasionally binding constraints. The endogenous grid point method proposed by Carroll [21] defines a grid on the next period's variables resulting in an "endogenous" grid on the current period variables, a trick that can avoid the root-finding procedure for solving the Euler equation. This method can also place grid points at non-differentiabilities, as extensions by Barillas and Villaverde [7], Hintermaier and Koeniger [37], and Fella [26] show. An alternative method that is more flexible, but does not avoid root-finding, is the adaptive simplicial interpolation by Brumm and Grill [12], which puts additional grid points at kinks and interpolates on the resulting irregular grid using Delaunay interpolation. While all these methods are able to approximate the non-differentiabilities induced by occasionally binding constraints very precisely for up to three continuous state variables, we can handle such models with state spaces of up to 20 dimensions.

When it comes to solving high-dimensional models, another widely used approach is to interpolate policy functions only on a simulation-based approximation to the ergodic set. Unfortunately, equilibrium conditions outside this set might then be substantially violated; moreover, such a procedure can be numerically unstable as interpolation coefficients on simulated data are often highly correlated with one another. Judd et al. [40] propose a generalized stochastic simulation method that addresses the instability issue by employing various tools, like principle component regressions, that can handle ill-conditioned problems. Such simulation-based methods can be combined effectively with SGs, as Judd et al. [41] show, and could be so as well with ASGs.

Finally, our paper is part of the emergent literature on parallel computing applications in economics. Aldrich et al. [2] use GPUs, which are special purpose devices that can speed up only particular sections of a code. Cai et al. [19] solve large-scale dynamic programming problems with a distributed memory parallelized code. Such implementations have limited scalability on contemporary hardware, as their memory consumption and communication overhead both tend to grow substantially as the number of processes increases. Cai et al. [18] apply a high

[3]We cite and discuss the related literature on our applications, in particular the menu-cost model, in the introduction and the respective sections.

throughput computing (HTC) parallelization scheme to solve dynamic programming problems. HTC can distribute embarrassingly parallel tasks very efficiently, yet it is not able to guarantee a fast time-to-solution if synchronization is needed regularly. In contrast to all these contributions, our implementation is fully hybrid parallel and thus able to use the available computation power of contemporary supercomputers in an efficient way.

3 From Full Grids to Adaptive Sparse Grids

This section proceeds in three main steps. First, we present piecewise linear hierarchical basis functions in one dimension and extend such bases to multiple dimensions via a tensor product construction. Second, we show how "classical" sparse grids can alleviate the curse of dimensionality. Third, we explain how the hierarchical structure of the basis functions and the associated sparse grid can be used to build an adaptation procedure that can better capture the local behavior of the functions to be interpolated. Additionally, we provide an example of a function with steep gradients and non-differentiabilities where ASGs outperform "classical" SGs by far.

3.1 Hierarchical Basis Functions

The sparse grid method introduced below is based on a hierarchical decomposition of the underlying approximation space [67, 16, 28]. Such a hierarchical structure is convenient for local adaptivity (see Sec. 3.3) and for the use of parallel computing (see Sec. 4.3); moreover, it is essential when both features are combined (see Sec. 4.3). To explain this hierarchical structure we assume for simplicity that we interpolate a function $f : \Omega \to \mathbb{R}$ with $\Omega = [0,1]^d$ that vanishes at the boundary of its domain (i.e., $f|_{\partial\Omega} = 0$). Differing domains can be handled by rescaling, and non-zero boundaries can be treated as explained in Appendix A.

In the 1-dimensional case, we interpolate on an equidistant grid Ω_l on $\Omega = [0,1]$ with mesh size $h_l := 2^{-l}$ and grid points $x_{l,i} := i \cdot 2^{-l}$, where $l \in \mathbb{N}, i \in \{0,1,...,2^l\}$. For instance, let $l = 2$, $i = 3$, then $x_{l,i} = 3 \cdot 2^{-2} = 0.75$. To generate a hierarchical basis for interpolating f, we rely on the hat function

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

to define for each grid point $x_{l,i}$ a basis function

$$\phi_{l,i}(x) := \phi\left(\frac{x - i \cdot h_l}{h_l}\right) \tag{2}$$

with $\phi_{l,i}(x_{l,i}) = 1$ and support $[x_{l,i} - h_l, x_{l,i} + h_l]$. The index l represents the refinement level, and the index i numbers the basis functions within a given level (see Fig. 1). To order the basis functions, we define the so-called hierarchical increment spaces

$$W_l := \operatorname{span}\{\phi_{l,i} : i \in I_l\}, \quad I_l = \{i \in \mathbb{N}, 1 \leq i \leq 2^l - 1, i \text{ odd}\}, \tag{3}$$

where the index set I_l ensures that a) the different basis functions $\phi_{l,i}$ have mutually disjoint support within a level l, and b) the support of a level l function nests the supports of two level $l+1$ functions. The top panels of Fig. 1 show the first three levels of these hierarchical, piecewise linear basis functions. The direct sum of the hierarchical increment spaces results in the function space

$$V_l = \bigoplus_{k \leq l} W_k. \tag{4}$$

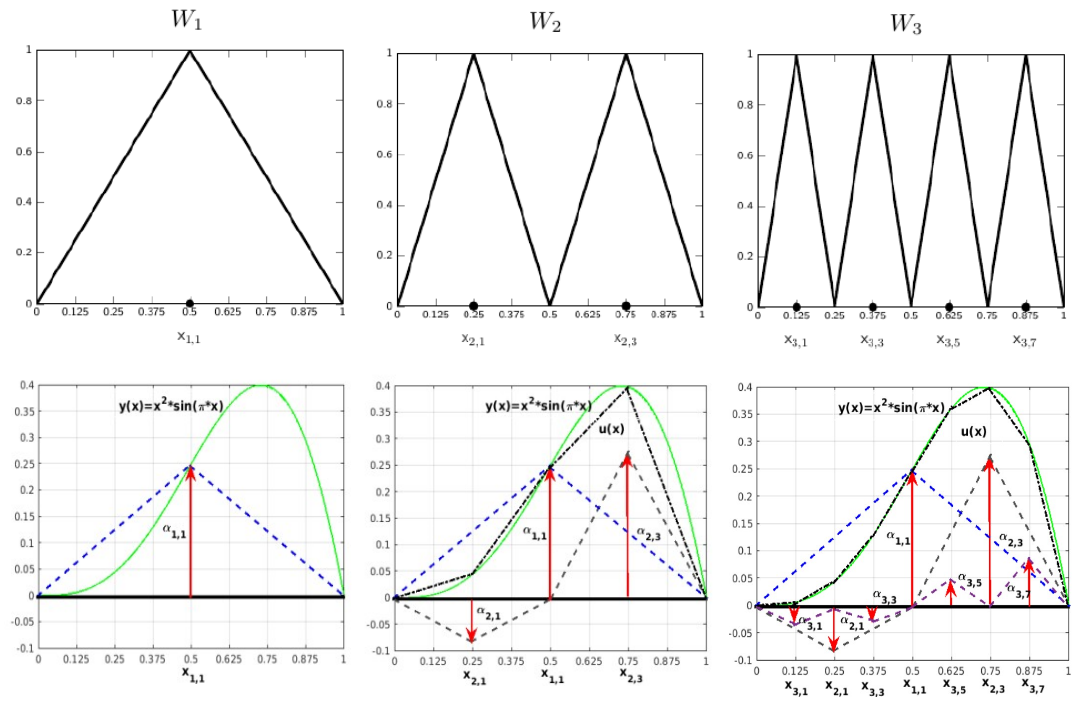

Figure 1: The top panels show the basis functions of V_3, going from level 1 to level 3. The lower panels show the construction of the interpolant u (dashed-dotted line) interpolating $y(x) = x^2 \cdot \sin(\pi \cdot x)$ (green, solid line) with hierarchical linear basis functions of levels 1, 2, and 3. The hierarchical surpluses $\alpha_{l,i}$ associated with the respective basis functions are indicated by (red) arrows.

Using this basis, a function f is approximated by a unique $u \in V_l$ with coefficients $\alpha_{k,i} \in \mathbb{R}$:

$$f\left(x\right) \approx u\left(x\right) = \sum_{k=1}^{l} \sum_{i \in I_k} \alpha_{k,i} \cdot \phi_{k,i}\left(x\right). \tag{5}$$

The coefficients $\alpha_{k,i}$ in Eq. 5 are called *hierarchical surpluses*. Due to the nested structure of the hierarchical grid they can easily be determined: $\alpha_{k,i}$ simply corrects the interpolant of level $k-1$ at the point $x_{k,i}$ to the actual value of $f\left(x_{k,i}\right)$, as displayed (by arrows) in the lower panels of Fig. 1.

The 1-dimensional hierarchical basis from above can be extended to a d-dimensional basis on the unit cube $\Omega = [0,1]^d$ by a tensor product construction. Our notation naturally extends to the d-dimensional case. 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 and the spatial position of a d-dimensional grid point $\vec{x}_{\vec{l},\vec{i}}$. Using this notation, we can define the full grid $\Omega_{\vec{l}}$ on Ω with mesh size $\left(2^{-l_1}, ..., 2^{-l_d}\right)$ and a generic grid point $\vec{x}_{\vec{l},\vec{i}} := (x_{l_1,i_1}, ..., x_{l_d,i_d})$, where $x_{l_t,i_t} := i_t \cdot 2^{-l_t}$, $i_t \in \{0, 1, ..., 2^{l_t}\}$, and $t \in \{1, \dots, d\}$. For each grid point, $\vec{x}_{\vec{l},\vec{i}}$, an associated piecewise d-linear basis function $\phi_{\vec{l},\vec{i}}\left(\vec{x}\right)$ is defined as the product of the 1-dimensional basis functions:

$$\phi_{\vec{l},\vec{i}}\left(\vec{x}\right) := \prod_{t=1}^{d} \phi_{l_t,i_t}\left(x_t\right). \tag{6}$$

Similar to the 1-dimensional case, the hierarchical increment spaces are defined by

$$W_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}, \quad I_{\vec{l}} := \{\vec{i} : 1 \leq i_t \leq 2^{l_t} - 1, i_t \text{ odd}, 1 \leq t \leq d\}. \tag{7}$$

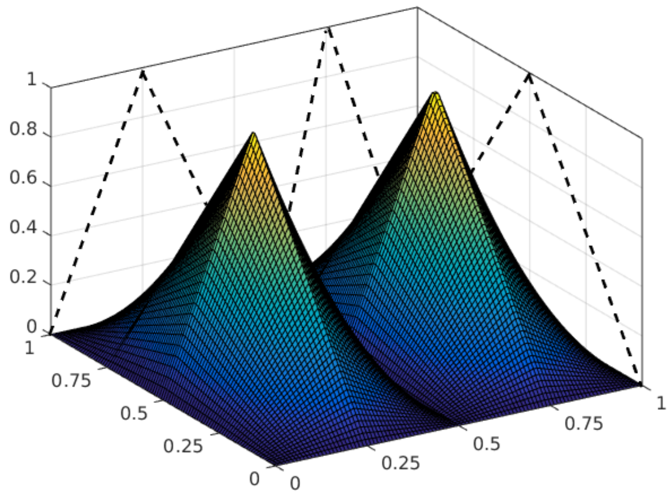

Figure 2: The 2-dimensional basis functions of the hierarchical increment space $W_{(2,1)}$.

An example of a 2-dimensional increment space—namely, $W_{(2,1)}$, is shown in Fig. 2. Extending the procedure followed in the 1-dimensional case, we take the direct sum of the hierarchical increment spaces up to certain levels $\vec{l} = (l_1, \ldots, l_d)$ to obtain the function spaces

$$V_{\vec{l}} := \bigoplus_{k_1=1}^{l_1} \cdots \bigoplus_{k_d=1}^{l_d} W_{\vec{k}}. \tag{8}$$

In line with the literature, we define the isotropic space $V_n := V_{(n,\ldots,n)}$ as a special case of Eq. 8. Consequently, the hierarchical increment spaces $W_{\vec{l}}$ are related in the following way to the function space V_n of piecewise d-linear functions with mesh width $h_n = 2^{-n}$ in each dimension:

$$V_n := \bigoplus_{|\vec{l}|_\infty \leq n} W_{\vec{l}}, \tag{9}$$

where $|\vec{l}|_\infty = \max_{1\leq t\leq d} l_t$. The full Cartesian grid V_n has $(2^n - 1)^d$ grid points; its decomposition into hierarchical increment spaces is depicted in the top panel of Fig. 3. The interpolant of f—namely, $u(\vec{x}) \in V_n$, can uniquely be represented by

$$f(\vec{x}) \approx u(\vec{x}) = \sum_{|\vec{l}|_\infty \leq n} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|\vec{l}|_\infty \leq n} u_{\vec{l}}(\vec{x}), \tag{10}$$

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

For a sufficiently smooth function f (stated more precisely in Sec. 3.2 and Appendix B) and its interpolant $u \in V_n$, we obtain an asymptotic error decay of $\|f(\vec{x}) - u(\vec{x})\|_{L_2} \in \mathcal{O}\left(2^{-2n}\right)$, but at the cost of $\mathcal{O}\left(2^{nd}\right)$ grid points, encountering the so-called *curse of dimensionality* (see, e.g., [16]). This is a prohibitive obstacle for the accurate solution of problems with more than four or five dimensions. For instance, a 10-dimensional problem with a resolution of 15 points in each dimension (i.e., V_4) needs $5.8 \cdot 10^{11}$ coefficients, which already brings us to the capacity limits of today's most advanced computer systems (see, e.g., [24]).

3.2 Classical Sparse Grids

To alleviate the curse of dimensionality we need to construct approximation spaces that are better than V_n in the sense that the same number of grid points leads to higher accuracy. The

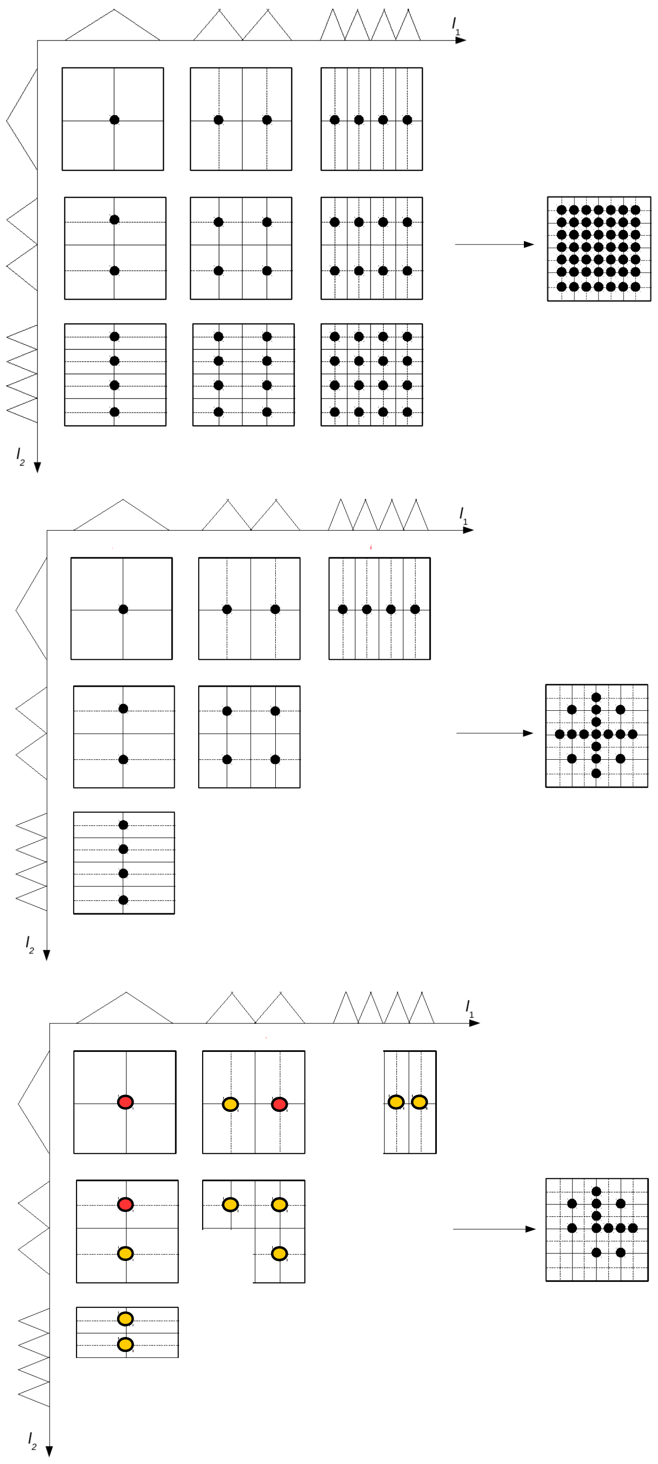

Figure 3: The top panel shows the construction of a full "Cartesian" grid (see Eq. 8). The middle panel shows the construction of a "classical" sparse grid (see Eq. 12). The bottom panel shows the construction of an adaptive sparse grid. Dark (red) dots represent points where the grid is refined, whereas light (yellow) dots indicate points where the grid is not further refined.

"classical" sparse grid construction arises from a "cost-benefit" analysis in function approximation (see Appendix B for details). For functions with bounded second-order mixed derivatives, it can be shown (see, e.g., [16]) that the hierarchical coefficients $\alpha_{\vec{l},\vec{i}}$ decay rapidly as the grid level $\vec{l}$ increases:

$$|\alpha_{\vec{l},\vec{i}}| = \mathcal{O}\left(2^{-2|\vec{l}|_1}\right), \tag{11}$$

where $|\vec{l}|_1 = \sum_{t=1}^{d} l_t$. The strategy for constructing a sparse grid is to leave out those subspaces within the full grid space V_n that only contribute little to the interpolant. Proposition 2 of Appendix B states that an optimization that minimizes the approximation error (in the L_2- or L_∞-norm) given a fixed number of grid points leads to the *sparse grid* space of level n, defined by

$$V_n^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}. \tag{12}$$

In contrast to the full grid, where the *maximum* of the grid refinement levels across dimensions is restricted (see Eq. 9), now it is the *sum* of these that is restricted. The middle panel of Fig. 3 displays the construction of V_3^S. The number of grid points required by the space V_n^S is given by

$$|V_n^S| = \mathcal{O}\left(2^n \cdot n^{d-1}\right), \tag{13}$$

which is a significant reduction compared to $\mathcal{O}\left(2^{nd}\right)$ for the full grid space V_n (see Appendix B for details and Tab. 1 for examples). Expressed in terms of the number of grid points in one dimension $M = 2^n - 1$, Eq. 13 implies

$$|V_n^S| = \mathcal{O}\left(M \cdot \log(M)^{d-1}\right), \tag{14}$$

compared to $\mathcal{O}\left(M^d\right)$ for the Cartesian grid. In analogy to Eq. 10, a function $f \in V_n^S \subset V_n$ is now approximated by

$$f(\vec{x}) \approx u(\vec{x}) = \sum_{|\vec{l}|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|\vec{l}|_1 \leq n+d-1} u_{\vec{l}}(\vec{x}), \tag{15}$$

where $u_{\vec{l}} \in W_{\vec{l}}$. The asymptotic accuracy of the interpolant deteriorates only slightly from $\mathcal{O}\left(2^{-2n}\right)$ in the case of the full grid to $\mathcal{O}\left(2^{-2n} \cdot n^{d-1}\right)$, as shown in [16, 28]. This demonstrates why SGs are so well suited to high-dimensional problems. In contrast to full grids, their size increases only moderately with dimensions, while they are only slightly less accurate than full grids.

Finally, note that Eq. 13 also holds for SGs with non-vanishing boundaries (see Appendix A) and that sparse grid methods with hierarchical basis functions are not restricted to the linear hat functions presented above. Several other basis functions are possible, including piecewise polynomials and wavelets (see [16, 58] and [32], respectively). A notable advantage of wavelets is that they form a Riesz basis, implying that their surpluses are sharper indicators of the local approximation error, and that a refinement strategy based on wavelets can thus be more accurate for a fixed number of points. A major drawback of wavelets is that determining their coefficients requires solving systems of equations. Overall, this leads to significantly higher computational costs, in particular in high dimensions. We focus on linear hat functions as they are simple and convenient, in particular when it comes to adaptive refinement procedures, which we present next.

3.3 Adaptive Sparse Grids

The sparse grid structure introduced in the previous section defines an a priori selection of grid points that is optimal for functions with bounded second-order mixed derivatives. However,

Dimension d	Full Grid $\lvert V_4 \rvert$	Sparse Grid $\lvert V_4^S \rvert$
1	15	15
2	225	49
3	3,375	111
4	50,625	209
5	759,375	351
10	$5.77 \cdot 10^{11}$	2,001
20	$3.33 \cdot 10^{23}$	13,201
50	$6.38 \cdot 10^{58}$	182,001
100	>Googol	1,394,001

Table 1: Number of grid points for increasing dimensions of a full Cartesian grid (second column) and "classical" sparse grid (third column), keeping the refinement level fixed at $l = 4$.

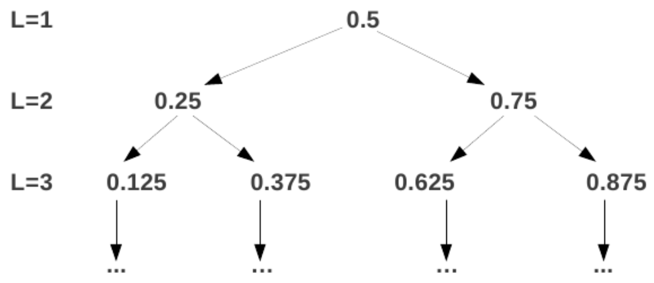

Figure 4: 1-dimensional tree-like structure of a "classical" sparse grid (see Sec. 3.2) for the first three hierarchical levels.

in many applications—including economic models with occasionally binding constraints (see Sec. 4) or discrete choices (see Sec. 5)—these prerequisites are not met as the functions of interest display kinks. Consequently, the sparse grid methods outlined thus far may fail to provide good approximations. Therefore, we need a method for efficiently approximating functions that do not fulfill the smoothness conditions underlying the theory of sparse grids (see Eq. 38 in Appendix B). A very effective strategy for achieving this goal is to adaptively refine the sparse grid in regions of high curvature and spend less points in regions of low function variation (see, e.g., [59, 58, 15, 51]). While there are various ways of refining sparse grids (see, e.g., [58], and references therein), we focus on the working principle of the algorithm that we use (see [51, 58] for details).

When approximating a function as a sum of piecewise linear basis functions, the main contributions to the interpolant most likely stem from comparatively few terms with big surpluses (see the lower panels of Fig. 1). The logic of the refinement strategy we employ is therefore to monitor the size of the hierarchical surpluses and refine the grid where those terms are larger than some threshold. Technically, the adaptive sparse grid refinement can be built on top of the hierarchical grid structure. Let us first consider the 1-dimensional case. The equidistant grid points form a tree-like data structure as displayed in Fig. 4. Going from one level to the next, we see that for each *parent* grid point there are two *children*, which are neighboring points of the parent. For example, the point 0.5 from level $l = 1$ is the parent of the points 0.25 and 0.75 from level $l = 2$. In the d-dimensional case, each grid point has two children in each dimension—thus $2d$ in total. In order to adaptively refine the grid, we use the hierarchical surpluses as an error indicator to detect the smoothness of the solution. We refine those hierarchical basis functions $\phi_{\vec{l},\vec{i}}$ that have a hierarchical surplus, $\alpha_{\vec{l},\vec{i}}$, that satisfies

$$|\alpha_{\vec{l},\vec{i}}| \geq \epsilon, \tag{16}$$

for a so-called *refinement threshold* $\epsilon \geq 0$. Whenever this criterion is satisfied, the children of the current point are added to the sparse grid. The construction of a 2-dimensional adaptive sparse grid is illustrated in the lowest panel of Fig. 3. This figure neatly demonstrates that the adaptivity can add a second layer of sparsity on top of the sparse grid structure.

Note that in our applications in Secs. 4 and 5, as in many economic applications, we interpolate several policies or value functions on one grid—that is to say, we interpolate a function

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

with $m > 1$. Therefore, we get m surpluses at each grid point and we thus have to replace the refinement criterion in Eq. 16 by

$$g\left(\alpha^1_{\vec{l},\vec{i}}, \ldots, \alpha^m_{\vec{l},\vec{i}}\right) \geq \epsilon, \tag{17}$$

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, which we will use in Sec. 4.4. However, depending on the application, more sophisticated criteria might need to be imposed for an efficient approximation. In our application in Sec. 5, the proper choice of g turned out to be non-trivial, yet essential for achieving efficient approximations.

Analytical Example

We now demonstrate the ability of the adaptive sparse grid algorithm to efficiently interpolate functions that exhibit steep gradients and kinks. As an analytical example of a (non-smooth) function $f : [0,1]^d \to \mathbb{R}$ we chose

$$f(x,y) = \frac{1}{|0.5 - x^4 - y^4| + 0.1}, \tag{18}$$

which is from a representative suite of test problems by [29]. As this function does not vanish at the boundary, and to show how to handle such functions, we use basis functions that can deal with non-zero boundaries (see Eq. 15 in Appendix A). We generate sparse grid as well as adaptive sparse grid approximations to the function f and measure their accuracy as follows: We randomly generate $N = 1,000$ test points from a uniform distribution on $[0,1]^2$, and compute the maximum error and the L_2 error, which are, respectively, given by

$$\max_{i=1,\ldots,N} |f(\vec{x_i}) - u(\vec{x_i})|, \quad \left(\frac{1}{N}\sum_{i=1,\ldots,N} |f(\vec{x_i}) - u(\vec{x_i})|^2\right)^{1/2}. \tag{19}$$

Fig. 5 displays an adaptive sparse grid interpolation of our test function f that was constructed with a refinement threshold of $\epsilon = 0.01$. This figure illustrates nicely that the non-differentiability in f is automatically detected by the adaptive sparse grid algorithm.

In Fig. 6 we show the convergence behavior of the adaptive sparse grid method for approximating f. The data points reported in Fig. 6 were obtained by computing the L_∞ and L_2 errors at levels 5, 8, 10, 12, and 15. These results are contrasted with the “classical” sparse grid counterparts of the respective refinement levels. Strikingly, to reach an L_2 error of around 10^{-4} the adaptive sparse grid needs $4,411$ points as opposed to $311,297$ points when using the conventional sparse grid.

Summing up, local adaptability can save enormous amounts of resources when applied to functions with non-differentiabilities. In Sec. 4 we show that this is still true when going to economic applications and also to higher dimensions.

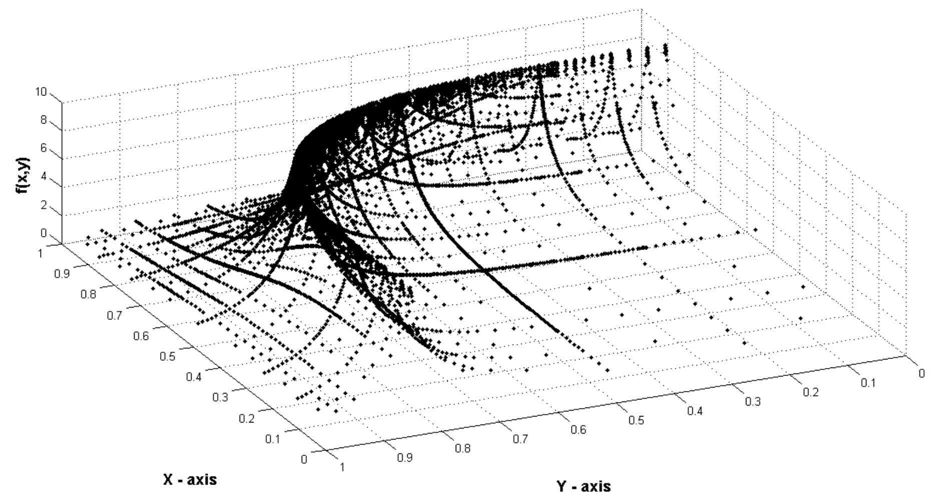

Figure 5: Evaluation of the non-differentiable test function given in Eq. 18 at the grid points obtained by the adaptive sparse grid algorithm at level 15.

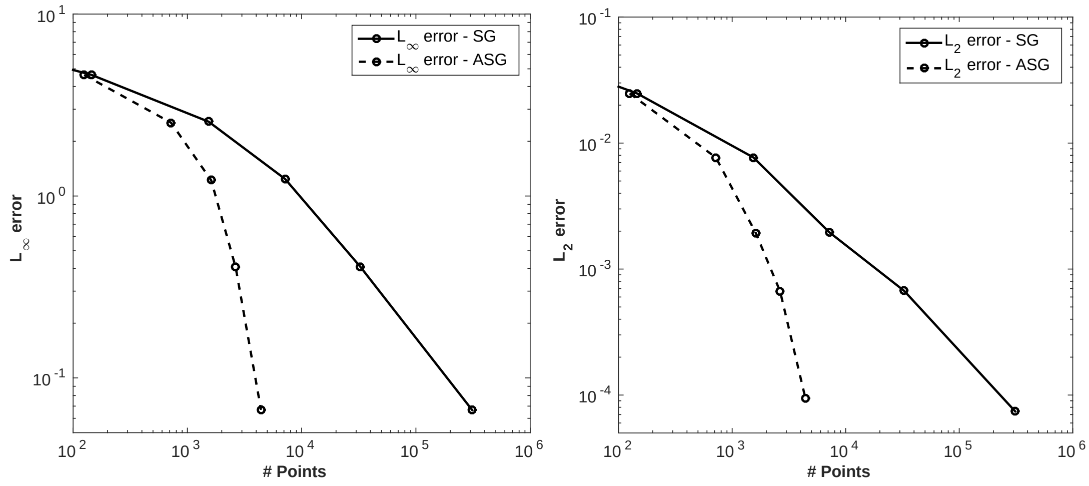

Figure 6: Comparison of the interpolation error for "classical" sparse grid (SG) and adaptive sparse grid (ASG) interpolation at different refinement levels. The left figure displays the L_∞ error; the right figure the L_2 error (see Eq. 19). The adaptive grids were obtained by applying a threshold $\epsilon = 10^{-2}$ and increasing maximum refinement levels.

4 High-Dimensional IRBC Application

We now apply the adaptive sparse grid method to an IRBC model with adjustment costs and irreversible investment. In Sec. 4.1, we introduce the model, while Sec. 4.2 outlines the time iteration algorithm for solving the model and Sec. 4.3 presents the parallelization of the code. Finally, Sec. 4.4 discusses the performance of our method, a task to which the IRBC model is perfectly suited as its dimensionality can be scaled up in a straightforward and meaningful way—it just depends linearly on the number of countries considered.

4.1 IRBC Model with Irreversible Investment

To focus on the problem of handling high-dimensional state spaces, we follow [23, 42, 46] and consider an IRBC model that is rather simple, except for the large number of countries. However, we add one crucial assumption: investment in each country is irreversible. The kinks in policy functions induced by these occasionally binding constraints constitute a perfect test case for our ASG method. We now briefly describe the IRBC model, its first-order conditions, and its parameterization.

The Model Description

There are N countries that differ from one another with respect to their preferences, their exogenous productivity, and their endogenous capital stock. They all produce, trade, and consume a single homogeneous good. Production of country j at time $t \in \mathbb{N}_0$ is given by

$$y_t^j = a_t^j \cdot f^j(k_t^j), \tag{20}$$

where $a_t^j \in \mathbb{R}_{++}$, $f^j : \mathbb{R}_+ \to \mathbb{R}_+$, and $k_t^j \in \mathbb{R}_+$ are productivity, a neoclassical production function, and the capital stock of country j, respectively. The law of motion of productivity is given by

$$\ln a_t^j = \rho \cdot \ln a_{t-1}^j + \sigma \left(e_t^j + e_t\right), \tag{21}$$

where the shock e_t^j is specific to country j, e_t is a global shock, and these shocks are all i.i.d. standard normal. The parameter σ is thus the standard deviation of both types of shocks to log-productivity. The law of motion of capital is given by

$$k_{t+1}^j = k_t^j \cdot (1-\delta) + i_t^j, \ i_t^j \geq 0, \tag{22}$$

where δ is the rate of capital depreciation, and i_t^j is investment. The irreversibly of investment in Eq. 22 can also be expressed as

$$k_{t+1}^j - k_t^j \cdot (1-\delta) \geq 0. \tag{23}$$

Moreover, there is a convex adjustment cost on capital, given by

$$\Gamma_t^j(k_t^j, k_{t+1}^j) = \frac{\phi}{2} \cdot k_t^j \cdot \left(\frac{k_{t+1}^j}{k_t^j} - 1\right)^2. \tag{24}$$

The aggregate resource constraint thus reads

$$\sum_{j=1}^{N} \left(a_t^j \cdot f^j(k_t^j) + k_t^j \cdot (1-\delta) - k_{t+1}^j - \Gamma_t^j(k_t^j, k_{t+1}^j) - c_t^j\right) \geq 0. \tag{25}$$

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,[4] 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

$$\max_{\{c_t^j, k_t^j\}} \mathbb{E}_0 \sum_{j=0}^{N} \tau^j \cdot \left(\sum_{t=1}^{\infty} \beta^t \cdot u^j(c_t^j)\right) \tag{26}$$

[4]We follow the comparison study [23, 42, 46] in assuming complete markets. However, the time iteration algorithm in Sec. 4.2 can also handle settings with incomplete markets.

subject to the constraints (23) and (25) given initial capital stocks $k_0 \in \mathbb{R}^N_{++}$ and productivity levels $a_0 \in \mathbb{R}^N_{++}$. Here and in the following, $\mathbb{E}_t$ denotes the expectation conditional on the information available at time t.
We assume the following standard functional forms for the production function and the utility function:

$$f^j(k^j_t) = A \cdot (k^j_t)^\zeta,\ u^j(c^j_t) = (1 - 1/\gamma_j)^{-1} \, (c^j_t)^{1-\frac{1}{\gamma_j}}, \tag{27}$$

where ζ is the capital share and γ_j is the elasticity of intertemporal substitution (EIS) for country j.

First-Order Conditions

We now state the first-order conditions (FOCs) of problem (26) for the functional forms given in (27). We denote the multiplier on the time t resource constraint by λ_t, the Kuhn–Tucker multipliers for the irreversibility constraints by μ^j_t, and the growth rate of capital by $g^j_t = k^j_t/k^j_{t-1} - 1$. The Euler equations and the irreversibility constraints for all $j \in \{1, ..., N\}$ are then given by

$$\begin{aligned} &\lambda_t \cdot \left[1 + \phi \cdot g^j_{t+1}\right] - \mu^j_t \\ &\quad - \beta \mathbb{E}_t \left\{ \lambda_{t+1} \left[a^j_{t+1} A \zeta (k^j_{t+1})^{\zeta - 1} + 1 - \delta + \frac{\phi}{2} g^j_{t+2} \left(g^j_{t+2} + 2 \right) \right] - (1-\delta) \mu^j_{t+1} \right\} = 0, \end{aligned} \tag{28}$$

$$0 \leq \mu^j_t \perp \left(k^j_{t+1} - k^j_t (1-\delta) \right) \geq 0. \tag{29}$$

The aggregate resource constraint (holding with equality due to the strictly increasing per-period utility assumed in Eq. 27) is given by

$$\sum_{j=1}^{N} \left(a^j_t \cdot A \cdot (k^j_t)^\zeta + k^j_t \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g^j_{t+1})^2 \right) - k^j_{t+1} - \left(\frac{\lambda_t}{\tau_j} \right)^{-\gamma^j} \right) = 0. \tag{30}$$

In Sec. 4.2 we describe how to solve for a recursive equilibrium of the IRBC model by iterating on recursive formulations of Eqs. 28–30.

Parameterization

With respect to the parameter choices we follow Juillard and Villemot [42], who provide the model specifications for the comparison study that uses the IRBC model to test several solution methods (see [23]). We choose an asymmetric specification where preferences are heterogeneous across countries. In particular, the EIS of the N countries is evenly spread over the interval $[0.25, 1]$. This corresponds to model A5 in Juillard and Villemot [42]. The only difference between our parameterization and theirs is that we use a depreciation rate of $\delta = 1\%$ (per quarter), while they write down the model such that they effectively have no depreciation. The parameters we use are reported in Tab. 2. Note that the welfare weights τ^j do not matter for the capital allocation, only for the consumption allocation, which we do not consider. The aggregate productivity parameter is chosen such that the capital of each country is equal to 1 in the deterministic steady state.

4.2 Time Iteration Algorithm

We solve for an equilibrium of the IRBC model that is recursive in the physical state of the economy, s, which is given by the productivity levels and capital stocks of all N countries. Thus, dropping time indices, we have

$$s = (a, k) = \left(a^1, \dots, a^N, k^1, \dots, k^N \right). \tag{31}$$

Parameter	Symbol	Value
Discount Factor	β	0.99
EIS of Country j	γ^j	a+(j-1)(b-a)/(N-1)
		with a=0.25, b=1
Capital Share	ζ	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
Aggregate Productivity	A	$(1-\beta(1-\delta))/(\zeta-\beta)$
Welfare Weights	τ^j	A^{1/γ^j}
Number of Countries	N	$\{2, ..., 50\}$

Table 2: Choice of parameters for the IRBC model, following Juillard and Villemot [42].

Denoting the state space by $S \subset \mathbb{R}^d$, where $d = 2N$, we solve for policies that map the current state into capital choices, into the Kuhn–Tucker multipliers on the irreversibility constaints, and into the Lagrange multiplier on the aggregate resource constraint. Thus, the policy function we solve for is

$$p : S \to \mathbb{R}^{2N+1},\ p(s) = \left(k^1_+(s), \dots, k^N_+(s), \mu^1(s), \dots, \mu^N(s), \lambda(s)\right), \tag{32}$$

where we indicate next period variables with a subscript "+". Such policies p represent an equilibrium of the IRBC model only if they satisfy the equilibrium conditions (see Eqs. 28–30) at all points in the state space. To compute policies that approximately satisfy these conditions, we use time iteration (see, e.g., [39], Sec. 17.8) and employ adaptive sparse grid interpolation in each of its iteration steps. The structure of the algorithm is as follows:[5]

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

   $$p_+ : S \to \mathbb{R}^{2N+1}, p_+(s_+) = \left(k^1_{++}(s_+), \dots, k^N_{++}(s_+), \mu^1_+(s_+), \dots, \mu^N_+(s_+), \lambda_+(s_+)\right).$$

   Choose an approximation accuracy $\bar{\eta}$.

2. Make one time iteration step:

   (a) Start with a coarse grid $G_{old} \subset S$, and generate G (of level L_0) by adding for each $x \in G_{old}$ all $2d$ neighboring points. Choose a refinement threshold ϵ and a maximal level $L_{max} > L_0$. Set $l = L_0$.

   (b) For each grid point

   $$x = \left(a^1, \dots, a^N, k^1, \dots, k^N\right) \in \begin{cases} G \text{ if } l = L_0 \\ G \setminus G_{old} \text{ if } l > L_0 \end{cases}$$

   solve for the optimal policies

   $$p(x) = \left(k^1_+(x), \dots, k^N_+(x), \mu^1(x), \dots, \mu^N(x), \lambda(x)\right)$$

   at grid point x by solving the system of equilibrium conditions given the next period's policy

   $$p_+(s_+) = \left(k^1_{++}(s_+), \dots, k^N_{++}(s_+), \mu^1_+(s_+), \dots, \mu^N_+(s_+), \lambda_+(s_+)\right).$$

[5]Note that this is the algorithm for the adaptive sparse grid. The "classical" sparse grid of level L is obtained as a special case by setting $\epsilon = 0$ and $L_{max} = L$.

More precisely, abbreviating $p(x)$ by p, and $p_+(s_+)$ by p_+, the equilibrium conditions (Eqs. 28–30) demand that for all countries $j \in \{1, ..., N\}$:

$$p^{2N+1} \cdot \left[1 + \phi \cdot (p^j/k^j - 1)\right] - p^{N+j} -$$

$$\beta \mathbb{E}_t \left\{ p_+^{N+1} \left[a_+^j A \zeta (p^j)^{\zeta - 1} + 1 - \delta + \frac{\phi}{2} \left(\frac{p_+^j}{p^j} - 1 \right) \left(\frac{p_+^j}{p^j} + 1 \right) \right] - (1-\delta) p_+^{N+j} \right\} = 0,$$

$$0 \leq p^{N+j} \perp \left(p^j - k^j(1-\delta)\right) \geq 0.$$

and also

$$\sum_{j=1}^{N} \left(a^j A (k^j)^{\zeta} + k^j \left((1-\delta) - \frac{\phi}{2} (p^j/k^j - 1)^2 \right) - p^j - \left(\frac{p^{2N+1}}{\tau_j} \right)^{-\gamma^j} \right) = 0.$$

(c) Generate G_{new} from G by adding for each $x \in G \setminus G_{old}$ its $2d$ neighboring points if

$$\|p(x) - \tilde{p}(x)\|_\infty > \epsilon,$$

where the policy $\tilde{p}(x)$ is given by interpolating between $\{p(x)\}_{x \in G_{old}}$.

(d) If $G_{new} = G$ or $l = L_{max}$, then set $G = G_{new}$ and go to (e); otherwise set $G_{old} = G$,$G = G_{new}$, $l = l + 1$, and go to (b).

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

(f) Calculate (an approximation for) the error, for example,

$$\eta = \|p - p_+\|_\infty.$$

If $\eta > \bar{\eta}$, set $p_+ = p$ and go to step 2; otherwise go to step 3.

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

In principle, one could set L_0 very small and L_{max} very large. These choices can, however, create practical problems. If L_0 is very low, the algorithm starts with a very coarse grid, increasing the risk that some irregularities will not be detected. If, in contrast, L_{max} is set too high, one can end up with far too many points, as there is no reasonable upper bound for the number of grid points created by the refinement procedure. To be on the safe side when solving a model that is not well understood, one should therefore choose L_0 not too small and L_{max} not too large.

4.3 Parallelization

In order to solve "large" problems in a reasonable amount of time, we make use of modern hybrid HPC architectures. Only the combination of multiple parallel programming paradigms enables us to use the massive compute power available on today's high-end systems.[6]

For this purpose, in each step of the above time iteration procedure the updated policy function is determined using a hybrid parallel algorithm, as illustrated in Fig. 7. Within each refinement step, we first distribute newly generated grid points among multiple processes by Message Passing Interface (MPI; see [62]). The points that are sent to one particular compute node are further distributed among different threads via Threading Building Blocks (TBB; see

[6]Note that there is a clear trend in hardware design (even for desktops) towards massively parallel compute units. Hence, if economists in the foreseeable future want to be able to make use of the still exponentially growing compute resources, they have to be aware of these technical possibilities and how these can be exploited when using standard algorithms like value function iteration or time iteration.

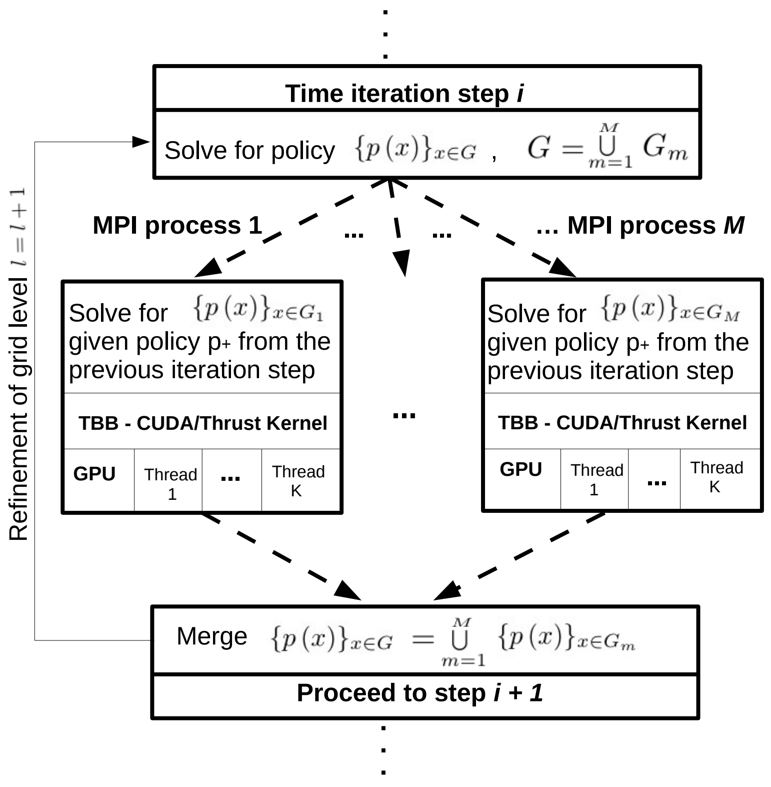
Time iteration step i
Solve for policy $\{p(x)\}_{x\in G}$, $G = \bigcup_{m=1}^{M} G_m$
Refinement of grid level $l = l + 1$
MPI process 1
...
...
... MPI process M
Solve for $\{p(x)\}_{x\in G_1}$ given policy p+ from the previous iteration step
TBB - CUDA/Thrust Kernel
GPU
Thread 1
...
Thread K
Solve for $\{p(x)\}_{x\in G_M}$ given policy p+ from the previous iteration step
TBB - CUDA/Thrust Kernel
GPU
Thread 1
...
Thread K
Merge $\{p(x)\}_{x\in G} = \bigcup_{m=1}^{M} \{p(x)\}_{x\in G_m}$
Proceed to step $i + 1$

Figure 7: Schematic representation of the hybrid parallelization of a time iteration step (see Sec. 4.2). Newly generated points are distributed among different MPI processes. Each MPI process allocates points to different threads via TBB and offloads parts of the interpolation to the available GPU via a CUDA/Thrust Kernel.

[61]). Each thread then solves a set of nonlinear equations for every single grid point assigned to it.[7] On top of this, we add an additional level of parallelism. When searching for the solution to the equation system at a given point, the algorithm has to frequently interpolate the function computed in the previous iteration step. These interpolations take up 99 percent of the computation time needed to solve the equation system. As they have a high arithmetic intensity—that is to say, many arithmetic operations are performed for each byte of memory transfer and access—they are perfectly suited for GPUs (see, e.g., [34, 55]). We therefore offload parts of the interpolation from the compute nodes to their attached accelerators via a CUDA/Thrust Kernel (see [9]).

Combining distributed memory parallelism with the on-node shared memory parallelism and the usage of GPUs, we are able to scale our code to at least $2,048$ compute nodes with more than 60% efficiency, resulting in an overall speedup of more than four orders of magnitude compared to running this numerical experiment on a single CPU. More details regarding the optimization of our code and the parallel implementation can be found in Appendix D and in our companion paper [14].

4.4 Performance and Accuracy

In this section we show how the algorithm presented in Secs. 4.2 and 4.3 performs in solving the IRBC model described in Sec. 4.1. We start our discussion by briefly considering the model

[7]The set of nonlinear equations is solved with `Ipopt` [65] (`http://www.coin-or.org/Ipopt/`).

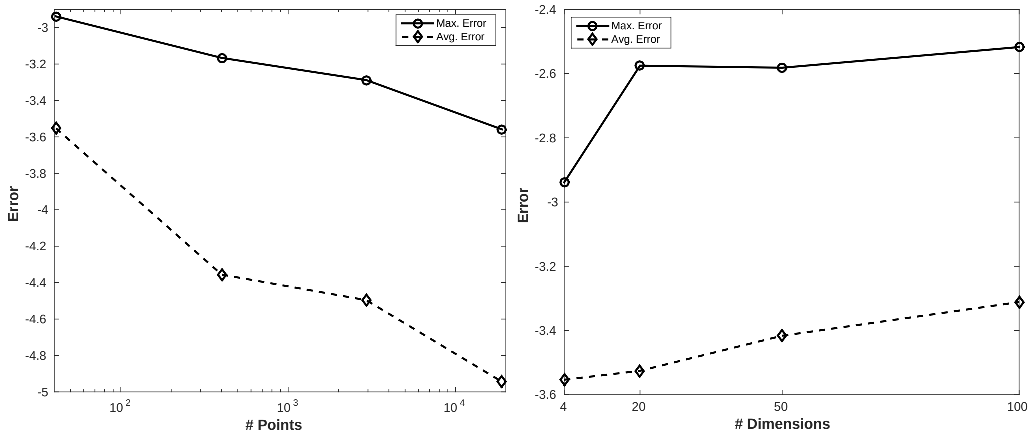

Figure 8: Both panels show maximum and average approximation errors for **SG solutions** of the **smooth IRBC** model. In the left panel, the dimension is fixed at $d = 2N = 4$, while the approximation level increases ($l \in \{3, 5, 7, 9\}$). In the right panel the level is fixed at $l = 3$, while the dimension of the problem increases ($d \in \{4, 20, 50, 100\}$). All errors are reported in $\log_{10}$-scale.

without the irreversibility constraints, which we call the smooth IRBC model. We then go on to discuss the model that includes the occasionally binding irreversibility constraints, which we call the non-smooth IRBC model. To evaluate the accuracy of our solutions we follow the previous literature and compute errors in the equilibrium conditions (see, e.g., [42]). Details of our error measure and the implementation of the algorithm can be found in Appendix C.

Results for the Smooth IRBC Model

To gain a first systematic understanding of how accuracy varies with the grid level and the dimensionality of the problem, we consider "classical" sparse grid solutions of the smooth IRBC model. The left panel of Fig. 8 shows the approximation errors as a function of the grid level while keeping the dimensionality fixed at four. The maximum and average errors fall as the level of the grid increases, which is exactly what one would expect. The errors are reasonably low even for a relatively small number of grid points. This is simply because the policy functions of the IRBC model are very smooth. For the same reason, the accuracy obtained by some smooth approximation methods used in the comparison study summarized by [46] is even higher than the accuracy of our solutions. However, we can go to much higher dimensions. As a next exercise we vary the dimensionality of the problem up to $d = 100$ while keeping the grid level fixed. The right panel of Fig. 8 shows that there is only a very moderate downward trend in accuracy as the dimensionality of the problem is increased massively. Details about the resources needed to solve high-dimensional smooth IRBC models are reported in Appendix D.

A severe drawback of "classical" sparse grids of all types, including Smolyak's method, is that the number of grid points grows very fast with the level of the approximation, as shown in the last three columns of Tab. 8 in Appendix A. It is, therefore, often not practical to increase accuracy by simply going to the next level. For instance, in 50 dimensions a level-3 SG has $5,001$ points while a level-4 grid already has $171,901$ points. Adaptive sparse grids can solve this problem, as intermediate grid sizes can be reached by choosing the refinement threshold and maximum refinement level appropriately. In Tab. 3 we present results for a 50-dimensional ASG with $12,551$ points, which achieves substantially higher accuracy compared to the level-3 SG with only moderately more points—about 3 times higher accuracy with 2.5 times more

Dimension	Grid type	Points	Max. Error	Avg. Error
50	SG: Level 3	5,001	-2.58	-3.42
50	ASG: Max. Level 5	12,551	-3.14	-3.90

Table 3: Maximum and average errors of **SG and ASG solutions** of the **smooth IRBC** model with 50 dimensions. To achieve much higher accuracy than the level-3 SG, the ASG needs only 2.5 times as many points, while the level-4 SG would need 34 times as many points—namely, $171,901$. The ASG has a refinement threshold $\epsilon = 0.001$ and a maximum refinement level $L_{max} = 5$, corresponding to an SG with $4,352,001$ points (see Tab. 8).

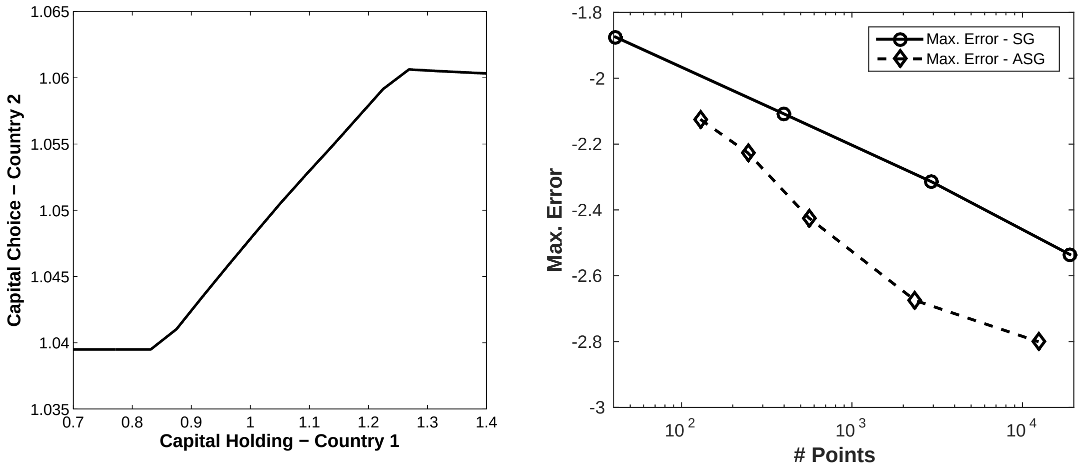

Figure 9: Both panels refer to the 2-country IRBC model with irreversible investment. The left panel shows the capital choice of country 2 as a function of the capital holding of country 1, while all three other state variables are kept fixed. The right panel compares the maximum error for SGs and ASGs as a function of the number of grid points. The data points for the SGs stem from models run with fixed levels 3, 5, 7, and 9. The respective ASGs arise from varying refinement thresholds ϵ ranging from 0.015 down to 0.001. This figure illustrates that, for similar sized grids, the ASG is much more accurate than the SG.

points. This example shows that ASGs are much more flexible than SGs when it comes to choosing the right balance between running times and accuracy. In the following, we show that ASGs are also much more efficient in reaching a desired level of accuracy, in particular in the case of non-smooth models.

Results for the Non-Smooth IRBC Model

We now turn to the IRBC model with irreversible investment, as described in Sec. 4.1. The assumption that investment is not reversible induces non-differentiabilities in the policy functions that we interpolate. Two such kinks can be observed in the left panel of Fig. 9. For the 2-country case, Fig. 9 plots the capital choice (i.e., the end-of-period capital) for country 2 as a function of the beginning-of-period capital holding of country 1 while keeping all other state variables fixed. If the capital of country 1 is low, then investment opportunities in that country are better than in country 2, and thus the irreversibility constraint for country 2 is binding. This explains the flat part of the policy function at the lower-left side of Fig. 9. If, on the other hand, the capital of country 1 is very high, its irreversibility constraint is binding and thus limits the transfer of resources to country 2. Therefore the policy function is non-increasing at

Dimension	Level	Points	Max. Error	Avg. Error
4	3	41	-1.87	-2.82
4	5	401	-2.11	-3.34
4	7	2,929	-2.32	-3.57
4	9	18,945	-2.45	-3.83

Table 4: Maximum and average errors of **SG solutions** of the **non-smooth IRBC** model with fixed dimension and increasing approximation level. All errors are reported in $\log_{10}$-scale. The number of grid points is also reported.

ϵ	Max. Level Reached	Points	Max. Error	Avg. Error
0.0150	6 (1,105)	129	-2.12	-3.06
0.01	7 (2,929)	245	-2.23	-3.12
0.005	9 (18,945)	559	-2.42	-3.22
0.0025	13 (643,073)	2,346	-2.68	-3.63
0.001	14 (3,502,081)	12,437	-2.80	-3.87

Table 5: Maximum and average errors for **ASG solutions** of the **non-smooth IRBC** model of different refinement thresholds ϵ. The size of the respective 4-dimensional ASGs is reported in the third column. The second column contains two numbers—the first is the highest resolution level that was reached for a particular ϵ; the second (in parentheses) indicates how many grid points a corresponding SG of that resolution level would comprise.

the very right. It is in fact slightly decreasing for consumption smoothing reasons. When looking at Fig. 9, one has to keep in mind that a kink that appears as a single point in that slice through the $2N$-dimensional state space is in fact a (2N-1)-dimensional hypersurface. Such high-dimensional kinks pose a substantial challenge to the construction of an accurate global approximation. Global (polynomial) basis functions are clearly not suited to approximating such distinct local features.[8]

In order to gain an understanding of how accurately the (adaptive) sparse grid method is able to solve high-dimensional, non-smooth IRBC models, we first focus on the 4-dimensional case. Tab. 4 reports errors for "classical" SGs of increasing levels. It reveals two important insights. First, both the maximum and the average errors of non-smooth IRBC models are considerably worse than those of smooth IRBC models of comparable resolution (see Fig. 8). The reason for this is that a relatively coarse grid can hardly capture kinks. The second insight is that increasing the global resolution level decreases the errors only very slowly—in particular, the maximum errors. The reason for this behavior is that even with a relatively high global resolution, the local resolution is not sufficiently high to roughly match the kinks. In contrast to SGs, ASGs are more efficient at reducing approximation errors as the right panel of Fig. 9 shows (see also Tabs. 4 and 5). For instance, an ASG with $12,437$ points (see Tab. 5) achieves higher accuracy than an SG with $18,945$ points (see Tab. 4). In Tab. 5, we report results for different refinement thresholds. The smaller the chosen threshold, the larger the maximum refinement level reached and the larger the number of grid points. With the lowest threshold considered ($\epsilon = 0.001$), the ASG algorithm continues to refine until level 14. Thus, we are able to locally mimic an interpolant that is of the same order of approximation as an SG of level

[8]In general, high polynomial degrees are required to match kinks, which in higher dimensions results in an enormous number of grid points. Moreover, when high-order polynomials are used, spurious oscillations occur, implying wrong derivatives, even in terms of their sign. This is particularly problematic if the interpolation procedure is employed within an iterative approach that involves solvers that rely on derivative information.

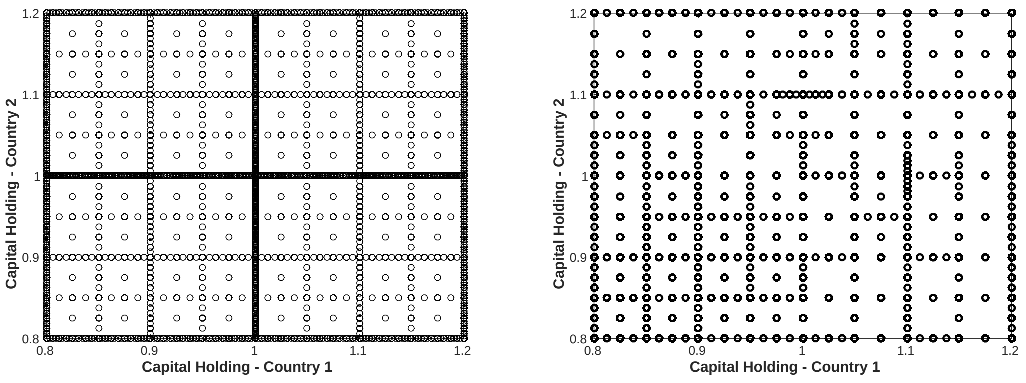

Figure 10: This figure displays 2-dimensional projections of two different 4-dimensional grids used to interpolate a policy function in a non-smooth, 2-country IRBC model. The left projection is from a "classical" SG of level 9 (see Tab. 4), the right from an ASG of refinement threshold $\epsilon = 0.001$ (see Tab. 5). The x-axis shows the capital holding of country 1, the y-axis shows the capital holding of country 2, while the productivities of the two countries are kept fixed at their unconditional means.

14. While the ASG consists of $12,437$ grid points, the SG would comprise more than 3 million points (see the second column of Tab. 5). This comparison shows that the adaptive sparse grid introduces a second layer of sparsity on top of the a priori sparse structure of the "classical" sparse grid. Making use of this sparsity, additional grid points can reduce the approximation error much more efficiently (see the right panel of Fig. 9). This is possible as grid points are placed only where high resolution is needed while only few points are put in areas where the policies to be approximated vary little. In order to illustrate this feature, we display—in Fig. 10—slices of 4-dimensional grids, one SG of fixed refinement level 9 (see Tab. 4) and one ASG with refinement threshold $\epsilon = 0.001$. In spite of having considerably fewer points, the solution provided by the adaptive sparse grid is of comparable accuracy. Note that the actual grid is 4-dimensional; thus the 2-dimensional projection can only give a rough idea of the sparse grid structure.

Let us now turn to higher-dimensional models. Fig. 11 reports errors for adaptive sparse grids of increasing dimension. The refinement threshold is set to $\epsilon = 0.01$ for dimensions $d = \{4, 6, 8, 10, 12\}$ since this setting provides decent errors at moderate costs (see Tab. 5). For dimensions 16 and 20, we chose $\epsilon = 0.001$ in order to obtain results of the desired accuracy. There seems to be a moderate downward trend in accuracy as a function of dimensionality. This behavior is not surprising, as the kinks, being $(2N-1)$-dimensional objects, become much harder to approximate as N increases. One way of counteracting the worsening errors is to lower ϵ moderately when the dimension of the problem is increased (as we did for dimensions 16 and 20).

Regarding the compute resources needed to solve the non-smooth IRBC model, the required CPU time grows less than exponentially as the dimension increases. Thus, we can—at least to some extent—control the increasing running times by using a larger number of compute nodes (for more details, see Sec. 4.3 and Appendix D). This stands in contrast to, for instance, [41, 53], who only provide serial algorithms.

To sum up, the presented adaptive sparse grid method can successfully compute global solutions of high-dimensional, non-smooth dynamic models. Resolving such non-smooth behavior is not possible with the methods for high-dimensional models that have so far been considered

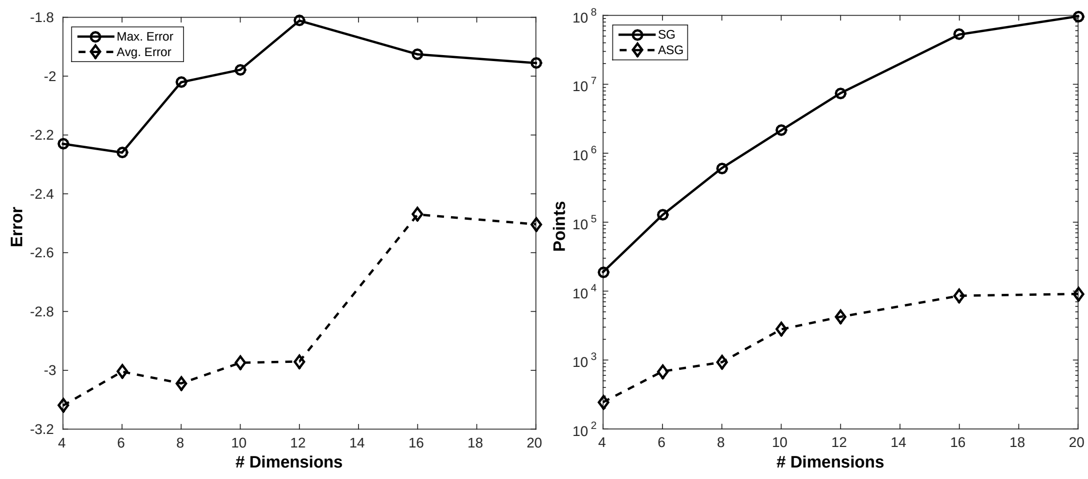

Figure 11: The left panel shows the errors for **ASG solutions** of the **non-smooth IRBC** model with increasing dimension. The right panel displays the number of grid points of the ASG solution, contrasted with the corresponding grid size of the SG of level 9, which is the maximum refinement level employed for the ASG.

in economics (see, e.g., [42, 47]). Methods that are designed to handle non-differentiabilities (see, e.g., those in [37, 12, 26, 7]) are in turn only capable of solving models with up to three or four continuous state variables.

5 Multiproduct Menu-Cost Application

To demonstrate the broad applicability of our method, we now solve a high-dimensional dynamic economic model that differs in important ways from the IRBC model above. We consider the price-setting behavior of a multiproduct firm facing fixed costs for adjusting prices (so-called menu-costs). Due to these costs, the firm has to make a discrete choice about whether to adjust prices. As a consequence, we have to solve the model by value function iteration and have to keep track of the values associated with each of the choices possible. The menu-cost application thus illustrates that our method also works with value function iteration and with discrete choices.

5.1 Menu-Cost Model

Menu-cost models are at the center of an ongoing debate about the real effects of monetary policy (see, e.g., [4, 10, 43, 3]). This debate started with Golosov and Lucas [31], who argue that in a menu-cost model that is consistent with the large average size of price changes in micro-price data, the real effects of monetary policy are very small. This is due to the so-called selection effect (going back to Caplin and Spulbar [20]): an inflationary monetary shock, while hardly increasing the number of price changes, substantially alters which prices are changed—more of the low prices are increased and less of the high prices are decreased. However, Midrigan [54] finds much stronger real effects of monetary policy than Golosov and Lucas [31] by considering a model with the following two features: first, firms have the option of temporarily deviating from a regular price at a certain cost; second, firms face economies of scope in setting the regular and temporary prices of multiple goods—in particular, the cost of changing prices is the same no matter how many prices are changed. The first feature can account for the sales periods present in the data.[9] The second feature in turn implies more small price changes, which are "for free" at times when a firm makes a large change to the price of another good. For that reason, the economies of scope assumption helps to match the high standard deviation of price changes in the data. However, the 2-goods setup of Midrigan [54] does not capture the high kurtosis of price changes, even though it assumes that good-specific shocks are fat-tailed (as in Gertler and Leahy [30]). Interestingly, Alvarez and Lippi [4] and Alvarez et al. [3]—in models that also include economies of scope in price setting but no temporary prices—derive quasi-analytical results stating that the kurtosis of price changes increases with the number of products. Motivated by these findings, we extend the model in Midrigan [54] to three goods and show that the fit to micro-price data is thereby substantially improved, while the real effects of monetary policy are even larger than with two goods.

Except for the number of goods, we consider the exact same setup as Midrigan [54]. We thus keep the exposition very concise and refer the reader to Midrigan [54] for details of the model and its parametrization. The economy consists of a perfectly competitive manufacturing sector, monopolistically competitive retailers (called simply firms), and a representative consumer. Each firm sells N goods, and receives two types of shocks influencing their optimal prices: one shock is discrete, transitory, and common across goods, thus driving most temporary price changes; the other type of shock is continuous, good-specific (yet correlated across goods), and persistent, thus causing most changes in regular prices. To change regular prices, the firm has to incur costs ϕ^R; to charge temporary prices that differ from the regular prices, the firm has to incur costs κ; finally, the firm can charge the previous period's regular prices at no cost. Due to economies of scope, the same costs ϕ^R or κ apply whether one or more than one price is changed. Let V^R be the firm's value conditional on changing regular prices in the current period; let V^T be the value for charging temporary prices; and let V^N be the value of simply charging the prevailing regular prices. All these values are functions of the vector of the firm's previous period markups $\boldsymbol{\mu}_{-1}^R$, of the discrete state of the firm e, of the money growth rate g, and of the distribution of markups across firms Λ. Denoting the (unconditional) value of the firm by $V = \max\left(V^R, V^T, V^N\right)$, Midrigan [54] characterizes the firm's problem by the following system of functional equations:

$$
\begin{aligned}
V^R(\boldsymbol{\mu}_{-1}^R, e; g, \Lambda) &= \max_{\mu^R} \left(\sum_{i=1}^{N} e^{1-\gamma} \left(\mu^R - 1\right) (\mu^R)^{-\gamma} \hat{P}^{\gamma-1} - \phi^R + \beta \mathbb{E} V((\boldsymbol{\mu}_{-1}^R)', e'; g', \Lambda') \right) \\
V^T(\boldsymbol{\mu}_{-1}^R, e; g, \Lambda) &= \max_{\mu^T} \left(\sum_{i=1}^{N} e^{1-\gamma} \left(\mu^T - 1\right) (\mu_i^T)^{-\gamma} \hat{P}^{\gamma-1} - \kappa + \beta \mathbb{E} V((\boldsymbol{\mu}_{-1}^R)', e'; g', \Lambda') \right) \\
V^N(\boldsymbol{\mu}_{-1}^R, e; g, \Lambda) &= \sum_{i=1}^{N} e^{1-\gamma} \left(\mu_{i,-1}^R - 1\right) (\mu_{i,-1}^R)^{-\gamma} \hat{P}^{\gamma-1} + \beta \mathbb{E} V((\boldsymbol{\mu}_{-1}^R)', e'; g', \Lambda').
\end{aligned}
$$

The parameter γ captures the elasticity of substitution between two identical goods sold by different firms, and $\hat{P}$ is the aggregate price level (de-trended by the money stock).

5.2 Numerical Approach

To make the above described problem tractable, Midrigan [54]—inspired by Krusell and Smith [48]—replaces the distribution of markups Λ by the previous period's aggregate price level $\hat{P}_{-1}$ and assumes that $\hat{P}$ is a log-linear function of $\hat{P}_{-1}$ and g.[10] The state space on which the above

[9] An alternative way of capturing sales periods is to assume that firms set price plans as in Eichenbaum et al. [25] or as in Alvarez and Lippi [5].

[10] Alvarez and Lippi [4] show that the general equilibrium feedback on decision rules is only weak and that there is thus not much accuracy lost by not making firms' decisions dependent on information about the distribution of markups across firms.

	Dimension of Grid	Points of ASG	Points of SG
2 Goods	4	52,723	3,502,081
3 Goods	5	187,707	12,765,185

Table 6: Number of points of the ASG (average over the two discrete states), contrasted by the size of a "classical" SG of the same maximum refinement level—namely level 14.

value functions are defined then comprises $N + 2$ continuous dimensions (for the N markups, g, and $\hat{P}_{-1}$) and one discrete dimension (for the two possible realizations of e). For $N = 2$, Midrigan [54] interpolates the value functions on this space with splines on tensor product grids—an approach that severely suffers from the curse of dimensionality.

To overcome the curse of dimensionality, we solve the above menu-cost model using a value function iteration algorithm that interpolates the value functions using ASGs. The iterative structure of this algorithm (which can be parallelized exactly as described in Sec. 4.3) is very similar to that of the time iteration algorithm in Sec. 4.2. The most obvious difference is that, in step 2(b), instead of updating policy functions by solving systems of nonlinear equations, the value functions are updated by solving the respective maximization problems. Another difference is that the state space, in addition to the continuous dimensions, now includes a discrete component, which we handle by building separate ASGs for each possible realization of the discrete state. A more subtle yet important difference is the adaptation criterion we employ (technically, the choice of $g(\cdot)$ in Eq. 17). The heuristic of our approach is to refine the grid in regions where a coarse grid appears to provide insufficient information about which of the three value functions has the highest value (i.e., about which discrete choice is optimal). Technically, when building the adaptive sparse grid interpolation of the three value functions V^R, V^T, and V^N, we refine the grid based on the minimal pairwise absolute differences between the hierarchical surpluses of the three value functions. This procedure ensures that we refine the grid in regions of the state space where the difference between two of the three value functions is small and changes substantially as the approximation level increases. It is in such regions that a further refinement could most likely provide a more precise picture of where each discrete choice is optimal. In Tab. 6, we report how much ASGs improve over "classical" SGs in our menu-cost application. Compared to the SGs with the same maximum local resolution, ASGs require almost 70 times less grid points, which results in a enormous reduction in computational cost. The reason for the usefulness of ASGs in the menu-cost application is twofold: First, the discrete choices in discrete time induce kinks in the value functions;[11] second, the curvature of the value functions varies substantially across different regions and dimensions of the state space. ASGs can adapt to both of these features.

5.3 Economic Results

Let us now turn to the results obtained by solving the menu-cost model with our adaptive sparse grid value function iteration algorithm. We first solve the 2-goods version of the model using the parameters that Midrigan [54] chooses to match the statistics of data from Dominick's Finer Foods retail chain. Tab. 7 shows that our algorithm generates similar results to those of Midrigan [54].[12] In particular, our computations confirm that the 2-goods model does not capture the high kurtosis of the price distribution.[13] Alvarez and Lippi [4] show that the fit of

[11]In contrast, value functions in continuous-time menu-cost models (like [4, 3]) do typically not exhibit kinks.

[12]Small deviations are to be expected due to the differences in the interpolation method, and also in some other details of the implementation, in particular the quadrature method.

[13]We concentrate on regular price changes, as these alone are mainly driven by good-specific shocks in Midrigan's framework; temporary price changes, in contrast, are mainly caused by firm-specific shocks. Thus,

Moments	Data Dominick's	2 Goods Midrigan [54]	2 Goods Sparse Grid	3 Goods Sparse Grid
Frequency of Price Changes	0.34	0.32	0.32	0.32
Frequency of Reg. Price Changes	0.029	0.025	0.025	0.027
Mean size of Price Changes	0.20	0.20	0.19	0.19
Mean size of Reg. Price Changes	0.11	0.10	0.13	0.11
Std. of Price Changes	0.18	0.15	0.14	0.15
Std. of Regular Price Changes	0.08	0.07	0.08	0.08
Kurtosis of Reg. Price Changes	4.0	2.5	2.5	3.2

Table 7: Statistics of price changes—first, data from Dominick's Finer Foods retail chain; second, results of the 2-goods model of Midrigan [54]; third and fourth, results of the 2- and 3-goods models solved with our adaptive sparse grid algorithm. We do not aim to match the data, but use the same parameters as Midrigan. We also do not try to match the kurtosis of regular price changes, but rather only show that with 3 goods much higher values can be reached.

the kurtosis can be improved by increasing the number of goods. Moreover, Alvarez et al. [3] show that one can generate a much higher kurtosis for a fixed number of goods by assuming menu costs to be random (as in Nakamura and Steinsson [56]). The latter assumption plays a similar role to that of the fat-tailed shocks in Midrigan [54]. It is thus very much in line with Alvarez et al. [3] that in our extension of Midrigan [54] from two to three goods the kurtosis reaches levels that are much closer to the data.[14] In the final column of Tab. 7, we report statistics from a simulation of the 3-goods model. This setup generates first and second moments that are close to those in the 2-goods model, yet at the same time a much higher kurtosis.[15] Note that our aim is not to match the kurtosis of Dominick's data, but rather to show that, in a framework with temporary price changes and fat-tailed shocks as in Midrigan [54], 3-goods models can generate much more reasonable values for the kurtosis.

Interestingly, the improved fit to the data implies even stronger real effects of monetary policy. We find that going from the 2-goods to the 3-goods setup increases the standard deviation of (HP-filtered) monthly consumption from 0.26 to 0.28 percent, reducing the distance to the value of 0.35 percent in the Calvo setup by more than 20 percent. This finding is in line with Alvarez et al. [3], who provide a quasi-analytical result stating that the real effects of monetary policy are proportional to the ratio of the kurtosis to the frequency of price changes. Thus, when keeping the frequency roughly constant, we should expect the real effects to rise proportionately with the kurtosis.[16] The intuition for this result is that the higher kurtosis of price changes weakens the above described selection effect.

The broader purpose of the menu-cost application in this section is to demonstrate the usefulness of ASGs for solving a class of models that is at the center of an ongoing debate of great economic importance. Many extensions of currently used menu-cost models—like relaxing the economies of scope assumption by assuming a fixed cost for each price change in

for a higher number of goods to have a substantial effect on the kurtosis of temporary price changes, the shock structure of the model would have to be changed.

[14]Alvarez et al. [3] provide a recent review of empirical measures of the kurtosis of price changes and conclude, in line with Midrigan [54], that it is around 4 in low-inflation countries.

[15]For the 3-goods model we use the same parameters as for the two goods model, except that we have to slightly recalibrate the good specific shocks in order to keep the first two moments of price changes approximately at their previous levels.

[16]Alvarez et al. [3] use the area under the impulse response of output as a measure of the real effect, while we use the standard deviation of consumption. However, these two measures move proportionately (see [56]).

addition to a fixed cost for any change—can be addressed with adaptive sparse grid methods.

6 Conclusion

We embed an adaptive sparse grid method in an iteration framework, either time iteration or value function iteration. In addition, we provide a fully hybrid parallel implementation of the resulting iterative adaptive sparse grid algorithm. With this implementation, we can efficiently use contemporary high-performance computing technology.

Our algorithm is highly flexible and scalable. First, by choosing the resolution level we can tightly control accuracy, and are thus able to strike the right balance between running times and the desired accuracy of the solution. Second, due to the highly parallelized implementation, we can speed up the computations tremendously by using a larger number of compute nodes. This allows us to solve hard problems in a relatively short amount of time and to tackle problems that have, thus far, been out of reach.

Our first application of ASGs are high-dimensional IRBC models. We solve smooth versions of these models with up to 100 continuous state variables and also non-smooth versions, with irreversible investment, of up to 20 dimensions. In the latter application, the comparative advantage of the adaptive sparse grid reveals its full potential, as it can efficiently capture the kinks induced by irreversibility without wasting additional grid points in regions of the state space where they are not needed. As another application, we solve a menu-cost model featuring temporary prices and economies of scope in price setting. This application demonstrates that adaptive sparse grids can also be used within a value function iteration framework and that they are particularly useful in the presence of discrete choices.

Being scalable and flexible, our iterative algorithm can make use of modern high-performance computing infrastructure and at the same time be applied to a broad variety of dynamic economic models. This tool thus offers the promise of being able to compute global solutions of economic models that include much more heterogeneity and non-smoothness than was previously possible. The need for such tools became obvious due to the financial crisis of 2008 with its large price fluctuations and tremendous spillover effects. Also, the oil price slump of 2014—with the associated spike in volatility and its repercussions across various sectors and countries of the world economy—was a distinct reminder of the urgent need for such methods.

Appendix

A Sparse Grids with Non-Zero Boundaries

In Sec. 3, we have assumed that the functions under consideration vanish at the boundary of the domain—that is, $f|_{\partial\Omega} = 0$. To allow for non-zero values at the boundary, the procedure one usually follows is to add additional grid points located directly on $\partial\Omega$ (see, e.g., [58, 45]). Doing this naively, one needs at least 3^d grid points, which makes the approach inapplicable to high-dimensional problems (see [45]). In what follows, we discuss two procedures that mitigate this problem.

The crucial idea when dealing with non-zero boundaries is to have only one grid point at the lowest level of approximation and to add support nodes at the boundaries only at the second level (see, e.g., [51, 45, 8, 57]). Technically, the difference from the sparse grid V_n^S is that the

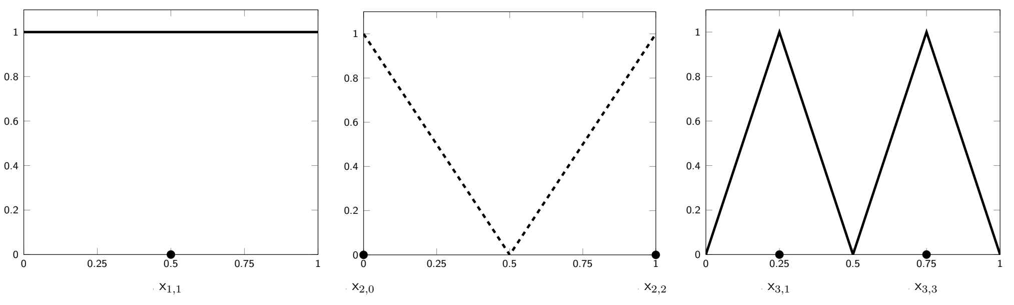

Figure 12: Shows 1-dimensional hierarchical basis functions of levels 1 (left), 2 (middle), and 3 (right). $V_3 = W_1 \bigoplus W_2 \bigoplus W_3$.

d	$\lvert V_4\rvert$	$\lvert V_4^S\rvert$	$\lvert V_4^{S,NZ}\rvert$	$\lvert V_5^{S,NZ}\rvert$	$\lvert V_6^{S,NZ}\rvert$
1	15	15	9	17	33
2	225	49	29	65	145
3	3,375	111	69	177	441
4	50,625	209	137	401	1,105
5	759,375	351	241	801	2,433
10	$5.77 \cdot 10^{11}$	2,001	1,581	8,801	41,265
20	$3.33 \cdot 10^{23}$	13,201	11,561	120,401	1,018,129
50	$6.38 \cdot 10^{58}$	182,001	171,901	4,352,001	88,362,321
100	>Googol	1,394,001	1,353,801	68,074,001	$2.74 \cdot 10^9$

Table 8: Number of grid points for several different grid types of level 4. First column—dimension; second column—full grid; third column— SG with no points at the boundaries; columns four to six — SG with non-zero boundaries (levels four to six).

index set of the support nodes is not given by the expression stated in Eq. 8, but rather by

$$I_{\vec{l}} := \begin{cases} \{\vec{i} : i_t = 1, 1 \leq t \leq d\} & \text{if } l = 1 \\ \{\vec{i} : 0 \leq i_t \leq 2,\ i_t \text{ even}, 1 \leq t \leq d\} & \text{if } l = 2 \\ \{\vec{i} : 1 \leq i_t \leq 2^{l_t - 1} - 1,\ i_t \text{ odd}, 1 \leq t \leq d\} & \text{else,} \end{cases} \tag{33}$$

and the 1-dimensional basis functions for non-zero (NZ) boundaries are given by

$$\phi^{NZ}_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 1 - 2 \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 0 \\ \left\{ \begin{array}{ll} 2 \cdot x_t - 1 & \text{if } x_t \in \left[\frac{1}{2}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 2 \\ \phi_{l,i}(x_t) & \text{else,} \end{cases} \tag{34}$$

where $\phi_{l,i}(x)$ is given by Eq. 2. The first three levels of these basis functions are displayed in Fig. 12. In line with Eq. 6, the multi-dimensional case can be obtained by a tensor product construction. An example of such a 2-dimensional sparse grid of level 3 is shown in Fig. 13. It is also worth mentioning that the number of grid points of the sparse grid with non-zero boundaries $|V_n^{S,NZ}|$ is always smaller than $|V_n^S|$ for a fixed grid refinement level n (see Tab. 8).

Another way of handling $f|_{\partial\Omega} \neq 0$ in high dimensions is to omit grid points on the boundary altogether and to instead modify the interior basis functions to extrapolate toward the boundary of the domain. This approach is especially well suited to settings where high accuracy

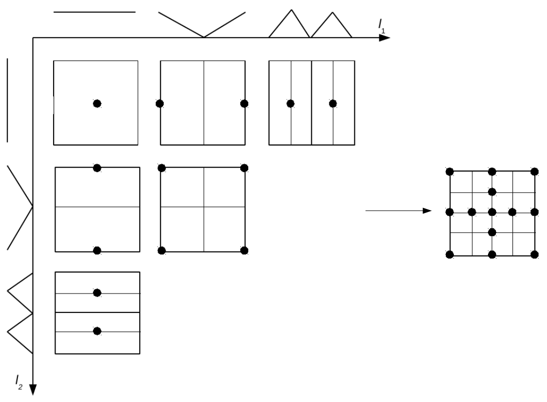

Figure 13: Schematic construction of a level-3 sparse grid with non-zero boundaries in two dimensions (see Eq. 33). $V_3^{S,NZ}$ consists of the hierarchical increment spaces $W_{(l_1,l_2)}$ that satisfy $|\vec{l}| \leq n+d-1$ (for $1 \leq l_1 \leq 3$, and $1 \leq l_2 \leq 3$). Note that in contrast to Fig. 3, there are grid points located at the boundary when (at least) one entry of the multi-index $\vec{l}$ has the value 2.

close to the boundary is not required and where the underlying function does not change too much toward the boundary (see [58]). An appropriate choice for the modified 1-dimensional basis functions is

$$\phi_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l=1 \wedge i=1 \\ \left\{ \begin{array}{ll} 2-2^l \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l>1 \wedge i=1 \\ \left\{ \begin{array}{ll} 2^l \cdot x_t + 1 - i & \text{if } x_t \in \left[1-\frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l>1 \wedge i=2^l-1 \\ \phi_{l,i}\left(x_t \cdot 2^l - i\right) & \text{else,} \end{cases} \tag{35}$$

where $\phi_{l,i}(x)$ is again given by Eq. 2, and where the support nodes have the same coordinates as in the "classical" sparse grid. The d-dimensional basis functions are again obtained by a tensor product construction (see Eq. 6).

B Details of "Classical" Sparse Grids

This Appendix provides important formal details and references for "classical" sparse grids, as developed in [67, 16]. We first present the main result on the size of SGs and then formally state the optimization problem from which SGs arise as the solution.

Proposition 1 *The number of grid points in the sparse grid space V_n^S is given by*

$$|V_n^S| = \sum_{i=0}^{n-1} 2^i \binom{d-1+i}{d-1} = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}(n^{d-2}) \right). \tag{36}$$

For the proof of this proposition we refer the reader to Lemma 3.6 of [16]. Note that Prop. 1 directly implies that

$$|V_n^S| = \mathcal{O}(2^n \cdot n^{d-1}). \tag{37}$$

As mentioned in the main part of the paper, SGs arise from a "cost-benefit" consideration: find the approximation space $V^{opt} \subset V := \bigcup_{n=1}^{\infty} V_n$ that provides the highest accuracy for a given number of grid points. Clearly, the answer to this question depends on the class of functions we would like to interpolate. The theory of SGs considers the Sobolev space of functions with bounded second-order mixed derivatives:

$$H_2(\Omega) := \{f : \Omega \to \mathbb{R} : D^{\vec{l}} f \in L_2(\Omega), |\vec{l}|_\infty \leq 2, f|_{\partial\Omega} = 0\}, \tag{38}$$

where

$$D^{\vec{l}} f := \frac{\partial^{|\vec{l}|_1}}{\partial x_1^{l_1} \cdots \partial x_d^{l_d}} f. \tag{39}$$

Furthermore, V^{opt} certainly depends on the norm $\|\cdot\|$ in which the interpolation error is measured, and also on the seminorm used to measure the "variability" of the functions to be interpolated. Proposition 2, below, holds for the L_2 and L_∞ norms as well as for the two seminorms

$$|f|_{\alpha,2} := \left(\int_\Omega |D^\zeta f|^2 d\vec{x}\right)^{\frac{1}{2}}, \quad |f|_{\alpha,\infty} := ||D^\alpha f||_\infty, \tag{40}$$

with $\alpha = 2$. In order to leverage on the hierarchical setting introduced in Sec. 3.1, we only allow discrete spaces of the type $U_{\vec{I}} := \bigoplus_{\vec{l} \subset \vec{I}} W_{\vec{l}}$ for an arbitrary index set $\vec{I}$ as candidates for the optimization process. We use $f_{U_{\vec{I}}} \in U_{\vec{I}}$ to denote the interpolant of f from the approximation space $U_{\vec{I}}$. We are now in the position to state precisely in which sense "classical" SGs are optimal.

Proposition 2 *The sparse grid approximation space*

$$V_n^S = \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}} \tag{41}$$

is the solution to the constrained optimization problem

$$\min_{U_{\vec{I}} \subset V : |U_{\vec{I}}| \leq |V_n^S|} \quad \max_{f \in H_2(\Omega) : |f|=1} \|f - u_{U_{\vec{I}}}\|. \tag{42}$$

In words, the sparse grid V_n^S minimizes among all approximation spaces $U_{\vec{I}}$ which do not have more grid points ($|U_{\vec{I}}| \leq |V_n^S|$) the maximal approximation error reached when interpolating functions with bounded second-order mixed derivatives ($f \in H_2(\Omega)$) and a given variability ($|f| = 1$). Proposition 2 is implied by the results in Sec. 3.2 of [16], who show, in particular, that V_n^S is the interpolation space that includes $W_{\vec{l}}$ with $\vec{l} = (n, 1, \ldots, 1)$ and all other increment spaces with equal or better "cost-benefit" ratio (between points and gained accuracy). Finally, note that "classical" SGs are also the solution to the reverse optimization problem of achieving some desired accuracy with the smallest possible number of grid points.

C Implementation Details

Error Measure

To evaluate the accuracy of our solutions we follow the previous literature (see, e.g., [42]) and compute (unit-free) errors in the $N + 1$ equilibrium conditions. In particular, there is one Euler equation error for each country $j \in \{1, ..., N\}$ and one error from the aggregate resource constraint. The (unit-free) Euler equation error for each country $j \in \{1, ..., N\}$ is—if the irreversibility constrraint is not binding today—given by

$$EE^j = \left[\beta \mathbb{E}_t \left\{\lambda_{t+1} \left[a_{t+1}^j A\zeta(k_{t+1}^j)^{\zeta-1} + 1 - \delta + \frac{\phi}{2} g_{t+2}^j \left(g_{t+2}^j + 2\right)\right] - (1-\delta)\mu_{t+1}^j\right\}\right] \cdot \left[\lambda_t \cdot \left(1 + \phi \cdot g_{t+1}^j\right)\right]^{-1} - 1, \tag{43}$$

and the error from the aggregate resource constraint is

$$\sum_{j=1}^{N}\left(a_t^j \cdot A \cdot (k_t^j)^\zeta + k_t^j \cdot \left((1-\delta) - \frac{\phi}{2}\cdot (g_{t+1}^j)^2\right) - k_{t+1}^j - \left(\frac{\lambda_t}{\tau_j}\right)^{-\gamma^j}\right) \cdot \left(\sum_{j=1}^{N}\left(a_t^j \cdot A \cdot (k_t^j)^\zeta + k_t^j \cdot \left(-\frac{\phi}{2}\cdot (g_{t+1}^j)^2\right)\right)\right)^{-1}. \quad (44)$$

These errors measure how far from optimal the combination of today's policy and the next period's policy is, when using the computed equilibrium policy function to calculate both. We compute these errors for $10,000$ points in the state space—visited during a long simulation path (for the smooth model) or drawn from a uniform distribution over the state space (in the case of the non-smooth model). For each of these points we get $N+1$ errors. We thus end up with $(N+1)\cdot 10,000$ errors of which we take both the maximum (Max. Error) and the average (Avg. Error), both of which we report in $\log_{10}$-scale.[17]

However, the presence of the occasionally binding irreversibility constraints complicates defining reasonable error measures. Suppose, for instance, that the optimal policy implies a binding constraint while the computed policy implies that the constraint is just very close to being binding. In such a case, the error in the Euler equation might be large while the computed policy can in fact only be improved slightly by moving from almost binding to binding. To take care of this problem, we define the percentage violation of the irreversibility constraint by

$$IC^j \equiv 1 - \frac{k_{t+1}^j}{k_t^j \cdot (1-\delta)} \quad (45)$$

and the error by

$$\max\left(EE^j, IC^j, \min\left(-EE^j, -IC^j\right)\right). \quad (46)$$

The first term within the max operator (namely, EE^j as defined in Eq. 43) is positive when the marginal cost of investing in country j today is lower than 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 today is higher than the discounted marginal benefit 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, if $-EE^j$ is positive, the error is not simply given by $-EE^j$ but by $\min\left(-EE^j, -IC^j\right)$.

Integration

One important detail of the implementation is the integration procedure used to evaluate the expectations operator, for example in Eq. 29. As we want to focus on the grid structure, we chose an integration rule that is simple and fast, yet not very accurate. In particular, we use a simple monomial rule that uses just two evaluation points per shock—that is to say,

[17]In the case of the non-smooth IRBC model the "Max. Error" is defined as the 99.9 percent quantile of the error distribution. In high-dimensional settings, occasionally binding constraints become increasingly harder to approximate and therefore the error distribution gradually becomes more fat-tailed. By removing the worst 0.1% of the Euler errors, we mitigate this effect. This "truncation" makes the "Max. Error" less dependent on individual draws and thus results in an error measure that is more comparable across dimensions and grid specifications.

$2(N+1)$ points in total (see, e.g., [39], Sec. 7.5). As we apply the same rule along the time iteration algorithm and 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 [51]), yet not over the same space that the policy functions are interpolated on. While this would make the use of sparse grid interpolation cumbersome, there is another drawback in the case of our application. When used to interpolate Gaussian densities, sparse grids of relatively low level l generally return probability densities with negative values (see, e.g. [58], Sec. 4.2), which results in inaccurate approximations of the integral. For these two reasons, we do not use sparse grid integration.

Acceleration

Another implementation detail that we would like to discuss is the possibility of accelerating the time iteration procedure by starting with coarse grids and later changing to finer grids. For instance, using "classical" SGs, the overall computation time can be reduced by one order of magnitude when using a level-2 grid for 200 iterations, followed by 80 iterations on a level-3 grid, before finally using a level-4 grid for 20 iterations instead of running 300 iterations with a grid of level 4. Our tests indicate that this approach yields the same accuracy as if the entire computation would have been carried out at level 4. As with integration, it is not the focus of this paper to discuss the most efficient acceleration strategy. Of course, when we compare results, they are achieved with comparable acceleration strategies.

D Scaling and Resource Requirements

In this Appendix, we report the strong scaling efficiency of our algorithm and discuss the resource requirements needed to solve high-dimensional models.

Our scaling tests were carried out on the 5,272-node Cray XC30 "Piz Daint" system installed at the Swiss National Supercomputing Centre (CSCS).[18] In Fig. 14, we display the speedup S_p and the efficiency E_p of the code [60]. These two quantities are defined by

$$S_p = \frac{T_1}{T_p}, \qquad E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}, \tag{47}$$

where T_1 is the execution time of the test benchmark and T_p is the execution time of the algorithm running with a multiple of p-times the processes of the benchmark. The test problem is a single time iteration step of a 16-dimensional IRBC model. In order to provide a consistent benchmark, we used an SG of level 6. It has a total of 3,183,561 variables per time step.[19] The test case was solved with increasingly larger numbers of nodes (from 1 to 2,048 nodes). Fig. 14 shows the execution time and the scaling on different refinement levels and their ideal speedups. We used one MPI process per SandyBridge node, of which each offloads part of the function evaluation to the GPU. The code scales nicely to about 2,048 nodes, as shown in Fig. 14. Even at this large node count, the overall efficiency of our code is above 60 percent. Thus, combined with the on-node shared memory parallelism, we obtain an overall speedup of more than four orders of magnitude in this example compared to running this numerical experiment on a single CPU.

We now discuss the resources needed to solve high-dimensional models with our ASG algorithm, starting with the smooth IRBC model. Clearly, the number of grid points increases

[18] "Piz Daint" is a supercomputer with a peak performance of 7.7 petaflops. Its compute nodes combine 8-core Intel Xeon E5-2670 (SandyBridge) CPUs with one NVIDIA Tesla K20X GPU. Our code is compiled with GNU compilers and CUDA Toolkit 6.5.

[19] The sparse grid under consideration consists of 353,729 points with $N+1=9$ unknown policy variables at each point.

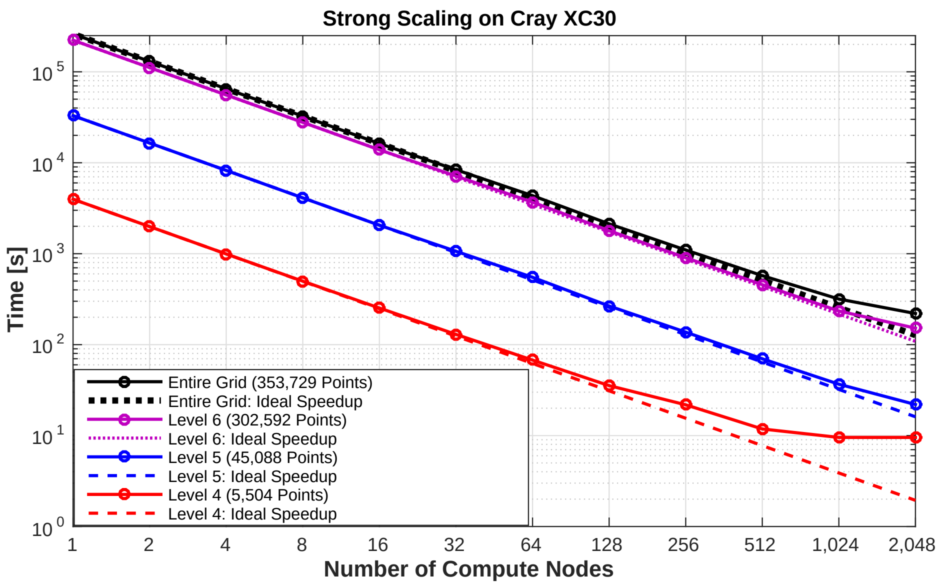

Figure 14: Strong scaling plots on Piz Daint for a 16-dimensional IRBC model using six levels of grid refinement and in total 353,729 points and 3,183,561 unknowns. "Entire Grid" shows the entire computation time as a function of the number of compute nodes employed. We also show execution times for the computational subcomponents on different refinement levels, e.g., for level 6 using 302,592 points, level 5 using 45,088 points, and level 4 using 5,504 points. Dashed lines show ideal speedups.

as the dimension of the problem goes up. However, as we show in the left panel of Fig. 15, this increase is far from exponential, which is the defining feature of sparse grids. Concerning CPU time, the right panel of Fig. 15 shows that it substantially increases as we go to higher dimensions. This effect, however, is known and has been previously reported, for example by [41, 53] who both used Smolyak sparse grids with global polynomials. We also observe that the compute time grows more rapidly than the number of grid points. This effect is largely driven by two factors. First, the memory access time required to interpolate on a sparse grid grows substantially when the dimension increases, as shown in great detail in [55]. Second, the size of the nonlinear equation systems scales up linearly with dimension, and thus exacerbates the computational burden. While computing the solution to a 4-dimensional model with an SG of level 3 consumes about 0.0003 node hours, roughly 560 node hours are needed for a 100-dimensional smooth IRBC model. All in all, however, the increase in CPU time is less than exponential as the right panel of Fig. 15 clearly shows. Thus, we can—at least to some extent—control the increasing running times by using a larger number of compute nodes (see Sec. 4.3).

We now turn our attention to the non-smooth IRBC models. Regarding solution time, it is clear that these models present a considerably larger burden compared to the smooth IRBC models. One reason for this is that the nonlinear system of equations to be solved is of size $2N + 1$ (see Eqs. 29 and 30), compared to $N + 1$ in the case of the smooth IRBC models. Moreover, the (local) resolution needs to be considerably higher in order to achieve decent errors (see Tab. 5), resulting in more grid points when comparing the smooth and non-smooth IRBC models of fixed dimension. Both factors increase the total computational burden of the problem substantially. The required CPU time for solving a 2-country model now consumes

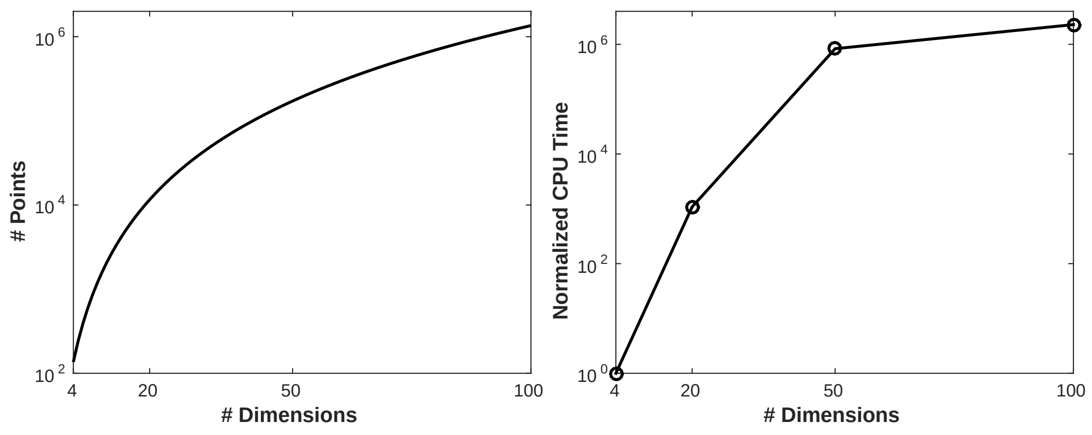

Figure 15: The left panel shows the number of grid points (when an SG of level 3 is used) as a function of the dimensionality. The right panel shows CPU time (normalized to the case $d = 4$, which takes 0.0003 hours on a single compute node) as a function of the dimensionality of the problem for such SGs. The y-axes are log-scale. The concave shape of these graphs thus implies that growth is less than exponential.

about 0.004 node hours, while a 20-dimensional model may require up to 160 node hours—a number that is comparable to solving a 50-dimensional smooth IRBC model. However, this is still not a roadblock to solving such complex models. CPU time again increases less than exponentially in the dimension of the problem. Therefore, we can still compute solutions of high-dimensional, non-smooth models in a reasonable time by using acceleration methods (see Appendix C) and employing a large number of nodes. For instance, using 256 nodes, we obtain results in less than one hour, even for the largest non-smooth models considered here.

References

[1] R. Aiyagari. Uninsured idiosyncratic risk and aggregate saving. *The Quarterly Journal of Economics*, 109(3):659–684, 1994.

[2] E. M. Aldrich, J. Fernández-Villaverde, A. R. Gallant, and J. F. Rubio-Ramírez. Tapping the supercomputer under your desk: Solving dynamic equilibrium models with graphics processors. *Journal of Economic Dynamics and Control*, 35(3):386–393, 2011.

[3] F. Alvarez, H. Le Bihan, and F. Lippi. The real effects of monetary shocks in sticky price models: A sufficient statistic approach. *American Economic Review*, 106(10):2817–51, 2016.

[4] F. Alvarez and F. Lippi. Price setting with menu cost for multiproduct firms. *Econometrica*, 82(1):89–135, 2014.

[5] F. Alvarez and F. Lippi. Price plans and the real effects of monetary policy. *Working Paper*, 2016.

[6] C. Arellano. Default risk and income fluctuations in emerging economies. *The American Economic Review*, 98(3):690–712, 2008.

[7] F. Barillas and J. Fernndez-Villaverde. A generalization of the endogenous grid method. *Journal of Economic Dynamics and Control*, 31(8):2698–2712, 2007.

[8] V. Barthelmann, E. Novak, and K. Ritter. High dimensional polynomial interpolation on sparse grids. *Advances in Computational Mathematics*, 12:273–288, 2000.

[9] N. Bell and J. Hoberock. Thrust: A productivity-oriented library for CUDA. *GPU Computing Gems*, 7, 2011.

[10] S. Bhattarai and R. Schoenle. Multiproduct firms and price-setting: Theory and evidence from US producer prices. *Journal of Monetary Economics*, 66:178–192, 2014.

[11] O. Bokanowski, J. Garcke, M. Griebel, and I. Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. *Journal of Scientific Computing*, 55(3):575–605, 2013.

[12] J. Brumm and M. Grill. Computing equilibria in dynamic models with occasionally binding constraints. *Journal of Economic Dynamics and Control*, 38:142–160, 2014.

[13] J. Brumm, M. Grill, F. Kubler, and K. Schmedders. Collateral requirements and asset prices. *International Economic Review*, 56(1):1–25, 2015.

[14] J. Brumm, D. Mikushin, S. Scheidegger, and O. Schenk. Scalable high-dimensional dynamic stochastic economic modeling. *Journal of Computational Science*, 11:12–25, 2015.

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

[16] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.

[17] 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*, 236(15):3741–3750, 2012.

[18] Y. Cai, K. Judd, G. Thain, and S. Wright. Solving dynamic programming problems on a computational grid. *Computational Economics*, 45(2):261–284, 2015.

[19] Y. Cai, K. L. Judd, and T. S. Lontzek. The Social Cost of Carbon with Economic and Climate Risks. *ArXiv e-prints*, 2015.

[20] A. S. Caplin, D. F. Spulber, et al. Menu costs and the neutrality of money. *The Quarterly Journal of Economics*, 102(4):703–725, 1987.

[21] C. D. Carroll. The method of endogenous gridpoints for solving dynamic stochastic optimization problems. *Economics Letters*, 91(3):312–320, 2006.

[22] S. Chatterjee, D. Corbae, M. Nakajima, and J.-V. Ríos-Rull. A quantitative theory of unsecured consumer credit with risk of default. *Econometrica*, 75(6):1525–1589, 2007.

[23] W. J. Den Haan, K. L. Judd, and M. 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, 2011.

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

[25] M. Eichenbaum, N. Jaimovich, and S. Rebelo. Reference prices, costs, and nominal rigidities. *The American Economic Review*, 101(1):234–262, 2011.

[26] G. Fella. A generalized endogenous grid method for non-smooth and non-concave problems. *Review of Economic Dynamics*, 17(2):329–344, 2014.

[27] J. Fernández-Villaverde, G. Gordon, P. Guerrón-Quintana, and J. F. Rubio-Ramirez. Non-linear adventures at the zero lower bound. *Journal of Economic Dynamics and Control*, 57:182–204, 2015.

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

[29] A. 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, 1984.

[30] M. Gertler and J. Leahy. A Phillips curve with an Ss foundation. *Journal of Political Economy*, 116(3):533–572, 2008.

[31] M. Golosov and R. E. Lucas Jr. Menu costs and phillips curves. *Journal of Political Economy*, 115(2):171–199, 2007.

[32] M. Gunzburger, C. G. Webster, and G. Zhang. *An Adaptive Wavelet Stochastic Collocation Method for Irregular Solutions of Partial Differential Equations with Random Input Data*, pages 137–170. Springer, Cham, 2014.

[33] C. Hager, S. Heber, and B. Wohlmuth. Numerical techniques for the valuation of basket options and its greeks. *Journal of Computational Finance*, 13(4):1–31, 2010.

[34] M. Heene, C. Kowitz, and D. Pflüger. Load balancing for massively parallel computations with the sparse grid combination technique. In *PARCO*, pages 574–583, 2013.

[35] A. Heinecke and D. Pflger. Emerging architectures enable to boost massively parallel data mining using adaptive sparse grids. *International Journal of Parallel Programming*, 41(3):357–399, 2013.

[36] F. Heiss and V. Winschel. Likelihood approximation by numerical integration on sparse grids. *Journal of Econometrics*, 144(1):62–80, 2008.

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

[38] M. Holtz. *Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance*. Lecture Notes in Computational Science and Engineering. Springer, Dordrecht, 2011.

[39] K. L. Judd. *Numerical methods in economics*. The MIT press, 1998.

[40] K. L. Judd, L. Maliar, and S. Maliar. Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. *Quantitative Economics*, 2(2):173–210, 2011.

[41] K. L. Judd, L. Maliar, S. Maliar, and R. Valero. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. *Journal of Economic Dynamics and Control*, 44:92–123, 2014.

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

[43] P. Kehoe and V. Midrigan. Prices are sticky after all. *Journal of Monetary Economics*, 75:35–53, 2015.

[44] A. Khan and J. K. Thomas. Credit shocks and aggregate fluctuations in an economy with production heterogeneity. *Journal of Political Economy*, 121(6):1055–1107, 2013.

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

[46] R. Kollmann, S. Maliar, B. A. Malin, and P. Pichler. Comparison of solutions to the multi-country real business cycle model. *Journal of Economic Dynamics and Control*, 35(2):186–202, 2011.

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

[48] P. Krusell and A. A. Smith, Jr. Income and wealth heterogeneity in the macroeconomy. *Journal of Political Economy*, 106(5):867–896, 1998.

[49] I. Livshits, J. MacGee, and M. Tertilt. Consumer bankruptcy: A fresh start. *The American Economic Review*, 97(1):402–418, 2007.

[50] D. 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.

[51] X. Ma and N. Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. *Journal of Computational Physics*, 228(8):3084–3113, 2009.

[52] L. Maliar and S. Maliar. Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model. *Quantitative Economics*, 6(1):1–47, 2015.

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

[54] V. Midrigan. Menu costs, multiproduct firms, and aggregate fluctuations. *Econometrica*, 79(4):1139–1180, 2011.

[55] A. Murarasu, J. Weidendorfer, G. Buse, D. Butnaru, and D. Pflüeger. Compact data structure and parallel algorithms for the sparse grid technique. In *16th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2011.

[56] E. Nakamura and J. Steinsson. Monetary non-neutrality in a multisector menu cost model. *The Quarterly Journal of Economics*, 125(3):961–1013, 2010.

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

[58] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems.* PhD thesis, München, 2010.

[59] D. Pflüger, B. Peherstorfer, and H.-J. Bungartz. Spatially adaptive sparse grids for high-dimensional data-driven problems. *Journal of Complexity*, 26(5):508–522, 2010.

[60] T. Rauber and G. Rünger. *Parallel Programming: for Multicore and Cluster Systems.* Springer, 2010.

[61] J. Reinders. *Intel Threading Building Blocks.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2007.

[62] A. Skjellum, W. Gropp, and E. Lusk. *Using MPI.* MIT Press, 1999.

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

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

[65] A. Waechter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[66] V. Winschel and M. Kraetzig. Solving, estimating, and selecting nonlinear dynamic models without the curse of dimensionality. *Econometrica*, 78(2):803–821, 2010.

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