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

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

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

May 4, 2015

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 piecewise multi-linear basis functions. With increasing dimensions, sparse grids grow much slower than standard tensor product grids. In addition, the grid scheme is automatically refined locally and can thus capture steep gradients and non-differentiabilities. To further increase the maximum problem size we can handle, our implementation is fully hybrid parallel: it uses a combination of distributed and shared memory parallelization schemes. This parallelization enables us to efficiently use high-performance computing architectures. To demonstrate its capabilities, we apply our method to two very different dynamic models: First, high-dimensional international real business cycle models with capital adjustment costs and irreversible investment. Second, multi-good menu-cost models with temporary sales and economies of scope in price-setting.

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

JEL Classification: C63, C68, F41

*We thank the co-editor Lars Peter Hansen and three anonymous referees for their extremely valuable comments. We are very grateful to Felix Kubler for helpful discussions and support. We thank Daniel Harenberg, Ken Judd, Karl Schmedders, and seminar participants at the University of Zürich, Stanford University, the University of Chicago, Argonne National Laboratory, Cornell University, USI Lugano, CEF 2013 in Vancouver, and at the BYU Computational Public Economics conference 2014 for valuable comments. Moreover, we thank Xiang Ma for very instructive e-mail discussions regarding the workings of adaptive sparse grids. We are grateful for the support of Olaf Schenk and the University of Zurich's S3IT team concerning HPC related issues. This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID s555. Additionally, we acknowledge CPU time granted on the University of Zurich's "Schrödinger" HPC cluster. Johannes Brumm gratefully acknowledges financial support from the ERC.

1 Introduction

Many important economic phenomena can only be captured by models 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. Most recently, this has become obvious in the financial crisis of 2008 with its large price fluctuations and tremendous spillover effects. Yet already in the nineties, more and more economists started to use global solution methods[1] and also included various kinds of heterogeneity in their models (see, e.g., [53, 40, 1, 38]). However, solving for the global solution of general-equilibrium rational-expectations model with substantial heterogeneity is very costly: using conventional solution methods, the computation time and storage requirements increase exponentially with the amount of heterogeneity—that is, with the dimensionality of the problem.

This paper demonstrates that global solutions of economic models can be computed accurately and fast even when the amount of heterogeneity is substantially larger than in the previous literature. We achieve this by employing a highly parallel implementation of a so-called adaptive sparse grid within an iteration algorithm. 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 models with up to 100 dimensions, and also for at least 20-dimensional models with non-differentiabilities.

Standard grid-based algorithms suffer from the curse of dimensionality: Starting with a one-dimensional discretization scheme that employs N grid points, a straightforward extension to d dimensions leads to N^d grid points. In contrast, sparse grids are able to alleviate this curse by reducing the number of grid points from the order $\mathcal{O}\left(N^d\right)$ to $\mathcal{O}\left(N \cdot (\log N)^{d-1}\right)$ with only slightly deteriorated accuracy if the underlying function is sufficiently smooth (see, e.g., [13], and references therein).

The sparse grid construction we are using was introduced by Zenger [56] for the solution of partial differential equations. However, the underlying principle, a sparse tensor product decomposition, goes back to the seminal work of Smolyak [52]. Sparse grids have been applied to a wide range of different research fields such as physics, visualization, data mining, and mathematical finance (see, e.g., [9, 13, 22, 26, 28, 30, 44]). Using the original formulation of Smolyak [52], Krueger and Kubler [37] were the first to solve dynamic economic models using sparse grids, while Winschel and Kraetzig [55] embed this method into a Bayesian estimation framework. In recent work, Judd et al. [32] 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. In contrast, our algorithm can accommodate non-smooth behavior. The reason for this is that we use hierarchical basis functions with an adaptive grid refinement strategy as in Ma and Zabaras [41]. More specifically, our basis functions are piecewise multi-linear functions with local support. The space of these basis functions is hierarchically structured into refinement levels. For a basis function at a given level l, there are several basis functions of the next finer level $l+1$ among which the support of the original function of level l is subdivided. Using the value of a level l function, an automatic grid adaptation algorithm decides whether the interpolation is to be refined locally, in which case the associated functions

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

of level $l+1$ are added. Importantly, this refinement scheme scales linearly with increasing dimension (see, e.g., [41, 47, 46]). Such an adaptive sparse grid with hierarchical local basis functions offers the promise of an efficient and accurate solution of economic problems that are both high-dimensional and non-smooth.

However, such problems often require substantial computation time even if an efficient solution method is applied. Therefore, in order to achieve a fast time-to-solution process, our implementation is able to access high-performance computing (HPC) facilities, whose performance nowadays reaches multiple petaflops (see [19]). A typical HPC hardware design features thousands of compute nodes each consisting of multiple central processing units (CPUs) with attached graphic processing units (GPUs). Hence, efficient parallel programming requires the use of multiple computational units by combining distributed memory parallelization among different nodes with shared memory parallelization inside each node (see, e.g., [48]). We address this challenge with a "hybrid" parallelization scheme that uses Message Passing Interface (MPI) (see [51]) and Threading Building Blocks (TBB) (see [50]). Moreover, it partially offloads the function evaluations to GPUs using CUDA/Thrust (see [7]). Our large scale numerical experiments performed on the 'Piz Daint' (Cray XC30) machine from the Swiss National Supercomputing Centre (CSCS) show that we are able to efficiently use at least $2,000$ nodes.[2]

To show that our algorithm can solve standard high-dimensional economic problems, we solve an international real business cycle (IRBC) model with adjustment costs. For this application the performance of alternative solution methods is well documented in a recent coordinated study (see [18, 33, 36]). Like many of these established methods, we use a time iteration[3] 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. As the policy functions in this model are very smooth, our linear interpolation scheme is at a disadvantage compared to global polynomial interpolation. Nevertheless, we can compute quite accurate solutions for models that are of a higher dimension than any that have been reported in the comparison study cited above. But the main purpose and comparative advantage of our algorithm lies in solving models that exhibit non-smooth behavior. To demonstrate its performance with respect to such models, we augment the IRBC model with irreversible investment. Despite the resulting non-differentiabilities (also called "kinks") in the policy functions, we are still able to compute accurate solutions for high-dimensional examples. The adaptivity of the grid now ensures that grid points are placed close to the kinks while the grid is coarse where policy functions are smoother.

To emphasize the broad applicability of adaptive sparse grids in dynamic economics, we additionally consider an application that differs in important ways from the IRBC model: A menu-cost model where multi-product firms face economies of scope in price-setting and have the possibility to set temporary prices (see Midrigan [43]). Due to the fixed costs of adjusting prices (so-called menu costs), firms make discrete choices. As a consequence, we have to solve the model by value function iteration and to keep track of the values associated with each of the discrete choices possible. In this context, adaptive sparse grids with piecewise multilinear hierarchical basis functions are a particularly suitable choice for interpolation, as discrete choices typically induce kinks in value functions. The menu-cost application thus illustrates that our method can also be combined with value function iteration and can handle discrete choices, which are central to many other important economic applications, for instance models of consumer default or sovereign default (see, e.g., [17, 39] and [4], respectively). In our menu-

[2]The CSCS's Cray XC30 that is used in our numerical experiments in Sec. 3.3 consists of Intel Xeon E5 processors with NVIDIA Tesla K20X GPUs attached to it. Its peak performance is 7.7 petaflops.

[3]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., [31]).

cost application, we show that extending the number of products relative to Midrigan [43] can substantially improve the fit to the high kurtosis of price changes found in the data.

In addition to the above-mentioned literature on sparse grids in mathematics and economics, our paper is closely related to two other strands of the literature. First, to papers that develop methods for solving dynamic economic models with occasionally binding constraints. The endogenous grid point method proposed by Carrol [16] defines a grid on next period's variables resulting in an "enodgenous" grid on current period variables, thereby avoiding the root finding step in solving the Euler equation. Moreover, this method can also place grid points at non-differentiabilities, as extensions by Barillas and Villaverde [5], Hintermaier and Koeniger [29], and Fella [20] show. An altenative method that is more flexible, yet does not avoid the root-finding step, is the adaptive simplicial interpolation by Brumm and Grill [10], 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. Second, our paper is part of the emergent literature on parallel computing applications in economics (see, e.g., [15, 2, 14]). To the best of our knowledge, we are the first to efficiently use current high-performance computing technology to solve dynamic economic models. We are able to do so as our implementation is fully hybrid parallel.

The remainder of the paper is organized as follows. In Section 2, we explain the construction of adaptive sparse grids and also provide simple test cases. In Section 3, we first embed adaptive sparse grid interpolation in a time iteration algorithm to solve high-dimensional IRBC models, both without and with irreversible investment. We go on to discuss how hybrid parallelization can speed up the computations and then report the performance of this algorithm. Finally, Section 3 also presents the application of our method to a menu-cost model. Section 4 concludes.

2 From Full Grids to Adaptive Sparse Grids

After introducing some notation, this section proceeds in four main steps. First, we present piecewise linear hierarchical basis functions in one dimension. Second, we extend such bases to multiple dimensions via a tensor product construction. Third, we show how "classical" sparse grids avoid the curse of dimensionality. Fourth, 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. Finally, we provide examples of functions with steep gradients and non-differentiabilities where adaptive sparse grids outperform "classical" sparse grids by far.

2.1 Notation

We first introduce some notation and definitions that we will require later (see [13, 22]). We will focus on the domain $\Omega = [0,1]^d$, where d is the dimensionality of the problem. Most domains occurring in economics can be transformed into such a cube by proper rescaling.
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

$$h_{\vec{l}} := (h_{l_1}, ..., h_{l_d}) = 2^{-\vec{l}} := \left(2^{-l_1}, ..., 2^{-l_d}\right), \tag{1}$$

and generic grid point

$$\vec{x}_{\vec{l},\vec{i}} := (x_{l_1,i_1}, ..., x_{l_d,i_d}), \tag{2}$$

where $x_{l_t,i_t} := i_t \cdot h_{l_t} = i_t \cdot 2^{-l_t}$, $i_t \in \{0, 1, ..., 2^{l_t}\}$, and $t \in \{1, \ldots, d\}$.[4] Along each dimension, the grid is equidistant. However, the mesh sizes, h_{l_t}, may differ across dimensions, in which case the grid is called anisotropic.
In addition, when dealing with d-dimensional multi-indices such as $\vec{l}$, we use relational operators component-wise,

$$\vec{l} \leq \vec{k} \Leftrightarrow l_t \leq k_t, \forall t \in \{1, \ldots, d\}. \tag{3}$$

Finally, we use the l_1-norm, $|\vec{l}|_1$, and the maximum norm, $|\vec{l}|_\infty$, given by

$$|\vec{l}|_1 := \sum_{t=1}^{d} l_t, \ |\vec{l}|_\infty := \max_{1 \leq t \leq d} l_t. \tag{4}$$

2.2 Hierarchical Basis Functions in One Dimension

The sparse grid method introduced below is based on a hierarchical decomposition of the underlying approximation space. Such a hierarchical structure is convenient both for local adaptivity (see Sec. 2.5) and for the use of parallel computing (see Sec. 3.3). We now explain this hierarchical structure starting with the one-dimensional case—that is, $\Omega = [0, 1]$.

Let us assume that a function $f : \Omega \to \mathbb{R}$ of interest is sufficiently smooth (see, e.g., [13]). For the time being we also assume that the function f vanishes at the boundary (i.e. $f|_{\partial\Omega} = 0$). We delegate the treatment of non-zero boundaries to Appendix A. An interpolation formula is then given by

$$f(x) \approx u(x) := \sum_i \alpha_i \phi_i(x) \tag{5}$$

with coefficients α_i and a set of appropriate piecewise linear basis functions $\phi_i(x)$. One standard approach is to use hat functions

$$\phi(x) = \begin{cases} 1 - |x| & \text{if } x \in [-1, 1] \\ 0 & \text{else} \end{cases} \tag{6}$$

to generate a family of basis functions $\phi_{l,i}$ with support $[x_{l,i} - h_l, x_{l,i} + h_l]$ by dilation and translation—that is,

$$\phi_{l,i}(x) := \phi\left(\frac{x - i \cdot h_l}{h_l}\right). \tag{7}$$

This basis is called *nodal basis*, and the respective nodal function spaces are

$$V_l := \text{span}\{\phi_{l,i} : 1 \leq i \leq 2^l - 1\}. \tag{8}$$

The hierarchical increment spaces W_l are defined by

$$W_l := \text{span}\{\phi_{l,i} : i \in I_l\}, \tag{9}$$

using the index set

$$I_l = \{i \in \mathbb{N}, 1 \leq i \leq 2^l - 1, i \text{ odd}\}. \tag{10}$$

The nodal spaces V_l are the direct sum of the hierarchical increment spaces W_l

$$V_l = \bigoplus_{k \leq l} W_k. \tag{11}$$

[4] A concrete example for $d = 2$ is the following: let $\vec{l} = (2, 2)$, $\vec{i} = (1, 2)$. Then, $\vec{x}_{\vec{l},\vec{i}} = (1 \cdot 0.25, 2 \cdot 0.25) = (0.25, 0.5)$.

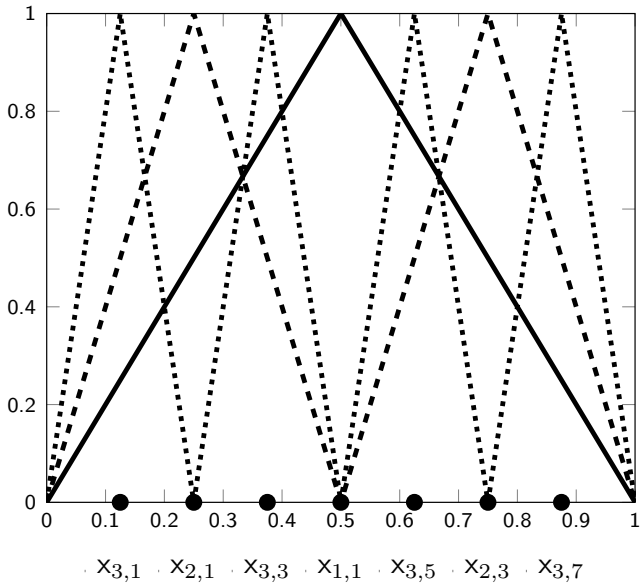

Figure 1: Hierarchical basis functions of V_3; level 1 (solid), level 2 (dashed), level 3 (dotted).

Fig. 1 shows the first three levels of these hierarchical, piecewise linear basis functions. Using this basis, a function f is approximated by a unique $u \in V_l$ with coefficients $\alpha_{k,i} \in \mathbb{R}$:

$$f(x) \approx u(x) = \sum_{k=1}^{l} \sum_{i \in I_k} \alpha_{k,i} \cdot \phi_{k,i}(x). \tag{12}$$

Note that the basis functions $\phi_{k,i}$ spanning W_k have mutually disjoined support, as can be seen in Fig. 1. The coefficients $\alpha_{k,i}$ in Eq. 12 can easily be determined due to the nested structure of the hierarchical grid: The set of points X^{l-1} at level $l-1$ is contained in X^l (i.e. $X^{l-1} \subset X^l$). The hierarchical coefficients $\alpha_{l,i}$, $l \geq 1$, i odd are given by [13, 22]:

$$\begin{aligned} \alpha_{l,i} &= f(x_{l,i}) - \frac{f(x_{l,i} - h_l) + f(x_{l,i} + h_l)}{2} \\ &= f(x_{l,i}) - \frac{f\left(x_{l-1,(i-1)/2}\right) + f\left(x_{l-1,(i+1)/2}\right)}{2}. \end{aligned} \tag{13}$$

In operator form, Eq. 13 can conveniently be rewritten as

$$\alpha_{l,i} = \left[-\frac{1}{2} \;\; 1 \;\; -\frac{1}{2}\right]_{l,i} f. \tag{14}$$

Note that the coefficients $\alpha_{l,i}$ are called *hierarchical surpluses* [13] since they correct the interpolant of level $l-1$ at the point $x_{l,i}$ to the actual value of $f(x_{l,i})$, as displayed in Fig. 2.

2.3 Hierarchical Basis Functions in Multiple Dimensions

The one-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 as well.

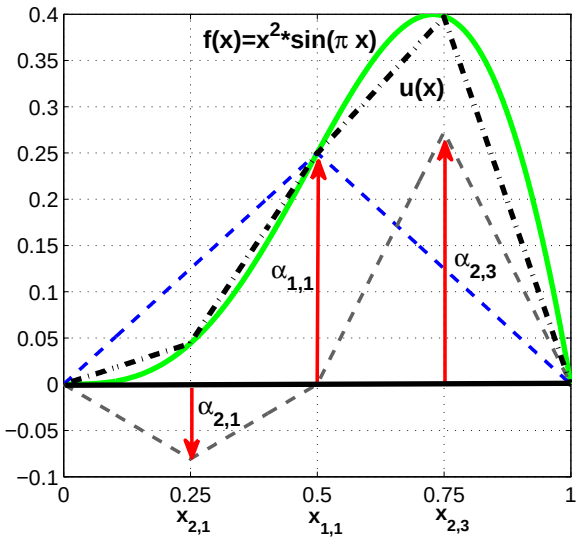

Figure 2: Construction of $u(x)$ interpolating $f(x) = x^2 \cdot \sin(\pi \cdot x)$ with hierarchical linear basis functions of levels 1 and 2. The hierarchical surpluses $\alpha_{l,i}$ associated with the respective basis functions are indicated by arrows (cf. Eq. 13). They are simply the difference between the function values at the current and the previous interpolation levels.

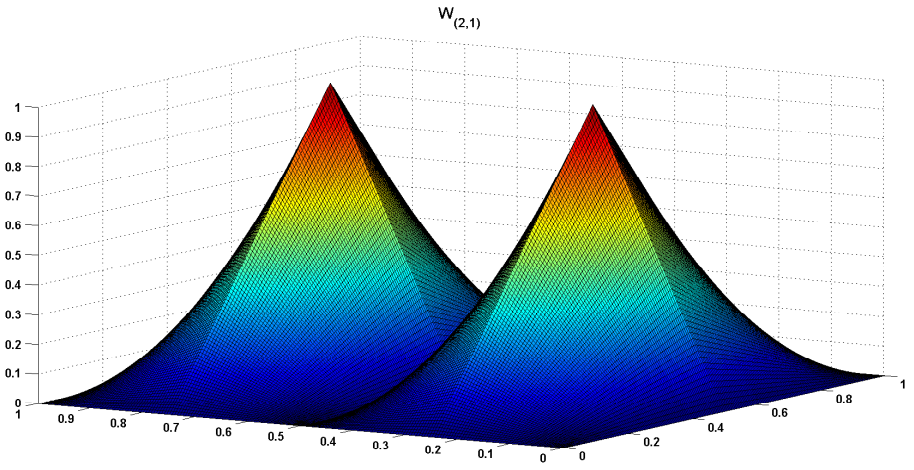

Figure 3: The 2-dimensional Basis functions of the hierarchical increment space $W_{(2,1)}$.

For each grid point, $\vec{x}_{\vec{l},\vec{i}}$, an associated piecewise d-linear basis function $\phi_{\vec{l},\vec{i}}(\vec{x})$ is defined as the product of the one-dimensional basis functions (see Eq. 6):

$$\phi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{t=1}^{d} \phi_{l_t,i_t}(x_t). \tag{15}$$

These basis functions are then used to define the function spaces $V_{\vec{l}}$ consisting of piecewise linear functions on Ω (with $f|_{\partial\Omega} = 0$):

$$V_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{1} \leq \vec{i} \leq 2^{\vec{l}} - \vec{1}\}. \tag{16}$$

The index set $I_{\vec{l}}$ is given by

$$I_{\vec{l}} := \{\vec{i} : 1 \leq i_t \leq 2^{l_t} - 1, i_t \text{ odd}, 1 \leq t \leq d\}. \tag{17}$$

and the hierarchical increment spaces are defined as

$$W_{\vec{l}} := \text{span}\{\phi_{\vec{l},\vec{i}} : \vec{i} \in I_{\vec{l}}\}. \tag{18}$$

An example of such an increment space is shown in Fig. 3. These hierarchical increment spaces now allow us to define a multilevel space decomposition. In line with the sparse grid literature (see, e.g., [46, 22, 13]), we define $V_n := V_{(n,\ldots,n)}$ as a direct sum of spaces. Consequently, the hierarchical increment spaces $W_{\vec{l}}$ are related to the nodal spaces $V_{\vec{l}}$ of piecewise d-linear functions with mesh width h_l in each dimension by

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

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

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

with $f_{\vec{l}} \in W_{\vec{l}}$ and $\alpha_{\vec{l},\vec{i}} \in \mathbb{R}$. The hierarchical surpluses are given by

$$\alpha_{\vec{l},\vec{i}} = \left(\prod_{t=1}^{d} \left[-\frac{1}{2} \quad 1 \quad -\frac{1}{2} \right]_{l_t,i_t} \right) f. \tag{21}$$

For a sufficiently smooth function f (which we will state more precisely in Sec. 2.4) and its interpolant $u \in V_n$ [13], we obtain an asymptotic error decay of

$$\| f(\vec{x}) - u(\vec{x}) \|_{L_2} \in \mathcal{O}\left(h_n^2\right) = \mathcal{O}\left(2^{-2n}\right), \tag{22}$$

but at the cost of

$$\mathcal{O}\left(h_n^{-d}\right) = \mathcal{O}\left(2^{nd}\right) \tag{23}$$

grid points, encountering the so-called *curse of dimensionality*. The exponential dependence of the overall computational effort on the number of dimensions is a prohibitive obstacle for the numerical treatment of high-dimensional problems. The curse of dimensionality typically prohibits an accurate solution of problems with more than 4 or 5 dimensions. A 10-dimensional problem with a resolution of 15 points in each dimension (i.e. $n = 4$) needs $0.58 \cdot 10^{12}$ coefficients, which already brings us to the capacity limits of today's most advanced computer systems [19].

2.4 Classical Sparse Grids

To alleviate the curse of dimensionality (see Sec. 2.3) 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 (see, e.g., [56, 13]). The "classical" sparse grid construction arises from a "cost-benefit" analysis (see, e.g., [56, 22, 13], and references therein) in function approximation. In particular, we consider the Sobolev space of functions with bounded second-order mixed derivatives

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

where

$$D^{\vec{l}} f := \frac{\partial^{|\vec{l}|_1}}{\partial x_1^{l_1} \cdots \partial x_d^{l_d}} f. \tag{25}$$

For functions in $H_2(\Omega)$ the hierarchical coefficients $\alpha_{\vec{l},\vec{i}}$ (see [13] and Eq. 20) rapidly decay—namely,

$$|\alpha_{\vec{l},\vec{i}}| = \mathcal{O}\left(2^{-2|\vec{l}|_1}\right). \tag{26}$$

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.
An optimization that minimizes the approximation error given a fixed number of grid points (or vice versa) leads to the *sparse grid* space $V_{0,n}^S$ of level n, defined by

$$V_{0,n}^S := \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}}, \tag{27}$$

where the index 0 in $V_{0,n}^S$ stands for $f|_{\partial\Omega} = 0$ (see [13], and references therein). Note that the actual choice of subspaces depends on the norm in which we measure the error. The result obtained in Eq. 27 is optimal for the L_2-norm and the L_∞-norm.
The number of grid points required by the space $V_{0,n}^S$ is given by (see [13])

$$|V_{0,n}^S| = 2^n \cdot \left(\frac{n^{d-1}}{(d-1)!} + \mathcal{O}\left(n^{d-2}\right)\right) = \mathcal{O}\left(h_n^{-1} \cdot \left(\log(h_n^{-1})\right)^{d-1}\right) = \mathcal{O}\left(2^n \cdot n^{d-1}\right), \tag{28}$$

which is a significant reduction of the number of grid points, and thus of the computational and storage requirements compared to $\mathcal{O}\left(2^{nd}\right)$ for the full grid space V_n (see Tabs. 1 and 9). In analogy to Eq. 20, a function $f \in V_{0,n}^S \subset V_n$ is now approximated by

$$f_{0,n}^S(\vec{x}) \approx u(\vec{x}) = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot \phi_{\vec{l},\vec{i}}(\vec{x}) = \sum_{|l|_1 \leq n+d-1} f_{\vec{l}}(\vec{x}), \tag{29}$$

where $f_{\vec{l}} \in W_{\vec{l}}$. As in the one-dimensional case, a hierarchical surplus $\alpha_{\vec{l},\vec{i}} \in \mathbb{R}$ is simply the difference between the function value at the current and the previous interpolation level (cf. Sec. 2.2). Since the grid points are nested (i.e. X^{l-1} at level $l-1$ is contained in X^l) the extension of the interpolation level from level $l-1$ to l only requires the evaluation of the function at grid points that are unique to X^l—that is, at $X_\Delta^l = X^l \backslash X^{l-1}$.

The asymptotic accuracy of the interpolant deteriorates only slightly from $\mathcal{O}\left(h_n^2\right)$ in case of the full grid (cf. Eq. 22) to

$$\mathcal{O}\left(h_n^2 \cdot \log(h_n^{-1})^{d-1}\right), \tag{30}$$

as shown in [13, 22]. Taken together, Eqs. 28 and 30 demonstrate why sparse grids 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.

Note that sparse grid methods are not restricted to piecewise linear basis functions; there are several other basis functions possible, including piecewise polynomials (see [13, 46], and

Dimension d	Full grid $\lvert V_4 \rvert$	Sparse grid $\lvert V_{0,4}^S \rvert$
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).

references therein). However, we focus on linear hat functions, since these are most convenient for adaptive refinement procedures as presented below in Sec. 2.5. Finally, note that Eq. 30 also holds for sparse grids with non-vanishing boundaries, which we present in Appendix A.

2.5 Adaptive Sparse Grids

The sparse-grid structure introduced in the previous section defines an a priori selection of grid points that is optimal for function with bounded second-order mixed derivatives (cf. Sec. 2.4 and Eq. 24). However, in many applications including economic models with occasionally binding constraints, 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, the next task is to find a way of efficiently approximating functions that do not fulfill the necessary smoothness conditions given in Eq. 24.

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., [47, 46, 12, 41]). By doing so, resources are only invested where needed. While there are various ways of refining a sparse grid (see, e.g., [46], and references therein), we outline only briefly the basic ideas behind the algorithms that we use. For details, we refer the reader to the original articles, namely the ones by [41] and [46].

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 (cf. Eq. 29 and Fig. 2). The logic of the refinement strategies of [41, 46] is therefore to monitor the size of the hierarchical surpluses. Recall from Sec. 2.3 and Sec. 2.4 that the interpolated function is represented by a linear combination of hierarchical, piecewise linear hat functions. The coefficients of the hat functions—the hierarchical surpluses—are just the hierarchical increments between two successive interpolation levels. The magnitude of the hierarchical surplus reflects the local irregularity of the function and thus serves as a natural error indicator.

Technically, the adaptive grid refinement can be built on top of the hierarchical grid structure. Let us first consider the one dimensional case. The equidistant grid points form a tree-like data structure (cf. 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

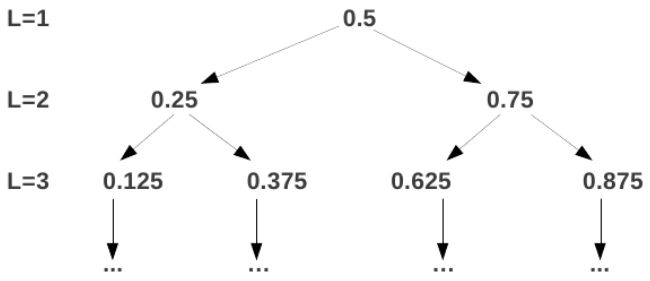

Figure 4: One-dimensional tree-like structure of a "classical" sparse grid (cf. Sec. 2.4) for the first three hierarchical levels.

have a hierarchical surplus, $\alpha_{\vec{l},\vec{i}}$, that satisfies

$$|\alpha_{\vec{l},\vec{i}}| \geq \epsilon, \tag{31}$$

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. Note that this refinement method scales linearly and thus does not suffer from the curse of dimensionality.
Note that in our application in Sec. 3, we interpolate several policies on one grid—that is to say, we interpolate a function

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

Therefore, we get m surpluses at each gridpoint and we thus have to replace the refinement criterion in Eq. 31 by

$$g\left(\alpha^1_{\vec{l},\vec{i}}, \ldots, \alpha^m_{\vec{l},\vec{i}}\right) \geq \epsilon, \tag{32}$$

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. 3.4. However, depending on the application, more sophisticated criteria might need to be imposed for an efficient approximation (see, e.g., Sec. 3.5).

Analytical Examples

We now demonstrate the ability of adaptive sparse grid algorithms to efficiently interpolate functions that exhibit steep gradients and kinks. We present analytical examples in one and two dimensions that were selected from a suite of test problems by [23]. This should foster an understanding of the adaptive sparse grid algorithms in use, namely the ones by [46] and [41].

For the testing, we proceed as follows: We pick a (non-smooth) function $f : [0,1] \to \mathbb{R}$, construct the interpolant $u(\vec{x})$ of $f(\vec{x})$ (cf. Eq. 29), then randomly generate $1,000$ test points from a uniform distribution on $[0,1]^d$, and finally compute the maximum error given by

$$e = \max_{i=1,\ldots,1000} |f(\vec{x_i}) - u(\vec{x_i})|. \tag{33}$$

Moreover, we also assess which choice of sparse grid and its respective basis functions suits our purposes best. More precisely, we compare the following two settings:

- "setting A" (algorithm by Ma and Zabaras [41]):
  - grid: Clenshaw–Curtis grid (cf. Eq. 59)
  - basis functions: modified linear basis functions (cf. Eq. 60)

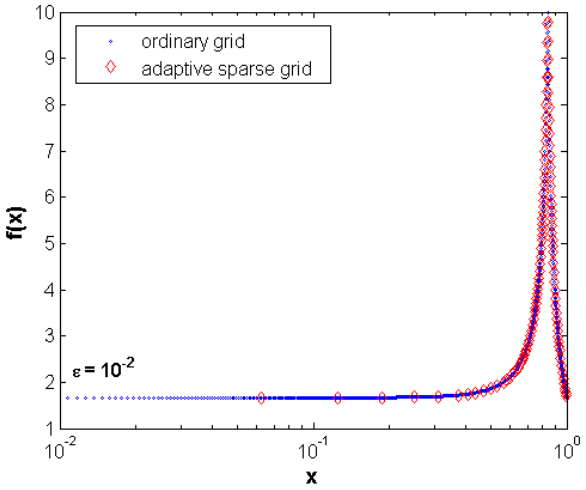

Figure 5: Evaluation of the function given by Eq. 34 at the points of the adaptive sparse grid (red diamonds) and the "full grid" (blue dots). Both grids attain a maximum error $\mathcal{O}\left(10^{-2}\right)$. The adaptive sparse grid reaches this accuracy with 109 points, whereas the full grid needs $1,023$ points.

- "setting B" (algorithm by Pflüger [46]):
  - grid: "standard" sparse grid (cf. Eq. 10)
  - basis functions: modified linear basis functions (cf. Eq. 61)

Note that these two settings, which we discuss in detail in Appendix A, can handle non-vanishing boundaries as our test cases below illustrate.

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

$$f(x) = \frac{1}{|0.5 - x^4| + 0.01}. \tag{34}$$

The refinement threshold for the adaptive sparse grid algorithm (cf. Eq. 31) is chosen to be $\epsilon = 10^{-2}$. With this setting, the maximum interpolation error reaches approximately 10^{-2} (cf. Eq. 33) using 109 grid points. In contrast, to attain the same level of accuracy, 1023 equidistant grid points have to be invested,[5] as shown in Fig. 5. From Fig. 5, it is obvious that the adaptive grid places points in regions where high resolution is needed, while putting only few points in areas where the function varies little. This fact makes adaptive sparse grid algorithms favorable over all other (sparse) grid interpolation methods if kinks or steep gradients have to be handled.

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

$$f(x, y) = \frac{1}{|0.5 - x^4 - y^4| + 0.1}, \tag{35}$$

as shown in Fig. 6. Note that the line of non-differentiability is automatically detected by the adaptive sparse grid algorithm (cf. Figs. 6 and 7). In Fig. 8, we provide the convergence rate of the adaptive sparse grid method for "setting A" and "setting B". The data points shown in

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

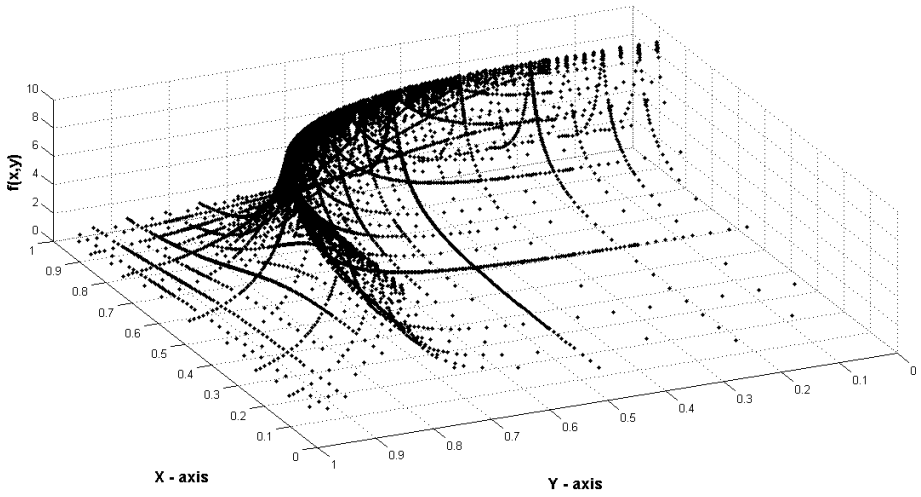

Figure 6: Evaluation of Eq. 35 at the grid points obtained by the adaptive sparse grid algorithm "setting B" at level 15.

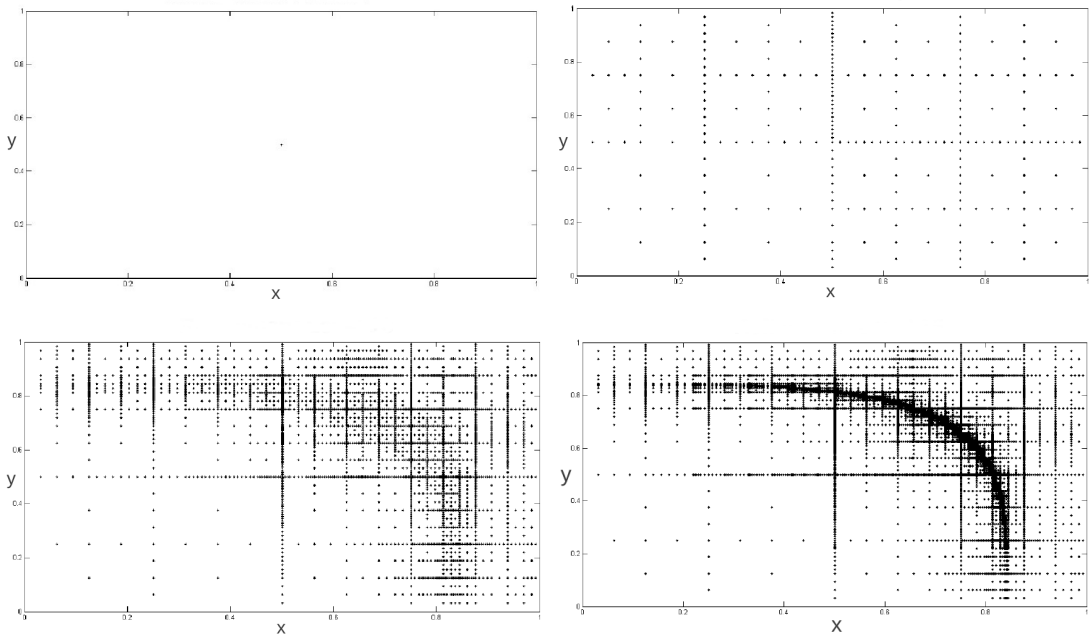

Figure 7: The evolution of an adaptive sparse grid on $[0,1]^2$ with a threshold $\epsilon = 10^{-2}$ ("setting B"). Levels $1, 5, 10$, and 15 are shown.

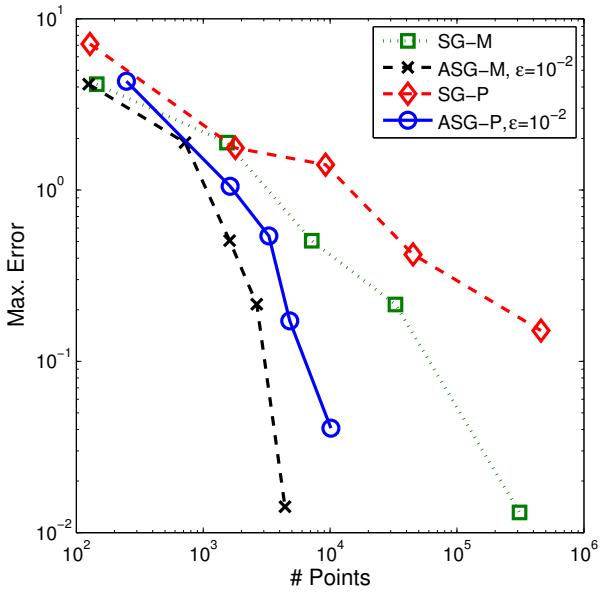

Figure 8: Comparison of the interpolation error (cf. Eq. 33) for conventional and adaptive sparse grid interpolation at different refinement levels. Note that the adaptive sparse grid algorithm "setting A" is labelled "ASG-M", while its corresponding "classical" sparse grid version is denoted by "SG-M". In analogy, "setting B" is denoted by "ASG-P" (and 'SG-P'). Both adaptive grids were obtained by applying a threshold $\epsilon = 10^{-2}$.

Fig. 8 were obtained by computing the errors (cf. Eq. 33) at levels 5, 8, 10, 12, and 15. These results are contrasted with the "classical" sparse grid counterparts of the respective refinement level.
Strikingly, "setting A" for example reaches an accuracy of $e \approx 1.4 \cdot 10^{-2}$, where the interpolation level is 15 and the number of points is $4,411$ as opposed to $311,297$ points using the same level of refinement in the conventional sparse grid. An exemplary evolution of the adaptive sparse grid is shown in Fig. 7. From Fig. 8, it also becomes apparent that "setting A" is converging faster than "setting B". Thus, we will use "setting A" for all our applications below.

3 Application to Dynamic Models

In this section, we apply the adaptive sparse grid method to economic examples, namely an IRBC model and a menu-cost model. We first introduce the IRBC model in Sec. 3.1, then outline the time iteration algorithm for solving it in Sec. 3.2, and subsequently present the parallelization of the code in Sec. 3.3. The IRBC model has become a standard for testing computational methods for solving high-dimensional dynamic models (see, e.g., [18]), as its dimensionality can be scaled up in a straightforward and meaningful way—it just depends linearly on the number of countries considered. This model is therefore perfectly suited to discuss the performance of our method, as we do in Sec. 3.4. Finally, Sec. 3.5 provides the menu-cost application and thereby shows that adaptive sparse grids can also be used within a value function iteration framework and are well suited to handle discrete choices.

3.1 International Real Business Cycle Model

To focus on the problem of handling high-dimensional state spaces, we follow [18, 33, 36] and consider an IRBC model that is simple in many other ways. Extending the model in directions that are standard in the literature, however, is not a serious challenge for our solution method. For example, solving a model with country specific goods (and thus meaningful exchange rates) would not increase the dimension of the state space, just the size of the equation system that has to be solved. We now briefly describe the IRBC model, its first-order conditions, its parametrization, and finally an extension of the model where investment is irreversible.

Model Description

There are N countries that differ from each other 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 is given by

$$y_t^j = a_t^j \cdot f^j(k_t^j), \tag{36}$$

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

$$\ln a_t^j = \rho \cdot \ln a_{t-1}^j + \sigma \left(e_t^j + e_t \right), \tag{37}$$

where the shock e_t^j is specific to country j, while e_t is a global shock. These shocks are all i.i.d. standard normal. The law of motion of capital is given by

$$k_{t+1}^j = k_t^j \cdot (1 - \delta) + i_t^j, \;\; k_{t+1}^j \geq 0, \tag{38}$$

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

$$\Gamma_t^j(k_t^j, k_{t+1}^j) = \frac{\phi}{2} \cdot k_t^j \cdot \left(\frac{k_{t+1}^j}{k_t^j} - 1 \right)^2. \tag{39}$$

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

$$\sum_{j=1}^{N} y_t^j \geq \sum_{j=1}^{N} \left(i_t^j + \Gamma_t^j(k_t^j, k_{t+1}^j) + c_t^j \right), \tag{40}$$

where $c_t^j \in \mathbb{R}_+$ denotes consumption of country j at time t. Substituting and rearranging, we get

$$\sum_{j=1}^{N} \left(a_t^j \cdot f^j(k_t^j) + k_t^j \cdot (1 - \delta) - k_{t+1}^j - \Gamma_t^j(k_t^j, k_{t+1}^j) - c_t^j \right) \geq 0. \tag{41}$$

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,[6] the decentralized competitive equilibrium allocation can be obtained as the solution to a social planner's problem, where the welfare weights, τ^j, of the various countries depend on their initial endowments. More precisely, the social planner solves

$$\max_{\{c_t^j, k_t^j\}} \mathbb{E}_0 \sum_{j=1}^{N} \tau^j \cdot \left(\sum_{t=1}^{\infty} \beta^t \cdot u^j(c_t^j) \right) \tag{42}$$

[6] We follow the comparison study [18, 33, 36] in assuming complete markets. However, the time iteration algorithm in Sec. 3.2 can also handle setting with incomplete markets.

subject to the aggregate resource constraint (41) given initial capital stocks $k_0 \in \mathbb{R}^N_{++}$.

We assume the following standard functional forms for the production function and the utility function:

$$f^j(k_t^j) = A \cdot (k_t^j)^\alpha,\ u^j(c_t^j) = (1 - 1/\gamma_j)^{-1} \left(c_t^j\right)^{1-\frac{1}{\gamma_j}}, \tag{43}$$

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

First Order Conditions

To get the first order conditions (FOCs) of problem (42), we differentiate the Lagrangian with respect to c_t^j:

$$\tau^j \cdot \frac{\partial u^j(c_t^j)}{\partial c_t^j} - \lambda_t = 0, \tag{44}$$

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

$$\lambda_t \left[-1 - \frac{\partial \Gamma_t^j(k_t^j, k_{t+1}^j)}{\partial k_{t+1}^j}\right] + \beta \cdot \mathbb{E}_t \left\{\lambda_{t+1} \cdot \left[a_{t+1}^j \cdot \frac{\partial f^j(k_{t+1}^j)}{\partial k_{t+1}^j} + (1-\delta) - \frac{\partial \Gamma_{t+1}^j(k_{t+1}^j, k_{t+2}^j)}{\partial k_{t+1}^j}\right]\right\} = 0, \tag{45}$$

where λ_t denotes the multiplier on the time t resource constraint.
Differentiating the adjustment cost function given in Eq. 39, defining the growth rate of capital by $g_t^j = k_t^j / k_{t-1}^j - 1$, and exploiting the functional forms in Eq. 43, we finally get the system of $N+1$ equilibrium conditions that have to hold at each t, namely for all countries $j \in \{1, ..., N\}$:

$$\lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j\right] - \beta \cdot \mathbb{E}_t \left\{\lambda_{t+1} \left[a_{t+1}^j \cdot A \cdot \alpha \cdot (k_{t+1}^j)^{\alpha-1} + (1-\delta) + \frac{\phi}{2} \cdot g_{t+2}^j \cdot \left(g_{t+2}^j + 2\right)\right]\right\} = 0, \tag{46}$$

and the aggregate resource constraint (holding with equality due to strictly increasing per-period utility assumed in Eq. 43):

$$\sum_{j=1}^N \left(a_t^j \cdot A \cdot (k_t^j)^\alpha + k_t^j \cdot \left((1-\delta) - \frac{\phi}{2} \cdot (g_{t+1}^j)^2\right) - k_{t+1}^j - \left(\frac{\lambda_t}{\tau_j}\right)^{-\gamma^j}\right) = 0. \tag{47}$$

In Sec. 3.2 we descibe how to solve for a recursive equilibrium of the IRBC model by iterating on recursive formulations of Eqs. 46 and 47.

IRBC Model with Irreversible Investment

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

$$k_{t+1}^j \geq k_t^j \cdot (1-\delta). \tag{48}$$

As a consequence, we have to solve a system of $2N+1$ equilibrium conditions. These conditions now include the Karush–Kuhn–Tucker multiplier, μ_t^j, for the irreversibility constraint $k_{t+1}^j -$

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	α	0.36
Depreciation	δ	0.01
Std. of log-productivity shocks	σ	0.01
Autocorrelation of log-productivity	ρ	0.95
Intensity of capital adjustment costs	ϕ	0.50
Aggregate productivity	A	$(1-\beta(1-\delta))/(\alpha-\beta)$
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 [33].

$k_t^j \cdot (1-\delta) \geq 0$. The Euler equations and the irreversibility constraints for all countries $j \in \{1, ..., N\}$ are

$$\begin{aligned}
&\lambda_t \cdot \left[1 + \phi \cdot g_{t+1}^j\right] - \mu_t^j \\
&\quad - \beta \mathbb{E}_t \left\{ \lambda_{t+1} \left[a_{t+1}^j A \alpha (k_{t+1}^j)^{\alpha-1} + 1 - \delta + \frac{\phi}{2} g_{t+2}^j \left(g_{t+2}^j + 2 \right) \right] - (1-\delta)\mu_{t+1}^j \right\} = 0, \\
&\qquad\qquad 0 \leq \mu_t^j \perp \left(k_{t+1}^j - k_t^j (1-\delta) \right) \geq 0. \quad (49)
\end{aligned}$$

The aggregate resource constraint is unchanged (see Eq. 46).

Parameterization

With respect to the parameter choices (for the model with and without irreversibility) we follow Juillard and Villemot [33], who provide the model specifications for the comparison study that uses the IRBC model to test several solution methods (see [18]). 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 [33]. 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 parameter A is chosen such that the capital of each country is equal to 1 in the deterministic steady state.

3.2 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

$$s = (a, k) = \left(a^1, \ldots, a^N, k^1, \ldots, k^N\right). \quad (50)$$

Denoting the state space by $S \subset \mathbb{R}^{2N}$, we solve for policies that map the current state into capital choices and into the Lagrange multiplier on the aggregate resource constraint

$$p : S \to \mathbb{R}^{N+1}, \; p(s) = \left(k_+^1(s), \ldots, k_+^N(s), \lambda(s)\right), \quad (51)$$

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 Eqs. 46 and 47 at all points in the state space. To compute policies that approximately satisfy these conditions, we use time iteration (see, e.g., [31]) and employ adaptive sparse grid interpolation in each of its iteration steps. The details of the implementation are discussed in Sec. 3.4. The structure of the algorithm is as follows:[7]

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

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

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

2. Make one time iteration step:

   (a) Start with a coarse grid $G_{old} \subset S$ (a "classical" sparse grid of a "low" level L_0), and generate G by adding for each $x \in G_{old}$ all $2d$ neighboring points. Choose a refinement threshold ϵ and a maximal level $L_{max} > L_0$ and set $l = 1$.

   (b) For each grid point

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

   solve for the optimal policies

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

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

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

   More precisely, abbreviating $p(x)$ by p, and $p_+(s_+)$ by p_+, the equilibrium conditions (Eqs. 46 and 47) demand that for all countries $j \in \{1, ..., N\}$:

$$p^{N+1} \cdot \left[1 + \phi \cdot (p^j/k^j - 1)\right] - \beta \mathbb{E}_t \left\{ p^{N+1}_+ \left[a^j_+ A \alpha (p^j)^{\alpha-1} + (1-\delta) + \frac{\phi}{2}\left(p^j_+/p^j - 1\right)\left(p^j_+/p^j + 1\right)\right]\right\} = 0, \quad (52)$$

   and also

$$\sum_{j=1}^{N} \left(a^j A (k^j)^\alpha + k^j \left((1-\delta) - \frac{\phi}{2}(p^j/k^j - 1)^2 \right) - p^j - \left(\frac{p^{N+1}}{\tau_j} \right)^{-\gamma^j} \right) = 0. \quad (53)$$

   (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 $\{p(x)\}_{x \in G_{old}}$.

[7] Note that we present the time iteration algorithm for the smooth IRBC model. However, the time iteration procedure for the non-smooth IRBC works analogously; Eq. 46 just need to be replaced by Eqs. 49, and the multipliers for the irreversibility constraint have to be interpolated in addition. Note also 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 $L_0 = 1$, $\epsilon = 0$, and $L_{max} = L$.

(d) If $G_{new} = G$ or $L_0 + l = L_{max}$, then set $G = G_{new}$ and go to (e), otherwise set $l = l + 1$ 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 the maximum level L_{max} to a very large value such that it is never reached for a given refinement threshold. However, this can create practical problems as there is no reasonable upper bound for the number of grid points created by the refinement procedure. Note that for the 4-dimensional models (see Sec. 3.4), we set L_{max} such that it is never reached.

3.3 Parallelization and Scaling

In order to solve "large" problems in a reasonable amount of time, we make use of modern HPC architectures. For this purpose, in each step of the above time iteration procedure the updated policy function is determined using a hybrid MPI/threads parallel algorithm, as illustrated in Fig. 9. Within each refinement step, we first distribute newly generated grid points among multiple (multi-threaded) processes by MPI. The points that are send to one particular compute node are further distributed among different threads. Each thread then solves a set of nonlinear equations for every single grid point assigned to it.[8] 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% 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., [27, 21, 44]). We therefore offload parts of the interpolation from the compute nodes to their attached accelerators. When comparing a single-threaded CPU with the case in which it also employs a GPU, we observe a speedup of roughly one order of magnitude for the interpolation. In the case where we use the entire node, the multi-threading is implemented with thread building blocks (TBB, see [50]). One of the TBB-managed threads exclusively uses the GPU, reducing the overall computation time by roughly 50%. Moreover, CPU and GPU threads leverage TBB's automatic workload balancing based on stealing tasks from slower threads. More details regarding the parallel implementation and optimization of our code can be found in the companion paper [11].

We now report strong scaling efficiency of our algorithm on the $5,272$-node Cray XC30 "Piz Daint" system that is installed at CSCS. Cray XC30 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 5.0. In Fig. 10, we display the speedup S_p and the efficiency E_p of the code [49]. These two quantities are defined by

$$S_p = \frac{T_1}{T_p}, \qquad E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}, \tag{54}$$

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

[8] The set of nonlinear equations is solved with `Ipopt` [54] (`http://www.coin-or.org/Ipopt/`).

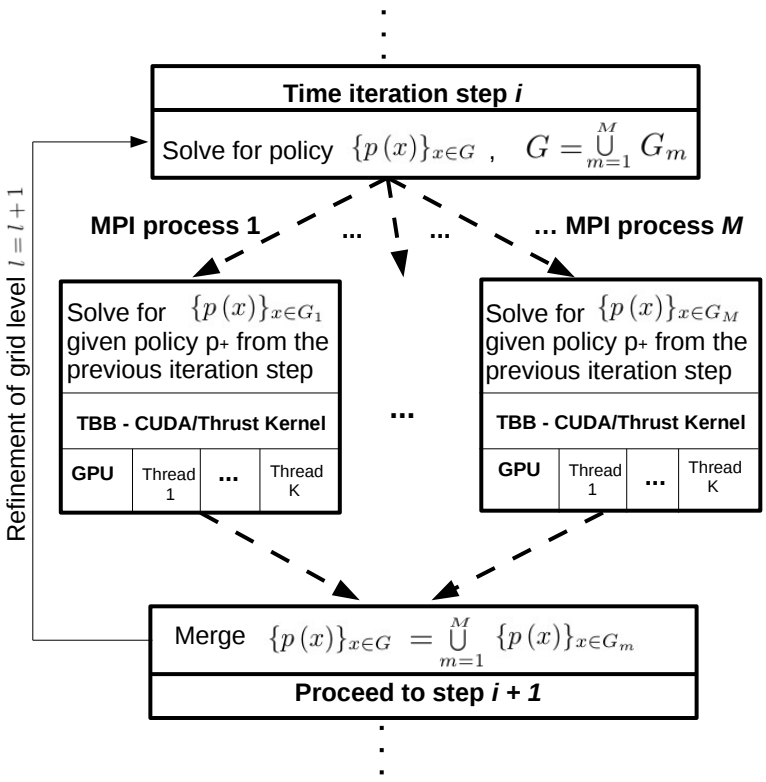

Figure 9: Schematic representation of the hybrid parallelization of a time iteration step (see Sec. 3.2). Newly generated points are distributed among different MPI processes. Each MPI process is using TBB and a CUDA/Thrust Kernel for the function evaluation.

is a single time iteration step of a 16-dimensional IRBC model. In order to provide a consistent benchmark, we used a non-adaptive "classical" Clenshaw–Curtis sparse grid of level 6. It has a total of $3,183,561$ variables per time step.[9] The economic test case was solved with increasingly larger numbers of nodes (from 1 to $2,048$ nodes). Fig. 10 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 K20x GPU. The code scales nicely to about $2,048$ nodes, as shown in Fig. 10. Even at this large node count, the overall efficiency of our code is beyond 60%. Note that the dominant limitation to the strong scaling stems from the fact that within the first few refinement levels, the workload can be very unbalanced. In the extreme, when a large number of nodes is used, the ratio of "points to be evaluated to MPI processes" is substantially smaller than one, resulting in idle MPI processes. The better parallel efficiency on the higher refinement levels is due to the fact we have many more points available on this refinement level, so the workload is distributed more equally among the different MPI processes.

3.4 Performance and Accuracy

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

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

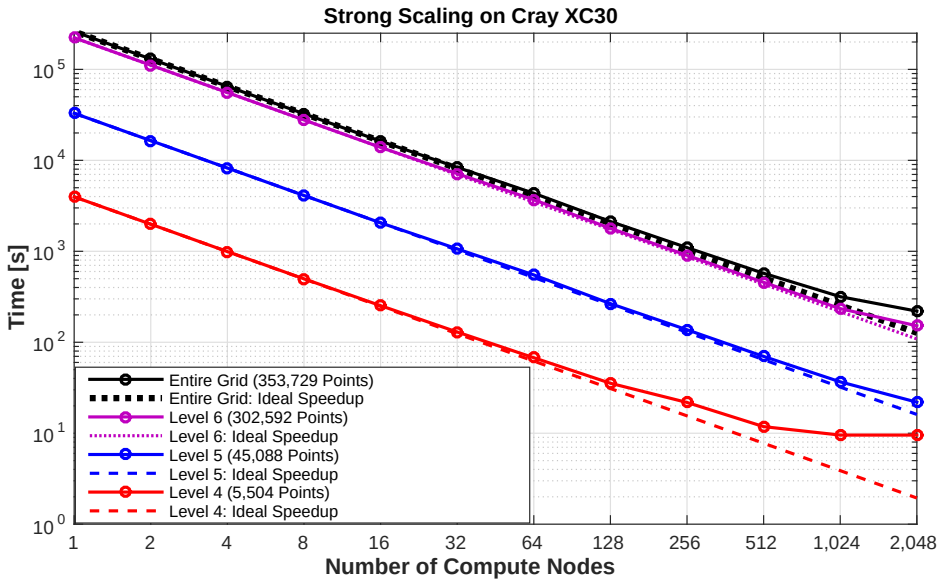

Figure 10: Strong scaling plots on Piz Daint for a 16-dimensional IRBC model using 5 levels of grid refinement and in total 353,729 points on which we solve Eqs. 52 and 53. "Entire Grid" shows the entire computation time up to 2,048 nodes. We also show execution times for the computational subcomponents on different refinement levels, e.g, for refinement level 5 using 302,592 points, refinement level 4 using 45,088 points, and refinement level 3 using 5,504 points. Dashed lines show ideal speedups.

Implementation Details

One important detail of the implementation is the integration procedure used to evaluate the expectations operator, for example in Eq. 46. 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, $2(N+1)$ points in total (see [31], and references therein). 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 Appendix B), yet not over the same space that the policy functions are interpolated on. Therefore, we view integration as a problem that is orthogonal to the choice of the grid structure, and thus do not focus on it.

Another implementation detail that we would like to touch upon is the possibility of accelerating the time iteration procedure by starting with coarse grids and later changing to finer grids. For instance, using "classical" sparse grids, the overall computation time can be reduced by one order of magnitude when using a level 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.

Error Measure

To evaluate the accuracy of our solutions we follow the previous literature (see, e.g., [33, 31]) and compute (unit-free) errors in the $N+1$ equilibrium conditions. Namely, we get one Euler equation error for each country $j \in \{1, ..., N\}$

$$\left[\beta \mathbb{E}_t \left\{ \lambda_{t+1} \left[a^j_{t+1} A \alpha (k^j_{t+1})^{\alpha-1} + 1 - \delta + \frac{\phi}{2} g^j_{t+2} \left(g^j_{t+2} + 2 \right) \right] \right\} \right] \cdot \left[\lambda_t \cdot \left(1 + \phi \cdot g^j_{t+1} \right) \right]^{-1} - 1, \quad (55)$$

and we get an additional error from the aggregate resource constraint

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

These expressions are evaluated by using the computed equilibrium policy function to calculate both today's policy and next period's policy. We compute these errors for 10 thousand 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 case of the non-smooth model). For each of these points we get $N+1$ errors. We then take the maximum over the absolute value of these errors, which results in one error for each point. Over all these errors, we compute both the maximum (Max. Error) and the average (Avg. Error), both of which we report in $\log_{10}$-scale. In the case of the IRBC model with irreversible investment there is one additional complication. Defining the percentage violation of the irreversibility constraint by[10]

$$IC^j \equiv 1 - \frac{k^j_{t+1}}{k^j_t \cdot (1-\delta)} \quad (57)$$

the error is now given by

$$\max \left(EE^j, IC^j, \min \left(-EE^j, -IC^j \right) \right). \quad (58)$$

The first term within the max operator, EE^j, 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, in the case that $-EE^j$ is positive, the error is not simply given by $-EE^j$ but by $\min \left(-EE^j, -IC^j \right)$.

Results for the Smooth IRBC Model

We first consider "classical" sparse grid solutions of the smooth IRBC model. Tab. 3 shows how the approximation errors decrease as we increase the resolution level of the grid while

[10] With irreversible investment, the Euler error EE^j contains an additional term within the expectations operator: $-(1-\delta)\mu^j_{t+1}$ (cf. Eq. (49)).

Dimension	Level	Points	Max. Error	Avg. Error
4	3	41	-2.95	-3.18
4	5	401	-3.36	-4.22
4	7	2,929	-3.53	-4.56
4	9	18,945	-3.64	-4.78

Table 3: Maximum and average errors of "classical" sparse grid solutions of the smooth IRBC model with fixed dimension and increasing approximation level. All errors are reported in $\log_{10}$-scale. In addition, the number of grid points is reported.

Dimension	Level	Points	Max. Error	Avg. Error
4	3	41	-2.95	-3.18
8	3	145	-3.04	-3.25
12	3	313	-2.81	-3.27
16	3	545	-2.59	-3.29
20	3	841	-2.93	-3.30
50	3	5,101	-2.64	-3.33
100	3	20,201	-2.79	-3.33
4	4	137	-3.04	-3.65
8	4	849	-3.26	-3.83
12	4	2,649	-3.04	-3.83
16	4	6,049	-2.72	-3.76
20	4	11,561	-3.00	-3.73

Table 4: Maximum and average errors for "classical" sparse grid solutions of the smooth IRBC model with increasing dimension—up to dimension 100 for level 3, and up to dimension 20 for level 4. All errors are given in $\log_{10}$-scale. Moreover, the number of grid points is reported.

keeping the dimensionality of the problem fixed at $2N = 4$. Recall that we report all errors in $\log_{10}$-scale. The maximal and average errors fall as the level of the grid, and thereby the number of grid points, increases. This is exactly what one would expect. Note that 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. Therefore, an adaptive grid cannot improve much on "classical" sparse grids. For the same reason, the accuracy of our solutions is lower than the accuracy obtained by some smooth approximation methods used in the comparison study summarized by [36].

Let us now consider higher dimensions. In Tab. 4, we vary the dimensionality of the problem while keeping the grid level fixed. We find that the accuracy fluctuates little as the dimension of the problem increases.[11] Importantly, there seems to be no clear downward trend in accuracy. Thus, for a given resolution level the accuracy of the solution does not deteriorate as the dimensionality of the problem is increased massively. Clearly, the number of grid points increases, but far from exponentially, which is the defining feature of sparse grids. Concerning CPU time, Fig. 11 shows that it substantially increases as we consider higher dimensions. This effect, however, is known and was previously reported, for example, by [32, 42], who were both using Smolyak sparse grids with global polynomials. We also observe that the compute time grows faster than the number of grid points as the dimensionality increases. This effect is largely driven by two additional factors. First, memory access time required by interpolating

[11] Some fluctuations are to be expected, especially because the preference heterogeneity across countries depends on the dimension of the problem (cf. Tab. 2)

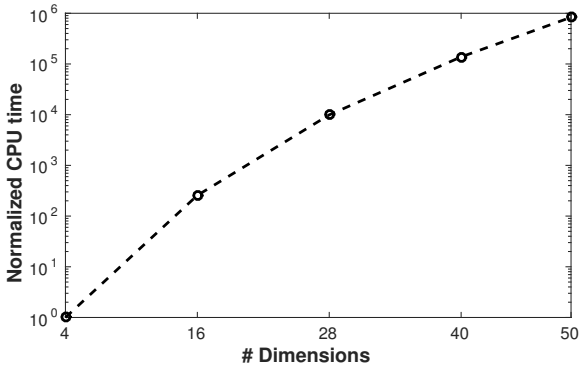

Figure 11: The figure shows how CPU time (in log-scale) increases with the dimensionality of the problem. CPU time is normalized to the computation time of the 4-dimensional case. The test problem was one representative time iteration step with a "classical" sparse grid of level 3.

on a sparse grid grows substantially when the dimension increases, as shown in great detail by [44]. Second, the size of the nonlinear equation systems scale up linearly with dimension, and thus exacerbate the computational burden. While an average time step using a "classical" sparse grid of level 3 consumes about 0.0002 node hours for $2N = 4$, it needs roughly 200 node hours for $2N = 50$. All in all, the increase in CPU time is less than exponential (cf. Fig. 11). Thus, we can—at least to some extent—control the increasing running times by using a larger number of compute nodes (cf. Sec. 3.3). This stands in contrast to, for instance, [32, 42], who only provide serial algorithms.

Results for the Non-Smooth IRBC Model

We now turn to the IRBC model with irreversible investment, as described in Sec. 3.1. The assumption that investment is not reversible induces non-differentiabilities in the policy functions that we interpolate. Two such kinks can be observed in Fig. 12. For the 2-country case, Fig. 12 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, 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. 12. On the other hand, if the capital of country 1 is very high, then its irreversibility constraint is binding and thus limits the transfer of resources to country 2. Therefore the policy function is non-increasing at the very right. It is in fact slightly decreasing for consumption smoothing reasons. When looking at Fig. 12, 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. Clearly, such high-dimensional kinks pose a substantial challenge to constructing an accurate global approximation.
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. 5 reports errors[12] for "classical" sparse grids of increasing resolution levels. It reveals

[12]Note that in this section, the "Max. Error" is defined as the 99.9% 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

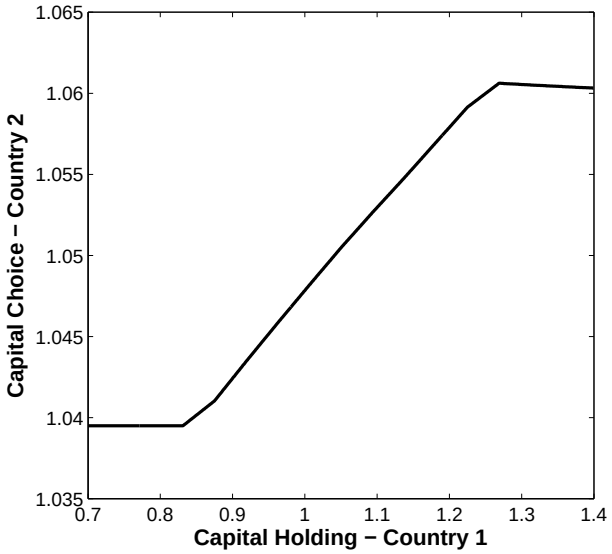

Figure 12: For the 2-country IRBC model with irreversible investment, this figure shows the capital choice of country 2 as a function of the capital holding of country 1. All three other state variables are kept fixed at their deterministic steady state levels. Thus, this figure shows a one dimensional cut through a 4-dimensional policy function.

Dimension	Level	Points	Max. Error	Avg. Error
4	3	41	-1.78	-2.42
4	5	401	-2.11	-2.93
4	7	2,929	-2.32	-3.12
4	9	18,945	-2.45	-3.32

Table 5: Maximum and average errors of "classical" sparse grid solutions of the **non-smooth** IRBC model with fixed dimension and increasing approximation level. Furthermore, the number of grid points is reported. All errors are reported in $\log_{10}$-scale, and the "Max. Error" is here given by the 99.9% quantile of the error distribution.

two important insights. First, both the maximum and the average errors of non-smooth IRBC models are considerably worse than for smooth IRBC models of comparable resolution (cf. Tab. 3). 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 error barely decreases as the number of grid points increases. 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.

With these findings in mind, we now turn our attention from "classical" sparse grids to adaptive sparse grids. Tab. 6 and Fig. 13 show that adaptive sparse grids are more efficient in reducing approximation errors, as they put additional resolution where needed, while not wasting resources in areas of smooth variation. For instance, an adaptive sparse grid with 2,346 points (cf. Tab. 6) reaches errors comparable to a fixed sparse grid with 18,945 points (see Tab. 5).

errors, we mitigate this effect. This "truncation" also 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.

ϵ	Max. Level Reached	Points	Max. Error	Avg. Error
0.0150	6 (1,105)	129	-2.12	-2.75
0.01	7 (2,929)	245	-2.23	-2.88
0.005	9 (1,945)	559	-2.42	-2.98
0.0025	13 (643,073)	2,346	-2.68	-3.32
0.001	14 (3,502,081)	14,226	-2.91	-3.73

Table 6: Maximum and average errors for an adaptive sparse grid solution of the **non-smooth** IRBC model of different refinement thresholds ϵ are reported. The size of the respective $2N = 4$-dimensional adaptive sparse grids is reported in the third column. The second column shows two numbers. The first is the highest resolution level that was reached for a particular ϵ. The second (in brackets) indicates how many grid points a corresponding "classical" sparse grid of that resolution level would comprise. All errors are reported in $\log_{10}$-scale, and the "Max. Error" is here given by the 99.9% quantile of the error distribution.

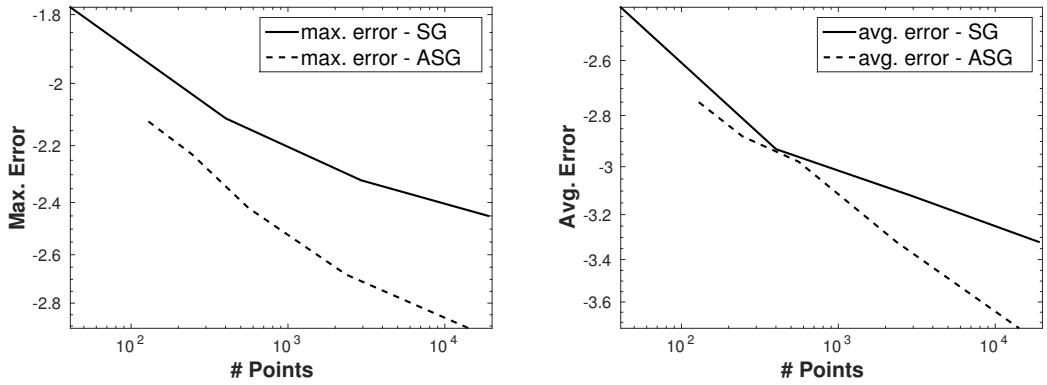

Figure 13: Comparison of the maximum error for "classical" and adaptive sparse grid solutions to the 4-dimensional **non-smooth** IRBC model as a function of grid points. The data points for the "classical" sparse grids (SG) stem from models run with fixed levels 4 to 9. The respective adaptive sparse grid (ASG) solutions arise from varying refinement thresholds ϵ ranging from 0.015 down to 0.001. This figure illustrates that, for similar sized grids, the "classical" sparse grid is much less accurate than the adaptive sparse grid. All errors are reported in $\log_{10}$-scale, and the "Max. Error" is here given by the 99.9% quantile of the error distribution.

In Tab. 6, we report results for different refinement thresholds ϵ. The smaller the chosen refinement threshold ϵ, the larger the maximum refinement level reached and the larger the number of grid points. The reported errors are in stark contrast to the results reported in Tab. 5, where we find that increasing the size of "classical" sparse grids improves the maximum error only very slowly. With the lowest threshold considered (0.001), the adaptive sparse grid 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 a "classical" sparse grid of level 14. While the adaptive sparse grid consists of $14,226$ grid points, the "classical" grid would comprise more than 3 million points (cf. the second column of Tab. 6). This comparison shows that the adaptive sparse grid introduces an extra 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, as one can see in Fig. 13. This is possible as grid points are placed only where high resolution is needed while just few points are put in areas where the policies to be approximated vary little. In order to illustrate this feature, we

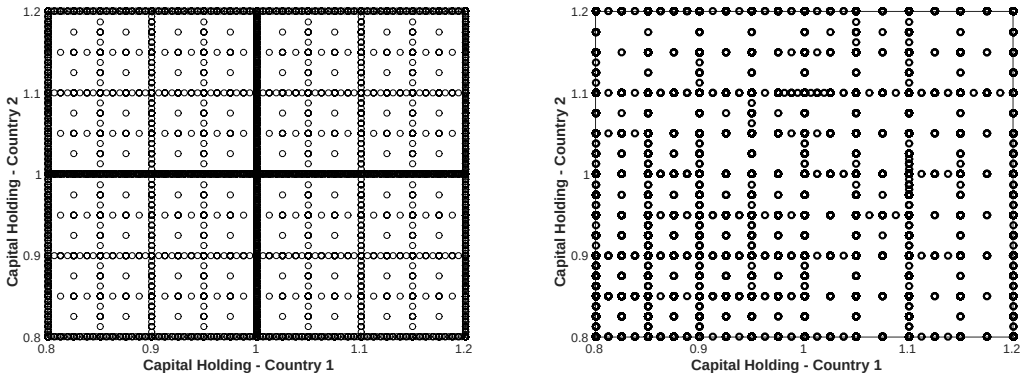

Figure 14: This figure displays 2-dimensional projections of two different grids. The left one is from a "classical" sparse grid of level 9 (cf. Tab. 5), the right one from an adaptive grid of refinement threshold $\epsilon = 0.001$ (cf. Tab. 6). Both grids were generated in the course of running a 4-dimensional simulation (2 countries). 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.

display in Fig. 14 slices of 4-dimensional grids, one "classical" of fixed refinement level 9 (cf. Tab. 3) and the other "adaptive". In spite of having fewer points, the solution provided by the adaptive sparse grid is of higher 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. Tab. 7 reports errors for adaptive sparse grids of increasing dimensionality. The refinement threshold is set to $\epsilon = 0.01$ for dimensions $d = \{4, ..., 12\}$ due to the fact that this setting provides decent errors at moderate costs (cf. Tab. 6). For dimensions 16 and 20, we chose $\epsilon = 0.0025$ in order to obtain results of the desired accuracy. We find that the accuracy weakly depends on the dimension. There seems to be a moderate downward trend in accuracy. This behavior is not surprising as the kinks, being $2N - 1$ dimensional objects, become much harder to approximate as N increases. Also, the maximum refinement level $L_{max} = 9$ is binding for dimensions 6 and larger, while it is not binding for dimension 4. Therefore, with a higher L_{max} the errors for higher dimensions would slightly improve relative to dimension 4. Another way to counteract 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 solution time, it is clear that the non-smooth IRBC models pose 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 $2N + 1$ (cf. Eqs. 49 and 47), compared to $N + 1$ in the case of the smooth IRBC models (cf. Eqs. 46 and 47). On top of this, the (local) resolution needs to be considerably higher in order to achieve decent errors (cf. Tab. 6), 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 a representative time iteration step in a 2-country model now consumes about 0.003 node hours, while a 20-d model may require up to 160 node hours—a number that is comparable to solving a 50-d smooth IRBC model. However, this is still not a roadblock for solving such complex models. All in all, 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 Sec. 3.4) and employing a large number of nodes (see Sec. 3.3). Hence, we can obtain

Dimension	Points	Max. Error	Avg. Error	$\lvert V_9^{S,CC}\rvert$
4	245	-2.22	-2.88	18,945
6	684	-2.26	-2.73	127,105
8	931	-2.02	-2.66	609,025
10	2,790	-1.97	-2.54	2,148,960
12	4,239	-1.81	-2.48	7,451,394
16	8,569	-1.94	-2.36	52,789,761
20	9,098	-1.96	-2.35	$\gg 10^8$

Table 7: Errors for an adaptive sparse grid solution of the **non-smooth** IRBC model with increasing dimension. The number of grid points of the adaptive sparse grid solution is contrasted by the corresponding grid size of the "classical" sparse grid of level 9 (last column). All errors are reported in $\log_{10}$-scale, and the "Max. Error" is here given by the 99.9% quantile of the error distribution.

results within one day, even for the largest non-smooth models considered here.

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 in economics (see, e.g., [33, 37]). On the other hand, methods that are designed to handle non-differentiabilities (see those, e.g., in [29, 10, 20, 5]) are only capable of solving models with up to 3 continuous state variables.

3.5 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 multi-product 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 to keep track of the values associated with each of the discrete choices possible. As discrete choices induce kinks in value functions, adaptive sparse grids with piecewise multi-linear hierarchical basis functions are a particularly suitable choice for interpolation. The menu-cost application thus illustrates that our method also works with value function iteration and with discrete choices.

Menu-cost models are at the center of an ongoing debate about the real effects of monetary policy (see, e.g., [34, 3]). This debate started with Golosov and Lucas [25] who argue that monetary policy has only small effects in a menu-cost model that is consistent with the (large) average size of price-changes in micro-price data. Midrigan [43] challenges their view by considering a model where multi-product firms face economies of scope in price-setting as well as the possibility of temporarily deviating from a regular price at a certain cost. The first feature helps to better capture the dispersion of price-changes in the data, while the second can account for sales periods. However, the 2-goods setup of Midrigan [43] does not capture the high kurtosis of the price-distribution, as Alveraz and Lippi [3] point out. Thus, we extend Midrigan [43] to 3 goods and show that the fit to micro-price data is thereby substantially improved. This result illustrates that our method allows for substantial extensions of currently used menu-cost models—extensions that are both economically interesting as well as out-of-reach for standard methods.

Except for the number of goods, we consider the exact same setup as Midrigan [43]; we thus keep the exposition very concise and refer the reader to Midrigan [43] for details of the model

and the parametrization. The firm sets two types of prices, regular and temporary, for N different goods. 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 regular prices at no cost. Due to (extreme) economies of scope, the same costs ϕ^R or κ apply, no matter whether one or more prices are 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 V^N the value of simply charging the prevailing regular prices. All these values are functions of the vector of the firm's previous period markups $\boldsymbol{\mu}^R_{-1}$, of the discrete cost state of the firm e, of the money growth rate g, and 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 [43] characterizes the firm's problem by the following system of functional equations:

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

The parameter γ captures the elasticity of substitution between two identical goods sold by different firms, and $\hat{P}$ is the aggregate price level (detrended by the money stock). To make this problem tractable, Midrigan [43]—inspired by Krusell and Smith [38]—replaces the distribution of markups Λ by 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. The state space on which the above 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 [43] interpolates the value functions on this space with splines on tensor product grids. As the latter suffer from the curse of dimensionality, a straightforward extension to higher dimensions is most likely infeasible.

To overcome the curse of dimensionality, we solve the above problem using a value function iteration algorithm that interpolates the value functions with adaptive sparse grids. The iterative structure of this algorithm is very similar to the time iteration algorithm in Sec. 3.2. The most obvious difference is that in step 2(b), instead of updating policy functions by solving systems of non-linear equations, the value functions are updated by solving the respective maximization problems. A more subtle yet important difference is adaptation criterion we employ. 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 the adaptive sparse grid interpolation of the three value functions V^R, V^T and V^N is build, 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 changes substantially as the approximation level increases. It is in such regions that a further refinement could most likely provide a more precise picture about where each discrete choice is optimal.

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 [43] chose to match the statistics of data from Dominick's Finer Foods retail chain. Tab. 8 shows that our algorithm generates very similar results as Midrigan [43]. Our computations also confirm that the 2-goods model does not capture the

Moments	**Data**	**2 Goods**	**2 Goods**	**3 Goods**
	Dominick's	Midrigan [43]	Sparse Grid	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	0.025
Mean size of price changes	0.20	0.20	0.18	0.18
Mean size of reg. price changes	0.11	0.10	0.11	0.10
Std. of price changes	0.18	0.15	0.13	0.14
Std. of regular price changes	0.08	0.07	0.07	0.07
Kurtosis of reg. price changes	4.0	2.5	2.5	5.3

Table 8: Statistics of price changes—First, data from Dominick's Finer Foods retail chain; second, results of the 2-goods model of Midrigan [43]; third and fourth, results of the 2- and 3-goods model 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, yet only show that with 3 goods much higher values can be reached.

high kurtosis of the price-distribution. In contrast, Alveraz and Lippi [3] show in an analytically tractable framework that the fit of the kurtosis can be improved by increasing the number of goods. However, in their framework, a very high number of goods is necessary, but may not be sufficient, to reach values that are close to the data.[13] We show that in a straightforward extension of Midrigan [43] from 2 to 3 goods, the kurtosis already reaches or even exceeds levels that are consistent with the data. In the final column of Tab. 8, we report statistics from a simulation of the three good model. This setup generates first and second moments that are very close to the ones in the 2 good model, yet at the same time a much higher kurtosis.[14] Note that our aim is not to match the kurtosis of Dominick's data,[15] yet to show that in a framework with correlated and fat-tailed shocks (as in Midrigan [43]) three goods are sufficient to generate reasonable values for the kurtosis, even if two are not. The more general purpose of this exercise is to demonstrate the usefulness of adaptive sparse grids for solving a class of models that is at the center of a current debate of great economic importance. Many other extensions of menu costs models—like relaxing the economies of scope assumption or assuming different economies of scope for regular and temporary prices—could be addressed with adaptive sparse grid methods, yet are beyond the scope of this paper.

4 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 and are, to the best of our knowledge, the first paper on dynamic economic models to do so.

Our algorithm is highly flexible and scalable. First, by choosing the resolution level we can

[13] We concentrate on the kurtosis of regular price changes as only these are mainly driven by good specific shocks in Midrigan's framework; temporary price changes, in contrast, are mainly caused by firm specific shocks. Thus, 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] For the 3 goods model we use the same parameters as for the two goods model, except that we change (recalibrate) the standard deviation of good specific shocks (from 8% to 9%) in order to keep the first two moments of price changes approximately at their previous levels.

[15] Other data sets used in the menu cost literature exhibit even higher kurtosis (see, e.g., Bhattarai and Schoenle [8]).

tightly control accuracy, thus being 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 as well as to tackle problems that were so far out of reach.

We apply this algorithm to an IRBC model with adjustment costs and up to 100 continuous state variables. We also solve an IRBC models with irreversible investment and up to 20 dimensions. In that application, the comparative advantage of the adaptive sparse grid displays 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. Lastly, 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 solve economic models that include much more heterogeneity than was previously possible.

Appendix

A Sparse Grids with Non-Zero Boundaries

In Sec. 2, we have assumed that the functions under consideration vanish at the boundary of the domain—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., [46, 35]). Doing this naively, one needs at least 3^d grid points, which makes the approach inapplicable to high-dimensional problems (see [35]). In what follows, we discuss two procedures that mitigate this problem.

One way of handling non-zero boundaries is the so-called Clenshaw–Curtis sparse grid $V_n^{S,CC}$ with equidistant support nodes (see, e.g., [41, 35, 6, 45]). The crucial idea is to have only one grid point at the lowest level of approximation. Technically, the difference to the sparse grid $V_{n,0}^S$ is that the index set of the support nodes is not given by Eq. 17, but rather by

$$I_{\vec{l}} := \begin{cases} \{\vec{i} : i_t = 1, 1 \le t \le d\} & \text{if } l = 1 \\ \{\vec{i} : 0 \le i_t \le 2,\ i_t \text{ even}, 1 \le t \le d\} & \text{if } l = 2 \\ \{\vec{i} : 1 \le i_t \le 2^{l_t - 1} - 1,\ i_t \text{ odd}, 1 \le t \le d\} & \text{else,} \end{cases} \tag{59}$$

and the one-dimensional basis functions are given by

$$\phi_{l_t,i_t}^{CC}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 1 - 2 \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 0 \\ \left\{ \begin{array}{ll} 2 \cdot x_t - 1 & \text{if } x_t \in \left[\frac{1}{2}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l = 2 \wedge i = 2 \\ \phi_{l,i}(x_t) & \text{else,} \end{cases} \tag{60}$$

where $\phi_{l,i}(x)$ is given by Eq. 7. The first three levels of these basis functions are displayed in Fig. 15. An example of a 2-dimensional "Clenshaw–Curtis" grid of level 4 is shown in Figs. 16 and 17. It is also worth mentioning that the number of grid points of the "Clenshaw–Curtis" grid $|V_n^{S,CC}|$ grows slightly slower than $|V_{0,n}^S|$ (cf. Tab. 9).

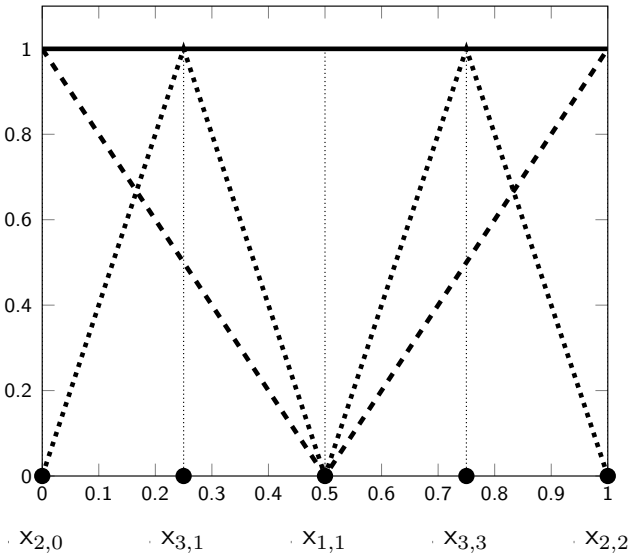

Figure 15: Hierarchical basis functions of the "Clenshaw–Curtis" grid in one dimension. Level 1 (solid), level 2 (dashed), and level 3 (dotted).

d	$\lvert V_4\rvert$	$\lvert V_{0,4}^S\rvert$	$\lvert V_4^{S,CC}\rvert$
1	15	15	9
2	225	49	29
3	3,375	111	69
4	50,625	209	137
5	759,375	351	241
10	$5.77 \cdot 10^{11}$	2,001	1,581
20	$3.33 \cdot 10^{23}$	13,201	11,561
50	$6.38 \cdot 10^{58}$	182,001	171,901
100	>Googol	1,394,001	1,353,801

Table 9: Number of grid points for several different grid types of level 4. First column—dimension; second column—full grid; third column—"classical" sparse grid with no points at the boundaries; last column—"Clenshaw–Curtis" sparse grid.

Another way to handle $f|_{\partial\Omega} \neq 0$ 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 close to the boundary is not required and where the underlying function does not change too much toward the boundary (see [46]). An appropriate choice for the modified one-dimensional basis functions is

$$\phi_{l_t,i_t}(x_t) = \begin{cases} 1 & \text{if } l = 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2 - 2^l \cdot x_t & \text{if } x_t \in \left[0, \frac{1}{2^{l-1}}\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 1 \\ \left\{ \begin{array}{ll} 2^l \cdot x_t + 1 - i & \text{if } x_t \in \left[1 - \frac{1}{2^{l-1}}, 1\right] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1 \wedge i = 2^l - 1 \\ \phi_{l,i}\left(x_t \cdot 2^l - i\right) & \text{else,} \end{cases} \tag{61}$$

where $\phi_{l,i}(x)$ is again given by Eq. 7, and where the support nodes have the same coordinates as in the "classical" sparse grid. The d-dimensional basis functions are obtained in analogy to

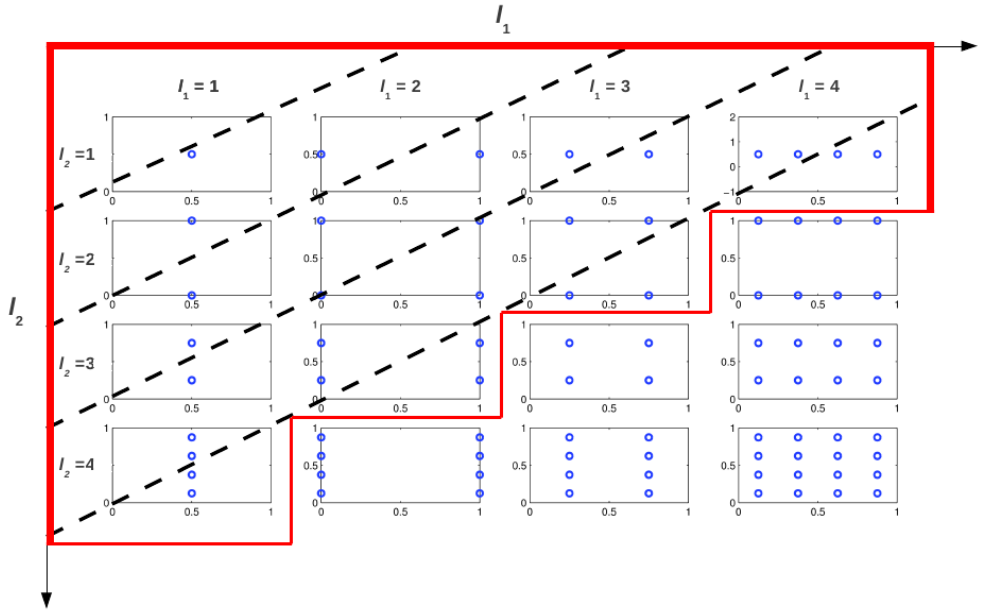

Figure 16: Schematic construction of a level 4 "Clenshaw–Curtis" sparse grid $V_4^{S,CC}$ in 2 dimensions (cf. Eq. 59). $V_4^{S,CC}$ consists of the hierarchical increment spaces $W_{(l_1,l_2)}$ that satisfy $|\vec{l}| \leq n+d-1$ (for $1 \leq l_1 \leq n$, and $1 \leq l_2 \leq n$). These increment spaces are enclosed by the red bold lines. The blue dots represent the grid points of the respective subspaces. Finally, the dashed black lines indicate the hierarchical increment spaces for constant $|\vec{l}|$.

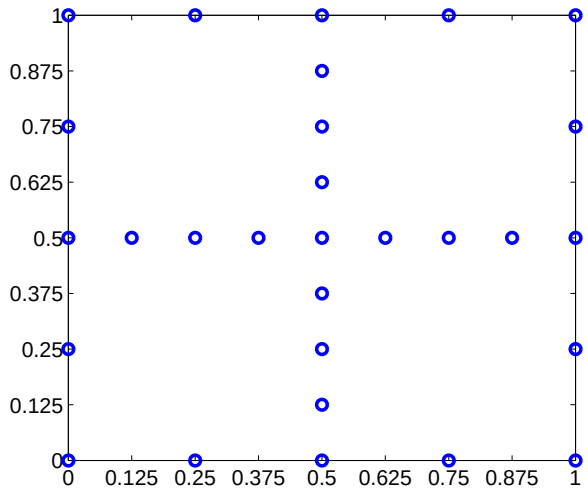

Figure 17: Sparse grid space $V_4^{S,CC}$ in 2 dimensions, constructed according to Eq. 27 from the increment spaces displayed in Fig. 16. Note that it contains only 29 support nodes, whereas a full grid of the same level would consist of 225 points.

Eq. 15 by a tensor product construction of the one-dimensional ones. We denote this function space by V_n^S, as it allows for non-zero boundaries.

B Hierarchical Integration

The (adaptive) sparse grid approach can also be used for high-dimensional numerical integration, such as the computation of expectations (see [24, 41, 45]).
Starting from Eq. 29, the expected value of the interpolant can be evaluated as follows:

$$\mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \int_{\Omega} \phi_{\vec{l},\vec{i}}(\vec{x}) d\vec{x}, \tag{62}$$

where we assume for simplicity that the probability density is 1 on $\Omega = [0,1]^d$. Using the basis functions described by Eqs. 59 and 60 as an example, the one dimensional integral can now easily be computed analytically by

$$\int_0^1 \phi_{l,i}\left(x\right) dx = \begin{cases} 1, & \text{if } l = 1 \\ \frac{1}{4} & \text{if } l = 2 \\ \\ 2^{1-l} & \text{else.} \end{cases} \tag{63}$$

The multi-dimensional integrals are simply given by the product of one dimensional integrals. Following [41], we denote $\int_{\Omega} \phi_{l,i}\left(\vec{x}\right) d\vec{x} = J_{\vec{l},\vec{i}}$, and can thus rewrite Eq. 62 by

$$\mathbb{E}\left[u(\vec{x})\right] = \sum_{|l|_1 \leq n+d-1} \sum_{\vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \cdot J_{\vec{l},\vec{i}}. \tag{64}$$

Eq. 64 states that the expectation is given by the arithmetic sum over all grid points of the product between the hierarchical surpluses and the integral weights.

References

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

[2] Eric M Aldrich, Jesús Fernández-Villaverde, A Ronald Gallant, and Juan 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] Fernando Alvarez and Francesco Lippi. Price setting with menu cost for multiproduct firms. *Econometrica*, 82(1):89–135, 2014.

[4] Cristina Arellano. Default risk and income fluctuations in emerging economies. *The American Economic Review*, pages 690–712, 2008.

[5] Francisco Barillas and Jesús Fernández-Villaverde. A generalization of the endogenous grid method. *Journal of Economic Dynamics and Control*, 31(8):2698 – 2712, 2007.

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

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

[8] Saroj Bhattarai and Raphael Schoenle. Multiproduct firms and price-setting: Theory and evidence from us producer prices. *Journal of Monetary Economics*, 66:178–192, 2014.

[9] Olivier Bokanowski, Jochen Garcke, Michael Griebel, and Irene Klompmaker. An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations. *Journal of Scientific Computing*, 55(3):575–605, 2013. also available as INS Preprint No. 1207.

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

[11] Johannes Brumm, Dmitry Mikushin, Simon Scheidegger, and Olaf Schenk. Scalable high-dimensional dynamic economic modeling using sparse grids. 2014, in preparation.

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

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

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

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

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

[17] Satyajit Chatterjee, Dean Corbae, Makoto Nakajima, and José-Víctor Ríos-Rull. A quantitative theory of unsecured consumer credit with risk of default. *Econometrica*, 75(6):1525–1589, 2007.

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

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

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

[21] Abhijeet Gaikwad and Ioane Muni Toke. Gpu based sparse grid technique for solving multidimensional options pricing pdes. In *Proceedings of the 2Nd Workshop on High Performance Computational Finance*, WHPCF '09, pages 6:1–6:9, New York, NY, USA, 2009. ACM.

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

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

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

[25] Mikhail Golosov and Robert E Lucas Jr. Menu costs and phillips curves. *Journal of Political Economy*, 115(2), 2007.

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

[27] Mario Heene, Christoph Kowitz, and Dirk Pflüger. Load balancing for massively parallel computations with the sparse grid combination technique. In *PARCO*, pages 574–583, 2013.

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

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

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

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

[32] Kenneth L Judd, Lilia Maliar, Serguei Maliar, and Rafael 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.

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

[34] Patrick Kehoe and Virgiliu Midrigan. Prices are sticky after all. *Journal of Monetary Economics, forthcoming*, 2014.

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

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

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

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

[39] Igor Livshits, James MacGee, and Michele Tertilt. Consumer bankruptcy: A fresh start. *The American Economic Review*, pages 402–418, 2007.

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

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

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

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

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

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

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

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

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

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

[50] James Reinders. *Intel Threading Building Blocks*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, first edition, 2007.

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

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

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

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

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

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