# Sparse grids for dynamic economic models *

Johannes Brumm,† Christopher Krause,‡ Andreas Schaab,§ Simon Scheidegger¶

December 2, 2021

**Abstract**

Dynamic economic models that capture salient real-world heterogeneity and non-linearity require the approximation of high-dimensional functions. To overcome the inherent curse of dimensionality—compute time and storage requirements growing exponentially with increasing complexity—sparse grid methods substantially reduce the number of interpolation nodes needed to achieve a desired level of accuracy. In particular, sparse grids accelerate a wide range of computational tasks arising in high-dimensional settings: interpolation, regression, classification, density estimation, quadrature, uncertainty quantification, and solving partial differential equations. The goal of this review article is three-fold. First, it provides an accessible introduction to the most popular versions of sparse grids used in economics: (spatially-adaptive) sparse grids with local basis functions, and (dimension-adaptive) Smolyak sparse grids that use global polynomial basis functions. Second, it illustrates how sparse grids work by applying them to two conceptually straightforward yet computationally demanding applications—an international real business cycle model with irreversible investment in discrete time, and an Aiyagari-type model with a jump in the capital-tax rate in continuous time. Third, it provides simple-to-use code examples, which constitute an easy starting point for applying sparse grid methods to dynamic economic models and which can be found under the following URL: `https://github.com/SparseGridsForDynamicEcon/SparseGrids_in_econ_handbook`.

Keywords: Sparse Grids, Smolyak Sparse Grids, Adaptive Sparse Grids, International Real Business Cycles, Occasionally Binding Constraints, Dynamic Programming, Time Iteration, Hamilton-Jacobi-Bellman Equations, Heterogeneous Agent Models
JEL Classification: C60, C63, C88, E20, E30, F44, G11

*This work was supported by the Swiss National Science Foundation (SNF), under project IDs "Can Economic Policy Mitigate Climate-Change?", and "New methods for asset pricing with frictions". Simon Scheidegger gratefully acknowledges support from the Enterprise for Society (E4S).

†Department of Economics and Management, Karlsruhe Institute of Technology, Germany, Email: johannes.brumm@kit.edu

‡Department of Economics and Management, Karlsruhe Institute of Technology, Germany, Email: christopher.krause@kit.edu

§Columbia Business School, Columbia University, USA, Email: ajs2428@columbia.edu

¶Corresponding author; Department of Economics, University of Lausanne, Switzerland; Email: simon.scheidegger@unil.ch

# 1 Introduction

**High-dimensional economic problems.** Dynamic economic modeling has for a long time heavily relied on two very convenient methodological approaches: First, maintaining assumptions that ensure the existence of representative households and firms. Second, approximating the dynamics of a model only locally around its steady state. Yet, being aware of the limitations of these approaches, economists have also gone beyond those by including meaningful heterogeneity in their models and by employing global solution methods. Doing both at the same time, however, remains a formidable challenge. The reason is simply that the computational cost of *naive* global solution methods increases exponentially with the dimensionality of the model, which in turn depends on the amount of heterogeneity included. This obstacle is referred to as the *curse of dimensionality* [Bellman, 1961]. We will explain below how sparse grid (SG) methods can alleviate this problem, which we now characterize more formally.

The dynamics of economic models can often be captured by *recursive equilibria* [Stokey et al., 1989, Ljungqvist and Sargent, 2004] where the potentially high-dimensional *state variable*, $x \in X \subset \mathbb{R}^d$, represents the state of the economy, $d$ is the dimensionality of the state space, and a time-invariant *equilibrium function*, $f : X \to Y \subset \mathbb{R}^m$, captures the model dynamics and can be characterized as the solution to a functional equation that reads:

$$\mathcal{E}(f) = \mathbf{0}. \tag{1}$$

This abstract description nests characterizations of recursive equilibria in discrete time, where $f$ is the value function, and the operator $\mathcal{E}$ captures the Bellman equation it has to satisfy—or the Hamilton-Jacobi-Bellman (HJB) equation in continuous time. It also nests characterizations where $\mathcal{E}$ captures discrete-time first-order equilibrium conditions, and where $f$ is the (multi-dimensional) policy function.[1] No matter which of these characterizations is used, the curse of dimensionality bites as soon as $X$ is of a higher dimension, and a global solution needs to be computed. In contrast to a local solution, which relies on equilibrium conditions (and their derivatives) at a single point in the state space, a global solution aims to satisfy the equilibrium conditions throughout the state space. A grid-based solution that relies on a tensor-product construction will require $M^d$ points when $M$ points are needed in each dimension. As this exponential growth makes tensor-product grids infeasible as soon as $M$ and $d$ reach moderate levels (say $d > 3$), there are basically two ways to proceed: Grid-free approximation methods[2] and SG methods. Figure 1 provides a classification of solution methods for dynamic economic models.

**Sparse grid methods.** SG methods are a very efficient and systematic way to tackle the numerical difficulties that arise in dynamic economic models with high-dimensional state spaces. While there exists a vast body of literature on how SGs can be constructed,[3] all approaches share the following fundamental construction principle: univariate interpolation formulae are extended to the multivariate case by choosing linear combinations of tensor products in a way that reduces the number of support nodes (that is, grid points) by orders of magnitude relative to an entire tensor-product grid without substantially worsening interpolation errors, thereby alleviating the curse of dimensionality. The most fundamental distinction between the various SG techniques proposed in the literature is between those using basis functions with local support and those using global basis functions. In the following, we refer to these as i) local sparse grid (LSG) and ii) global sparse grid (GSG) techniques, respectively. LSGs are suitable for handling non-smooth functions with locally distinct behavior such as kinks,

[1] For a thorough review on how to numerically solve dynamic models in general, see the excellent handbook chapter by Maliar and Maliar [2014].

[2] Grid-free approaches have been proposed, e.g., by Den Haan and Marcet [1990], Maliar and Maliar [2014] and have recently become more powerful by leveraging developments in machine learning—see for instance Duffy and McNelis [2001], Norets [2012], Maliar et al. [2021], Azinovic et al. [2019], Fernández-Villaverde et al. [2019], Scheidegger and Bilionis [2019]. We doubt, however, that SGs will be replaced entirely by any of those methods anytime soon. In contrast to most machine learning approaches, SGs have well-understood convergence properties. They can help researchers to scale up any model (if numerically formulated on a grid) to higher dimensions without the need to replace the entire solution technique. Furthermore, they can relatively easily be parallelized [Brumm et al., 2015], which can speed up the time-to-solution by orders of magnitude. So, all in all, SGs can provide a relatively straightforward way to scale up existing modeling and solution frameworks and, in turn, provide a simple means to extend the boundaries of research.

[3] These include the Smolyak algorithm (see Smolyak [1963]), the *classical* SG method (see, e.g., Zenger [1991], Bungartz and Griebel [2004], and references therein), the combination technique (see, e.g., Griebel et al. [1992], Obersteiner and Bungartz [2021]), and dimension- as well as spatially-adaptive methods (see, e.g., Nobile et al. [2008], Pflüger [2010], and references therein).

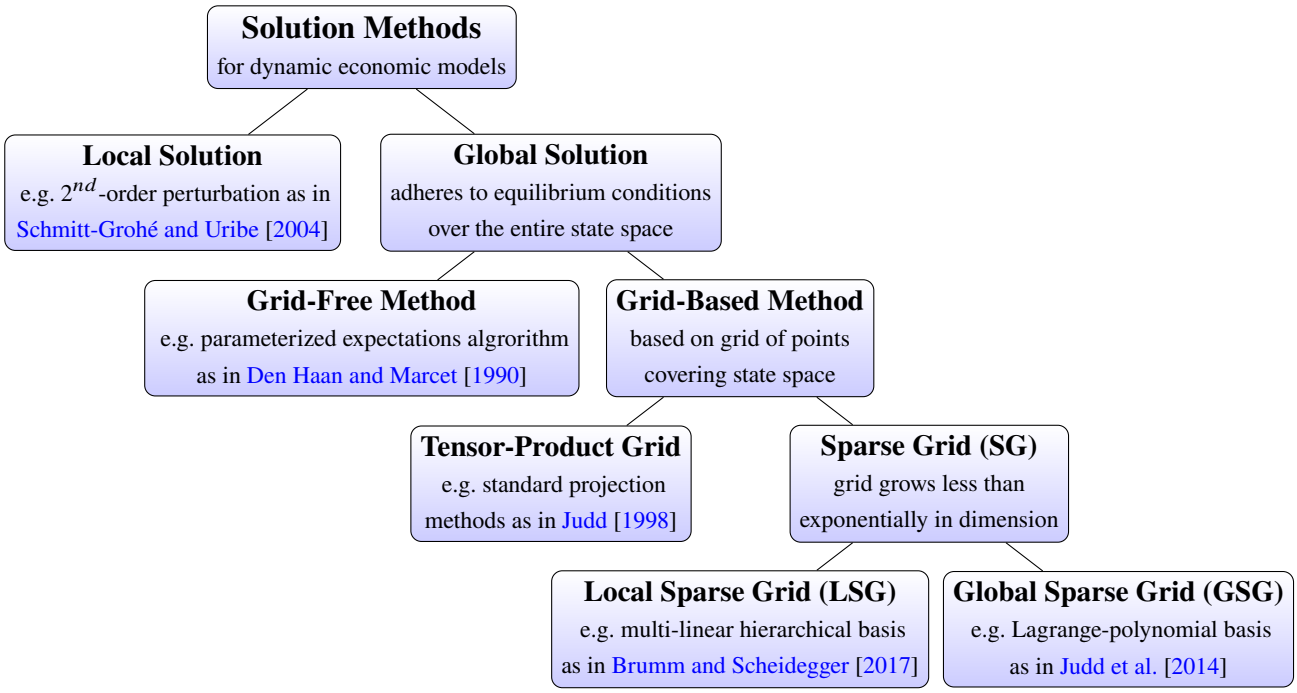



**Fig. 1** Overview of numerical solution methods for dynamic stochastic economic models with popular examples. SG methods provide grid-based global solutions that do not suffer from the curse of dimensionality like tensor-product grids. We distinguish between LSGs and GSGs depending on whether the basis functions have local or global support.

whereas GSGs are more suitable for approximating smooth functions. For these two cases, we choose and present below the specific methods that are arguably most useful and popular in economics: i) LSGs based on hierarchical, multi-linear basis functions, and ii) GSGs based on Lagrange characteristic polynomials (to which the economics literature often refers to as the *Smolyak* method).[4] Throughout this review article, we apply those methods to problems of relatively low dimension—for illustrative purposes and also to show that (adaptive) SGs are of high added value even under such circumstances.

**Sparse grids in economics and finance.** In economics and finance, both GSGs and LSGs have gained substantial popularity for solving and estimating high-dimensional models in both discrete and continuous time, as this by no means complete survey of the literature shows. Krueger and Kubler [2004, 2006] introduced GSG (the *Smolyak* SG) to economics in the context of (discrete-time) overlapping generations (OLG) models, whereas Fernández-Villaverde et al. [2015] use that very same type of GSGs to study non-linear dynamics in a New Keynesian model with a zero lower bound on the nominal interest rate. Judd et al. [2014] improve the Smolyak method in various directions to substantially enhance its performance in economic applications. LSGs were introduced to economics by Brumm and Scheidegger [2017] in the context of an international real business cycle (IRBC) model with irreversible investment, and a menu-cost model that generalizes the setup by Midrigan [2011]. Brumm et al. [2017] use LSGs to tackle annually-calibrated OLG models with aggregate shocks, whereas Usui [2019] apply those grids to investigate rare natural disasters and adaptation decisions in a dynamic stochastic economy in discrete time. Cao et al. [2020] use LSGs to solve (discrete-time) DSGE models with strong nonlinear features. Recent work, for example, by Garcke and Ruttscheidt [2019] introduces continuous-time adaptive LSG methods to macroeconomics in the context of heterogeneous agent models. Schaab [2020] solves HJB-type equations with LSGs to study the interaction between micro and macro

[4]For a complete list of SG algorithms, see Stoyanov [2015], and references therein. SGs have a long-standing success story of enabling and accelerating the solution of a wide range of computational problems in physics, chemistry, visualization, and data mining, including classification (see, e.g., Bungartz et al. [2008]), density estimation and sampling (see, e.g., Frommert et al. [2010]), numerical quadrature (see, e.g., Holtz [2010], and references therein), uncertainty quantification (see, e.g., Rehme et al. [2021]), partial differential equations in general (see, e.g., Babuška et al. [2007], Gunzburger et al. [2014]), optimal control and HJB-type of problems (see, e.g., Bokanowski et al. [2013]) to name just a few. For a thorough review on SGs, see, e.g., Bungartz and Griebel [2004], Pflüger [2010].

uncertainty in a globally solved heterogeneous agent new Keynesian model with aggregate risk, counter-cyclical unemployment risk, and a zero lower bound constraint on monetary policy. In finance, a series of authors used LSGs (both in discrete- as well as in continuous-time settings) to study, for instance, high-dimensional option-pricing problems (see, e.g., Reisinger and Wittum [2007], Bungartz et al. [2012], Scheidegger and Treccani [2018]) as well as dynamic portfolio choice models with transaction costs as in Schober et al. [2021]. Heiss and Winschel [2008] integrate likelihood functions on GSGs. Winschel and Krätzig [2010] embed GSGs in a Bayesian estimation framework, whereas Gilch et al. [2021] apply LSGs in the context of generalized method of moments.

**Goals and outline.** The three main goals of this review article are as follows. First, to provide a concise and accessible introduction to SGs focusing on those LSG and GSG techniques most commonly employed in economic applications. Second, to illustrate the working of those methods by applying them to conceptually simple yet computationally demanding economic test cases—an IRBC model with irreversible investment in discrete-time and an Aiyagari [1994]-type model with a jump in the capital-tax rate in continuous time. Third, to provide simple to use publicly available codes posted under the following link: `https://github.com/SparseGridsForDynamicEcon/SparseGrids_in_econ_handbook`.

The remainder of this review is organized as follows: Section 2 introduces LSGs and GSGs. Section 3 demonstrates how LSGs can be used to solve high-dimensional non-linear models in discrete time settings. Section 4 illustrates how LSGs can be applied to continuous-time models. Finally, section 5 briefly presents some of the most common drawbacks SGs can suffer from, how they can be overcome, and then makes some concluding remarks.

## 2 Sparse grids in a nutshell

Consider the task of approximating a (policy or value) function $f : \Omega \to \mathbb{R}$ which maps some input state $\mathbf{x} \in \mathbb{R}^d$ to some output value $f(\mathbf{x})$, where $d$ is the dimensionality of the problem at hand. When $d$ is larger than one yet of moderate size (e.g., $d < 4$), such problems are often solved on full (Cartesian) grids. Full grids, however, become very expensive in a computational sense when used to solve models of higher dimension, as their size grows exponentially in $d$. SGs, in contrast, alleviate this curse and allow for the solution of the problems with much smaller effort and at the cost of slightly deteriorated errors. We now present SGs in a compact yet rigorous way, starting from the basic setting common to all SG techniques.

In the following, we restrict the domain of interest to the compact sub-volume $\Omega = [0, 1]^d$. This situation can be achieved for most other settings by re-scaling and possibly carefully truncating the original domain. Furthermore, we assume that the *true* $f$ can be evaluated at arbitrary points in $\Omega$, and it is *expensive* to evaluate, which means that the numerical evaluations of $f$ might take much time. Thus, we want to replace $f$ with another function $u$ that approximates $f$ well but is much cheaper to evaluate:

$$f(\mathbf{x}) \approx u(x) = \sum_{i=1}^{N} \xi_i \cdot \Phi_i(\mathbf{x}), \tag{2}$$

with $N$ basis functions $\Phi_i$ and coefficients $\xi_i$. The $\Phi_i(\mathbf{x})$ are basis functions with local or global support, and the weights $\xi_i$ are computed from the values $f(\chi_i)$ at so-called collocation points $\{\chi_i\}_{i=1}^{N} \subset \Omega$.

In this section, we proceed in three steps: Section 2.1 presents LSGs based on hierarchical, multi-linear basis functions. Section 2.2 presents GSGs based on Lagrange characteristic polynomials. Section 2.3 moves beyond regular grids—which treat different dimensions and regions of the domain with the same level of importance—by discussing dimension as well as spatial adaptivity: adding grid points and thus the computational resources where they are needed most for the particular problem at hand. Furthermore, analytical examples to foster the basic intuition are provided. Finally, note that there exist highly performant and easy-to-use open-source implementations of SG methods in a broad range of programming languages, with some of the most popular ones

probably being *SG++*,[5] the sparse grids Matlab kit,[6] spinterp,[7] and TASMANIAN.[8] Furthermore, to facilitate the replicability of our examples, all of those below were generated with TASMANIAN and that can be found in the following repository: `https://github.com/SparseGridsForDynamicEcon/SparseGrids_in_econ_handbook`.

## 2.1 Local sparse grids

In this section, we follow Brumm and Scheidegger [2017] in introducing LSGs based on multi-linear, hierarchical basis functions.

**One-dimensional functions.** The fundamental building blocks are 1-dimensional piecewise linear functions,[9]

$$\phi_{l,i}(x) = \begin{cases} 1, & l = i = 1, \\ \max(1 - 2^{l-1} \cdot |x - x_{l,i}|, 0), & i = 0, \ldots, 2^{l-1}, \ l > 1, \end{cases} \tag{3}$$

which depend on a refinement level $l \in \mathbb{N}$ and index $i \in \mathbb{N}$. The corresponding grid points are distributed as

$$x_{l,i} = \begin{cases} 0.5, & l = i = 1, \\ i \cdot 2^{1-l} = i \cdot h_l, & i = 0, ..., 2^{l-1}, \ l > 1. \end{cases} \tag{4}$$

Thus, in the 1-dimensional case stated here, we interpolate on an equidistant grid $\Omega_l$ on $\Omega = [0, 1]$ with mesh width $h_l = 2^{1-l}$ and grid points $x_{l,i} = i \cdot h_l$, where $l \in \mathbb{N}, i \in \{0, 1, ..., 2^{l-1}\}$. For instance, let $l = 3$, $i = 3$, then $x_{l,i} = 3 \cdot 2^{1-3} = 0.75$, as shown in panel (a) of figure 2. Next, we want to ensure that the construction of the approximator (cf. equation (2)) makes it possible to utilize all the previous results when increasing the refinement level $l$ to improve the interpolation. By choosing appropriate points for interpolating the 1-dimensional function, it can be ensured that the sets of points are nested to the extent that the interpolation from level $l - 1$ to $l$ demands that only the function at the grid points that are unique to the *new* level $l$ have to be evaluated. Formally, such a hierarchical decomposition of the underlying approximation space can be achieved by introducing hierarchical index sets $I_l$:

$$I_l = \begin{cases} \{i = 1\}, & \text{if } l = 1, \\ \{0 \le i \le 2, i \text{ even}\} & \text{if } l = 2, \\ \{0 \le i \le 2^{l-1}, i \text{ odd}\} & \text{else}, \end{cases} \tag{5}$$

that lead to so-called hierarchical subspaces $W_l$ spanned by the corresponding basis $\phi_l = \{\phi_{l,i}(x), i \in I_l\}$. Panel (a) of figure 2 depicts the basis functions up to level 3. The direct sum of the hierarchical increment spaces results in the function space:

$$\mathcal{V}_l = \bigoplus_{k \le l} W_k. \tag{6}$$

Using this hierarchical basis, a given function $f$ can then be uniquely approximated by $u \in \mathcal{V}_l$ with coefficients $\alpha_{k,i} \in \mathbb{R}$, that is,

$$f(x) \approx u(x) = \sum_{k=1}^{l} \sum_{i \in I_k} \alpha_{k,i} \cdot \phi_{k,i}(x). \tag{7}$$

The coefficients $\alpha_{k,i}$ in equation (7) are commonly termed *hierarchical surpluses*. Due to the nested structure of the hierarchical grid they can easily be determined: $\alpha_{k,i}$ simply corrects the interpolant of level $k - 1$ at the point $x_{k,i}$ to the actual value of $f(x_{k,i})$, as displayed (by arrows) in panel (a) of figure 2. One of the key

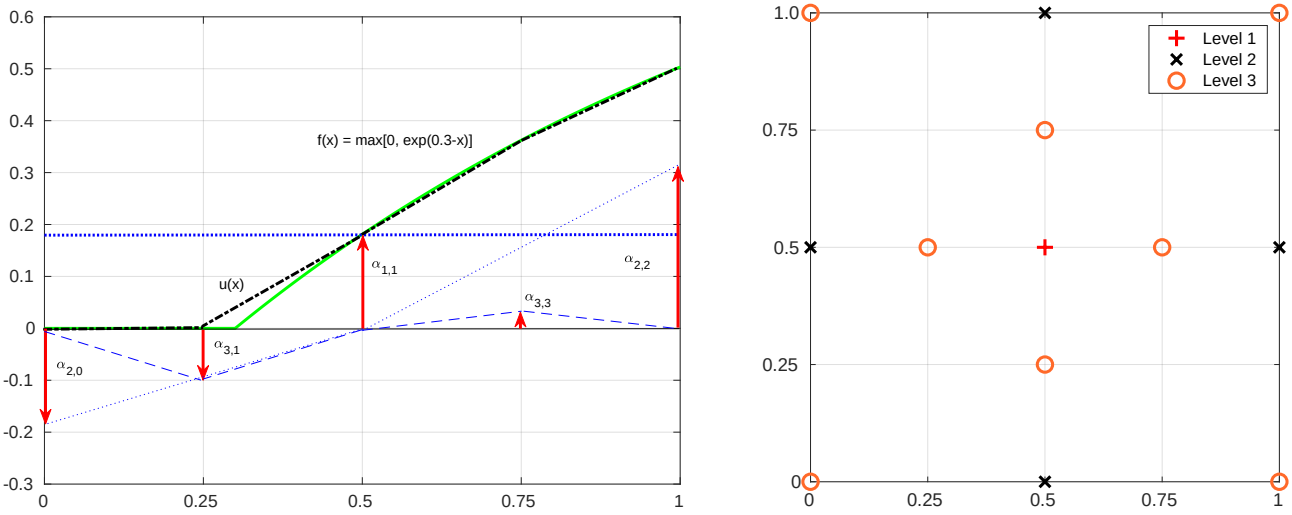
[5] see `https://sgpp.sparsegrids.org`.

[6] see `https://sites.google.com/view/sparse-grids-kit`.

[7] see `https://people.math.sc.edu/Burkardt/m_src/spinterp/spinterp.html`.

[8] see `https://tasmanian.ornl.gov`.

[9] Following for instance Ma and Zabaras [2009], we choose basis functions that can handle non-zero boundaries, as needed in most economic applications.




**(a)** Construction of $u(x)$ interpolating $f(x)$ **(b)** SG of level 3

**Fig. 2** Panel (a): construction of $u(x)$ (dashed-dotted black line) interpolating $f(x) = \max[0, 1 - \exp(0.3 - x)]$ (green line) with hierarchical linear basis functions of levels 1, 2, and 3. The hierarchical surpluses $\alpha_{l,i}$ that belong to the respective basis functions are indicated by arrows. They are simply the difference between the function values at the current and the previous interpolation levels. Panel (b): construction of $\mathcal{V}_3^S$ (see equation (14)), consisting of all the points contained in the respective hierarchical increment spaces $W_{(l_1,l_2)}$ for $1 \leq l_1, l_2 \leq n = 3$.

observations when looking at panel (a) of figure 2 is that the size of the support (basically the set where the function is non-zero) of $\phi_{k,i}$ is smaller in higher levels: The size of the support is proportional to $2^{-l}$, that is, it halves in size for each level.

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

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \prod_{t=1}^{d} \phi_{l_t,i_t}(x_t), \tag{8}$$

They span the multivariate subspaces by

$$W_{\mathbf{l}} = \operatorname{span}\{\phi_{\mathbf{l},\mathbf{i}} : \mathbf{i} \in I_{\mathbf{l}}\}, \tag{9}$$

with the index set $I_{\mathbf{l}}$ given by a multidimensional extension to equation (5):

$$I_{\mathbf{l}} = \begin{cases} \{\mathbf{i} : i_t = 1, 1 \leq t \leq d\} & \text{if } l = 1, \\ \{\mathbf{i} : 0 \leq i_t \leq 2, i_t \text{ even}, 1 \leq t \leq d\} & \text{if } l = 2, \\ \{\mathbf{i} : 0 \leq i_t \leq 2^{l_t-1}, i_t \text{ odd}, 1 \leq t \leq d\} & \text{else.} \end{cases} \tag{10}$$

The space of piecewise multi-linear functions $\mathcal{V}_n$ on a Cartesian grid with mesh size $h_n$ for a given level $n$ is

| d | $\lvert\mathcal{V}_4\rvert$ | $\lvert\mathcal{V}_4^S\rvert$ | $\lvert\mathcal{V}_5^S\rvert$ | $\lvert\mathcal{V}_6^S\rvert$ |
|---|---|---|---|---|
| 1 | 9 | 9 | 17 | 33 |
| 2 | 81 | 29 | 65 | 145 |
| 3 | 729 | 69 | 177 | 441 |
| 4 | 6,561 | 137 | 401 | 1,105 |
| 5 | 59,049 | 241 | 801 | 2,433 |
| 10 | $3.49 \cdot 10^9$ | 1,581 | 8,801 | 41,265 |
| 20 | $1.22 \cdot 10^{19}$ | 11,561 | 120,401 | 1,018,129 |
| 50 | $5.15 \cdot 10^{47}$ | 171,901 | 4,352,001 | 88,362,321 |
| 100 | $2.66 \cdot 10^{95}$ | 1,353,801 | 68,074,001 | $2.74 \cdot 10^9$ |

**Table 1** Number of points for several different grid types. First column: dimension; second column: full grid of refinement level 4; columns three to five: SG with non-zero boundaries (levels four to six).

then defined by the direct sum of the hierarchical increment spaces (cf. equation (9)):

$$\mathcal{V}_n = \bigoplus_{|l|_\infty \leq n} W_{\mathbf{l}}, \quad |l|_\infty = \max_{1 \leq t \leq d} l_t. \tag{11}$$

The interpolant of $f$, namely, $u(\mathbf{x}) \in \mathcal{V}_n$, can now uniquely be represented by

$$f(\mathbf{x}) \approx u(\mathbf{x}) = \sum_{|l|_\infty \leq n} \sum_{\mathbf{i} \in I_{\mathbf{l}}} \alpha_{\mathbf{l},\mathbf{i}} \cdot \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}). \tag{12}$$

For a sufficiently smooth function $f$ (see Zenger [1991], Bungartz and Griebel [2004], and references therein) and its interpolant $u \in \mathcal{V}_n$, we obtain an asymptotic error decay of $\|f(\mathbf{x}) - u(\mathbf{x})\|_{L_2} \in O(2^{-2n})$, but at the cost of $O(2^{nd})$ grid points, encountering the so-called *curse of dimensionality*. This is a prohibitive obstacle for the accurate solution of problems with more than four or five dimensions. For instance, a 10-dimensional problem with a resolution of 9 points in each dimension (i.e., $\mathcal{V}_4$) needs almost $10^{10}$ coefficients, as shown in the second column of table 1.

**Sparse grid construction.** To alleviate the curse of dimensionality, we need to construct approximation spaces that are better than $\mathcal{V}_n$ in the sense that a much lower number of grid points leads to similar accuracy. To this end, one needs to find a way of using tensor products in a special way. The classical SG construction arises from a cost-benefit analysis in function approximation (see, e.g., Bungartz and Griebel [2004] for details). For functions with bounded second-order mixed derivatives, it can be shown that the hierarchical coefficients $\alpha_{\mathbf{l},\mathbf{i}}$ decay rapidly with increasing grid level $\mathbf{l}$, that is,

$$|\alpha_{\mathbf{l},\mathbf{i}}| = O\left(2^{-2|\mathbf{l}|_1}\right), \tag{13}$$

where $|\mathbf{l}|_1 = l_1 + \ldots + l_d$. The strategy for constructing an SG is to leave out those subspaces within the full grid space $\mathcal{V}_n$ that only contribute little to the interpolant. It can be shown that an optimization that minimizes the approximation error (in the $L_2$- or $L_\infty$-norm) given a fixed number of grid points leads to the *sparse grid* space of level $n$, defined by

$$\mathcal{V}_n^S = \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}}. \tag{14}$$

In contrast to the full Cartesian grid, where the maximum of the grid refinement levels across dimensions is restricted (see equation (11)), now it is the sum of these that is restricted – yet, it is in line with equation (13), which states that the hierarchical surpluses decay with growing $|\mathbf{l}|_1$. Panel (b) of figure 2 displays the construction

of $\mathcal{V}_3^S$ in two dimensions. The number of grid points required by the space $\mathcal{V}_n^S$ is given by

$$|\mathcal{V}_n^S| = O\left(2^n \cdot n^{d-1}\right), \tag{15}$$

which is a massive reduction compared to $O(2^{nd})$ for the full grid space $\mathcal{V}_n$ (see table 1 for concrete examples). In analogy to equation (12), a function $f \in \mathcal{V}_n^S \subset \mathcal{V}_n$ can now be approximated by

$$f(\mathbf{x}) \approx u(\mathbf{x}) = \sum_{|\mathbf{l}|_1 \leq n+d-1} \sum_{\mathbf{i} \in I_\mathbf{l}} \alpha_{\mathbf{l},\mathbf{i}} \cdot \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}). \tag{16}$$

The asymptotic accuracy of the interpolant deteriorates only slightly from $O(2^{-2n})$ in the case of the full grid to $O(2^{-2n} \cdot n^{d-1})$, as shown, for instance, in Bungartz and Griebel [2004]. This demonstrates why SGs are so well suited to high-dimensional problems. In contrast to full grids, their size increases only moderately with dimensions, while they are only slightly less accurate than full grids.

Finally, note that LSGs are not restricted to the multi-linear hat functions presented above. Several other local basis functions have been proposed in the literature, including piecewise polynomials (see, e.g., Pflüger [2010], and references therein), wavelets (see, e.g., Gunzburger et al. [2014]), or B-splines (see, e.g., Schober et al. [2021]). Those alternative basis functions can, in certain situations, provide superior approximation performance over the ones presented in this section. We deliberately focus on linear hat functions, as they are simple and convenient, in particular when it comes to adaptive refinement procedures, which we present in section 2.3.

## 2.2 Global sparse grids

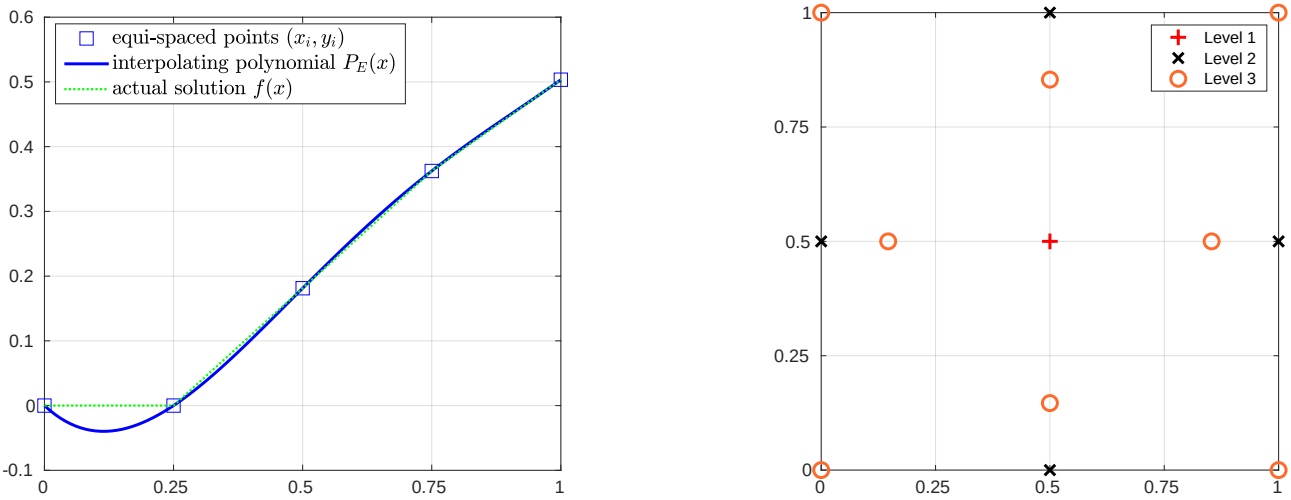
We now introduce—following the notation of Ma and Zabaras [2009]—an SG construction that relies on global basis functions, that is, polynomials which have the whole domain $[0, 1]$ as their support. In economics, such GSGs are widely used and, as mentioned before, typically referred to as Smolyak SGs (see, e.g., Krueger and Kubler [2004], Judd et al. [2014]), since their construction relies on the algorithm of Smolyak [1963]. The latter provides, as above, a way to construct multi-dimensional interpolators based on using tensor products in a combinatorial way, thus alleviating the curse of dimensionality. In the context of economic applications, Smolyak SGs turn out to be very useful in situations where a global solution is required, but the model at hand does not include distinct local features such as kinks. In those situations, global polynomials can provide a superior approximation performance over piecewise multi-linear basis functions.

**One-dimensional functions.** The fundamental building blocks for Smolyak SGs are the univariate Lagrange characteristic polynomials, that is,

$$\mathcal{L}_{i,l}(x) = \begin{cases} 1, & \text{for } l = 1, \\ \prod_{k=1, k\neq i}^{m_l} \frac{x - x_{k,l}}{x_{i,l} - x_{k,l}}, & \text{for } l > 1 \text{ and } i = 1, \ldots, m_l, \end{cases} \tag{17}$$

where $m_l$ is again the number of elements in a set of collocation points. Panel (a) of figure 3 displays the approximation of a 1-dimensional test function by using a Lagrange polynomial with five uniformly distributed grid points. It becomes visible that the polynomial approximation with uniform grid spacing slightly suffers from the so-called Runge's phoenomenon.[10] As in the case of LSGs, it is advantageous to choose the collocation points in a nested fashion. One popular choice is the Clenshaw-Curtis grid at the non-equidistant extrema of the Chebyshev polynomials (see, e.g., Xiu and Hesthaven [2005], Nobile et al. [2008]). Thus, for any $m_l > 1$,

[10] Runge's phenomenon is the problem of oscillation at the edges of an interval that occurs when using polynomial interpolation with polynomials of a high degree over a set of equispaced interpolation points.




**(a)** Construction of $P_E(x)$ interpolating $f(x)$ **(b)** Smolyak SG of level 3

**Fig. 3** Panel (a): interpolation of $f(x) = \max[0, 1-\exp(0.3-x)]$ using Lagrange polynomials with equi-spaced grid points (denoted $P_E(x)$). Note that Runge's phenomenon becomes visible for this latter setting, resulting in undershooting the true solution for $x < 0.25$. Panel (b): construction of a Smolyak SG of level 3, with the incremental contributions of grid points from all lower levels (cf. equation (21)).

the sets of points $\chi_l = \{x_{1,l}, \cdots, x_{m_l,l}\}$ are characterized by

$$\begin{aligned} m_l &= \begin{cases} 1, & \text{if } l = 1, \\ 2^{l-1}+1, & \text{if } l > 1, \end{cases} \\ x_{i,l} &= \begin{cases} \left(-\cos\left(\pi(i-1)/(m_l-1)\right)+1\right)/2, & \text{for } i = 1,\ldots,m_l, \quad \text{if } m_l > 1, \\ 0.5, & \text{for } i = 1, \quad \text{if } m_l = 1. \end{cases} \end{aligned} \tag{18}$$

Consequently, the univariate interpolator is now given by

$$P_l(x) = \sum_{i=1}^{m_l} y_i \cdot \mathcal{L}_{i,l}(x), \tag{19}$$

where $y_i = f(x_{i,l})$ is the function value at this particular grid point.

**Multi-dimensional functions.** The 1-dimensional interpolator can be extended to the $d-$dimensional case by forming a full tensor product that reads as

$$P(\mathbf{x}) = \sum_{i_1=1}^{m_1} \cdots \sum_{i_d=1}^{m_d} f\left(x_{i_1,l_1}, \ldots, x_{i_d,l_d}\right) \cdot \left(\mathcal{L}_{i_1,l_1} \otimes \cdots \otimes \mathcal{L}_{i_d,l_d}\right). \tag{20}$$

where, from now on, again $\mathbf{x} \in \mathbb{R}^d$.

**Sparse grid construction.** Equation (20) now serves as a key ingredient for the so-called Smolyak algorithm [Smolyak, 1963]. The latter namely constructs the sparse interpolant $u_{n,d}$ using combinatorial products of 1-dimensional functions (cf. equation (19)) as follows (see, e.g., Wasilkowski and Wozniakowski [1995]):

$$u_{n,d}(\mathbf{x}) = \sum_{n-d+1 \leqslant |\mathbf{l}|_1 \leqslant n} (-1)^{n-|\mathbf{l}|_1} \cdot \begin{pmatrix} d-1 \\ n-|\mathbf{l}|_1 \end{pmatrix} \cdot \left(P_{l_1}(\cdot) \otimes \cdots \otimes P_{l_d}(\cdot)\right), \tag{21}$$

where $n \geqslant d, u_{d-1,d} = 0$ holds, the multi-index $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}^d$, and $|\mathbf{l}|_1 = l_1 + \ldots + l_d$. As above, $l_t$ (for $1 \leq t \leq d$) is the level of interpolation along the $t$-th dimension. Therefore, the Smolyak algorithm builds the interpolation function by adding a combination of 1-dimensional functions of order $l_t$ with the constraint that

the sum total ($|\mathbf{l}|_1 = l_1 + \cdots + l_d$) across all dimensions is between $n-d+1$ and $n$.[11] Panel (b) of figure 3 displays a Smolyak SG of level 3. Moreover, note that the number of points contained in a Smolyak SG increases with levels and dimensions similar to the numbers presented in table 1, that is, at a sub-exponential rate (see also Judd et al. [2014] for detailed tables).

Finally, notice that both LSGs, as well as GSGs, possess analytical properties that allow for highly efficient numerical quadrature of functions over the entire function space (see, e.g., Holtz [2010]). Unfortunately, in economic applications, where integration often does not take place over the same spaces that the policy functions are interpolated on, SG quadrature cannot be applied directly.[12]

## 2.3 Adaptive sparse grids & analytical examples

In the two types of SG constructions that we have seen above, the support nodes were pre-determined. Thus, they treat every dimension and region of the computational domain $[0,1]^d$ as equally important. However, if the function $f$ that needs to be approximated has different characteristics in the different dimensions or specific regions of the domain, trying to increase the resolution of the approximator via adaptivity to exploit these asymmetries might lead to better performance, that is, a smaller approximation error with the same number of grid points. Conversely, fewer grid points are necessary to achieve the same error. In the following, we will briefly discuss the basic notion behind the two most important adaptation strategies: dimension-wise and spatial adaptivity. Furthermore, to foster the basic intuition on how to use adaptivity in practice to approximate high-dimensional functions efficiently, we provide an analytical example on spatially-adaptive SGs (ASGs).[13]

**Dimension-adaptive sparse grids.** Recall that regular LSGs or GSGs are constructed using a diagonal cut, as indicated in the left panel of figure 4 (cf. equation (14)), where the dashed red diagonal lines indicate the corresponding function space which treats all dimensions equally. However, there might be situations where one dimension might be more important than another. Consider for instance the simple bivariate function $f(x,y) = \cos(20x) \cdot \sin(y)$. The latter oscillates much faster in the first than in the second dimension. Consequently, it would be important to place more points along the x-axis than the y-axis to deal with this particularity. To lift this obstacle, dimension-adaptive SGs come to the rescue. Recall that SGs can be built via the combination technique. The latter composes an SG as a combination of full grids and applies both to SG formulations with local and global basis functions. By slightly adjusting the Smolyak algorithm (cf. equation (21)), formulas can be derived where the resulting SG is indeed anisotropic, that is, some dimensions obtain a higher resolution of points than others. For more details, see, for example, Hegland [2003], Gerstner and Griebel [2003], and references therein for the general case of dimensionally-adaptive SGs, and Judd et al. [2014] for their application to economics.

**Spatially-adaptive sparse grids.** Dimensional adaptivity is useful, but sometimes it cannot provide satisfactory performance in the case of highly non-linear functions that need to be approximated. Imagine for instance a multi-variate test function $f_d : [0,1]^d \to \mathbb{R}$ that reads

$$f_d(\mathbf{x}) = \max\left(0, 1 - e^{\frac{1}{2} - \left(\prod_{t=1}^{d} (x_t + \frac{1}{5})\right)^{\frac{1}{d}}}\right). \tag{22}$$

As visible in the leftmost panel of figure 5 for the 2-dimensional case, this function has flat regions, low-curvature regions, high-curvature regions, and even a kink.

Regular or dimensionally adaptive SGs would have as many grid points in those regions where the function is (almost) flat as in the other regions. As the corresponding hierarchical surpluses (in the context of the

[11] Notice that the classical SG can also be written in terms of the Smolyak algorithm stated here (see, e.g., Griebel et al. [1992], Ma and Zabaras [2009], Obersteiner and Bungartz [2021], which is sometimes also termed as the *combination technique*. Furthermore, it is important to mention that there are sometimes significant computational gains when constructing an SG expressed in terms of full grid contributions instead of hierarchical contributions on the incremental grids. This means that existing algorithms that work on full grids (such as highly tuned solvers for partial differential equations) may be reused to compute SGs. The different and independent full grids can be processed independently (or even in parallel), and after the computation has finished, the full grid solutions can be combined according to the combinatorial formula.

[12] For more details on the numerical quadrature in dynamic models, see section 3 below.

[13] We restrict ourselves here deliberately to the use of ASGs, as the performance of dimension-adaptive GSGs was extensively discussed in the previous literature, including a provision for example codes (see, e.g., Judd et al. [2014]).

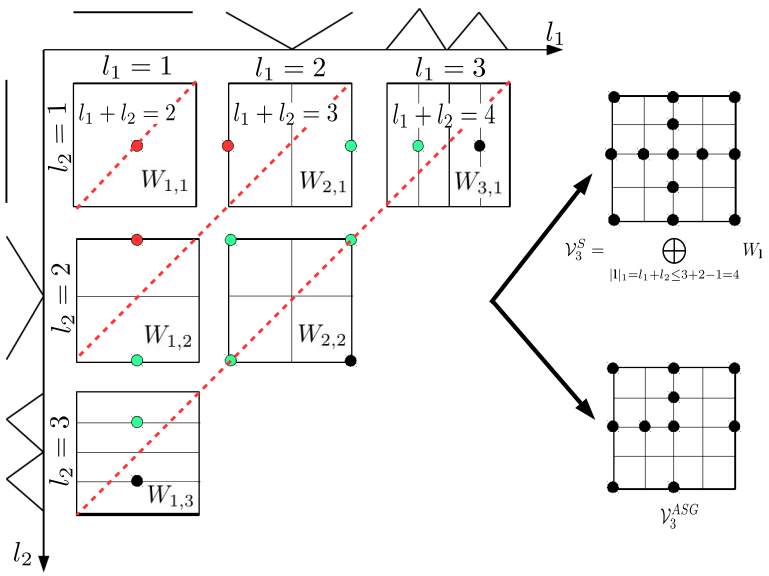
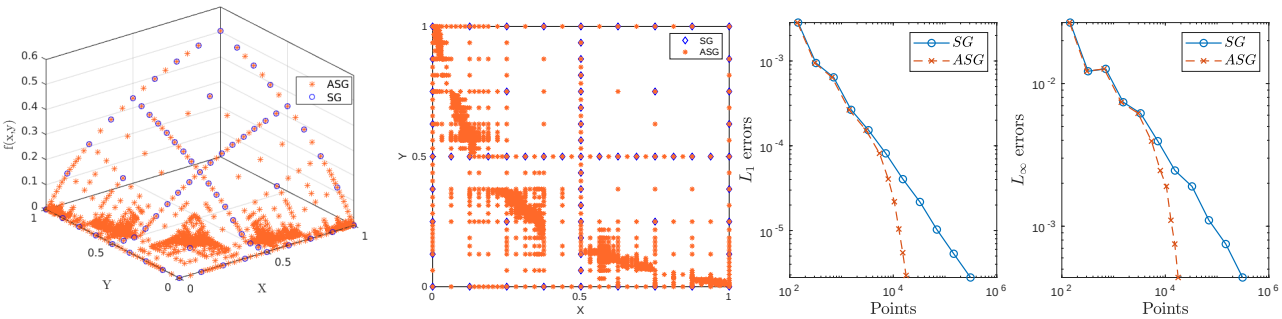



**Fig. 4** The left side of this figure displays hierarchical increment spaces $W_{(l_1,l_2)}$ for $1 \leq l_1, l_2 \leq n = 3$ with their corresponding grid points and one-dimensional piecewise linear basis functions of levels 1, 2, and 3. The regular SG $\mathcal{V}_3^S$ (see equation (14)) in the top right corner consists of all the points displayed in the left figure. An ASG $\mathcal{V}_3^{ASG}$, as shown in the bottom right corner, does not. The red dots in the left panel symbolically represent points that are refined, that is, $g(\alpha_{\mathbf{l},\mathbf{i}}) \geq \epsilon$ holds, whereas the green ones indicate points where the grid is not refined. The black points in the left panel are only contained in $\mathcal{V}_3^S$, and not $\mathcal{V}_3^{ASG}$.

LSGs presented in section 2.1) would almost vanish, this would be a waste of grid points and computational resources. It would make more sense to place these points near the kink or in high-curvature regions. Spatial adaptivity within LSGs makes this possible. Instead of adding entire incremental grids for an increasing grid level, spatially-adaptive grids only choose to add a subset of the points of the incremental grids to avoid wasting points. Recall that the regular LSG introduced in equation (14) is based on an a priori selection of grid points that are optimal for functions with bounded second-order mixed derivatives. An adaptive (a posteriori) refinement does, in addition, use local error estimators to select which grid points in the SG structure should be refined. The most common way to do this is to add $2d$ children in the hierarchical structure with increasing grid refinement level if the hierarchical surpluses satisfy $g(\alpha_{\mathbf{l},\mathbf{i}}) \geq \epsilon$ for a so-called refinement threshold $\epsilon \geq 0$. The basic intuition behind this procedure is the following: when approximating a function as a sum of piecewise linear basis functions, the main contributions to the interpolant most likely stem from comparatively few terms with big surpluses (see the left panel of figure 2 and equation (13)). Therefore, the logic of the refinement strategy presented here is to monitor the size of the hierarchical surpluses and refine the grid where those terms are larger than some threshold. Technically, the ASG refinement can be built on top of the hierarchical grid structure, as illustrated in the right panel of figure 4. The lower right panel of figure 4 illustrates a qualitative example of how a 2-dimensional SG is refined adaptively, adding a second layer of sparsity to the SG. For more details regarding spatially-adaptive SGs, we refer the reader, for example, to Bungartz and Dirnstorfer [2003], Ma and Zabaras [2009], Pflüger [2010].

**Numerical example.** Given this basic intuition on the workings of spatial adaptivity, we next apply it to the example of approximating a 2-dimensional (non-smooth) analytical function $f : [0,1]^2 \to \mathbb{R}$, as specified in equation (22) with LSGs, and using TASMANIAN.[14] In particular, we generate SG as well as adaptive SG approximations to the function $f$ and measure their accuracy as follows: We randomly generate $N = 10{,}000$ test points from a uniform distribution on $[0,1]^2$, and compute the maximum error and the $L_1$ error, which are,

[14] To test the performance of function approximators in general, a representative suite of test problems was proposed by Genz [1984] (see also `https://www.sfu.ca/~ssurjano/integration.html`).




**(a)** Level 5 SG vs. level 16 ASG **(b)** Level 5 SG vs. level 16 ASG **(c)** Comparison of the interpolation error

**Fig. 5** Panels (a) and (b): comparison of a level 5 SG with a level 16 ASG and a refinement threshold $\epsilon = 10^{-3}$; Panel (c): Comparison of the interpolation error for an ordinary SG and ASG interpolation at different refinement levels, ranging from from level 6 to 16; The ASGs were obtained by applying a threshold $\epsilon = 10^{-6}$.

respectively, given by

$$L_\infty = \max_{i=1,\ldots,N} |f(\mathbf{x}_i) - u(\mathbf{x}_i)|, \quad L_1 = \frac{1}{N} \sum_{i=1,\ldots,N} |f(\mathbf{x}_i) - u(\mathbf{x}_i)|. \tag{23}$$

The refinement criterion we apply here for the ASG is given by

$$g(\alpha_{\mathbf{l},\mathbf{i}}) = \frac{|\alpha_{\mathbf{l},\mathbf{i}}|}{f_{max}} > \epsilon, \tag{24}$$

where $f_{max} = \max |f(\mathbf{x})|$ is the maximum of the function values used so far in the ASG construction up to the current grid level. In the case of vector-valued functions, grid points are added isotropically, i.e., if any of the outputs has normalized hierarchical surpluses larger than $\epsilon$.[15]

Panels (a) and (b) of figure 5 display both an SG as well as an ASG interpolation of our test function $f$, where the non-adaptive SG is of level 5, and the ASG is refined up to level 16, given a refinement threshold of $\epsilon = 0.001$. This figure illustrates nicely that the ASG algorithm automatically detects the non-differentiability in $f$. Panel (c) of figure 5 displays the convergence behavior of the two methods for approximating $f$. The data points reported for the regular SG were obtained by computing the $L_\infty$ and $L_1$ errors at levels 6 through 16. These results are in contrast with the data points obtained from ASGs starting from level 3 and gradually increasing the maximum refinement level from 6 to 16 with $\epsilon = 10^{-6}$. Strikingly, to reach an $L_1$ error of around $10^{-5}$ the ASG needs 12,836 points as opposed to 69,633 points when using the conventional SG. This example illustrates that ASGs are not only able to capture strong non-linearities but also non-differentiabilities that SGs (which are constructed to approximate functions with bounded second-order mixed derivatives) are, in general, not able to capture. In sections 3 and 4 below, we show that this is still true when applied to economic applications and also to higher dimensions. For more details, see `https://github.com/SparseGridsForDynamicEcon/SparseGrids_in_econ_handbook`.

## 3 Solving discrete time-dynamic stochastic models with sparse grids

SG approximation techniques need to be integrated into the recursive methods that are typically employed to solve discrete-time dynamic economic models. Krueger and Kubler [2004] were the first to do so, applying GSGs in a time iteration algorithm to solve stochastic OLG models of medium size, whereas Judd et al. [2014] employ (anisotropic) GSGs in a fixed-point iteration algorithm to solve single- and multi-country growth models. This section follows Brumm and Scheidegger [2017] in showing how to embed adaptive LSGs in a time iteration

[15]For more details as well as alternative refinement criteria used in the literature, see Stoyanov [2015].

algorithm and to solve a IRBC model with constraints that occasionally bind, namely irreversible investment (sections 3.1 and 3.2). Finally, it discusses the use of SGs in the context of value function iteration algorithms (section 3.3).

### 3.1 Time iteration with sparse grids

The so-called time iteration algorithm, introduced by Coleman [1990], solves for a recursive equilibrium of a dynamic economic model by guessing a policy function and iteratively updating it using the first-order equilibrium conditions of the model. Let $\mathbf{x} \in X \subset \mathbb{R}^d$ denote the state of the economy at a given point in time and let $p : X \to Y$, where $Y \subset \mathbb{R}^m$ denotes the policy function. The stochastic transition of the economy to the next period can then be represented by the distribution of next period's state $\mathbf{x}'$, which depends on the current state and policy:

$$\mathbf{x}' \sim F\left(\cdot|\mathbf{x}, p(\mathbf{x})\right). \tag{25}$$

While the distribution $F$ given $\mathbf{x}$ and $p(\mathbf{x})$ follows from the assumptions of the model, the policy function $p$ needs to be determined from equilibrium conditions. When using time iteration, these conditions include the agents' first-order optimality conditions, budget constraints, and market clearing conditions. Taken together, these conditions constitute a functional equation that the policy function $p$ has to satisfy, namely, that for all $\mathbf{x} \in X$,

$$0 = \mathbb{E}\left\{E\Big(\mathbf{x}, \mathbf{x}', p\left(\mathbf{x}\right), p\left(\mathbf{x}'\right)\Big)|\mathbf{x}, p(\mathbf{x})\right\}, \tag{26}$$

where the expectation, represented by the operator $\mathbb{E}$, is taken with respect to the distribution $F\left(\cdot|\mathbf{x}, p(\mathbf{x})\right)$ of next period's state $\mathbf{x}'$. The function $E : X^2 \times Y^2 \to \mathbb{R}^m$ represents the period-to-period equilibrium conditions of the model. Below, we present an IRBC model as an example for this abstract characterization of a recursive equilibrium. For now, we stick with the abstract characterization to describe most clearly how to solve for time-invariant equilibrium policy functions using an ASG-time-iteration algorithm. The basic idea of time iteration is as follows: Solve the equilibrium conditions of the model for today's policy $p : X \to Y$ taking as given an initial guess for the function that represents next period's policy, $p_{next}$; then, use $p$ to update the guess for $p_{next}$ and iterate the procedure until convergence. Thus, time iteration creates many successive approximations of $p$. We employ ASGs in each iteration step to do so. The structure of the resulting ASG-time-iteration algorithm is displayed in algorithm 1. Before we turn to a concrete example, we first comment on several crucial aspects of this algorithm in more detail:

- *Maximal refinement level*, $L_{max}$: Note first that the classical SG of level $L$ is obtained as a special case by setting $L_{max} = L_0 = L$, so that all the grid points within a given level are added to the LSG. Note also that one could, in principle, set the maximum refinement level $L_{max}$ to a very large value that is never reached for a given refinement threshold. However, this might imply, in the case of a function with high curvature or non-differentiabilities, that the algorithm may not stop to refine until a very high interpolation level.

- *Conditional expectation*, $\mathbb{E}\left\{\cdot|\cdot\right\}$: The evaluation of the conditional expectation in the main step of the algorithm does, of course, depend on the specification of the exogenous shock process. If the exogenous states are assumed to be continuous, as was (implicitly) the case above, the standard approach is to choose a quadrature rule that transforms the integral into a sum over an appropriate set of quadrature points and respective weights. If the exogenous process is discrete with realizations $z \in Z$, then one can write the state as $\tilde{x} \in \tilde{X} = Z \times X$ and let the algorithm create a separate ASG, $G_z$, over $X$ for each $z \in Z$, as in Brumm et al. [2017].

- *Missing solutions*: In the main step of the algorithm, a policy vector is assigned to a given grid point by solving the equilibrium conditions. If the numerical solver does not find a solution, a fallback value needs to be provided to the SG algorithm in order to proceed. Potential fallback values are the function value at the same grid point in the previous iteration or an interpolated value based on neighboring points from the current iteration.

**Input:** Initial guess $p_{next}$ for next period's policy function. Approximation accuracy $\bar{\eta}$. Maximal refinement level $L_{max}$. Starting refinement level $L_0 \leq L_{max}$. Refinement threshold $\epsilon$.
**Output:** The (approximate) equilibrium policy function $p$.
**while** $\eta > \bar{\eta}$ **do**
  Set $l = 1$, set $G \subset X$ to be the level 1 grid on $X$, and set $G_{old} = \emptyset, G_{new} = \emptyset$.
  **while** $G \neq G_{old}$ **do**
    **for** $g \in G \setminus G_{old}$ **do**
      Given $p_{next}$, compute $p(g)$ by solving

$$0 = \mathbb{E}\left\{E\left(g, \mathbf{x}', p\left(g\right), p_{next}\left(\mathbf{x}'\right)\right)|g, p(g)\right\}, \quad \mathbf{x}' \sim F\left(\cdot|g, p(g)\right).$$

      Define the policy $\tilde{p}(g)$ by interpolating $\{p(g)\}_{g \in G_{old}}$.
      **if** ($l < L_{max}$ and $\|p(g) - \tilde{p}(g)\|_\infty > \epsilon$) or $l < L_0$, **then**
        Add the next-level neighboring points of $g$ to $G_{new}$.
      **end**
    **end**
    Set $G_{old} = G$, set $G = G_{old} \cup G_{new}$, set $G_{new} = \emptyset$, and set $l = l + 1$.
  **end**
  Define $p$ as the SG interpolation of $\{p(g)\}_{g \in G}$.
  Calculate (approximate) error: $\eta = \|p - p_{next}\|_\infty$. Set $p_{next} = p$.
**end**

**Algorithm 1:** Overview of the ASG-time-iteration algorithm.

### 3.2 Example: IRBC with occasionally binding constraints

To demonstrate the capabilities of ASGs, we consider an IRBC where investment in each country's capital stock is irreversible as in Brumm and Scheidegger [2017]. As these occasionally binding constraints induce non-differentiabilities in policy functions, they represent a challenge that adaptivity can address. To focus on the dimensionality of the state space and the occasionally binding constraints, we keep the model simple in many other ways.

**IRBC model.** In the model, there are $M$ countries, $j = 1, \ldots, M$, each using its capital stock, $k_j$, to produce output, which can be used for investment, $\chi_j$, or for consumption, $c_j$, generating utility through an additive separable utility function with discount factor $\beta$ and per-period utility function

$$u_j(c_j) = c_j^{1-\gamma_j}/(1-\gamma_j), \quad \gamma_j > 0. \tag{27}$$

Investment is subject to adjustment costs equal to $k_j \cdot \phi/2 \cdot g_j^2$, where $g_j \equiv k_j'/k_j - 1$ and $\phi > 0$. Furthermore, investment is irreversible, $\chi_j \geq 0$, and depreciates at a rate $\delta > 0$, thus following the law of motion:

$$k_j' = k_j \cdot (1-\delta) + \chi_j. \tag{28}$$

The amount produced by each country is given by $a_j \cdot A \cdot (k_j)^\kappa$, where $A > 0$, and $a_j$ is the country specific productivity level. The latter follows this law of motion:

$$\ln a_j' = \rho \cdot \ln a_j + \sigma \cdot \left(e_j + e_{M+1}\right). \tag{29}$$

The parameters $\rho$ and $\sigma$ determine the persistence and volatility in productivity. The country-specific shocks, $e^j \sim \mathcal{N}(0,1)$, as well as the global shock, $e_{M+1} \sim \mathcal{N}(0,1)$, are assumed to be independent from each other and across time. We follow the comparison study by Kollmann et al. [2011] in assuming complete markets,[16]

[16]However, as we do not directly solve the social planner problem, but iterate on the period-to-period equilibrium conditions given in (32)–(34). An incomplete markets version of this model could be solved in a similar way.

which implies that the decentralized competitive equilibrium allocation can be obtained as the solution to a social planner's problem: Subject to aggregate resource constraints and irreversibility constraints, maximize the weighted sum of country utility, weighted by welfare weights, $\tau^j$, which depend on the initial capital stocks of the countries.

**Recursive structure.** We now briefly present the recursive structure of the model, while we refer the reader to Brumm and Scheidegger [2017] for the derivation of the equilibrium conditions. The state variables of the IRBC model with $M$ countries consist of

$$\mathbf{x} = (a_1, \ldots, a_M, k_1, \ldots, k_M) \in X \subset \mathbb{R}^{2M}, \tag{30}$$

where $a_j$ and $k_j$ are the productivity and capital stock of country $j$, respectively. The policy function $p : \mathbb{R}^{2M} \to \mathbb{R}^{2M+1}$ maps the current state into investment choices, $\chi_j$, the multipliers for the irreversibility constraints, $\mu_j$, and the multiplier of the aggregate resource constraint, $\lambda$:

$$p(\mathbf{x}) = (\chi_1, \ldots, \chi_M, \mu_1, \ldots, \mu_M, \lambda)\,. \tag{31}$$

The investment choices determine next period's capital stock in a deterministic way through (28). In contrast, the law of motion of productivity, (29), is stochastic. Taken together, (28) and (29) specify the distribution of $x'$ (corresponding to (25) in the general problem above). The period-to-period equilibrium conditions of this model (corresponding to (26)) consist of three types of equations. First, the optimality conditions for investment in capital in each country $j$:

$$\lambda \cdot \left[1 + \phi \cdot g_j\right] - \mu_j - \beta \cdot \mathbb{E}_t \left\{ \lambda' \left[ a'_j \cdot A \cdot \kappa \cdot (k'_j)^{\kappa-1} + 1 - \delta + \frac{\phi}{2} \cdot g'_j \cdot \left(g'_j + 2\right) \right] - \mu'_j \cdot (1-\delta) \right\} = 0. \tag{32}$$

Second, the irreversibility constraint and the associated complementary condition for each country $j$,

$$\chi_j \geq 0, \mu_j \geq 0, \chi_j \cdot \mu_j = 0. \tag{33}$$

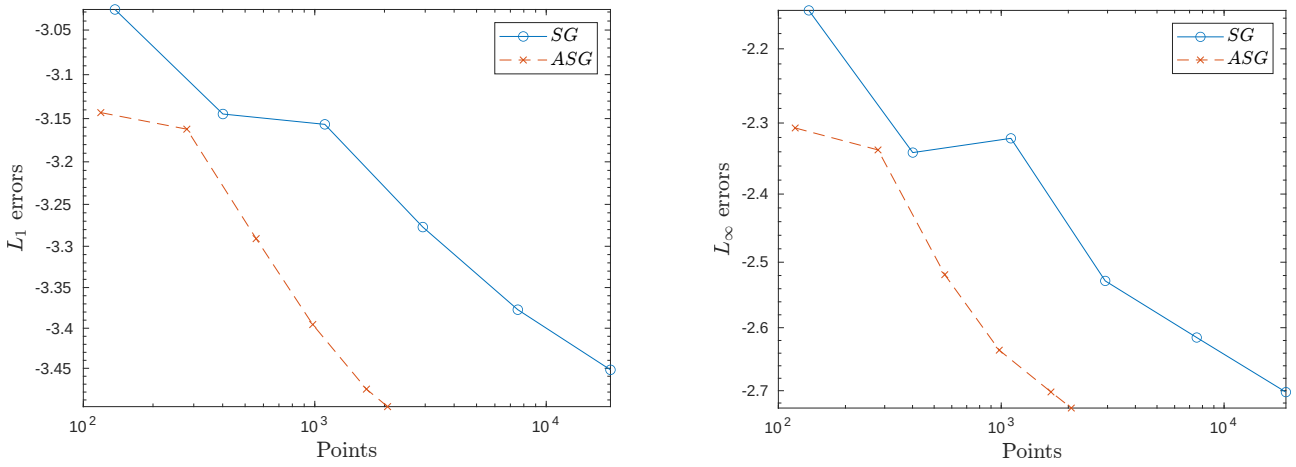
Finally, using $c_j = (\lambda/\tau_j)^{-\gamma_j}$, the aggregate resource constraint is

$$\sum_{j=1}^{M} \left( a_j \cdot A \cdot (k_j)^{\kappa} - k_j \cdot \frac{\phi}{2} \cdot (g_j)^2 - \chi_j - c_j \right) = 0. \tag{34}$$

**Time iteration solution.** Given the above's characterization of a recursive equilibrium, two important choices need to be made before algorithm 1 can be directly applied. First, the conditions in (33) include inequality constraints. To handle these, we either need to transform them into equations (as in Brumm and Grill [2014]), to include them as inequality constraints and use an appropriate solver, or to use a solver that directly handles complementarity conditions. Second, the expectation term in equation (32) has to be evaluated by integrating over the (normally distributed) productivity shocks. A careful selection of the quadrature rule is of key importance to the overall performance of algorithm 1, as the solution of the individual nonlinear sets of equations within the time iteration algorithm can take up more than 90% of the entire runtime [Scheidegger et al., 2018]. Thus, if a solution of moderate quality is sufficient, a simple and fast quadrature rule like a monomial rule that grows linearly in the number of shocks can be employed. In contrast, if the expectation needs to be computed to a high degree of precision, for example, a Gauss-Hermite quadrature rule can be employed. The latter, however, grows exponentially in dimension and thus will dominate the overall runtime in a high-dimensional model, despite the fact that SGs in a standalone mode would alleviate the curse of dimensionality.[17] For a thorough discussion of quadrature procedures, see, for example, Judd [1998].

Figure 6 reports numerical results for running LSG-time-iteration algorithms to solve the 2-country (i.e., four-dimensional) version of the above IRBC model. The employed parameters of the model and the Euler error measure are chosen as in Brumm and Scheidegger [2017]. Similar to the analytic example in figure 5, we

[17] Judd et al. [2017] recently devised a method to pre-compute integrals in a broad range of dynamic models so that this step in the time iteration algorithm can be accelerated substantially.




**(a)** $L_1$ errors **(b)** $L_\infty$ errors

**Fig. 6** Panels (a) and (b): Comparison of the average and maximum Euler errors in a 4-dimensional—that is, a 2-country IRBC model with irreversible investment, and that was solved via the time iteration algorithm. In the case of the ASGs, we started with a fixed SG of level 3 and then successively increased the maximum refinement level from 4 to 9, with $\epsilon = 10^{-3}$. The results are compared with corresponding ordinary SGs of level 4 up to level 9. Notice that ASGs need substantially fewer points compared to SGs for a given level of accuracy.

find that ASGs substantially outperform SGs. In addition to this fundamental result, Brumm and Scheidegger [2017] provide the following three additional findings: First, the comparative advantage of ASGs over SGs becomes even more critical as the dimensionality of the problem increases. Second, even for smooth models, in particular, when they are of very high dimension, ASGs are much more flexible than SGs when it comes to choosing the right balance between running times and accuracy. Third, while the compute time for SGs and ASGs increases substantially with the increasing dimensionality of the model, this burden can to some extent be alleviated by leveraging the high level of parallelism inherent to SG techniques in combination with contemporary high-performance computing systems (see section 5 for more details).

### 3.3 Value function iteration with sparse grids

Many dynamic problems in economics and finance can be formulated recursively by the Bellman equation [Bellman, 1961], and solved by *value function iteration*. Restricting the exposition to the infinite-horizon case without time-dependence of fundamentals, the Bellman equation reads as

$$V(\mathbf{x}) = \Gamma(V'(\mathbf{x})) \;=\; \max_{p \in \mathcal{D}(\mathbf{x})} \left[ u(\mathbf{x}, p) + \beta \cdot \mathbb{E}\{V'(\mathbf{x}'|\mathbf{x}, p)\} \right], \text{ with } \mathbf{x}' \sim F(\mathbf{x}, p). \tag{35}$$

$\Gamma(\cdot)$ is the Bellman operator that maps next period's value function $V'$ into this period's value function $V$. $\mathbf{x} \in \mathbb{R}^d$ is the vector of (continuous) state variables, and $p$ is the policy, which is constrained by $p \in \mathcal{D}(\mathbf{x})$. Furthermore, next period's state $\mathbf{x}'$ is distributed according to $F(\mathbf{x}, p)$, $u(\cdot)$ is the per-period payoff function, and $\beta \in (0, 1)$ and $\mathbb{E}(\cdot)$ are the discount factor and the conditional expectation, respectively.

Under assumptions that make the Bellman operator a contraction, that is, $\|\Gamma(V) - \Gamma(W)\|_\infty \leq \beta \cdot \|V - W\|_\infty$, value function iteration—iterating on the value function by applying the Bellman operator—converges to the unique value function representing a recursive equilibrium, as two consecutive value functions will eventually satisfy $\|V' - V\|_\infty \leq \epsilon$ for any $\epsilon > 0$ (see, e.g., Stokey et al. [1989], Rust [1996]). There are, however, two caveats when it comes to using SGs to solve high-dimensional problems with value function iteration. First, while value function iteration is, in many contexts, a robust and reliable convergence procedure, it is often much slower than alternative methods, which is of particular relevance when it comes to high-dimensional problems. Second, in theory, the convergence of value function iteration pre-supposes shape-preserving interpolation—in particular, that today's value function is concave if the next period's value function is also concave. Unfortunately, SG

methods are generally not concavity-preserving, which may result in convergence problems in practice.

The numerical implementation of value function iteration with ASGs is quite similar to the time iteration algorithm.[18] Looking at algorithm 1, only three aspects need to be adjusted. First, value functions instead of policy functions are approximated. In both cases, however, the fact that high-quality approximations need to be repeatedly constructed (i.e., for $V$ on the LHS of (35)) and evaluated (i.e., $V'$ on the RHS of (35)) necessitates an efficient approximation method like SGs. Second, the value function that has to be repeatedly approximated over the high-dimensional space is only scalar-valued in contrast to the vector-valued policy in time iteration. Note, however, that after convergence, even for value function iteration, the implied policy needs to be stored and then approximated on the SG such that the model can be simulated. Third, instead of a system of equations, a constrained optimization problem needs to be solved at each point in the SG. In doing so, using non-differentiable LSGs to approximate the value function may be problematic, as these methods do not lend themselves for gradient-based optimization and automatic differentiation. This problem can be overcome by the hierarchical B-spline approach outlined in Schober et al. [2021], which combines piecewise polynomial interpolation with the flexibility of multi-linear basis functions and thus generates smooth approximations of the value function. However, a drawback of this approach is that the determination of the coefficients of the B-Spline interpolant is computationally relatively expensive.

# 4 Solving continuous time dynamic stochastic models with sparse grids

This section demonstrates how LSGs can be used to solve multivariate continuous-time dynamic models, a setting that is at the heart of many economic applications. Dynamic stochastic models are usually recursively formulated in continuous time as HJB partial differential equations (PDEs). We begin our discussion in section 4.1 by very briefly outlining a generalized finite-difference scheme to solve PDEs on LSGs. In section 4.2, we then demonstrate this method by solving a variant of the standard incomplete markets Aiyagari [1994] model with a non-convex capital income tax.

## 4.1 A consistent finite-difference scheme for HJB equations on sparse grids

We consider a continuous-time dynamic stochastic optimization problem of an agent whose state is given by $\mathbf{x} = (x_1, \ldots, x_d) \in X \subset \mathbb{R}^d$. The agent's state is assumed to follow a controlled Itô process, that is,

$$d\mathbf{x} = \mu(p, \mathbf{x})d\tau + \sigma(p, \mathbf{x})dB, \tag{36}$$

where $\mu : Y \times \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : Y \times \mathbb{R}^d \to \mathbb{R}^{d\times k}$, and $B$ is $k$-dimensional Brownian motion. We use $\tau$ and $d\tau$ to denote time, since $t$ indexes dimensions of the state space in our notation. Mirroring our notation from section 3, we denote by $p \in Y \subset \mathbb{R}^m$ the vector of choice variables under the agent's control which may lie in the admissible set $Y$. The agent's problem is to choose $p$ to maximize the expected discounted sum of future payoff flows, which we denote by $u(p)$. The stationary value function $V : X \to \mathbb{R}$ associated with the agent's control problem is a solution to the HJB equation:[19]

$$\rho V(\mathbf{x}) = \max_{p\in Y} \left\{ u(p) + \sum_{t=1}^{d} \mu_t(p, \mathbf{x}) V_{x_t}(\mathbf{x}) + \frac{1}{2} \sum_{t,s=1}^{d} \sigma_{ts}(p, \mathbf{x}) V_{x_t x_s}(\mathbf{x}) \right\}, \tag{37}$$

where $\rho$ is the agent's discount rate, $\mu_t$ is the $t$-th element of the vector $\mu$, and, abusing notation slightly, $\sigma_{ts}$ is the $(t, s)$-entry of the matrix $\sigma\sigma^T$.[20] As above, we use $1 \le t, s \le d$ to index the dimensions. Furthermore, we apply the shorthand notation $V_{x_t}$ to denote the partial derivative $\partial/\partial x_t V(x)$, and analogously for $V_{x_t x_s}$.

After the formal introduction of LSGs by Zenger [1991], they were quickly recognized as a powerful tool for solving PDEs. Subsequent work initially applied LSGs in the context of (Galerkin) finite-element

[18] For more details on value function iteration with ASGs, see Scheidegger and Treccani [2018], and references therein.

[19] For the formal derivation of equation (37), see, for example, Oksendal [1992].

[20] A formal definition of the HJB equation (37) also requires the determination of the boundary conditions along the boundary of $X$. For a formal treatment of boundary conditions in the context of SGs, see Schaab and Zhang [2021].

methods. Schiekofer [1998] then extended the application of LSGs to the class of finite-difference methods, formally showed that the use of regular finite-difference stencils on LSGs leads to inconsistent discretization schemes, and proved the consistency of a generalized sparse finite-difference scheme for a particular class of LSGs.[21] In more recent work, Garcke and Ruttscheidt [2019] propose an alternative approach to construct consistent finite-difference schemes on LSGs using ghost node interpolation. In what follows, we provide a very brief overview of the finite-difference method developed in Schaab and Zhang [2021] to solve PDEs such as equation (37) on an LSG $G$ over the compact region $X$.[22]

Consider a function $u(x)$ of one variable. Its derivative $u'(x)$ can be numerically approximated by $h^{-1}[u(x+h)-u(x)]$ for small $h$. On a full grid, we can represent the numerical approximation to the derivative at a point $x_i$ as $D_x^+ \circ u(x_i) = h^{-1}[u(x_i+h)-u(x_i)]$, where $h = x_{i+1} - x_i$ is the grid's mesh size.[23] This *finite-difference* approximation can be written in terms of the shorthand stencil notation $h^{-1}[0 \quad -1 \quad 1]$, where the entries in brackets refer to the grid points $[x_{i-1}, x_i, x_{i+1}]$. On full grids, both the *forward* stencil $D_x^+$ and the *backward* stencil $D_x^- = h^{-1}[-1 \quad 1 \quad 0]$ represent *consistent* discretizations of the first-order derivative $u'(x)$. Similarly, the second-order finite-difference matrix $D_{xx}$ defined via the stencil $h_t^{-2}[1 \quad -2 \quad 1]$ represents a consistent discretization of $u''(x)$ on a full grid.

Defining a consistent discretization scheme for partial derivatives on LSGs is not trivial. In particular, the standard finite-difference approximations introduced above for the first- and second-order derivatives are generally no longer consistent on LSGs (see, e.g., Schiekofer [1998] and Schaab and Zhang [2021]). That is, we can no longer use the regular finite-difference matrices to solve equation (37) on LSGs numerically.

To develop a consistent finite-difference scheme, we build on our discussion in Section 2 and introduce a set of operators that map basis function coefficients $\xi_{\mathbf{i}}$ , equation (2), to the hierarchical surpluses $\alpha_{\mathbf{l},\mathbf{i}}$, equation (7).[24] Following Schiekofer [1998], we define the one-dimensional *hierarchization operator* as

$$H_{x_{l_t,i_t},h_t} : \mathcal{V}_{l_t} \rightarrow \bigoplus_{j_t \leq l_t} W_{j_t} \tag{38}$$

via the stencil $[-1/2 \quad 1 \quad -1/2]$, applied at a point $x_{l_t,i_t}$ and with mesh size $h_t$. That is, for $u : \mathbb{R} \rightarrow \mathbb{R}$, we have

$$H_{x_{l_t,i_t},h_t} \circ u(x_{l_t,i_t}) = u(x_{l_t,i_t}) - \frac{u(x_{l_t,i_t}-h_t) + u(x_{l_t,i_t}+h_t)}{2}, \tag{39}$$

where $1 \leq i_t \leq 2^{l_t} - 1$. For the multivariate case, $u : \mathbb{R}^d \rightarrow \mathbb{R}$, the following holds:

$$H_{x_{l_t,i_t},h_t} \circ u(\mathbf{x}_{\mathbf{l},\mathbf{i}}) = u(\mathbf{x}_{\mathbf{l},\mathbf{i}}) - \frac{u(x_{l_1,i_1},\ldots,x_{l_t,i_i}-h_t,\ldots,x_{l_d,i_d}) + u(x_{l_1,i_1},\ldots,x_{l_t,i_i}+h_t,\ldots,x_{l_d,i_d})}{2}. \tag{40}$$

Furthermore, we define the $d$-dimensional hierarchization operator as

$$\boldsymbol{H}_{\mathbf{x}_{\mathbf{l},\mathbf{i}},\mathbf{h}} = \prod_{t \leq d} H_{x_{l_t,i_t},h_t}. \tag{41}$$

We can indeed confirm that the hierarchization operator maps the standard basis function coefficients into hierarchical surpluses, in the sense that $\alpha_{\mathbf{l},\mathbf{i}} = \boldsymbol{H}_{\mathbf{x}_{\mathbf{l},\mathbf{i}},\mathbf{h}} \cdot \xi_{\mathbf{i}}$. For proofs and a more detailed discussion, see, for example, Bungartz and Griebel [2004], and references therein.

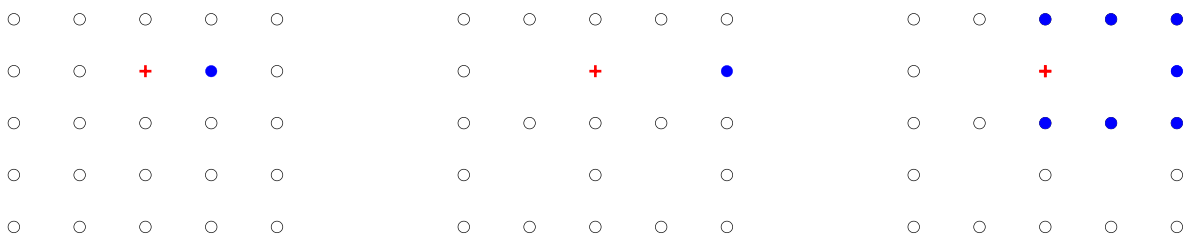
The hierarchization operator introduced above applies at a given grid point, $\mathbf{x}_{\mathbf{l},\mathbf{i}}$. It is convenient to define a composite operator that performs one-dimensional hierarchization at every grid point. Let $\boldsymbol{H}^t$ denote the operator that performs hierarchization in dimension $t$, $H_{x_{l_t,i_t},h_t} \circ u(\mathbf{x}_{\mathbf{l},\mathbf{i}})$, at all grid points of a given

[21] In the context of PDEs, a discretization scheme is called consistent when its local truncation error tends towards 0 as the grid's resolution tends towards 0. Intuitively, consistency implies that the approximation error in the partial derivative operators converges to 0 as the grid becomes finer [LeVeque, 2007].

[22] The formal results on consistency we summarize in this subsection apply for regular LSGs. We confirm numerically in the following subsection that consistency and convergence are also satisfied on ASGs in the context of our application. Schaab and Zhang [2021] perform numerical convergence analysis on ASGs across a broad range of applications.

[23] For a textbook treatment, see e.g. LeVeque [2007].

[24] We use $\xi_{\mathbf{i}}$ here to refer to the standard multi-linear basis function coefficients that correspond to evaluating a given function on a set of grid points. The mappings we discuss in the following have been previously discussed, e.g., by Bungartz and Griebel [2004].



**(a)** Dense grid: consistent **(b)** Sparse grid: inconsistent **(c)** Sparse grid: consistent

**Fig. 7** Consistent and inconsistent finite-difference stencils. Figures (a) through (c) display the grid points that a regular and sparse finite-difference forward stencil load on in case of a full grid, (a), and a regular LSG, (b) and (c). The stencil is applied at the grid point colored red. The regular stencil $[0 \quad -1 \quad 1]$ searches for the direct right neighbor. In the case of a dense grid in Panel (a), the red node has an immediate right-neighbor and the regular stencil consequently leads to a consistent discretization. In the case of a LSG in Panel (b), the red node does not have an immediate right-neighbor. The regular stencil therefore loads on a point that is far away. This results in an inconsistent scheme. Panel (c) displays a consistent sparse forward stencil which loads on all the nodes surrounding the grid point colored red and its right-neighbor. These points contain valuable information about variation in the underlying function.

grid. Similarly, let $\boldsymbol{H}_t$ denote the operator that performs hierarchization in all dimensions *but* $t$, and $\boldsymbol{H}$ the operator that performs hierarchization in all dimensions. We also introduce the inverse operators that map from hierarchical to nodal coefficients and are given by $\boldsymbol{E}^t = (\boldsymbol{H}^t)^{-1}$ and $\boldsymbol{E}_t = \boldsymbol{H}_t^{-1}$. We refer to these as *de-hierarchization* operators.

With these basis transformation operators in hand, we can now define a set of generalized sparse finite-difference operators. In particular, in a given dimension $t$, let $D_t^{\pm,\mathrm{S}} = \boldsymbol{E}_t \circ D_t^{\pm} \circ \boldsymbol{H}_t$ and $D_{tt}^{\mathrm{S}} = \boldsymbol{E}_{tt} \circ D_{tt} \circ \boldsymbol{H}_{tt}$, where $D_t^{\pm}$ and $D_{tt}$ are the standard finite-difference matrices. We illustrate these finite-difference matrices in figure 7. Formally, then, applying $D_t^{\pm,\mathrm{S}}$ and $D_{tt}^{\mathrm{S}}$ to the function $V(x)$ (cf. equation (37)) yields a consistent discretization of the first- and second-order partial derivatives $V_{x_t}$ and $V_{x_t x_s}$ on a regular LSG $G$, as long as $V$ has bounded mixed derivatives. A proof along these lines was originally developed in Schiekofer [1998].[25] See Schaab and Zhang [2021] for further details.

The algorithm to solve dynamic programming problems in continuous time using ASGs is summarized in algorithm 2 below. See Schaab and Zhang [2021] for additional details.

**Stability, monotonicity, and convergence.** The consistency of a discretization scheme does not yet imply convergence to the true solution. In the classical analysis of smooth partial differential equations, convergence typically requires at least stability in addition to consistency.[26] In the context of viscosity solutions, on the other hand, the seminal contribution of Barles and Souganidis [1991] establishes that convergence typically requires both stability and an appropriate notion of monotonicity. General results on the convergence of sparse finite-difference schemes in the context of viscosity solutions have remained elusive in part because SG interpolation does not generically satisfy monotonicity.[27] The numerical convergence analysis we perform in

[25] The consistency proofs of Schiekofer [1998] not only assume smoothness conditions on $V$, but also require that $G$ is a *regular* LSG, as defined, for example, in Bungartz and Griebel [2004].

[26] Intuitively, stability implies that truncation or local approximation errors do not propagate too quickly. A formal treatment of these topics in the context of LSGs is beyond the scope of this paper. For additional details, see, for example, the Lax-Richtmyer (Equivalence) Theorem and LeVeque [2007] for a textbook treatment.

[27] Hemker [2000] points out that interpolation on LSGs may not satisfy monotonicity. Garcke and Ruttscheidt [2019] show that even in the case of concave, monotonically increasing functions, interpolation on LSGs is not generically guaranteed to be monotone. In particular, they show that when the hierarchical representation of a concave and monotonically increasing function leads to negative hierarchical coefficients, the resulting SG interpolation may be non-monotone. Garcke and Ruttscheidt [2019] propose adaptive grid refinement in areas of the state space where non-monotonicities arise. This can be achieved by either checking the hierarchical coefficients for positivity or by verifying that the numerically approximated derivatives satisfy the concavity and monotonicity conditions of the

**Input:** Initial guess $V^0$ for the value function. Approximation accuracy $\bar{\eta}$. Refinement thresholds for grid adaptation. Starting refinement level $\boldsymbol{l}^0$.
**Output:** The (approximate) equilibrium value function $V$ and the associated ASG $G$.

Given index $\boldsymbol{l}^0$, construct the associated grid $G^0$, hierarchization matrix $\boldsymbol{H}^0$, and the sparse finite-difference matrices $D_t^{\pm,\mathrm{S},0}$ and $D_{ts}^{\mathrm{S},0}$ $\forall t, s$, on grid $G^0$.

**for** $k \geq 0$ **do**
  **while** $\eta > \bar{\eta}$ **do**
    Compute policy functions $p^k$ using guess $V^k$ and first-order conditions
    Solve HJB for $V^{k,new}$
    Set $\eta = ||V^{k,new} - V^k||$ and $V^k = V^{k,new}$
  **end**
  Compute hierarchical surplus $\boldsymbol{\alpha}^k = \boldsymbol{H}^k \circ V^k$
  If adaptation criterion met, adapt grid $G^{k+1}$; else stop, and set $V = V^k$ and $G = G^k$
  Reconstruct $\boldsymbol{H}^{k+1}$ and sparse FD matrices, $D_t^{\pm,\mathrm{S},k+1}$ and $D_{ts}^{\mathrm{S},k+1}$ $\forall t, s$, on grid $G^{k+1}$
**end**

**Algorithm 2:** Overview of the ASG value function iteration algorithm.

section 4.2 below suggests, however, that concerns about non-monotonicity and non-convergence do not apply in the context of our application. Indeed, Schaab and Zhang [2021] perform numerical convergence analysis along these lines for a wide range of SG applications in continuous time. The convergence properties of SG discretization schemes remain an active area of research.

## 4.2 Example: Aiyagari model with non-convex capital income tax

In this section, we present a variant of the standard incomplete markets model along the lines of Aiyagari [1994] to showcase how ASGs can be used in the context of HJB equations. In this model, the economy is populated by a continuum of households with measure one. Household preferences over consumption are given by

$$\mathbb{E}_0 \int_0^\infty e^{-\rho\tau} u(c_\tau) d\tau, \tag{42}$$

where $c_\tau$ denotes the rate of consumption, and $\rho$ is the discount rate. Households face the budget constraint

$$\dot{k}_\tau = \Big(1 - T(k_\tau)\Big) r k_\tau + e^{z_\tau} - c_\tau, \tag{43}$$

where $k_\tau$ is the household's stock of owned capital and $z_\tau$ captures idiosyncratic earnings risk. Households take as given a constant real interest rate $r$. Capital cannot be sold short, $k_\tau \geq 0$.

The households pay a constant tax rate on their financial income once their wealth exceeds a certain level $k^*$. Concretely, we set

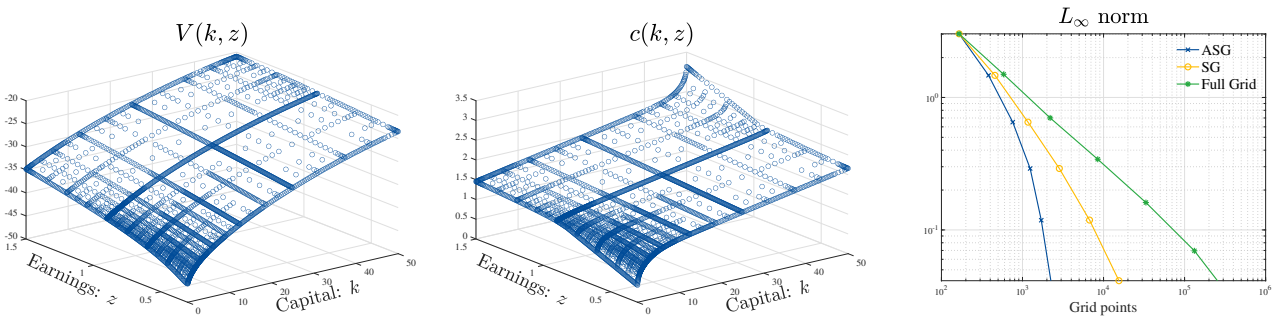
$$T(k) = \begin{cases} 0 & \text{if } k < k^* \\ \kappa & \text{if } k \geq k^* \end{cases}$$

Finally, we assume that $z_\tau$ follows a diffusion process, with

$$dz_\tau = -\theta z_\tau d\tau + \sigma dB, \tag{44}$$

where $dB$ is a standard Brownian motion.

underlying value function.




**(a)** Value Function **(b)** Consumption Policy **(c)** Relative Errors

**Fig. 8** This figure displays the numerical analysis for the Aiyagari model with diffusive earnings risk. Panels (a) and (b) display the value function $V(k,z)$ and consumption policy function $c(k,z)$ for an ASG with maximum refinement level $\boldsymbol{l}^{\max} = (11,8)$. Panel (c) plots the norm $L_\infty^j = \max|V^{\text{fine}} - V^j|$ for a sample of 30,000 grid points, where $V^{\text{fine}}$ is solved on a fine full grid of 2,1 million grid points and $\{V^j\}$ is a sequence of solutions on increasingly fine sets of full grids, LSGs and ASGs. For all three types of grids in Panel (c), we start with a full grid of level $\boldsymbol{l}^0 = (5,2)$ and consider 5 successive refinement levels. For ASG, we adapt the grid using thresholds $\epsilon^{\text{add}} = 10^{-5}$ and $\epsilon^{\text{keep}} = 10^{-6}$.

**Recursive formulation.** The households face the dynamic stochastic optimization problem of choosing $\{c_\tau\}_{\tau\geq 0}$ to maximize (42) subject to (43) and (44), as well as the short-sale constraint on capital. It will be convenient to work directly with the recursive formulation of this problem over a compact region $X = \{(k,z) \mid k \in [0,\bar{k}] \text{ and } z \in [\underline{z},\bar{z}]\} \subset \mathbb{R}^2$, which we refer to as the state space of the household problem.[28] In the notation of Section 4.1, the state variables of the household problem are therefore $\mathbf{x} = (k,z) \in X$ and the consumption policy function is $p(\mathbf{x}) = c(k,z)$.

Formally, the value function that maximizes the household problem solves a PDE of HJB form, which is given by

$$\rho V(k,z) = \max_{\hat{c}} \left\{ u(\hat{c}) + \Big((1-T(k))rk + e^z - \hat{c}\Big)V_k(k,z) - \theta z V_z(k,z) + \frac{\sigma^2}{2}V_{zz}(k,z) \right\}, \tag{45}$$

where we use the shorthand $V_k = \partial/\partial k\ V$ and so on to denote partial derivatives. The consumption policy function then solves the first-order condition $u'(c(k,z)) = V_k(k,z)$. Plugging in for the first-order condition, we can rewrite equation (45) to obtain

$$\rho V(k,z) = u\Big(c(k,z)\Big) + \Big((1-T(k))rk + e^z - c(k,z)\Big)V_k(k,z) - \theta z V_z(k,z) + \frac{\sigma^2}{2}V_{zz}(k,z). \tag{46}$$

Formally, equation (46) holds everywhere in the interior of the state space.[29] To pin down the value function over the entire compact space $X$ requires a set of boundary conditions. In particular, the short-sale constraint on capital gives rise to a state-constrained boundary condition that takes the form $u'(c(\underline{k},z)) \geq V_k(\underline{k},z)$.[30]

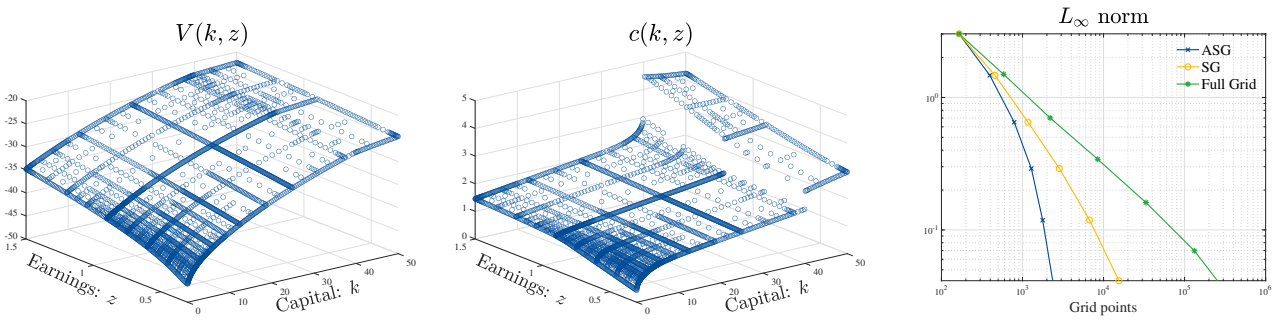
**Calibration.** Adopting CRRA utility, with $u(c) = (1-\gamma)^{-1}c^{1-\gamma}$, we set the discount rate of households to $\rho = 0.02$ and the coefficient of relative risk aversion to $\gamma = 2$. We calibrate the earnings process using $\theta = 0.10$ and $\sigma = 0.04$. The real interest rate $r$ is the only price that households face. We set it to 1.93% in the baseline model without tax and to 1.94% with the capital income tax.[31] Finally, we set $\kappa = 0.02$ and $k^* = 40$.

[28] We first construct the (sparse) grid and the associated (sparse) finite-difference matrices on the domain $[0,1]^2$ and then map these to economic units. See also our discussion in Section 2.

[29] The value function that solves equation (46) has a point of non-differentiability at $k = k^*$ due to the non-convexity of the capital income tax. We confirm this in our numerical analysis below. As a result, there exists no solution of equation (46) in the classical sense, and we consequently focus on the solution concept of a viscosity solution. See Achdou et al. [2021] for further details.

[30] See, e.g., Achdou et al. [2021] for further details. In the earnings dimension, we assume reflecting boundaries at $\underline{z}$ and $\bar{z}$.

[31] Our main numerical experiment is in partial equilibrium. That is, we hold constant the interest rate $r$ across the different grids we




**(a)** Value Function **(b)** Consumption Policy **(c)** Relative Errors

**Fig. 9** This figure shows the numerical analysis of Aiyagari model with diffusive earnings risk and non-convex capital income tax. The details of the numerical exercise are parallel to figure 8.

**Performance and accuracy.** Our main numerical exercise in this section is to compare the accuracy of solutions for $V(k,z)$ across different grids that discretize the state space $X$.[32] As discussed in Achdou et al. [2021], there is no readily available accuracy metric in continuous time—similar to the Euler condition error in discrete-time—which would let us assess the residual error implied by a particular numerical solution of the value function.[33] Instead, we solve the household problem on a fine uniform grid with 2.1 million grid points, which we denote $V^{\text{fine}}$. We consider a sequence of grids $\{G^j\}$ and solve for the household value function $V^j(k,z)$ on these grids using multi-linear hat functions as introduced in section 2.1. To compute error metrics, we sample 30,000 points from $X$ and project both $V^{\text{fine}}$ and $V^j$ on these points. Figures 8 and 9 then plot the sup norm, that is,

$$L_\infty^j = \max \left| V^{\text{fine}} - V^j \right|. \tag{47}$$

Our main results are displayed in figure 8 for the baseline Aiyagari model without taxes, and in figure 9 for the case with non-convex capital income tax. Both figures illustrate that ASGs substantially outperform both LSGs and especially uniform grids in terms of numerical accuracy. Panels (a) and (b) of both figures display the value and consumption policy functions on an ASG with maximum refinement level $\boldsymbol{l}^{\max} = (11, 8)$, with 2,840 grid points for the baseline model and 2,966 points for the variant with capital income tax.[34] Figure 9 shows that the non-convex capital income tax introduces a kink into the value function and a discontinuity into the consumption policy function. Grid adaptation places additional points close to the kink, especially for higher earnings levels.[35] Panels (c) in both figures plot the relative numerical error $L_\infty^j$ across differently structured grids with increasing numbers of grid points.

In this section, we illustrated the power of ASGs in the context of occasionally binding constraints and non-convexities. In both cases, grid adaptation is powerful because grid points can be placed in regions of the state space where the function $V$ is very concave or non-differentiable. LSG methods also excel in the context of free boundary problems, life-cycle models, high-dimensional dynamic programming problems, and many other

consider and only recompute the household value function $V(k,z)$. In particular, an interest rate of $r = 1.93\%$ (1.94%) corresponds to the rate that would clear the market for capital in an extension of the model without (with) capital income tax that also features a representative firm that uses capital and labor in production.

[32] The non-convex capital income tax introduces a *kink* in the household value function $V(k,z)$ at $k^*$. The appropriate formal solution concept for equation (46) is, therefore, that of a viscosity solution (see Achdou et al. [2021] for more details). Furthermore, the formal consistency proof developed originally in Schiekofer [1998] applies to functions of bounded mixed derivatives, a condition that is violated at $k^*$ whenever $\kappa > 0$. However, our numerical analysis confirms that our finite-difference method yields an accurate solution of $V(k,z)$ despite the non-differentiability.

[33] This is also the main reason why we focus on a relatively simple model in this section featuring only a two-dimensional state space.

[34] In the model without capital income tax, the numerical approximation of the consumption policy function becomes non-concave close to $(\bar{k}, \bar{z})$. This is an artifact of the boundary condition we use at $\bar{k}$, which forces households to dissave. We confirm numerically that, for the calibration we use, the stationary distribution has vanishingly small mass in this region of the state space.

[35] Unlike in Sections 2 and 3, we allow our grid refinement algorithm here to both add and remove grid points. For details, see e.g., Ma and Zabaras [2009], Brumm and Scheidegger [2017], or Schaab and Zhang [2021]. We use refinement thresholds $\epsilon^{\text{add}} = 10^{-5}$ for adding and $\epsilon^{\text{keep}} = 10^{-6}$ for removing grid points.

applications. Schaab and Zhang [2021] illustrate the power of ASGs across a wide range of continuous-time applications and develop an online repository with pedagogical sample codes.

# 5 Advanced topics & outlook

SGs are a very efficient and mathematically well-studied way to tackle the numerical difficulties that arise in dynamic stochastic models with high-dimensional state spaces. Therefore, they have become one of the most popular methods for solving and estimating dynamic models in discrete and continuous-time settings with up to dozens of continuous dimensions. Although the various formulations of SGs that we summarized above successfully allow to scale applications towards richer, more nonlinear settings while being only taxed moderately in terms of increased compute time with increasing dimensionality, they are no silver bullet. In the following, we outline three of the most common problems that may arise when applying SGs as solution techniques, and we discuss how to deal with them in practice.

First, on LSGs and GSGs, the value functions or policy functions need to be approximated on a hypercube. However, the ergodic set of dynamic economic models is often ex ante unknown and seldom hyper-cubic. Thus, it constitutes only a small fraction of the encompassing hypercube. Consequently, approximating the economy on a hypercubic domain can be wasteful and, in some cases, render the computation of solutions to models with irregularly-shaped ergodic sets very expensive, if not impossible. This problem can be ameliorated by merging simulation-based approaches with SG-based approaches, as proposed by Judd et al. [2014], Maliar and Maliar [2015].

Second, in many estimation tasks in economics and finance, some of the fundamental probability density functions that need to be approximated are multi-variate Gaussians. While a one-dimensional Gaussian can be approximated well by a piecewise-linear basis, multi-linear functions can have a hard time approximating multi-dimensional Gaussians. Spending more and more grid points can lead to significant approximation errors for densities with low variance, as LSGs tend to under- and overshoot in practical applications (see, e.g., Pflüger [2010]). In contrast, a straightforward measure that was shown to lift this issue is to approximate the logarithm of the Gaussian likelihood, where only a few basis functions are typically sufficient to obtain small interpolation errors [Frommert et al., 2010].

Third, even though SG methods alleviate the curse of dimensionality, their algorithmic complexity still grows polynomially with increasing refinement level and dimension.[36] This mathematical reality is worsened in practice by the fact that the interpolation tasks, when solving, for instance, nonlinear sets of equations (cf. section 3.2), take up to 99% of the computation time needed to solve the equation system, which is partly a result of the data structures used. These imparted limitations can be alleviated by choosing suitable data structures (to accelerate the interpolation tasks) and leveraging the high degree of intrinsic parallelism present in SGs, which can lead to several orders of magnitude of speedup on contemporary high-performance computing architectures (see, e.g., Scheidegger et al. [2018], and references therein). In addition, if the above measures do not provide sufficient performance gains for the model at hand, SGs can be coupled with so-called high-dimensional model representation (HDMR) techniques (see, e.g., Eftekhari et al. [2017], and references therein). The core idea of HDRM is to approximate high-dimensional functions by decomposing them into a nested summation of lower-dimensional functions [Holtz, 2010], which themselves are represented with SGs, thus imposing an additional layer of sparsity. Furthermore, such a dimensional decomposition also increases computational separability, which is ideal for parallelization. Combining SGs with HDMR has successfully been applied in the context of dynamic stochastic models containing up to 300 continuous state variables [Eftekhari and Scheidegger, 2020].

In summary, solving complex high-dimensional dynamic stochastic economic models numerically in reasonable time imposes several roadblocks. However, the rapidly-evolving field of SG methods provides a powerful framework that can handle a broad class of dynamic models up to a very high level of heterogeneity, non-linearity, and uncertainty—key features that, motivated by empirical observations, have become vital ingredients for capturing the salient features in contemporary dynamic economic models.

[36] See Judd et al. [2014] for a thorough discussion of the increasing computational complexity in the context of Smolyak SGs.

## Bibliographies

Yves Achdou, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll. Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach. The Review of Economic Studies, 04 2021. ISSN 0034-6527. doi: 10.1093/restud/rdab002. URL https://doi.org/10.1093/restud/rdab002. rdab002.

S. Rao Aiyagari. Uninsured idiosyncratic risk and aggregate saving. The Quarterly Journal of Economics, 109 (3):659–684, 1994. ISSN 00335533, 15314650. URL http://www.jstor.org/stable/2118417.

Marlon Azinovic, Luca Gaegauf, and Simon Scheidegger. Deep equilibrium nets. 2019. URL http://dx.doi.org/10.2139/ssrn.3393482.

Ivo Babuška, Fabio Nobile, and Raúl Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. SIAM Journal on Numerical Analysis, 45(3):1005–1034, 2007.

Guy Barles and Panagiotis E Souganidis. Convergence of approximation schemes for fully nonlinear second order equations. Asymptotic analysis, 4(3):271–283, 1991.

Richard E. Bellman. Adaptive Control Processes: A Guided Tour. 'Rand Corporation. Research studies. Princeton University Press, 1961. URL http://books.google.ch/books?id=POAmAAAAMAAJ.

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. ISSN 0885-7474. doi: 10.1007/s10915-012-9648-x. also available as INS Preprint No. 1207.

Johannes Brumm and Michael Grill. Computing equilibria in dynamic models with occasionally binding constraints. Journal of Economic Dynamics and Control, 38:142 – 160, 2014. ISSN 0165-1889. doi: https://doi.org/10.1016/j.jedc.2013.09.007. URL http://www.sciencedirect.com/science/article/pii/S0165188913001954.

Johannes Brumm and Simon Scheidegger. Using adaptive sparse grids to solve high-dimensional dynamic models. Econometrica, 85(5):1575–1612, 2017. ISSN 1468-0262. doi: 10.3982/ECTA12216. URL http://dx.doi.org/10.3982/ECTA12216.

Johannes Brumm, Dmitry Mikushin, Simon Scheidegger, and Olaf Schenk. Scalable high-dimensional dynamic stochastic economic modeling. Journal of Computational Science, 11:12 – 25, 2015. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2015.07.004. URL http://www.sciencedirect.com/science/article/pii/S1877750315300053.

Johannes Brumm, Felix Kubler, and Simon Scheidegger. Computing equilibria in dynamic stochastic macro-models with heterogeneous agents. In Advances in Economics and Econometrics: Volume 2: Eleventh World Congress, volume 59, page 185. Cambridge University Press, 2017.

H.-J. Bungartz and S. Dirnstorfer. Multivariate quadrature on adaptive sparse grids. Computing, 71:89–114, 2003. ISSN 0010-485X. doi: 10.1007/s00607-003-0016-4. URL http://dx.doi.org/10.1007/s00607-003-0016-4.

H. J. Bungartz, D. Pflüger, and S. Zimmer. Adaptive sparse grid techniques for data mining. In Hans Georg Bock, Ekaterina Kostina, Hoang Xuan Phu, and Rolf Rannacher, editors, Modeling, Simulation and Optimization of Complex Processes, pages 121–130, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-79409-7.

Hans-Joachim Bungartz and Michael Griebel. Sparse grids. Acta numerica, 13:147–269, 2004.

Hans-Joachim Bungartz, Alexander Heinecke, Dirk Pflüger, and Stefanie Schraufstetter. Option pricing with a direct adaptive sparse grid approach. Journal of Computational and Applied Mathematics, 236(15):3741–3750, 2012. ISSN 0377-0427. doi: http://dx.doi.org/10.1016/j.cam.2011.09.024. URL http://www.sciencedirect.com/science/article/pii/S0377042711005036.

Dan Cao, Wenlan Luo, and Guangyu Nie. Global DSGE models. Available at SSRN 3569013, 2020.

Wilbur John Coleman. Solving the stochastic growth model by policy-function iteration. Journal of Business & Economic Statistics, 8(1):27–29, 1990.

Wouter J Den Haan and Albert Marcet. Solving the stochastic growth model by parameterizing expectations. Journal of Business & Economic Statistics, 8(1):31–34, 1990.

John Duffy and Paul D. McNelis. Approximating and simulating the stochastic growth model: Parameterized expectations, neural networks, and the genetic algorithm. Journal of Economic Dynamics and Control, 25(9):1273–1303, September 2001. URL https://ideas.repec.org/a/eee/dyncon/v25y2001i9p1273-1303.html.

Aryan Eftekhari and Simon Scheidegger. High-dimensional dynamic stochastic model representation. Available at SSRN 3603294, 2020.

Aryan Eftekhari, Simon Scheidegger, and Olaf Schenk. Parallelized dimensional decomposition for large-scale dynamic stochastic economic models. In Proceedings of the Platform for Advanced Scientific Computing Conference, PASC '17, pages 9:1–9:11, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5062-4. doi: 10.1145/3093172.3093234. URL http://doi.acm.org/10.1145/3093172.3093234.

Jesús Fernández-Villaverde, Grey Gordon, Pablo Guerrón-Quintana, and Juan F Rubio-Ramirez. Nonlinear adventures at the zero lower bound. Journal of Economic Dynamics and Control, 57:182–204, 2015.

Jesús Fernández-Villaverde, Samuel Hurtado, and Galo Nuño. Financial frictions and the wealth distribution. Working Paper 26302, National Bureau of Economic Research, September 2019. URL http://www.nber.org/papers/w26302.

M. Frommert, D. Pflüger, T. Riller, M. Reinecke, H.-J. Bungartz, and T. A. Enßlin. Efficient cosmological parameter sampling using sparse grids. Monthly Notices of the Royal Astronomical Society, 406(2):1177–1189, 07 2010. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2010.16788.x. URL https://doi.org/10.1111/j.1365-2966.2010.16788.x.

Jochen Garcke and Steffen Ruttscheidt. Finite Differences on Sparse Grids for Continuous Time Heterogeneous Agent Models. 2019.

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. ISBN 0-444-87570-0. URL http://dl.acm.org/citation.cfm?id=2837.2842.

Thomas Gerstner and Michael Griebel. Dimension-adaptive tensor-product quadrature. Computing, 71:2003, 2003.

Alexandros Gilch, Michael Griebel, and Jens Oettershagen. Sparse tensor product approximation for a class of generalized method of moments estimators. International Journal for Uncertainty Quantification, 2021. doi: 10.1615/Int.J.UncertaintyQuantification.2021037549. Also available as INS Preprint No. 2006.

M. Griebel, M. Schneider, and C. Zenger. A combination technique for the solution of sparse grid problems. In P. de Groen and R. Beauwens, editors, Iterative Methods in Linear Algebra, pages 263–281. IMACS, Elsevier, North Holland, 1992. also as SFB Bericht, 342/19/90 A, Institut für Informatik, TU München, 1990.

Max Gunzburger, Clayton G. Webster, and Guannan Zhang. An adaptive wavelet stochastic collocation method for irregular solutions of partial differential equations with random input data. In Jochen Garcke and Dirk Pflüger, editors, Sparse Grids and Applications - Munich 2012, pages 137–170, Cham, 2014. Springer International Publishing. ISBN 978-3-319-04537-5.

M. Hegland. Adaptive sparse grids. In K. Burrage and Roger B. Sidje, editors, Proc. of 10th Computational Techniques and Applications Conference CTAC-2001, volume 44, pages C335–C353, April 2003. [Online] http://anziamj.austms.org.au/V44/CTAC2001/Hegl [April 1, 2003].

Florian Heiss and Viktor Winschel. Likelihood approximation by numerical integration on sparse grids. Journal of Econometrics, 144(1):62–80, 2008.

Pieter W Hemker. Application of an adaptive sparse-grid technique to a model singular perturbation problem. Computing, 65(4):357–378, 2000.

Markus Holtz. Sparse Grid Quadrature in High Dimensions with Applications in Finance and Insurance. Lecture Notes in Computational Science and Engineering. Springer, 2010. ISBN 9783642160042.

Kenneth L Judd. Numerical methods in economics. The MIT press, 1998.

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.

Kenneth L. Judd, Lilia Maliar, Serguei Maliar, and Inna Tsener. How to solve dynamic stochastic models computing expectations just once. Quantitative Economics, 8(3):851–893, 2017. doi: https://doi.org/10.3982/QE329. URL https://onlinelibrary.wiley.com/doi/abs/10.3982/QE329.

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. ISSN 0165-1889. doi: http://dx.doi.org/10.1016/j.jedc.2010.09.013.

Dirk Krueger and Felix Kubler. Computing equilibrium in olg models with stochastic production. Journal of Economic Dynamics and Control, 28(7):1411 – 1436, 2004. ISSN 0165-1889. doi: https://doi.org/10.1016/S0165-1889(03)00111-8. URL http://www.sciencedirect.com/science/article/pii/S0165188903001118.

Dirk Krueger and Felix Kubler. Pareto-improving social security reform when financial markets are incomplete!? American Economic Review, 96(3):737–755, June 2006. doi: 10.1257/aer.96.3.737. URL http://www.aeaweb.org/articles?id=10.1257/aer.96.3.737.

Randall J. LeVeque. Finite Difference Methods for Ordinary and Partial Differential Equations. Society for Industrial and Applied Mathematics, 2007. doi: 10.1137/1.9780898717839. URL https://epubs.siam.org/doi/abs/10.1137/1.9780898717839.

Lars Ljungqvist and Thomas J Sargent. Recursive macroeconomic theory. Mit Press, 2004.

Xiang Ma and Nicholas Zabaras. An adaptive hierarchical sparse grid collocation algorithm for the solution of stochastic differential equations. Journal of Computational Physics, 228(8):3084–3113, 2009. ISSN 0021-9991. doi: 10.1016/j.jcp.2009.01.006. URL http://dx.doi.org/10.1016/j.jcp.2009.01.006.

Lilia Maliar and Serguei Maliar. Chapter 7 - numerical methods for large-scale dynamic economic models. In Karl Schmedders and Kenneth L. Judd, editors, Handbook of Computational Economics Vol. 3, volume 3 of Handbook of Computational Economics, pages 325–477. Elsevier, 2014. doi: https://doi.org/10.1016/B978-0-444-52980-0.00007-4. URL https://www.sciencedirect.com/science/article/pii/B9780444529800000074.

Lilia Maliar and Serguei Maliar. Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model. Quantitative Economics, 6(1):1–47, 2015. doi: 10.3982/QE364.

Lilia Maliar, Serguei Maliar, and Pablo Winant. Deep learning for solving dynamic economic models. Journal of Monetary Economics, 122:76–101, 2021. ISSN 0304-3932. doi: https://doi.org/10.1016/j.jmoneco.2021.07.004. URL https://www.sciencedirect.com/science/article/pii/S0304393221000799.

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

F. Nobile, R. Tempone, and C. G. Webster. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. SIAM Journal on Numerical Analysis, 46(5):2411–2442, 2008. ISSN 00361429. URL http://www.jstor.org/stable/25663069.

Andriy Norets. Estimation of dynamic discrete choice models using artificial neural network approximations. Econometric Reviews, 31(1):84–106, 2012. doi: 10.1080/07474938.2011.607089.

Michael Obersteiner and Hans-Joachim Bungartz. A generalized spatially adaptive sparse grid combination technique with dimension-wise refinement. SIAM Journal on Scientific Computing, 43(4):A2381–A2403, 2021. doi: 10.1137/20M1325885. URL https://doi.org/10.1137/20M1325885.

Bernt Oksendal. Stochastic Differential Equations (3rd Ed.): An Introduction with Applications. Springer-Verlag, Berlin, Heidelberg, 1992. ISBN 3387533354.

Dirk Pflüger. Spatially Adaptive Sparse Grids for High-Dimensional Problems. PhD thesis, München, 2010. URL http://www5.in.tum.de/pub/pflueger10spatially.pdf.

Michael F. Rehme, Fabian Franzelin, and Dirk Pflüger. B-splines on sparse grids for surrogates in uncertainty quantification. Reliability Engineering System Safety, 209:107430, 2021. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2021.107430. URL https://www.sciencedirect.com/science/article/pii/S0951832021000016.

Christoph Reisinger and Gabriel Wittum. Efficient hierarchical approximation of high-dimensional option pricing problems. SIAM J. Sci. Comput., 29(1):440–458, January 2007. ISSN 1064-8275. doi: 10.1137/060649616. URL http://dx.doi.org/10.1137/060649616.

John Philip Rust. Numerical dynamic programming in economics. In H. M. Amman, D. A. Kendrick, and J. Rust, editors, Handbook of Computational Economics, volume 1, chapter 14, pages 619–729. Elsevier, 1 edition, 1996. URL http://EconPapers.repec.org/RePEc:eee:hecchp:1-14.

Andreas Schaab. Micro and macro uncertainty. 2020.

Andreas Schaab and Allen Tianlun Zhang. Essays in Macroeconomics, Chapter 2: Dynamic Programming in Continuous Time with Adaptive Sparse Grids. PhD thesis, 2021.

Simon Scheidegger and Ilias Bilionis. Machine learning for high-dimensional dynamic stochastic economies. Journal of Computational Science, 33:68 – 82, 2019. ISSN 1877-7503. doi: https://doi.org/10.1016/j.jocs.2019.03.004. URL http://www.sciencedirect.com/science/article/pii/S1877750318306161.

Simon Scheidegger and Adrien Treccani. Pricing American Options under High-Dimensional Models with Recursive Adaptive Sparse Expectations*. Journal of Financial Econometrics, 19(2):258–290, 10 2018. ISSN 1479-8409. doi: 10.1093/jjfinec/nby024. URL https://doi.org/10.1093/jjfinec/nby024.

Simon Scheidegger, Dmitry Mikushin, Felix Kubler, and Olaf Schenk. Rethinking large-scale economic modeling for efficiency: Optimizations for gpu and xeon phi clusters. In 2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 610–619, 2018. doi: 10.1109/IPDPS.2018.00070.

Thomas Schiekofer. Die Methode der Finiten Differenzen auf d unnen Gittern zur L osung elliptischer und parabolischer par PhD thesis, PhD thesis, Universit at Bonn, 1998.

Stephanie Schmitt-Grohé and Martın Uribe. Solving dynamic general equilibrium models using a second-order approximation to the policy function. Journal of economic dynamics and control, 28(4):755–775, 2004.

Peter Schober, Julian Valentin, and Dirk Pflüger. Solving high-dimensional dynamic portfolio choice models with hierarchical b-splines on sparse grids. Computational Economics, pages 1–40, 2021.

Sergey A. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. Soviet Mathematics, Doklady, 4:240–243, 1963.

Nancy L. Stokey, Robert E. Lucas, and Edward C. Prescott. Recursive Methods in Economic Dynamics. Harvard University Press, 1989. ISBN 9780674750968. URL http://www.jstor.org/stable/j.ctvjnrt76.

Miroslav Stoyanov. User manual: Tasmanian sparse grids. Technical Report ORNL/TM-2015/596, Oak Ridge National Laboratory, One Bethel Valley Road, Oak Ridge, TN, 2015.

Takafumi Usui. Adaptation to rare natural disasters and global sensitivity analysis in a dynamic stochastic economy. Available at SSRN 3462011, 2019.

G.W. Wasilkowski and H. Wozniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. Journal of Complexity, 11(1):1–56, 1995. ISSN 0885-064X. doi: https://doi.org/10.1006/jcom.1995.1001. URL https://www.sciencedirect.com/science/article/pii/S0885064X85710011.

Viktor Winschel and Markus Krätzig. Solving, estimating, and selecting nonlinear dynamic models without the curse of dimensionality. Econometrica, 78(2):803–821, 2010. doi: https://doi.org/10.3982/ECTA6297. URL https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA6297.

Dongbin Xiu and Jan S. Hesthaven. High-order collocation methods for differential equations with random inputs. SIAM Journal on Scientific Computing, 27(3):1118–1139, 2005. doi: 10.1137/040615201. URL https://doi.org/10.1137/040615201.

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. URL http://www5.in.tum.de/pub/zenger91sg.pdf.