Dynamic Programming in Continuous Time with Adaptive Sparse Grids*

Andreas Schaab† Allen T. Zhang‡

May, 2022

Abstract

This paper proposes a new approach to dynamic programming in continuous time using adaptive sparse grids. The standard finite-difference method, used to solve differential equations on uniform grids, fails on sparse grids. Our paper presents a sparse finite-difference method that leads to consistent solutions on a broad class of non-uniform grids. We demonstrate the power and versatility of our approach across a wide range of economic applications that feature high-dimensional state spaces, occasionally-binding constraints, life-cycles and overlapping generations, kinks and non-convexities, discrete choice, free boundaries, and dynamic games. A code repository accompanies this paper and provides a general-purpose library to solve continuous-time dynamic programming problems on adaptive sparse grids.[1]

JEL codes: C61, C63, E21, G11, L13

Keywords: dynamic programming, continuous time, adaptive sparse grids, occasionally-binding constraints, OLG, free boundary, non-convexities, dynamic oligopoly

*Appendix available on request.

†Columbia Business School. Email: ajs2428@columbia.edu. Website: https://andreasschaab.com.

‡Harvard University.

For many helpful comments and discussions, we are grateful to Johannes Brumm, Emmanuel Farhi, Xavier Gabaix, Greg Kaplan, Robin Lee, Matteo Maggiori, Benjamin Moll, Ariel Pakes, Chris Rycroft, Simon Scheidegger, Ludwig Straub, Adi Sunderam, as well as many seminar and conference participants.

[1] This repository is available at: https://github.com/schaab-lab/SparseEcon.

Introduction

Dynamic programming is widely used in economic applications that feature dynamic optimization problems. Its numerical implementation usually employs uniform grids, which suffer from the curse of dimensionality. Regular sparse grids help overcome this challenge and have already gained some popularity in economics, e.g., following Smolyak (1963)'s construction principle. It is adaptive sparse grids, however, that promise maximal efficiency by refining the grid to best suit the underlying economic application, placing grid points where increased resolution is valuable and removing them elsewhere.

In this paper, we forge a new approach to dynamic programming in continuous time leveraging adaptive sparse grids. Our starting point is a negative result: In continuous time, dynamic programming is based on the solution of differential equations. And the standard finite-difference method—used to discretize differential equations on uniform grids—breaks down on sparse grids. We develop a new value function iteration algorithm—based on a *sparse finite-difference method*—and generalize continuous-time dynamic programming to a broad class of non-uniform grids. We show that the versatility of adaptive sparse grids in particular makes them well-suited to tackle a wide range of economic problems that feature high-dimensional state spaces, occasionally-binding constraints, overlapping generations, kinks and non-convexities, discrete choice, free boundaries, and dynamic games. Our applications showcase that sparse grid dynamic programming substantially outperforms the standard approach in each case. A code repository accompanies this paper, which provides a general-purpose library for continuous-time dynamic programming on adaptive sparse grids and features code and tutorials for many applications across several fields of economics.

The protagonists of this paper are grids. In numerical analysis, a grid discretizes the domain of a function or the state space of an economic application. We focus on three classes of differently structured grids: Uniform grids feature regularly spaced nodes. In higher dimensions, they are constructed as the full tensor product of one-dimensional grids. Uniform grids consequently suffer from the curse of dimensionality, as the number of grid points associated with the full tensor product grows exponentially. Sparse grids aim to remove grid points, i.e., restrict the full tensor product, without compromising numerical accuracy. They emerge from a cost-benefit analysis that judges the value of each grid point in terms of its marginal contribution to the accuracy of function approximations. Regular sparse grids are efficient from the ex-ante perspective of accurately representing an entire class of functions, e.g., all functions that are continuously differentiable. They remain agnostic about the details of a specific economic problem. Adaptive sparse grids, on the other hand, use such information to locally refine the grid, placing nodes where the economic application benefits from increased resolution and removing them elsewhere. In particular, we develop a value function iteration algorithm that automates local grid adaptation based on the residual approximation error in the value function.

In continuous time, dynamic programming gives rise to differential equations that characterize

the policy and value functions associated with the dynamic optimization problem—the analog of the Bellman equation in discrete time. The most popular approach to solve these equations numerically is based on finite-difference methods. On a uniform grid with mesh size h, the first-order derivative $f'(x)$ can be discretized in terms of the function values at a grid point x and its right neighbor x^+. By construction, the distance between two neighboring nodes on a uniform grid is h, with $x^+ = x + h$, and so the approximation of the derivative becomes $\frac{f(x+h)-f(x)}{h}$, which converges to the true mathematical derivative $f'(x)$ as the grid becomes finer. A finite-difference method that satisfies this property is called *consistent*.

On sparse grids, however, it is no longer the case that the distance between a grid point x and its direct right neighbor x^+ is of order h. In fact, sparse grids of arbitrary resolution generically contain grid points whose direct neighbors are far away. As a result, discretizing derivatives using the standard finite-difference method, with only looks at a grid point's direct neighbors, no longer leads to consistent approximations. It is consequently not possible to numerically solve continuous-time dynamic programming problems on sparse grids using the standard finite-difference method.

In this paper, we develop a sparse finite-difference method to overcome this challenge. This method draws on information from a d-dimensional ball around the grid point in question, which we show leads to a consistent discretization of derivatives. In fact, we prove that the sparse finite-difference method is equivalent to interpolating onto exactly those points that would be used by the standard finite-difference if the grid were uniform. This interpolation error itself is of order $\mathcal{O}(h)$, and so the scheme overall is of order $\mathcal{O}(h)$ as well. Our discretization method consequently takes the form $\boldsymbol{E}_j \boldsymbol{D}_j \boldsymbol{H}_j$, where $\boldsymbol{D}_j$ is the standard finite-difference matrix in dimension j, and $\boldsymbol{E}_j$ and $\boldsymbol{H}_j$ are easily computable projection matrices. We also develop a general method to handle boundary conditions and show that the resulting boundary-adjusted finite-difference method remains consistent.

Leveraging our sparse finite-difference method, we present a value function iteration algorithm on adaptive sparse grids. Intuitively, the standard algorithm is augmented by an outer fixed point, in which the algorithm automatically adapts the grid to increase (decrease) resolution where the numerical approximation error in the value function remains large (small). Our algorithm starts with a coarse grid and carries over the numerical solution of the value function from one outer iteration to the next as an initial guess.

Use cases. We demonstrate the power and versatility of our approach in the context of many applications, showcasing that adaptive sparse grids substantially outperform uniform and even regular sparse grids. Section 5 of the main text presents five applications. In the repository that accompanies this paper, we additionally develop code and provide pedagogical tutorials for many dynamic programming applications drawn from several fields of economics, including macroeconomics, asset pricing, and industrial organization.

Our applications are judiciously chosen to highlight the use cases for sparse grid dynamic

programming. Adaptive sparse grids are powerful across a wide range of economic problems that either feature high-dimensional state spaces or require localized grid resolution:

1. *High dimensionality*: Dynamic optimization problems on high-dimensional state spaces are notoriously challenging to solve. On uniform grids, the curse of dimensionality makes the numerical solution of problems with more than 10 state variables all but intractable. The rapidly growing heterogeneous-agent literature again highlights the pertinence of this obstacle: In dynamic general equilibrium models with rich heterogeneity, the high-dimensional cross-sectional distribution of agents becomes part of the state space. In Section 5.1, we illustrate the power of adaptive sparse grids in this context by globally solving a variant of the Krusell and Smith (1998) model with a 14-dimensional distribution representation. Despite the large state space, our value function iteration algorithm requires less than 5 minutes on a personal computer.

2. *Occasionally-binding constraints*: Many economic applications feature occasionally-binding constraints such as borrowing and collateral constraints, or the zero lower bound on monetary policy. When agents face such constraints, their behavior often becomes highly sensitive to, i.e., non-linear in, changes in fundamentals. Sparse grid methods excel by placing grid points and adding resolution in the region of the state space where these constraints bind. We illustrate this in Sections 5.2 through 5.4, which present variants of the standard incomplete markets model in macroeconomics in the spirit of Huggett (1993) and Aiyagari (1994).

3. *Kinks and non-convexities*: When economic applications feature a locally non-linear solution, i.e., when behavior becomes highly sensitive in a local region of the state space, solving the problem on a uniform grid is highly inefficient. Using adaptive sparse grids allows us to match the local grid resolution to the demands of the underlying application. We illustrate their potency in the context of an Aiyagari (1994) economy with a non-convex capital income tax schedule in Section 5.3. The resulting consumption policy function becomes locally non-convex at the tax threshold and the value function features a kink. Our grid adaptation algorithm automatically increases the grid resolution near the tax threshold, while leaving the grid relatively coarse elsewhere.

4. *Finite horizons*: Many important economic problems feature finite horizons, including life-cycle and overlapping generations models. Solving these models accurately often requires high grid resolution near the endpoints of the time dimension. We show in Section 5.4 that a life-cycle portfolio choice problem requires substantial localized resolution to accurately characterize households' behavior near the time of retirement or death.

5. *Free-boundary problems*: Economic applications with free boundary problems abound. We show in Section 5.5 that adaptive sparse grids are powerful in this context for two reasons. First, the value and policy functions often exhibit large gradients near the free boundary.

Local grid refinement is therefore useful to place grid points close to the free boundary while removing them elsewhere. Second, solving free boundary problem oftentimes does not require characterizing agents' behavior beyond the free boundary. In this case, our grid adaptation algorithm automatically removes grid points in that region of the state space once the free boundary is accurately pinned down. In Section 5.5 of the main text, we present the well-known optimal stopping problem of a firm that chooses the time of plant closure. In our code repository, we develop several applications featuring consumer and sovereign default.

6. *Discrete-choice*: Discrete-choice models are widely used in industrial organization and other fields of economics. A rapidly expanding literature in macroeconomics focuses on household finance, with interest in discrete home purchase and mortgage decisions. Historically, discrete-time dynamic programming has been the method of choice to solve such problems. We show how sparse grids can be leveraged to realize great efficiency gains in continuous-time discrete-choice applications, where the analog of the discrete-time Bellman equation takes the form of a Hamilton-Jacobi-Bellman variational inequality. We illustrate this by solving a variant of the Ericson and Pakes (1995) model in continuous time, extended to feature a continuous quality dimension. We also solve a household portfolio choice problem with a discrete housing choice.

7. *Dynamic games*: In dynamic N-player games, the state variables of each of the N players become part of the state space of every other player. Computing best-response strategies therefore requires solving high-dimensional dynamic programming problems. Adaptive sparse grids again excel by realizing substantial efficiency gains relative to uniform grids on high-dimensional state spaces. In our repository, we solve the N-player game in a continuous-time variant of the seminal Ericson and Pakes (1995) model of quality ladders under dynamic oligopolistic competition.

Related literature. Following their introduction in a series of papers by Zenger (1991), Griebel (1991), Bungartz (1992), and Griebel et al. (1992), sparse grids have become an active research area in applied mathematics.[2,3] These initial contributions focused especially on the finite-element method to solve partial differential equations. Sparse grid methods were extended to the class of finite-difference methods by Schiekofer (1998) and Griebel (1998). Schiekofer (1998) was first

[2] Closely related ideas were discussed and used in the context of other problems much earlier. Babenko (1960) discusses a similar approach—known as hyperbolic crosses—in the approximation of periodic functions based on restrictions on Fourier coefficients. In his seminal contribution, Smolyak (1963) proposes a tensor product approach that leads to efficient non-uniform grids in the context of numerical integration. "Smolyak-grids" have frequently been employed in economics. For further details on these and related approaches such as Boolean methods (Delvos, 1982) and discrete blending methods (Baszenski et al., 1992), we refer readers to Bungartz and Griebel (2004).

[3] Sparse grid methods use function representations in the hierarchical basis (Faber, 1909). Yserentant (1986, 1992) introduced the hierarchical (multi-grid) basis for the numerical solution of partial differential equations. Zenger (1991)'s original definition of sparse grids is based on a generalization of Yserentant (1986)'s hierarchical basis to a tensor product-based approach that allows for subspace splitting. Relative to previous work, Zenger (1991) defines sparse grids formally as emerging from an a priori selection of approximating function subspaces based a multi-grid approach.

to show that the standard finite-difference method, which leads to the consistent discretization of partial derivatives on uniform grids, can fail on regular sparse grids. Schiekofer (1998)'s main contribution was a consistency proof for a generalized finite-difference method on a particular class of regular sparse grids.[4] Koster (2002), Ruttscheidt (2018), and Garcke and Ruttscheidt (2019) have proposed an alternative interpolation-based approach to solve differential equations on sparse grids using finite-difference methods.[5] Relative to this extensive body of work in applied mathematics, this paper demonstrates the power of adaptive sparse grids in the context of continuous-time dynamic programming applications in economics. Our approach builds on Schiekofer (1998)'s original sparse finite-difference method. We present a value function iteration algorithm using adaptive sparse grids, develop a general treatment of boundary conditions, extend the sparse finite-difference consistency result to a broader class of sparse grids, and prove the equivalence between Schiekofer (1998)'s construction and an interpolation stencil.

Our paper also builds on previous work that has leveraged sparse grids in economics. Krueger and Kubler (2004) propose a computational method based on Smolyak sparse grids to solve stochastic equilibria in OLG economies. Judd et al. (2014) use sparse grids together with a stochastic simulation approach to solve a high-dimensional multi-country model. The seminal contribution of Brumm and Scheidegger (2017) introduces adaptive sparse grids to economics. The main difference from earlier work is that they rely on local basis functions rather than global ones, which allows for adaptive grid refinement based on a measure of residual local approximation error. Schober (2018) uses this method to solve a high-dimensional dynamic portfolio choice model. The main distinction between these papers and ours is that we focus on continuous-time dynamic programming, which requires the solution of (partial) differential equations. The application of sparse grid methods in continuous time faces different challenges than in discrete time.

The first applications of continuous-time sparse grid methods in economics were in the area of finance and option pricing. Pflüger (2010) and Heinecke et al. (2012) solve Black Scholes equations for option pricing problems using an adaptive sparse grid finite-element method. Garcke and Ruttscheidt (2019), building on Ruttscheidt (2018), solve a six-dimensional portfolio choice problem in partial equilibrium. Schaab (2020) uses the adaptive sparse grid method developed in this paper to globally solve a heterogeneous-agent New Keynesian model with aggregate uncertainty and an occasionally-binding zero lower bound consdtraint.

[4] Schiekofer (1998)—as well as subsequent work—also discusses and numerically explores the stability and convergence properties of the resulting sparse finite-difference schemes. Formal stability and convergence results have proven elusive, however. Proving general results on the convergence of sparse finite-difference schemes in the context of viscosity solutions has been challenging because the Barles and Souganidis (1991) monotonicity condition is generally not satisfied. Hemker (2000) points out that interpolation on sparse grids does not generally satisfy monotonicity. Garcke and Ruttscheidt (2019) show that even in the case of concave, monotonically increasing functions, interpolation on sparse grids is not 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 sparse grid interpolation may be non-monotone.

[5] The literature on sparse grids in applied mathematics has since rapidly grown. The power of sparse grid methods has been demonstrated in the context of numerous applications, including differential equations, integration and quadrature, dimensional adaptation, integral equations, data mining, and uncertainty quantification among others.

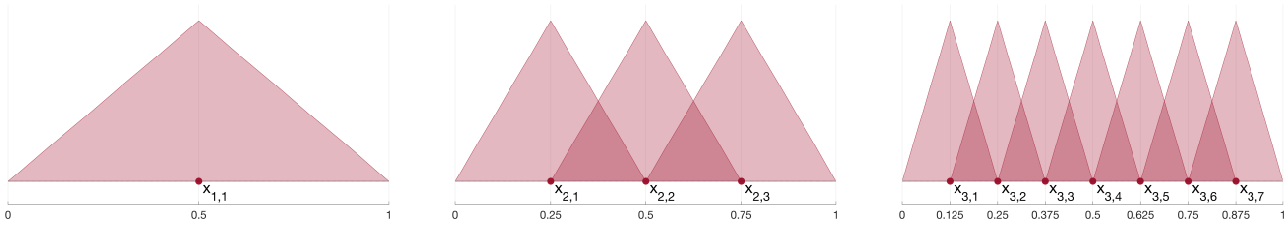
Layout. The rest of this paper proceeds as follows. Section 1 introduces hierarchical basis functions and shows how to construct sparse grids. Section 2 reviews the standard approach to continuous-time dynamic programming on uniform grids, setting the stage for our discussion of sparse grids. Section 3 then shows that this standard approach breakds down on sparse grids and introduces a sparse finite-difference method to solve the differential equations that emerge from continuous-time dynamic programming. Section 3 also presents a value function iteration algorithm on adaptive sparse grids and concludes with a discussion of monotonicity and convergence. Section 4 develops a general treatment of boundary conditions. Sections 5 and 6 respectively present our main applications and introduce the SparseEcon code repository that accompanies this paper. Finally, Section 7 concludes.

1 Function Approximations, Hierarchical Basis, and Sparse Grids

Our object of interest is a real-valued function $f : \Omega \to \mathbb{R}$ that represents the solution of an economic model. In this section, we consider as our domain the d-dimensional hypercube $\Omega = [0,1]^d$ and start by studying the one-dimensional case with $d = 1$ for ease of exposition. Our goal is to approximate f on some grid $G \subset \Omega$ that discretizes the domain. Denoting by l the *level* and i the *index*, we construct a grid point as the numeric value $x_{l,i} = i \cdot 2^{-l}$. Grid G is a collection of J such grid points and may be represented as a $J \times d$ matrix. We set the *local mesh size* to $h_l = 2^{-l}$. At a given level l, the maximum index that can be active is $2^l - 1$.[6]

A natural question is whether we can find the most efficient grid structure for approximating a given class of functions. Sparse grids emerge as the answer to this question (Zenger, 1991). To formalize this argument, we develop a cost-benefit analysis that determines the relative value of grid points and allows us to compare one grid to another. Intuitively, a grid is efficient if achieving a desired accuracy in approximating f is possible with relatively few grid points, and the value of a single grid point is then its marginal contribution to this approximation accuracy. Formally, approximation error corresponds to the distance between f and its approximation in an underlying function space $\mathcal{V}$, to which f belongs. Since the approximation of f on grid G is typically represented as a $J \times 1$ vector of function values, however, our first task is to associate this vector with an *interpolant* in $\mathcal{V}$. In Sections 1.1 and 1.2, we develop a basis function representation that allows us to interpolate the approximation of f onto points in the domain Ω that are not members of G. After formally characterizing the value of a grid point, we show in Section 1.3 how sparse grids systematically remove grid points whose marginal contribution is low and add them where the local function approximation error remains large.

[6] We adopt the construction principle of *dyadic grids*.



(a) $l = 1$ and $I_1 = \{1\}$ (b) $l = 2$ and $I_2 = \{1, 2, 3\}$ (c) $l = 3$ and $I_3 = \{1, \ldots, 7\}$

Figure 1: Nodal basis functions

Note. Figure 1 displays the nodal basis functions associated with levels $l \in \{1, 2, 3\}$ and illustrates the grid construction principle adopted in this paper. At level $l = 1$ in panel (a), the nodal index set is simply $I_1 = \{1\}$ and thus associated with a single grid point, $x_{1,1} = i \cdot 2^{-l} = 0.5$. The nodal basis function $\phi_{1,1}(x)$ is centered on $x_{1,1}$ and has symmetric support on $[0, 1]$. At higher levels l, illustrated in panels (b) and (c), the nodal index set is defined as $I_l = \{i \in \mathbb{N} \mid 1 \leq i \leq 2^l - 1\}$. Each grid point $x_{l,i}$ is associated with a nodal basis function that is centered on it and has symmetric support over $x \in [x_{l,i} - h_l, x_{l,i} + h_l]$.

1.1 The Approximation Space of Piecewise Linear Interpolants

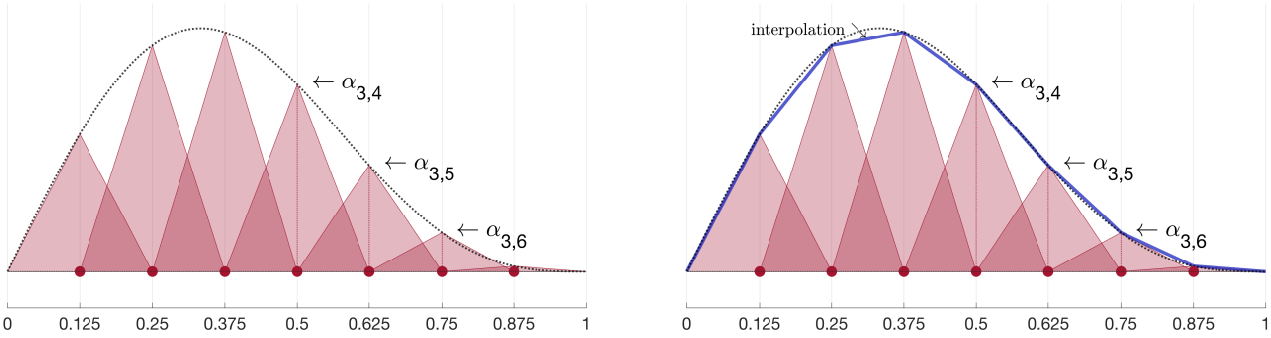
We focus on linear interpolation and use piecewise linear *hat functions* as basis functions.[7] Any one-dimensional symmetric hat function can be defined via translation and dilation of the *mother hat function* ϕ as

$$\phi_{l,i}(x) = \phi\left(\frac{x - ih_l}{h_l}\right), \qquad \text{where } \phi(x) = \begin{cases} 1 - |x| & \text{for } x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

for all $x \in [0, 1]$. The subscript (l, i) indicates that the hat function is centered around the grid point $x_{l,i}$.

We associate a collection of grid points or basis functions with an *index set*. In particular, we define the nodal index set of level l as $I_l = \{i \in \mathbb{N} \mid 1 \leq i \leq 2^l - 1\}$. The set of nodal basis functions

[7] Most work on sparse grids is based on d-dimensional piecewise-linear basis functions. Bungartz (1992) and Bungartz and Griebel (1999) show that, for tensor products based on the 1-dimensional piecewise linear basis function, it can be proven that the relative contribution of basis function coefficients in a hierarchical representation decays at rate $2^{-2|l|_1}$. Subsequent research has extended the sparse grid apparatus to a wide range of basis functions. Bungartz (1998) introduces the hierarchical Lagrange interpolation approach, which uses hierarchical bases of piecewise polynomials of arbitrary degree, allowing one degree of freedom per node. This results in a higher order of approximation. Bungartz and Griebel (2004) show that using polynomial basis functions of degree p and optimizing the relative contributions of the associated approximating subspaces results in an interpolation error of order $\mathcal{O}(h_n^{p+1} \cdot |\log_2 h_n|^{d-1})$ in the L_2 and L_∞ norms. The interpolet family introduced by Deslauriers and Dubuc (1989) represents an alternative set of higher-order basis functions that can be leveraged in the context of sparse grids. Bungartz and Griebel (2004) discuss how to use the 1-dimensional higher-order interpolet function—instead of the hat functions we use here—in the tensor product construction of hierarchical subspaces and sparse grids. While higher-order approaches such as hierarchical Lagrange interpolation or interpolets can leverage stricter regularity assumptions on the underlying function to achieve better approximation rates, they are not stable in the context of subspace splitting in the sense of Oswald (2013). The family of basis functions based on wavelets and prewavelets (Daubechies, 1992) can be used to circumvent this issue. For details see Bungartz and Griebel (2004).



(a) Function values $\alpha_{l,i} = f(x_{l,i})$

(b) Piecewise linear interpolation of $f(x)$ on G

Figure 2: Interpolation in the nodal basis

Note. Figure 2 illustrates the nodal basis function representation of the piecewise linear interpolant $f_l(x)$ that approximates $f(x) = \frac{4}{5}(\sin(\pi x) + \frac{1}{2}\sin(2\pi x))$ (*dashed black line*) on grid of level $l = 3$, $G = \{x_{3,i} \mid i \in I_3\}$. Panel (a) again displays the collection of hat functions that comprise the nodal basis of level $l = 3$, each now weighted by the function values $\alpha_{3,i} = f(x_{3,i})$. Panel (b) also plots the piecewise linear interpolant $f_3(x) = \sum_{i \in I_3} \alpha_{3,i}\, \phi_{3,i}(x)$ (*solid blue line*).

of level l is then given by $\{\phi_{l,i}(x) \mid i \in I_l \text{ and } x \in \Omega\}$. We call the span of this set of basis functions the *nodal basis*. Figure 1 displays the hat functions $\{\phi_{l,i}(x)\}$ for levels 1, 2, and 3 and illustrates the grid construction principle we adopt in this paper. For example, grid point $x_{3,2}$ in panel (c) takes the value $x_{3,2} = i \cdot 2^{-l} = 2 \cdot 2^{-3} = 0.25$. The associated hat function $\phi_{3,2}(x)$ is centered on $x = 0.25$ and has symmetric support over $x \in [x_{3,2} - 2^{-l}, x_{3,2} + 2^{-l}] = [0.125, 0.375]$. Each node $x_{l,i}$ in grid G can therefore be associated with a hat function $\phi_{l,i}(x)$ that is centered on it.

Interpolation. We can fit function f on grid G by evaluating it at all the grid points, and it will be notationally convenient to denote by $\alpha_{l,i} = f(x_{l,i})$ the function values of f on grid G. We can use the hat functions comprising the nodal basis for interpolation by using the function values $\alpha_{l,i}$ as basis function coefficients. That is, we can approximate f with the piecewise linear interpolant f_l on grid G of level l, i.e.,

$$f(x) \approx f_l(x) = \sum_{i \in I_l} \alpha_{l,i}\, \phi_{l,i}(x). \tag{2}$$

Crucially, the interpolant f_l associated with grid G is now also a member of the underlying function space $\mathcal{V}$, and we can formalize the resulting approximation error as the distance between f and f_l in $\mathcal{V}$ according to some norm $\|\cdot\|$.

Since we use the nodal index set I_l to construct this basis function representation, we also refer to equation (2) as the *nodal representation* of the piecewise linear interpolant f_l. Figure 2 illustrates piecewise linear interpolation in the nodal basis. In panel (a), we again plot the nodal hat functions associated with a grid of level $l = 3$ and scale each hat by its associated function value $\alpha_{l,i}$. The blue line in panel (b) depicts $f_l(x)$ using the nodal basis representation (2). The residual approximation

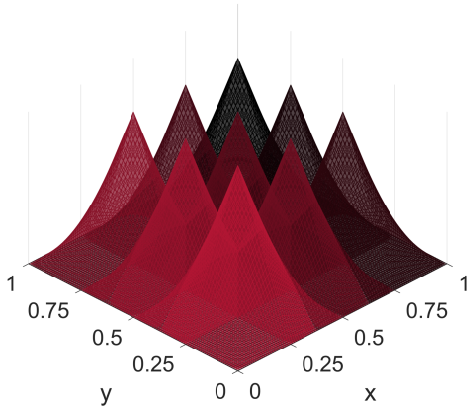


Figure 3: Tensor-product construction of d-piecewise linear basis functions

Note. Figure 3 illustrates the tensor product construction of d-piecewise linear hat functions in two dimensions according to equation (4). In particular, the figure displays the hat functions comprising the nodal basis of level $\boldsymbol{l} = (2,2)$, associated with the nodal index set $I_{\boldsymbol{l}} = \{(i_1, i_2) \mid i_k \in \mathbb{N}, 1 \leq i_k \leq 3, \text{ for } k \in \{1,2\}\}$.

error is the distance between the black-dotted line representing $f(x)$ and its interpolant $f_l(x)$.

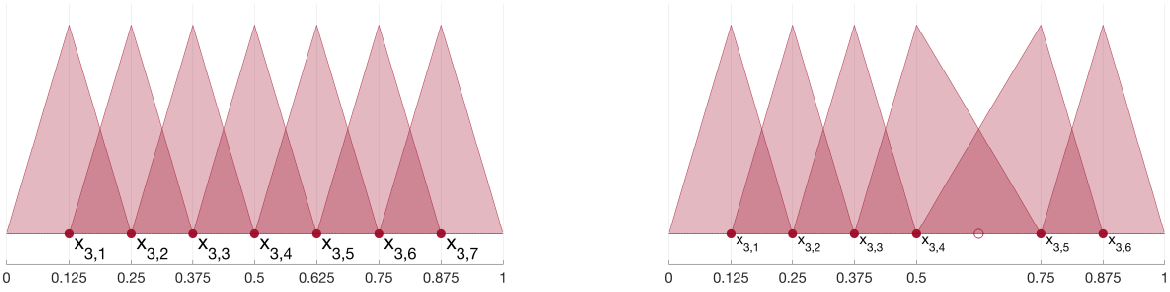
Approximation spaces. The basis of hat functions we have developed implicitly induces an approximation space of piecewise linear functions,

$$\mathcal{V}_l = \operatorname{span}\Big\{\phi_{l,i}(x) \mid i \in I_l \text{ and } x \in \Omega\Big\}. \tag{3}$$

For $l < \infty$, $\mathcal{V}_l \subset \mathcal{V}$. That is, approximating $f \in \mathcal{V}$ with its piecewise linear interpolant $f_l \in \mathcal{V}_l$ incurs approximation error. In particular, for any given $l < \infty$, $\mathcal{V}_l$ is finite-dimensional while $\mathcal{V}$ is infinite-dimensional.

Approximating f on a grid G can therefore equivalently be thought of as solving for an interpolant in an approximation space. Any grid is consequently associated with the approximation space it induces when endowed with a particular set of interpolating basis functions. Our focus in this paper is on hat functions and the resulting d-piecewise linear approximation (or interpolating) spaces. Crucially, associating the vector of function values $\alpha_{l,i} = f(x_{l,i})$ with the piecewise linear interpolant f_l now enables us to characterize the distance between f and f_l in the underlying space $\mathcal{V}$, which formalizes the approximation accuracy of grid G.

Tensor product construction in higher dimensions. It is straightforward to extend our discussion to the multi-dimensional case with $d > 1$ and $\Omega = [0,1]^d$. We introduce the multi-indices $\boldsymbol{l} = \{l_1, \ldots, l_d\}$ and $\boldsymbol{i} = \{i_1, \ldots, i_d\}$ to represent the grid point $\boldsymbol{x_{l,i}} = \{x_{l_1,i_1}, \ldots, x_{l_d,i_d}\}$ analogously to the one-dimensional case. The local mesh size is $\boldsymbol{h} = 2^{-\boldsymbol{l}}$. $I_{\boldsymbol{l}}$ is the nodal index set, with $I_{\boldsymbol{l}} = \{\boldsymbol{i} \mid i_k \in \mathbb{N}, 1 \leq i_k \leq 2^{l_k} - 1, \text{ for all } 1 \leq k \leq d\}$. A d-dimensional grid G is again a collection



(a) $l = 3$ and $I_3 = \{1, \ldots, 7\}$

(b) Remove grid point $x_{3,5}$

Figure 4: Nodal basis functions are not invariant to grid structure

Note. Figure 4 illustrates that the hat functions comprising the nodal basis associated with a grid change as the grid is adapted. Panel (a) displays the symmetric hat functions that comprise the nodal basis associated with a uniform grid of level $l = 3$. In panel (b), we remove grid point $x_{3,5}$. The nodal basis associated with the adapted grid now features two asymmetric hat functions whose respective neighbors changed.

of J grid points $\boldsymbol{x}_{l,i}$ and may be represented as a $J \times d$ matrix.

We construct d-piecewise linear hat functions centered on grid point $\boldsymbol{x}_{l,i}$ using a tensor product,

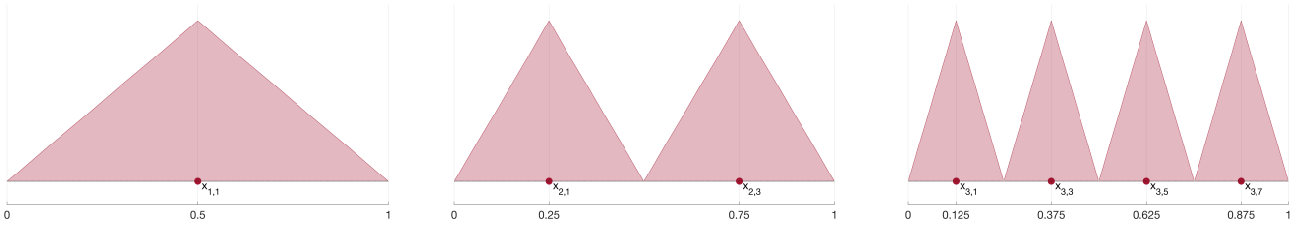
$$\phi_{l,i}(x) = \prod_{k=1}^{d} \phi_{l_k,i_k}(x_k) \tag{4}$$

where x_k is the k-coordinate of $x \in [0,1]^d$. We illustrate the tensor product construction principle in Figure 3, which displays the set of two-dimensional nodal basis functions of level $\boldsymbol{l} = (2,2)$.

1.2 The Hierarchical Basis

The nodal basis has two important drawbacks in the context of constructing sparse grids. First, it is not straightforward to characterize the marginal contribution of a grid point and its associated nodal basis function to the overall accuracy of the function approximation. Determining the value of grid points is essential in order to both characterize the overall efficiency of a grid and systematically refine a grid to make it more efficient.

Second, nodal basis functions are not invariant to the underlying grid structure, which we illustrate in Figure 4. Panel (a) again plots the hat functions that comprise the nodal basis of level $l = 3$. In panel (b), we remove the grid point $x_{3,5}$ and plot the resulting hat functions. In order to span the hole at $x = 0.625$, the two basis functions centered on 0.5 and 0.75 must be extended asymmetrically to their respective new neighbors. Since our goal is to develop an efficient procedure to systematically add grid points with large contributions and remove others, working with basis functions that change whenever the grid is locally adapted is unappealing.



(a) $l = 1$ and $I_1^{\mathrm{H}} = \{1\}$ (b) $l = 2$ and $I_2^{\mathrm{H}} = \{1, 3\}$ (c) $l = 3$ and $I_3^{\mathrm{H}} = \{1, 3, 5, 7\}$

Figure 5: Hierarchical basis functions

Note. Figure 5 illustrates the multi-level construction principle of the hierarchical basis. At each level l, the hierarchical space $\mathcal{W}_l$ is the span of those hat functions of level l that have *odd* indices; relative to the nodal basis of level l, those hat functions with even indices are dropped. For example, the basis functions active in the hierarchical space $\mathcal{W}_2$ of level $l = 2$ are $\phi_{2,1}(x)$ and $\phi_{2,3}(x)$ as depicted in panel (b); hat function $\phi_{2,2}(x)$ is dropped. The hierarchical basis of level l is then the union of the hierarchical spaces of all lower levels, i.e., $\mathcal{W}_k$ for $1 \leq k \leq l$. The hierarchical basis owes its name to this multi-level construction principle.

We now introduce the hierarchical basis and show that it addresses both concerns.[8]

Definition 1. (**Hierarchical Basis / Hierarchical Spaces**) *The hierarchical basis of level l comprises the set of piecewise linear hat functions*

$$\Big\{\phi_{k,i}(x) \mid k \leq l,\ i \in I_l^{\mathrm{H}} \text{ and } x \in \Omega\Big\}$$

where $I_l^{\mathrm{H}} = \big\{i \in \mathbb{N} \mid 1 \leq i \leq 2^l - 1 \text{ and } i \text{ odd}\big\}$ *is the hierarchical index set. We define the hierarchical space of level l as*

$$\mathcal{W}_l = \operatorname{span}\Big\{\phi_{l,i}(x) \mid i \in I_l^{\mathrm{H}} \text{ and } x \in \Omega\Big\}.$$

Figure 5 illustrates the multi-level construction principle of the hierarchical basis, to which it owes its name. At level $l = 1$, the hierarchical space $\mathcal{W}_1$ is the span of the level 1 hat function $\phi_{1,1}(x)$, just as in the nodal basis depicted in Figure 1. At level $l = 2$, however, the only basis functions active in hierarchical space $\mathcal{W}_2$ are those with odd indices, i.e., $\phi_{2,1}(x)$ and $\phi_{2,3}$. Relative to the nodal basis, we drop the hat function $\phi_{2,2}(x)$. Similarly, at level $l = 3$, the hierarchical space $\mathcal{W}_3$ is spanned by the basis functions of odd indices, namely $\{\phi_{3,i}(x)\}$ for $i \in I_3^{\mathrm{H}} = \{1, 3, 5, 7\}$. The hierarchical basis of level $l = 3$ is then obtained by combining the hierarchical basis functions of all lower levels — in other words, by taking the union of the hierarchical spaces $\mathcal{W}_k$ for $k = 1, 2, 3$.

We want to make several important remarks. First, the nodal and hierarchical bases span the same function spaces. Formally, we have $\mathcal{V}_l = \bigoplus_{k \leq l} \mathcal{W}_k$. That is, any function in the span of the

[8] The hierarchical basis will also be at the heart of constructing a consistent finite-difference scheme on sparse grids, as we show in Section 3. See for example Zenger (1991), Griebel (1998) and Bungartz and Griebel (2004), as well as more recently Brumm and Scheidegger (2017) and Ruttscheidt (2018).

nodal basis is also in the span of the hierarchical basis. This observation allows us to alternatively use and switch between the nodal and hierarchical representations of functions.

Second, the hierarchical spaces $\mathcal{W}_l$ are disjoint. Suppose we take the union of two disjoint hierarchical spaces, say $\tilde{\mathcal{V}} = \mathcal{W}_1 \bigoplus \mathcal{W}_2$. The resulting function space $\tilde{\mathcal{V}}$ is spanned by the set of basis functions that span each of the component spaces. That is, $\tilde{\mathcal{V}} = \text{span}\{\phi_{1,1}(x), \phi_{2,1}(x), \phi_{2,3}(x)\}$. The function spaces spanned by nodal basis functions are not disjoint and, as a result, the nodal basis functions are not invariant to the underlying grid structure. Thanks to its multi-level construction, the hierarchical basis does not suffer from this drawback. In particular, if we start with a grid G of level $l = 3$ and remove one of the nodes associated with the highest level, e.g., $x_{3,5}$ as in Figure 4, the remaining hierarchical basis functions would not be affected.

Third, the multi-dimensional case with $d > 1$ again obtains by using tensor product constructions as before.

Fourth, consider a grid G and the function values $\alpha_{l,i} = f(x_{l,i})$. We can characterize piecewise linear interpolants on this grid, as well as the associated approximation spaces, by using either the nodal or the hierarchical basis. Both sets of basis functions induce the same approximation space of piecewise linear interpolants. Formally, for any piecewise linear function $f_l \in \mathcal{V}_l$ we also have $f_l \in \bigoplus_{k \leq l} \mathcal{W}_k$, and its representation in the hierarchical basis is given by

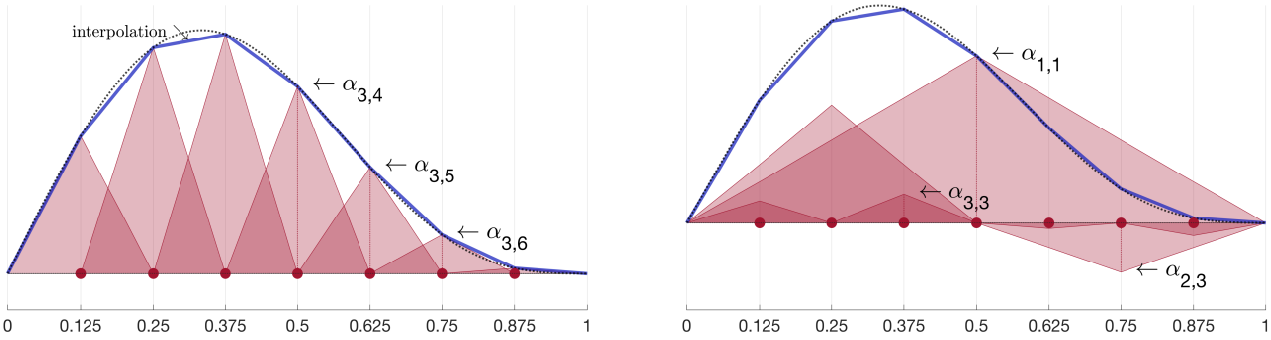
$$f_l(x) \quad = \underbrace{\sum_{i \in I_l} \alpha_{l,i}\, \phi_{l,i}(x)}_{\text{nodal representation}} \quad = \underbrace{\sum_{k \leq l} \sum_{i \in I_k^{\mathrm{H}}} \alpha_{k,i}^{\mathrm{H}}\, \phi_{k,i}(x)}_{\text{hierarchical representation}} \tag{5}$$

where $\alpha_{k,i}^{\mathrm{H}}$ are called the *hierarchical surpluses*. We graphically compare the nodal and hierarchical basis function representations in Figure 6, which illustrates that both bases span the same approximation space of piecewise linear interpolants.

Hierarchization. According to equation (5) and as illustrated in Figure 6, the interpolant f_l can be represented using nodal or hierarchical basis functions. In the nodal representation, we take as basis function coefficients simply the function values $\alpha_{l,i} = f(x_{l,i})$. Having shown that the hierarchical basis can be constructed using a particular reordering of nodal basis functions, we now explain how to obtain the hierarchical surpluses $\alpha_{l,i}^{\mathrm{H}}$ from the function values $\alpha_{l,i}$. In particular, we characterize projection operators that correspond to basis transformations between the nodal and hierarchical bases.[9] The resulting basis transformation operators will play a crucial role in our discussion of continuous-time dynamic programming on sparse grids in Section 3.

Intuitively, the function values $\alpha_{l,i}$ simply fit the function at the grid point on which the basis function is centered, i.e., $\alpha_{l,i} = f(x_{l,i})$. In Panel (a) of Figure 6, we see that the hat functions just touch the black-dotted line that represents $f(x)$. The hierarchical representation depicted in panel

[9] These projection operators have been previously discussed by Griebel (1998), Schiekofer (1998), Bungartz and Griebel (2004) and Ruttscheidt (2018) among others. We adopt the same notation as in Schiekofer (1998).



(a) Interpolation in the nodal basis

(b) Interpolation in the hierarchical basis

Figure 6: Nodal and hierarchical function approximations

Note. Figure 6 illustrates the nodal and hierarchical basis function representations of the piecewise linear interpolant $f_l(x)$ (*solid blue line*) that approximates $f(x) = \frac{4}{5}(\sin(\pi x) + \frac{1}{2}\sin(2\pi x))$ (*dashed black line*) on a full grid of level $l = 3$, $G = \{x_{3,i} \mid i \in I_3\}$. Panel (a) recalls the nodal representation from Figure 2 for convenience. Panel (b) plots the hierarchical basis functions, each weighted by the associated *hierarchical surplus*, as well as the linear interpolant $f_l(x)$ (*solid blue line*). The two interpolants in panels (a) and (b) are identical, which signifies that the nodal and hierarchical bases span the same approximation space.

(b), on the other hand, follows a multi-level construction. We first fit the function on level 1 by setting $\alpha^{\mathrm{H}}_{1,1} = f(x_{1,1}) = f(0.5)$. On level l and index i, we now fit the *residual*. That is, instead of setting $\alpha^{\mathrm{H}}_{l,i} = f(x_{l,i})$, we set $\alpha^{\mathrm{H}}_{l,i} = f(x_{l,i}) - \sum_{k\le l-1}\sum_{i\in I^{\mathrm{H}}_k} \alpha^{\mathrm{H}}_{k,i}\,\phi_{k,i}(x)$. For example, at the grid point of level $l = 2$ index $i = 1$, i.e., at $x_{2,1} = 0.25$, we set $\alpha^{\mathrm{H}}_{2,1} = f(x_{2,1}) - \alpha^{\mathrm{H}}_{1,1}\,\phi_{1,1}(x)$.

Due to this multi-level construction principle, we can always represent the hierarchical surplus at a grid point in terms of its hierarchical parents, i.e., those grid points closest to it at the previous level. We denote by $\mathcal{H}^{\pm}_1(x_{l,i})$ the *left* and *right parent* of $x_{l,i}$ in dimension 1. In particular, without grid point $x_{l,i}$ the local function approximation in the hierarchical representation is simply a linear interpolation between the two parents of $x_{l,i}$, i.e., $\frac{1}{2}f(\mathcal{H}^{-}_1(x_{l,i})) + \frac{1}{2}f(\mathcal{H}^{+}_1(x_{l,i}))$. So we find the hierarchical surplus as the residual by subtracting the interpolated value from the local function value

$$\alpha^{\mathrm{H}}_{l,i} = f(x_{l,i}) - \frac{1}{2}f(\mathcal{H}^{l}_1(x_{l,i})) - \frac{1}{2}f(\mathcal{H}^{+}_1(x_{l,i})).$$

We call the linear map that transforms the function value at grid point $x_{l,i}$ into the associated hierarchical surplus the *hierarchization operator*.

Definition 2. (**Hierarchization**) *The one-dimensional hierarchization operator in dimension k, denoted $H_{x_{l_k,i_k},h_k} : \mathcal{V}_{l_k} \to \bigoplus_{j_k\le l_k} \mathcal{W}_{j_k}$, is defined via the stencil $[-\frac{1}{2} \quad 1 \quad -\frac{1}{2}]_{h_k}$, applied at point x_{l_k,i_k} with step size h_k, where $h_k = \mathcal{H}^{+}_k(x_{l,i}) - x_{l,i}$ denotes the distance to the hierarchical parent in dimension k. That is, for $f : \mathbb{R} \to \mathbb{R}$,*

$$H_{x_{l,i},h} \circ f(x_{l,i}) = f(x_{l,i}) - \frac{f(x_{l,i}-h) + f(x_{l,i}+h)}{2} \tag{6}$$

for $1 \leq i \leq 2^l - 1$. *And for* $f : \mathbb{R}^d \to \mathbb{R}$,

$$H_{x_{l_k,i_k},h_k} \circ f(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}}) = f(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}}) - \frac{f(x_{l_1,i_1}, \ldots, x_{l_k,i_i} - h_k, \ldots, x_{l_d,i_d}) + f(x_{l_1,i_1}, \ldots, x_{l_k,i_i} + h_k, \ldots, x_{l_d,i_d})}{2}. \quad (7)$$

The d-dimensional hierarchization operator is defined by the tensor product

$$H_{\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}},\boldsymbol{h}} = \prod_{k \leq d} H_{x_{l_k,i_k},h_k}.$$

It can easily be verified that the hierarchization operator represents a basis transformation from the nodal into the hierarchical basis. Formally, we have $\alpha^{\mathrm{H}}_{\boldsymbol{l},\boldsymbol{i}} = H_{\boldsymbol{x}_{l,i},\boldsymbol{h}} \circ \alpha_{\boldsymbol{l},\boldsymbol{i}}$. See Bungartz (1992) for a formal proof and further discussion.

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

$$H = \prod_{k \leq d} H^k$$

denote the d-dimensional hierarchization operator that performs the basis transformation in all dimensions. Finally, let H_k denote the operator that performs hierarchization in all *but* dimension k. We also introduce the inverse operators that map from the hierarchical to the nodal basis and are given by $E = H^{-1}$, $E^k = (H^k)^{-1}$ and $E_k = H_k^{-1}$. We refer to these as *de-hierarchization* operators. In the following, we denote the matrices that discretize these operators on a given grid using bold-faced notation, i.e., $\boldsymbol{H}$ and $\boldsymbol{E}$.

Taking stock, the hierarchization operators allow us to switch between nodal and hierarchical basis representations, if necessary on a dimension-by-dimension basis. These projection matrices are the key ingredient in the generalized finite-difference scheme we introduce below for continuous-time dynamic programming on sparse grids.

Benefits of the hierarchical basis. We have motivated the hierarchical basis and its multi-level construction by observing that its basis functions at level l are independent from all its parent basis functions. Indeed, the hierarchical spaces themselves are disjoint.

An even more instrumental property of hierarchical basis representations is that hierarchical surpluses directly measure the contribution of basis functions to the accuracy of the local function approximation. By construction, the hierarchical surplus $\alpha^{\mathrm{H}}_{l,i}$ tells us how much grid point $x_{l,i}$ contributes to the local function approximation *residually*, relative to the interpolant that would obtain in the absence of this grid point. Hierarchical surpluses therefore play a key role in the construction of adaptive sparse grids, where adaptive grid refinement directly leverages the hierarchical surpluses as measures of residual local approximation error.

As a result, hierarchical surpluses decay with their level. This property is intuitive in the context of a multi-level construction whereby the hierarchical surplus at a given level l is used to fit the residual approximation error from lower levels. The following Lemma, originally due to Bungartz (1992), characterizes an estimate or bound on hierarchical surpluses of level l.

Lemma 1. (**Estimate of Hierarchical Surplus**) *Let $f : \Omega \to \mathbb{R}$ be a function that vanishes on the boundary $\partial\Omega$ and has bounded mixed derivatives up to order two. The hierarchical surpluses $\alpha^H_{\boldsymbol{l},\boldsymbol{i}}$ associated with the linear interpolant of f are bounded by*

$$\left| \alpha^H_{\boldsymbol{l},\boldsymbol{i}} \right| \leq C \cdot 2^{-d} \cdot 2^{-2|\boldsymbol{l}|_1}$$

for a constant C that does not depend on $\boldsymbol{l}$.

In conclusion, the hierarchical surplus $\alpha^{\mathrm{H}}_{l,i}$ directly encodes the marginal contribution of a grid point $x_{l,i}$ to the accuracy of our local function approximation. And for functions that are sufficiently smooth, hierarchical surpluses $\alpha^{\mathrm{H}}_{l,i}$ are guaranteed to decay in their level l. We exploit this property in Section 1.3 to construct grids adaptively by dropping grid points with low hierarchical surpluses and adding more grid points where hierarchical surpluses remain large.

1.3 Sparse Grids as Optimal Approximation Spaces

Having shown how to associate any grid G with the piecewise linear approximation space it induces, we now use this framework to study the efficiency properties of three classes of grids, which we formally introduce and define below: uniform grids, regular sparse grids, and adaptive sparse grids. We associate the cost of a grid with the dimensionality of the approximation space it induces. Similarly, we define the value of a grid point as its marginal contribution to the accuracy of the function approximation. Formally, we compare the projection of $f \in \mathcal{V}$ onto two different approximation spaces and then characterize the respective distances from f in a desired norm on $\mathcal{V}$. Indeed, Zenger (1991) first introduced sparse grids and showed that they emerge from an optimization problem over the active hierarchical spaces in a function representation.

Uniform grids. A uniform grid of level $\boldsymbol{l}$ is associated with the approximation space

$$v \in \mathcal{V}_{\boldsymbol{l}}^{\mathrm{ug}} = \bigoplus_{|\boldsymbol{k}| \leq \boldsymbol{l}} \mathcal{W}_{\boldsymbol{k}} \implies v(x) = \sum_{\boldsymbol{i} \in I_{\boldsymbol{l}}} \alpha_{\boldsymbol{l},\boldsymbol{i}}\, \phi_{\boldsymbol{l},\boldsymbol{i}}(x)\,, \quad \text{where } \alpha_{\boldsymbol{l},\boldsymbol{i}} = v(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}}).$$

That is, the uniform grid corresponds to the usual space of piecewise d-linear functions on grids with equidistant mesh size $\boldsymbol{h} = 2^{-\boldsymbol{l}}$.

To formalize the efficiency properties of $\mathcal{V}_{\boldsymbol{l}}^{\mathrm{ug}}$, it will be convenient in the main text to focus

on uniform grids with the same mesh size in each dimension.[10] That is, we set $\boldsymbol{l} = (l, \dots, l)$ and $h = 2^{-l}$. We can then show that the dimensionality, i.e., cost, of a uniform grid is

$$\left| \mathcal{V}_l^{\text{ug}} \right| = (2^l - 1)^d = \mathcal{O}(2^{d\,l}) = \mathcal{O}(h^{-d}). \tag{8}$$

Similarly, the accuracy of function approximations on uniform grids for functions of bounded mixed derivatives up to order two is given by

$$\|f - f_l^{\text{ug}}\|_\infty \leq C \cdot \frac{d}{6^d} \cdot 2^{-2l} = \mathcal{O}(h^2), \tag{9}$$

for some constant C that is independent from h. Uniform grids suffer from the curse of dimensionality as the number of grid points required to achieve accuracy $\mathcal{O}(h^2)$ grows exponentially according to equation (8).

Regular sparse grids. We now ask whether we can construct an optimal approximation space for a given class of functions. We follow the exposition in Bungartz and Griebel (2004). In particular, we can systematically construct finite-dimensional approximation spaces $\mathcal{U} \subset \mathcal{V}$ via the *subspace selection* index $\boldsymbol{I} \subset \mathbb{N}^d$ and set

$$\mathcal{U} = \bigoplus_{\boldsymbol{l} \in \boldsymbol{I}} \mathcal{W}_{\boldsymbol{l}}, \quad \text{with } f_{\mathcal{U}} = \sum_{\boldsymbol{l} \in \mathcal{I}} f_{\boldsymbol{l}}, \;\; f_{\boldsymbol{l}} \in \mathcal{W}_{\boldsymbol{l}}.$$

That is, the selection index $\boldsymbol{I}$ determines the active grid points or basis functions in the span of our desired approximation space $\mathcal{U}$.

Our goal now is to find the most efficient space $\mathcal{U}$ in terms of some norm $\|f - f_{\mathcal{U}}\| = \|\sum_{\boldsymbol{l}} f_{\boldsymbol{l}} - \sum_{\boldsymbol{l} \in \boldsymbol{I}} f_{\boldsymbol{l}}\| \leq \sum_{\boldsymbol{l} \notin \boldsymbol{I}} \|f_{\boldsymbol{l}}\|$. That is, we look for some optimal finite index set $\boldsymbol{I} \subset \mathbb{N}^d$ and characterize the resulting approximation error $\sum_{\boldsymbol{l} \notin \boldsymbol{I}} \|f_{\boldsymbol{l}}\|$. Selecting a particular $\mathcal{W}_{\boldsymbol{l}}$ to be included in the approximation space is equivalent to selecting the associated nodes to be included in the associated grid.

Regular sparse grids emerge from an *a priori* optimization problem that considers a broad class of functions f, while adaptive sparse grids, which we study next, emerge from problem-dependent optimization.

While different norms $\|\cdot\|$ lead to differently structured sparse grids, we define regular sparse grids as those that obtain under the L^2 norm (Zenger, 1991; Bungartz, 1992). Regular sparse grids then induce the approximation space

$$\mathcal{V}_n^{\text{rsg}} = \bigoplus_{|\boldsymbol{l}|_1 \leq n+d-1} \mathcal{W}_{\boldsymbol{l}} \tag{10}$$

We illustrate the construction principle of regular sparse grids in Figure 7. The cost of regular

[10] These grids are sometimes call isotropic grids.

(a) One-dimensional grids

(b) Tensor product construction of regular sparse grid

Figure 7: Constructing regular sparse grids

Note. Figure 7 illustrates the construction principle of regular sparse grids in terms of a restricted tensor product. Panel (a) shows the elementary grids of levels up to $\boldsymbol{l} \leq (3,3)$. A tensor product (or union) of all elementary grids would result in the uniform grid of level $(3,3)$ and the associated approximation space $\mathcal{V}_{\boldsymbol{l}}^{\text{ug}}$. Panel (b) displays the analogous regular sparse grid, which is the tensor product of all elementary grids of level $|\boldsymbol{k}|_1 \leq 3 + d - 1 = 4$. Concretely, the three elementary grids that are transparent in panel (a) are dropped relative to the full tensor product that results in the uniform grid. These three grids are associated with the levels $(3,2)$, $(2,3)$, and $(3,3)$, whose $|\cdot|_1$ norms are, respectively, 5, 5, and 6.

sparse grids, i.e., the dimensionality of the regular sparse grid approximation space, can be shown to be of order

$$\left| \mathcal{V}_n^{\text{rsg}} \right| = \mathcal{O}(2^n \, n^{d-1})$$

which is substantially smaller than the cost of a uniform grid $\mathcal{O}(2^{n\,d})$. At the same time, the accuracy of the regular sparse grid approximation space is only slightly worse than that of the uniform grid approximation space. See Bungartz (1992) and Bungartz and Griebel (2004) for additional details.

Adaptive sparse grids. Regular sparse grids are efficient under the L^2 norm for the class of functions of bounded mixed derivatives up to order two. They consequently represent a much better starting point for most economic applications than uniform grids. However, allowing the grid optimization problem to condition on additional information from the economic application in question can lead to further substantial efficiency gains. The resulting grids are called adaptive sparse grids.

Hierarchical surpluses directly measure the local residual error of a given function interpolation in a piecewise linear approximation space. In particular, the hierarchical surplus $\alpha_{l,i}^{\text{H}}$ associated with grid point $\boldsymbol{x}_{l,i}$ tells us the marginal contribution of this node to a piecewise linear approximation space. Crucially, it is easy to compute hierarchical surpluses. Given a vector of function values $\boldsymbol{\alpha} = \{\alpha_{l,i}\}_{l,i}$, we obtain the associated hierarchical surpluses by applying the hierachization matrix, $\boldsymbol{\alpha}^{\text{H}} = \{\alpha_{l,i}^{\text{H}}\}_{l,i} = \boldsymbol{H}\boldsymbol{\alpha}$. We can then adaptively refine our grid and the associated approximation

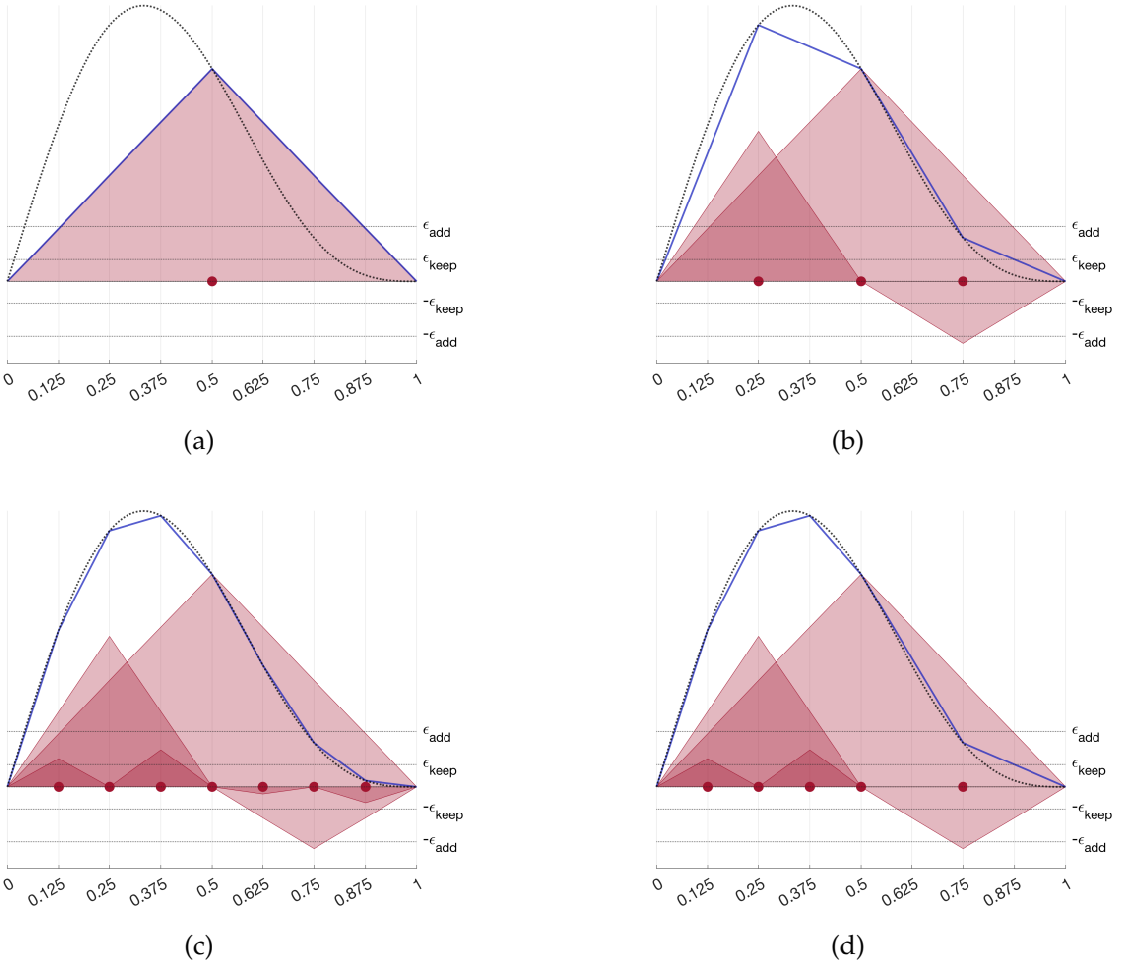


Figure 8: Local grid adaptation

Note. The four panels of Figure 8 illustrate a complete grid adaptation procedure to approximate the function $f(x) = \frac{4}{5}(\sin(\pi x) + \frac{1}{2}\sin(2\pi x))$ (*dashed black line*) on the interval $\Omega = [0, 1]$. Each panel also plots the adaptation thresholds ϵ^{add} and ϵ^{keep}. The red-tinted triangles correspond to the hierarchical basis functions, which are weighted by the associated hierarchical surpluses $\alpha^{\text{H}}_{l,i}$. In each refinement step, add the children of nodes whose hierarchical surpluses are larger than ϵ^{add} in absolute value and remove those whose hierarchical surpluses are smaller than ϵ^{keep}. The adaptation procedure converges in four steps and the resulting adaptive sparse grid is displayed in panel (d).

space via the following procedure:

$$\text{add the children of node } \boldsymbol{x}_{l,i} \text{ if:} \quad |\alpha^{\mathrm{H}}_{l,i}| > \epsilon^{\text{add}} \tag{11}$$

$$\text{keep node } \boldsymbol{x}_{l,i} \text{ if:} \quad |\alpha^{\mathrm{H}}_{l,i}| > \epsilon^{\text{keep}}, \tag{12}$$

for some $0 < \epsilon^{\text{keep}} < \epsilon^{\text{add}}$

We illustrate this grid adaptation procedure in Figure 8 for the function $f(x) = \frac{4}{5}(\sin(\pi x) + \frac{1}{2}\sin(2\pi x))$, plotted as a dashed-black line, on the interval $\Omega = [0, 1]$. Panels (a) through (d) present four successive steps of grid adaptation. Each panel also plots the adaptation thresholds ϵ^{keep} and ϵ^{add} as horizontal lines. Panel (a) plots the hierarchical basis function of level 1, $\phi_{1,1}(x)$, centered on the grid point $x_{1,1} = 0.5$. We obtain the hierarchical surplus $\alpha^{\mathrm{H}}_{1,1}$ by fitting the function, i.e., $\alpha^{\mathrm{H}}_{1,1} = f(x_{1,1}) = f(0.5)$. Since the hierarchical surplus is larger than the threshold ϵ^{add}, we adapt the grid around node $x_{1,1}$ by adding its children, $x_{2,1} = 0.25$ and $x_{2,3} = 0.75$ in panel (b). The associated hierarchical surpluses $\alpha^{\mathrm{H}}_{2,1}$ and $\alpha^{\mathrm{H}}_{2,3}$ are obtained by fitting the function $f(x)$ *residually*, i.e., by fitting the distance between the dashed black and solid blue lines in panel (a). In each case, the resulting hierarchical surplus is again larger than the threshold ϵ^{add} in absolute value, and so we add each node's children in panel (c). Panel (c) illustrates that, for sufficiently smooth functions, hierarchical surpluses become smaller at higher levels. In particular, the hierarchical surpluses $\alpha^{\mathrm{H}}_{3,1}$ and $\alpha^{\mathrm{H}}_{3,3}$ are larger than ϵ^{keep}, so we retain these nodes in the grid, but they are smaller than ϵ^{add}, so we do not refine the grid further. The hierarchical surpluses $\alpha^{\mathrm{H}}_{3,5}$ and $\alpha^{\mathrm{H}}_{3,7}$, on the other hand, are smaller than ϵ^{keep} in absolute value, signifying that the associated nodes are not required to fit $f(x)$ to the desired accuracy. Panel (d) drops these two nodes. The grid adaptation procedure has converged because all remaining nodes feature hierarchical surpluses larger than ϵ^{keep}, and no additional nodes need to be added.

2 Dynamic Programming on Uniform Grids

In this section, we review the theory of stochastic dynamic programming in continuous time and its numerical application on uniform grids. In the interest of broad accessibility, we frame the exposition in terms of the well-known neoclassical stochastic growth model. After a brief description of the model in Section 2.1, we state the Principle of Optimality and prove the convergence of a homogenized representation as a sequence of linear equations in Section 2.2. In Section 2.3, we prove the convergence properties of a popular numerical method on uniform grids. By reviewing this well-understood approach, we set the stage for the introduction of sparse grids in Section 3, where we then show that the standard approach fails.

2.1 The Neoclassical Stochastic Growth Model

Our notation follows closely that of Stokey and Lucas (1989), except that we write the model in continuous time, with $t \in [0, \infty)$. The economy is populated by a representative household that operates the economy's production technology $F(k_t, z_t)$, where k_t denotes capital and z_t exogenous productivity risk. We assume the production function to be well-behaved, with $F(0, z) = 0$, $F_k, F_z > 0$, and $F_{kk} < 0$. The economy's initial state at time $t = 0$ is given by (k_0, z_0).

The representative household derives utility from consumption and faces the infinite-horizon optimization problem

$$V(k_0, z_0) = \max_{\{c_t\}_{t\geq 0}} \mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t) dt, \tag{13}$$

where ρ is the discount rate, $u(\cdot)$ denotes instantaneous flow utility, and $\{c_t\}_{t\geq 0}$ is the stochastic consumption process. $\mathbb{E}_0$ denotes the expectation over future productivity realizations.[11] We assume labor supply to be inelastic and require that consumption be non-negative, $c_t \geq 0$ for all t. The economy's stock of capital, owned by the household, evolves according to

$$\frac{d}{dt} k_t = F(k_t, z_t) - c_t - \delta k_t. \tag{14}$$

Finally, productivity follows the mean-reverting diffusion process

$$dz_t = -\theta z_t dt + \sigma dB_t, \tag{15}$$

where B_t is a standard Brownian motion.

To solve the neoclassical growth model means to solve program (13) subject to (14) and (15) for the value function $V(k, z)$ over a compact region $\bar{\mathcal{X}} \subset \mathbb{R}^2$. We call $\bar{\mathcal{X}}$ the state space and, for the neoclassical growth model, take it to be given by $\bar{\mathcal{X}} = \{(k, z) \mid k \in [0, \bar{k}] \text{ and } z \in [\underline{z}, \bar{z}]\}$, for $\bar{k}, \bar{z} \leq \infty$ and $\underline{z} \geq -\infty$. We now state the continuous-time analog of the classical Principle of Optimality (see, e.g., Stokey and Lucas, 1989, p. 241-259).

Lemma 2. (**Principle of Optimality**) *The value function defined in equation (13) solves a partial differential equation of the Hamilton-Jacobi-Bellman (HJB) form, given by*

$$\rho V(k, z) = \max_c \left\{ u(c) + \Big(F(k, z) - c - \delta k\Big) V_k(k, z) - \theta z V_z(k, z) + \frac{\sigma^2}{2} V_{zz}(k, z) \right\}, \tag{16}$$

for all $(k, z) \in \mathcal{X}$.

Lemma 2 adopts a shorthand notation for partial derivatives, using $V_k = \frac{\partial}{\partial k} V(k, z)$, $V_z = \frac{\partial}{\partial z} V(k, z)$,

[11] Our exposition in the main text is kept deliberately light on the formal details of the measure theoretic assumptions that underlie this dynamic stochastic optimization problem. We present a more formal account in Appendix F, as well as in our proofs in Appendix B.

and $V_{zz} = \frac{\partial^2}{\partial z^2} V(k,z)$. Equation (16) is known as a Hamilton-Jacobi-Bellman (HJB) equation and takes the form of a parabolic partial differential equation (PDE). Formally, it is only defined on the interior of the state space, which we denote $\mathcal{X}$, and a set of boundary conditions is required to characterize $V(k,z)$ along the boundary, which we denote $\partial\mathcal{X}$. We postpone a formal discussion of boundary conditions until Section 4. In Appendix B.1, we present both a heuristic proof of Lemma 2, which many readers will be familiar with, as well as several formal proofs.

From equation (16), the consumption policy function $c(k,z)$ satisfies the first-order optimality condition

$$u'(c(k,z)) = V_k(k,z). \tag{17}$$

Plugging the optimal policy back into equation (16), the HJB can be rewritten without the max operator as

$$\rho V(k,z) = u\Big(c(k,z)\Big) + \Big(F(z,k) - c(k,z) - \delta k\Big) V_k(k,z) - \theta z V_z(k,z) + \frac{\sigma^2}{2} V_{zz}(k,z), \tag{18}$$

where the optimal policy function $c(k,z)$ is given by equation (17) as a function of $V(k,z)$. Solving the partial differential equation (18) for the value function $V(k,z)$ over the state space therefore represents a solution of the dynamic optimization problem (13) - (15) that constitutes the neoclassical growth model.

2.2 Homogenization via Time-Marching

Solving equation (18) directly would require resolving the optimal consumption policy, $c(k,z) = (u')^{-1}[V_k(k,z)]$. Plugging back into (18) yields a highly non-linear partial differential equation.[12] Non-linear partial differential equations are difficult to solve in practice. However, there is an alternative approach that can be used to transform the fully non-linear HJB equation into a system of linear equations via *homogenization*.

In particular, consider a sequence of functions $\{V^n\}$, with $V^n : \bar{\mathcal{X}} \to \mathbb{R}$, that are recursively defined by

$$\frac{V^{n+1} - V^n}{\Delta t} + \rho V^{n+1} = u\Big(c^n\Big) + \Big(F - c^n - \delta k\Big) V_k^{n+1} - \theta z V_z^{n+1} + \frac{\sigma^2}{2} V_{zz}^{n+1}, \tag{19}$$

where, for notational convenience, we suppress the dependence of V^n, c^n and F on the state variables. Crucially, the consumption policy is resolved using $c^n = (u')^{-1} V_k^n$, which implies that, given V^n, equation (19) is now linear in V^{n+1}. In the applied math literature, equation (19) is often referred to as a (semi-implicit) *time-marching* scheme, a popular method to solve time-homogeneous,

[12] Formally, the resulting HJB of the neoclassical growth model is quasi-linear because the terms corresponding to the highest-order derivatives, here V_{zz}, remain linear. Consider, alternatively, a portfolio choice problem such as Merton (1973), where the risky asset portfolio share multiplies the second-order term. The associated HJB would then be fully non-linear. The homogenization approach we take here is nontheless very useful in both cases.

i.e., stationary, non-linear differential equations (see, e.g., LeVeque, 2007, for a textbook treatment). Solving recursively for the sequence of functions $\{V^n\}$ requires an initial condition V^0.

Proposition 3. (**Contraction mapping**) *The sequence of functions* $\{V^n\}$ *characterized by* (19) *converges uniformly with* $V^n \to \bar{V}$, *and we have* $\bar{V} = V$ *so that the iterative scheme* (19) *converges to the stationary value function defined by* (18).

Proposition 3 establishes that the homogenized time-marching scheme (19) represents a convergent contraction mapping, whose fixed point is the unique solution of the neoclassical growth model. In other words, Proposition 3 is the continuous-time analog of the result that applying the functional Bellman operator in discrete time yields a convergent contraction mapping. (See, e.g., Stokey and Lucas, 1989, Theorem 4.6.) Crucially, the time-marching scheme (19) requires solving a linear equation at every iteration n, rather than the fully non-linear equation (18). The proof of Proposition 3 is in Appendix B.2.

Dynamic programming problems in economics seldom yield an analytical closed-form solution. In the remainder of this section, we therefore discuss a popular class of numerical methods to solve equation (19) recursively for $\{V^n\}$. In particular, this paper focuses on *finite-difference methods* for partial differential equations.[13] In Section 2.3, we discuss a well-known finite-difference scheme, with which to solve equation (19) on uniform grids, and we prove its convergence properties in Section 2.4. We thereby set the stage to introduce sparse grids in Section 3, where we first show that the standard finite-difference scheme fails and then develop a generalized finite-difference method to solve partial differential equations on non-uniform grids.

2.3 Finite Differences on Uniform Grids

Our goal in what follows is to develop a finite-difference method to solve equation (19) numerically for the sequence $V^n(k,z)$ and show that it converges to the solution of the neoclassical growth model, with $V^n(k,z) \to V(k,z)$.

To solve equation (19) numerically, we introduce a grid, a collection of points that discretize the state space. The state space of the neoclassical growth model is given by the compact set $\bar{\mathcal{X}} = [0,\bar{k}] \times [\underline{z},\bar{z}] \subset \mathbb{R}^2$. We also introduce its *torus*, which corresponds to the interior of the state space and is defined as $\mathcal{X} = (0,\bar{k}) \times (\underline{z},\bar{z})$. Finally, we denote the *boundary* of the state space by $\partial\bar{\mathcal{X}} = \bar{\mathcal{X}} \setminus \mathcal{X}$. For the remainder of Sections 2 and 3, we will work directly with $\mathcal{X}$, where equation (19) is well-defined. We postpone a formal treatment of boundary conditions until Section 4.

Let $G \subset \bar{\mathcal{X}}$ be a uniform grid of level $\boldsymbol{l} = (l,l)$, with mesh size $\boldsymbol{h} = 2^{-\boldsymbol{l}} = (h,h)$, containing J grid points, $\boldsymbol{x}_{l,i}$.[14] Our goal is to compute the $J \times 1$ vector of function values $\boldsymbol{V}^n = \{V^n_{l,i}\}_{l,i}$ on G

[13] Alternative methods that we do not explore in this paper include the (Galerkin) finite element method and spectral methods. See, e.g., Bungartz (1992).

[14] For consistency, we stay in the framework developed in Section 1 based on the dyadic grid construction principle.

to approximate the function V^n defined in equation (19), i.e., $V^n_{\boldsymbol{l},\boldsymbol{i}} \approx V^n(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}})$. We consistently use bold-faced notation to denote vectors and matrices.

Using matrix notation, the discretization of equation (19) on grid G can thus be written as

$$\frac{1}{\Delta t}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n) + \rho \boldsymbol{V}^{n+1} = u\Big(\boldsymbol{c}^n\Big) + \Big(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\Big)\boldsymbol{V}^{n+1}_k - \theta \boldsymbol{z}\boldsymbol{V}^{n+1}_z + \frac{\sigma^2}{2}\boldsymbol{V}^{n+1}_{zz}, \tag{20}$$

where products are elementwise. The discretized first-order condition $u'(\boldsymbol{c}^n) = \boldsymbol{V}^n_k$ defines the consumption policy. What remains to be specified in representation (20) is, of course, how to compute the discretized partial derivatives, $\boldsymbol{V}^{n+1}_k$ and so on, which is the key step of the finite-difference method. Indeed, our objective can now be formulated as follows: For which discretizations of $\boldsymbol{V}^{n+1}_k$ and so on does the sequence $\{\boldsymbol{V}^n\}$ derived from scheme (20) converge, i.e., $\boldsymbol{V}^n \to \boldsymbol{V} = \{V_{\boldsymbol{l},\boldsymbol{i}}\}$, and yield a good approximation of the true value function, i.e., $V_{\boldsymbol{l},\boldsymbol{i}} \approx V(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}})$?

Since derivatives are linear maps, their discretizations can be represented by matrices, i.e., $\boldsymbol{V}^{n+1}_k = \boldsymbol{D}_k \boldsymbol{V}^{n+1}$, $\boldsymbol{V}^{n+1}_z = \boldsymbol{D}_z \boldsymbol{V}^{n+1}$, and $\boldsymbol{V}^{n+1}_{zz} = \boldsymbol{D}_{zz}\boldsymbol{V}^{n+1}$, for some $J \times J$ matrices $\boldsymbol{D}_k$, $\boldsymbol{D}_z$, and $\boldsymbol{D}_{zz}$ that we refer to as *finite-difference matrices*. Plugging back into equation (20), we obtain the numerical finite-difference scheme

$$\frac{1}{\Delta t}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n) + \rho \boldsymbol{V}^{n+1} = u\Big(\boldsymbol{c}^n\Big) + \Big(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\Big)\boldsymbol{D}_k\boldsymbol{V}^{n+1} - \theta \boldsymbol{z}\boldsymbol{D}_z\boldsymbol{V}^{n+1} + \frac{\sigma^2}{2}\boldsymbol{D}_{zz}\boldsymbol{V}^{n+1},$$

which admits a more compact representation given by

$$\Big(\rho + \frac{1}{\Delta t} - \boldsymbol{A}^n\Big)\boldsymbol{V}^{n+1} = u\Big(\boldsymbol{c}^n\Big) + \frac{1}{\Delta t}\boldsymbol{V}^n, \tag{21}$$

where $\boldsymbol{A}^n = (\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k})\boldsymbol{D}_k - \theta \boldsymbol{z}\boldsymbol{D}_z + \frac{\sigma^2}{2}\boldsymbol{D}_{zz}$. This representation of the finite-difference scheme underscores that a linear system of equations must be solved at each step n. That is, $\boldsymbol{V}^{n+1} = (\rho + \frac{1}{\Delta t} - \boldsymbol{A}^n)^{-1}\big[u(\boldsymbol{c}^n) + \frac{1}{\Delta t}\boldsymbol{V}^n\big]$. For given matrices $\boldsymbol{D}_k$, $\boldsymbol{D}_z$, and $\boldsymbol{D}_{zz}$, we say that equation (21) defines a numerical finite-difference scheme. This recursive scheme represents the continuous-time analog of iteratively applying the Bellman operator on discretized vectors $\boldsymbol{V}^n$ in discrete time. In the remainder of this section, we introduce the *standard* finite-difference operators and matrices, and show that, using these, scheme (21) satisfies desired convergence properties on uniform grids.

Definition 2. **(Standard Finite-Difference Operators / Matrices)** *Consider a function $f : \bar{\mathcal{X}} \to \mathbb{R}$. Let $\mathcal{N}^{\pm}_j(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}})$ denote the right (+) or left (-) neighbor of $\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}}$ in dimension j. The first- and second-order*

A common alternative notational convention for isotropic uniform grids is the following: Denote grid points by $x_{i,j} = (k_i, z_j)$, and set $t_n = n\Delta t$, $k_i = i\Delta k = ih$ and $z_j = j\Delta z = jh$. The grid mesh size is then denoted by $h = \Delta k = \Delta z$, where $\Delta k = k_{i+1} - k_i$ and $\Delta z = z_{j+1} - z_j$ for all i, j. Letting J denote the number of grid points, grid G can again be thought of as a $J \times 2$ matrix, where the row correspond to the (k, z) coordinates of the J grid points.

standard finite-difference operators are defined as

$$
\begin{aligned}
D_z^+ \circ f(\boldsymbol{x}_{l,i}) &= \frac{f(\mathcal{N}_z^+(\boldsymbol{x}_{l,i})) - f(\boldsymbol{x}_{l,i})}{\left| \mathcal{N}_z^+(\boldsymbol{x}_{l,i}) - \boldsymbol{x}_{l,i} \right|} \\
D_z^- \circ f(\boldsymbol{x}_{l,i}) &= \frac{f(\boldsymbol{x}_{l,i}) - f(\mathcal{N}_z^-(\boldsymbol{x}_{l,i}))}{\left| \boldsymbol{x}_{l,i} - \mathcal{N}_z^-(\boldsymbol{x}_{l,i}) \right|} \\
D_{zz} \circ f(\boldsymbol{x}_{l,i}) &= \frac{f(\mathcal{N}_z^+(\boldsymbol{x}_{l,i})) - 2f(\boldsymbol{x}_{l,i}) + f(\mathcal{N}_z^-(\boldsymbol{x}_{l,i}))}{\left| \mathcal{N}_z^+(\boldsymbol{x}_{l,i}) - \boldsymbol{x}_{l,i} \right| \cdot \left| \boldsymbol{x}_{l,i} - \mathcal{N}_z^-(\boldsymbol{x}_{l,i}) \right|}
\end{aligned}
\tag{22}
$$

and similarly for $D_k^\pm$. The first- and second-order standard finite difference matrices discretize these operators and are denoted $\boldsymbol{D}_z^\pm$, $\boldsymbol{D}_k^\pm$, and $\boldsymbol{D}_{zz}$.

The definition of standard finite differences we present here is sufficiently general to apply to all grid structures we consider in this paper. On isotropic uniform grids with mesh size $h = \left| \mathcal{N}_j^\pm(\boldsymbol{x}_{l,i}) - \boldsymbol{x}_{l,i} \right|$ for all j, equations (22) simply collapse to

$$
\begin{aligned}
D_j^+ \circ f(\boldsymbol{x}_{l,i}) &= \frac{f(\ldots, x_{l_j,i_j} + h, \ldots) - f(\boldsymbol{x}_{l,i})}{h} \\
D_j^- \circ f(\boldsymbol{x}_{l,i}) &= \frac{f(\boldsymbol{x}_{l,i}) - f(\ldots, x_{l_j,i_j} - h, \ldots)}{h} \\
D_{jj} \circ f(\boldsymbol{x}_{l,i}) &= \frac{f(\ldots, x_{l_j,i_j} + h, \ldots) - 2f(\boldsymbol{x}_{l,i}) + f(\ldots, x_{l_j,i_j} - h, \ldots)}{h^2}
\end{aligned}
$$

where the mesh size h is directly used to identify a grid point's neighbors.

2.4 Consistency, Stability, Monotonicity, and Convergence

We now develop the well-known result that using the standard finite-difference matrices in equation (21) leads to a convergent finite-difference scheme. To formally state this result, we first introduce the important concepts of consistency, stability, and monotonicity. Our notation and exposition closely follow LeVeque (2007) and Barles and Souganidis (1991).

Our overarching goal is to estimate or bound the error that is incurred by solving the discretized scheme (21) for $\{\boldsymbol{V}^n\}$ relative to the sequence of functions $\{V^n\}$ defined by the recursion (19). Conceptually, we follow a two-step strategy that is an integral component in the analysis of finite-difference methods. First, we develop an estimate of the local error in discretizing derivatives using finite differences. Second, we provide conditions under which this local error does not propagate too much, which allows us to bound the global error in terms of the local error.

The *global error* of the finite-difference scheme *at a fixed point* is our ultimate object of interest. Consider the point in the state space $x \in \mathcal{X}$ associated with grid point $\boldsymbol{x}_{l,i}$ and fix n. The global error in scheme (21) is then given by $E_{l,i}^n = V_{l,i}^n - V^n(\boldsymbol{x}_{l,i})$, where $V_{l,i}^n$ is obtained from (21) and

$V^n(\boldsymbol{x}_{l,i})$ from (19).

The *local truncation error* (LTE) of a finite-difference scheme measures the local accuracy of the discretization itself and is obtained by evaluating the finite-difference scheme in question using the true function. For scheme (21), we have $\tau^n_{l,i} = \tau^n(\boldsymbol{x}_{l,i})$, with

$$\begin{aligned}\tau^n(k,z) = \frac{1}{\Delta t}(V^{n+1}(k,z) - V^{n+1}(k,z)) + \rho V^{n+1}(k,z) - u\Big(c^n(k,z)\Big) \\ - \Big(F(k,z) - c^n(k,z) - \delta k\Big) D_k \circ V^{n+1}(k,z) + \theta \boldsymbol{z} D_z \circ V^{n+1}(k,z) - \frac{\sigma^2}{2} D_{zz} \circ V^{n+1}(k,z).\end{aligned} \tag{23}$$

The difference between equations (21) and (23) is that the latter evaluates the finite-difference scheme by plugging in the true function values $V^{n+1}(k,z)$. In the context of differential equations that admit a closed-form solution, the LTE can be directly computed using its definition (23). In most applications, however, the true function values $V^{n+1}(k,z)$ are not available. Nonetheless, equation (23) can still be used to develop error estimates, for example when $V^n(\cdot)$ is sufficiently smooth.

Definition 3. (**Consistency**) *A finite-difference method is consistent on uniform grids if* $\tau^n_{l,i} \to 0$ *for all* $\boldsymbol{i} \in I_{\boldsymbol{l}}$ *and* n *as* $\boldsymbol{l} \to \infty$ *and* $\Delta t \to 0$.

Consistency requires that the finite-difference discretization of partial derivatives is sensible, resulting in a local error that decays in the resolution of the grid, i.e., as $\boldsymbol{l} \to \infty$, and the time step, i.e., as $\Delta t \to 0$. A local error bound is of course necessary for the global error $E^n_{l,i}$ at the fixed point $\boldsymbol{x}_{l,i}$ and time step n to decay as the grid and time steps become finer. The key to establish a link between local and global errors are the stability and monotonicity of the finite-difference scheme, which we define next.

Definition 4. (**Stability**) *A linear finite-difference scheme of the form* (21) *is (Lax-Richtmyer) stable on uniform grid* G *if* $(\rho + \frac{1}{\Delta t} - \boldsymbol{A}^n)$ *is invertible and, for each time step* T*, there exists a constant* $C_T > 0$ *such that*

$$\left\| \left(\rho + \frac{1}{\Delta t} - \boldsymbol{A}^n\right)^{-1} \frac{1}{\Delta t} \right\| \leq C_T,$$

for all $h > 0$ *and* n *for which* $h \cdot n \leq T$.

To introduce monotonicity, we rewrite the finite-difference scheme (21) following Barles and Souganidis (1991) in the form

$$\begin{aligned}\mathcal{S}\Big(\Delta t, G, \boldsymbol{V}^{n+1}, \big\{V^n_{l,i}, V^{n+1}_{l,i}\big\}\Big) = \frac{1}{\Delta t}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n) + \rho \boldsymbol{V}^{n+1} \\ - u\Big(\boldsymbol{c}^n\Big) - \Big(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\Big) \boldsymbol{D}_k \boldsymbol{V}^{n+1} + \theta \boldsymbol{z} \boldsymbol{D}_z \boldsymbol{V}^{n+1} - \frac{\sigma^2}{2} \boldsymbol{D}_{zz} \boldsymbol{V}^{n+1}\end{aligned} \tag{24}$$

where $\mathcal{S}(\cdot)$ defines a $J \times 1$ vector on grid G. The third and fourth arguments of $\mathcal{S}(\cdot)$ deserve notational clarification. We say that the element $\mathcal{S}_{\boldsymbol{l},\boldsymbol{i}}$ associated with grid point $\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}}$ depends on $V_{\boldsymbol{l},\boldsymbol{i}}^{n+1}$ directly through the third argument, and on all *other* elements of $\boldsymbol{V}^n$ and $\boldsymbol{V}^{n+1}$, i.e., all elements of these two vectors other than $V_{\boldsymbol{l},\boldsymbol{i}}^{n+1}$, through the fourth argument.

Definition 5. (**Monotonicity**) *A linear finite-difference scheme of the form* (21) *is monotonic on grid G if $\mathcal{S}(\cdot)$ is decreasing in its fourth argument up to order $\mathcal{O}(h)$, i.e.,*

$$\mathcal{S}(\Delta t, G, r, p) \geq \mathcal{S}(\Delta t, G, r, q) + \mathcal{O}(h)\,, \quad \text{if } p \leq p'.$$

Having introduced the concepts of consistency, stability, and monotonicity, we can now formally define convergence of finite-difference schemes and present the main result of this section. Note that the concept of convergence at stake here concerns the behavior of the finite-difference method's global error $E_{\boldsymbol{l},\boldsymbol{i}}^n$ at a *fixed point* as the grid and time step become finer. The notion of a convergent finite-difference scheme in this sense is distinct from the convergence of the homogenized time-marching scheme (19), which we proved in Proposition 3.

Definition 6. (**Convergent Finite-Difference Scheme**) *We say that a linear finite-difference scheme of the form* (21) *is convergent on grid G if the global error converges in $\|\cdot\|_2$ in the resolution of grid and time steps, i.e., if $\|E_{\boldsymbol{l},\boldsymbol{i}}^n\|_2 \to 0$ as $\boldsymbol{l} \to \infty$ and $\Delta t \to 0$ for $n \cdot 2^{-|\boldsymbol{l}|_\infty} \leq T$.*

We are now ready to state the well-known result that linear scheme (21) is consistent, stable, monotonic, and convergent on uniform grids when using standard finite-difference matrices $\boldsymbol{D}_z^{\pm}$, $\boldsymbol{D}_k^{\pm}$, and $\boldsymbol{D}_{zz}$.

Proposition 4. (**Convergence of Standard Finite-Difference Scheme on Uniform Grids**) *Let G be a uniform grid with mesh size h over the torus $\mathcal{X}$. Construct the sequence $\{\boldsymbol{V}^n\}$, where $\boldsymbol{V}^n \in \mathbb{R}^J$, recursively according to* (21) *using standard finite-difference matrices $\boldsymbol{D}_z^{\pm}$, $\boldsymbol{D}_k^{\pm}$, and $\boldsymbol{D}_{zz}$. Then:*

(i) (Consistency): *The local truncation error (LTE) converges uniformly, with $\tau_{\boldsymbol{l},\boldsymbol{i}}^n \to 0$ as $\boldsymbol{l} \to \infty$ and $\Delta t \to 0$.*

(ii) (Stability): *Scheme* (21) *is stable if we use an upwind scheme for $\boldsymbol{D}_k$ and $\boldsymbol{D}_z$.*[15]

(iii) (Monotonicity): *Scheme* (21) *is monotonic.*

(v) (Convergence): *For a fixed T, the global error $E_{\boldsymbol{l},\boldsymbol{i}}^n$ decays in $\|\cdot\|_2$ as $\boldsymbol{l} \to \infty$ and $\Delta t \to 0$ for $n \cdot 2^{-|\boldsymbol{l}|_\infty} \leq T$.*

[15] See, e.g., LeVeque (2007) for a textbook treatment of the upwind method or Achdou et al. (2021) for a discussion in the context of economic applications.

(vi) (Existence and uniqueness): *In iteration n, there exists a unique solution $\boldsymbol{V}^{n+1}$ to (21).*

(vii) (Contraction mapping): *Scheme (21) is a contraction mapping. There exists a $J \times 1$ vector $\boldsymbol{V}$ such that $\boldsymbol{V}^n \to \boldsymbol{V}$.*

From (v) and (vii), the piecewise linear interpolant of $\boldsymbol{V}$ on $\mathcal{X}$ converges to the unique stationary value function $V(\cdot)$ as $\boldsymbol{l} \to \infty$.

3 Dynamic Programming on Sparse Grids

The goal of Section 3 is to develop a finite-difference method for dynamic programming problems and the resulting differential equations on sparse grids. To set the stage, we show in Section 3.1 that the standard method introduced in Section 2 is no longer consistent and breaks down on sparse grids. Section 3.2 then develops a consistent finite-difference method by leveraging the hierarchical basis transformations introduced in Section 1. Finally, we develop a value function iteration algorithm on adaptive sparse grids in Section 3.3, and conclude with a discussion of monotonicity and convergence in Section 3.4. We postpone a formal treatment of boundary conditions until Section 4.

3.1 What Goes Wrong on Sparse Grids

On sparse grids, the standard finite-difference stencils no longer lead to consistent discretizations of partial derivatives and, in particular, scheme (21) breaks down. The failure of the standard method in the context of sparse grids is the principal motivation for this paper. We start our discussion of dynamic programming on sparse grids by formalizing this negative result.

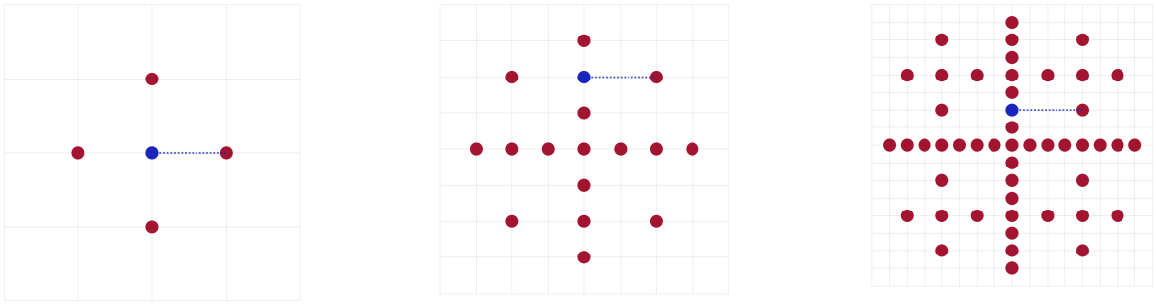
Proposition 5. (**Inconsistency of Standard Finite-Difference Schemes on Sparse Grids**) *Let G be a regular sparse grid of level $\boldsymbol{l}$. Construct the sequence $\{\boldsymbol{V}^n\}$ recursively according to (21) using the standard finite-difference matrices $\boldsymbol{D}_k^{\pm}$, $\boldsymbol{D}_z^{\pm}$, and $\boldsymbol{D}_{zz}$. Then:*

(i) In iteration n, there exist at least $d \cdot 2^{|\boldsymbol{l}|_\infty - 1}$ grid points with discretization error (LTE) of order $\mathcal{O}(1)$.

(ii) The discretization using standard finite-difference matrices is not consistent. That is, $\tau_{\boldsymbol{l},\boldsymbol{i}}^n \not\to 0$ as $\boldsymbol{l} \to \infty$.

(iii) If $\boldsymbol{V}^n \to \boldsymbol{V}$ converges as $n \to \infty$ on grid $G_{\boldsymbol{l}}$, then the interpolant of $\boldsymbol{V}$ does not converge to the unique solution V of equation (18) as $\boldsymbol{l} \to \infty$.

We present the proof of Proposition 5 in Appendix C.2.



(a) $\boldsymbol{l} = (2,2)$ (b) $\boldsymbol{l} = (3,3)$ (c) $\boldsymbol{l} = (4,4)$

Figure 9: Inconsistency of Standard Finite-Difference Method on Sparse Grids

Note. Figure 9 illustrates the failure of standard finite-difference schemes on sparse grids. Panels (a) through (c) plot a sequence of two-dimensional regular sparse grids of increasing level $\boldsymbol{l}$. The figure highlights that for arbitrary levels of resolution, i.e., arbitrarily large $\boldsymbol{l}$, regular sparse grids feature nodes at which the local truncation error of a standard finite-difference discretization of the first-order partial derivative remains of order $\mathcal{O}(1)$. Examples of such nodes are colored blue in each panel. The dashed blue line plots the distance between these focal nodes and their right neighbors, which does not decay despite the increasing resolution of the grid.

The standard finite-difference discretization typically used on uniform grids fails to yield a consistent discretization scheme on sparse grids. That is, one cannot simply use the discretizations $D_k^{\pm}$, for example, to approximate the partial derivative operator $\frac{\partial}{\partial k}$ on sparse grids. In particular, part (i) of Proposition 5 establishes that on regular sparse grids of level $\boldsymbol{l}$, there is always a minimum number of grid points — at least $d \cdot 2^{|\boldsymbol{l}|_\infty - 1}$ of them — at which the local truncation error of a standard finite-difference stencil remains of order $\mathcal{O}(1)$. In other words, the local discretization error at these points does not decay as the grid becomes finer, i.e., $\boldsymbol{l} \to \infty$.

We illustrate the intuition behind this negative result in Figure 9. Panels (a) through (c) display two-dimensional regular sparse grids of increasing level $\boldsymbol{l}$. Each panel contains a focal grid point, colored in blue, at which we apply the standard finite-difference forward operator in the horizontal dimension. The dashed blue lines connect these focal points to their right neighbors, illustrating the standard forward stencil. In panel (a), the focal point is $(0.5, 0.5)$ and the distance to its right neighbor, $(0.75, 0.5)$, is $h_2 = 0.25$. A consistent discretization of the first derivative requires that this distance shrinks as the grid becomes finer. Panel (b) demonstrates that a finer regular sparse grid indeed adds grid points on the main axis, with the distance between point $(0.5, 0.5)$ and its right neighbor shrinking to $h_3 = 0.125$, as required for a consistent discretization scheme.

Panel (b) also underscores, however, that refining the regular sparse grid introduces new grid points, at which the distance to the right neighbor remains 0.25. We color one such focal point in blue. As the sparse grid becomes finer, with $\boldsymbol{l} \to \infty$, there will always be a minimum number of grid points that are 0.25 removed from their right neighbor. In panel (c), we highlight one such new grid point in blue. Even for the finest sparse grids, with $\boldsymbol{l}$ large, applying the standard finite-difference

stencil at such grid points leads to a local discretization error that remains of order $\mathcal{O}(1)$. Even more problematically, the discretization error incurred at grid points in a local neighborhood remains of different order. For example in panel (c), the approximation error of the standard forward stencil applied at point $(0.5, 0.5)$ has shrunk to $h_4 = 0.0675$ — indeed, it is of order $2^{-|\boldsymbol{l}|_\infty}$ on this main axis and thus decays as the grid becomes finer. At the nearby focal point $(0.625, 0.5)$ colored in blue, however, it remains proportional to 0.25 and thus of order $\mathcal{O}(1)$. As a result, the discretization of partial derivatives becomes increasingly non-smooth as they locally converge to their true value at some grid points but remain inaccurate at other points in a vanishingly small local neighborhood.

3.2 A Finite-Difference Scheme for Sparse Grids

We are now ready to develop a *sparse finite-difference method* and prove that it is consistent on non-uniform grids, which is the main result of this section. The hierarchization operators introduced in Section 1 are at the heart of the sparse finite-difference operators defined below.

Definition 7. (**Sparse Finite-Difference Operators / Matrices**) *The first- and second-order sparse finite-difference operators are defined as*

$$D_k^{\pm,\mathrm{S}} = E_k \circ D_k^{\pm} \circ H_k \,, \quad D_z^{\pm,\mathrm{S}} = E_z \circ D_z^{\pm} \circ H_z \,, \quad \text{and} \quad D_{zz}^{\mathrm{S}} = E_z \circ D_{zz} \circ H_z, \tag{25}$$

where $D_\bullet^\pm$ and D_{zz} are the standard finite-difference operators introduced in (22), and $E_\bullet$ and $H_\bullet$ are hierarchization operators. The matrix discretizations of the sparse finite-difference operators are defined as

$$\boldsymbol{D}_k^{\pm,\mathrm{S}} = \boldsymbol{E}_k \, \boldsymbol{D}_k^{\pm} \, \boldsymbol{H}_k, \quad \boldsymbol{D}_z^{\pm,\mathrm{S}} = \boldsymbol{E}_z \, \boldsymbol{D}_z^{\pm} \, \boldsymbol{H}_z, \quad \text{and} \quad \boldsymbol{D}_{zz}^{\mathrm{S}} = \boldsymbol{E}_z \, \boldsymbol{D}_{zz} \, \boldsymbol{H}_z, \tag{26}$$

where $\boldsymbol{D}_\bullet^\pm$ and $\boldsymbol{D}_{zz}$ are the standard finite-difference matrices introduced in (22), and $\boldsymbol{E}_\bullet$ and $\boldsymbol{H}_\bullet$ are hierarchization matrices.

Sparse finite-difference matrices are still based on the standard stencils $\boldsymbol{D}_k^\pm$, $\boldsymbol{D}_z^\pm$, and $\boldsymbol{D}_{zz}$, but interact these with hierarchization and dehierchization. In simple words, the sparse finite-difference matrix $\boldsymbol{D}_j^{+,\mathrm{S}}$ in dimension j applied to a vector of function values $\boldsymbol{V}$ comprises three steps: First, hierarchize the function values $\boldsymbol{V}$ in all dimensions but j (apply $\boldsymbol{H}_j$). Second, use the standard finite-difference stencil for the desired derivative (apply $\boldsymbol{D}_j^+$). Third, dehierarchize the resulting vector in all dimensions but j (apply $\boldsymbol{E}_j$).

With these definitions in hand, we can generalize the finite-difference scheme (21) for the HJB equation to sparse grids, which yields

$$\frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}^n}{\Delta} + \rho \boldsymbol{V}^{n+1} = u\Big(\boldsymbol{c}^n\Big) + \Big(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\Big) \boldsymbol{D}_k^{\mathrm{S}} \boldsymbol{V}^{n+1} - \theta \boldsymbol{z} \, \boldsymbol{D}_z^{\mathrm{S}} \boldsymbol{V}^{n+1} + \frac{\sigma^2}{2} \boldsymbol{D}_{zz}^{\mathrm{S}} \boldsymbol{V}^{n+1}. \tag{27}$$

After rearranging and defining the composite operator $\boldsymbol{A}^{n,\mathrm{S}} = (\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}) \, \boldsymbol{D}_k^{\mathrm{S}} - \theta \boldsymbol{z} \boldsymbol{D}_z^{\mathrm{S}} + \frac{\sigma^2}{2} \boldsymbol{D}_{zz}^{\mathrm{S}}$,

we arrive at the sequence of linear systems

$$\left(\rho + \frac{1}{\Delta t} - \boldsymbol{A}^{n,\mathrm{S}}\right)\boldsymbol{V}^{n+1} = u\left(\boldsymbol{c}^{n}\right) + \frac{1}{\Delta t}\boldsymbol{V}^{n}.$$

Finally, we define the local truncation error associated with the sparse finite-difference scheme (27) as $\tau^{n,\mathrm{S}}_{\boldsymbol{l},\boldsymbol{i}} = \tau^{n,\mathrm{S}}(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}})$, where

$$\begin{aligned}\tau^{n,\mathrm{S}}(k,z) = \frac{1}{\Delta t}(V^{n+1}(k,z) - V^{n+1}(k,z)) + \rho V^{n+1}(k,z) - u\Big(c^{n}(k,z)\Big) \\ - \Big(F(k,z) - c^{n}(k,z) - \delta k\Big)D^{\mathrm{S}}_{k} \circ V^{n+1}(k,z) + \theta \boldsymbol{z} D^{\mathrm{S}}_{z} \circ V^{n+1}(k,z) - \frac{\sigma^2}{2} D^{\mathrm{S}}_{zz} \circ V^{n+1}(k,z).\end{aligned} \tag{28}$$

The following proposition summarizes the main result of this section: sparse finite-difference matrices lead to consistent discretizations of first- and second-order partial derivatives on sparse grids.[16]

Proposition 6. (**Consistency of Sparse Finite-Difference Schemes**) *Let G be a regular sparse grid of level $\boldsymbol{l}$ over the torus $\mathcal{X}$. Consider the sequences of function values $\{\boldsymbol{V}^n\}$ and LTEs $\{\boldsymbol{\tau}^{n,S}\}$ defined by (27) and (28). Then:*

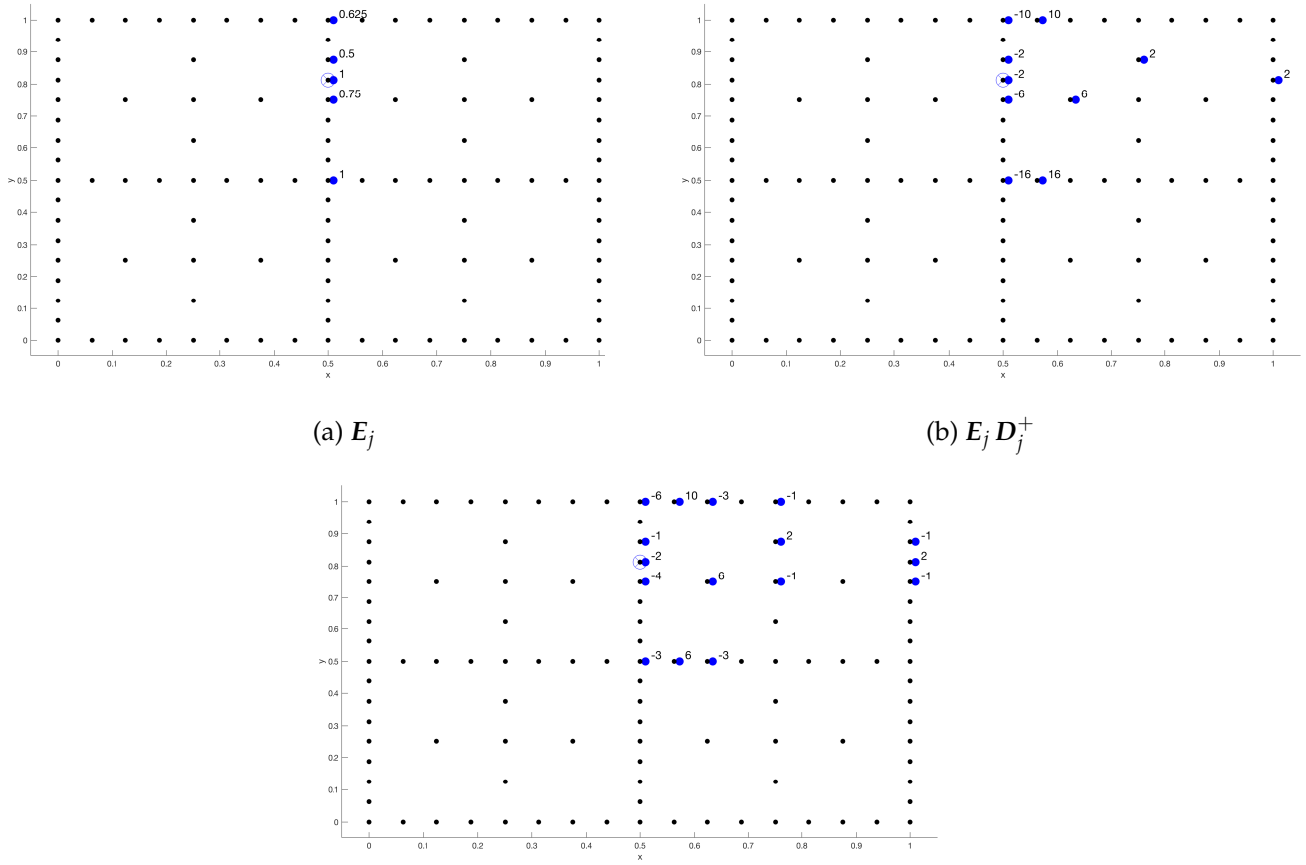
(i) The sparse finite-difference operators $D^{\pm,S}_{j}$ and D^{S}_{jj} lead to consistent discretizations of first- and second-order partial derivatives in dimension j.

(ii) Scheme (27) is a consistent discretization of the HJB equation (18). That is, $\tau^{n,S}_{\boldsymbol{l},\boldsymbol{i}} \to 0$ for all $\boldsymbol{i} \in I^{H}_{\boldsymbol{l}}$ and n as $\boldsymbol{l} \to \infty$ and $\Delta t \to 0$.

The proof of Proposition 6 is in Appendix C.3. In the remainder of this subsection, we explain and unpack why the sparse finite-difference method is consistent while the standard one is not. We discuss three perspectives that are useful to understand this result.

Information from off-dimensions. Sparse finite-difference stencils draw valuable information from off-dimensions, i.e., from neighbors in a d-dimensional ball. Recall the source of inconsistency in the negative result of Section 3.1: When applying the standard forward stencil at one of the focal points colored blue in Figure 9, the standard stencil looks for a right neighbor that is far removed. At the same time, however, there are other grid points in a d-dimensional ball around the focal node that are much closer and that, intuitively, should provide important information regarding the derivative of the function.

[16] In addition to the definition and construction of generalized sparse finite-difference operators, it is also important to define an underlying *function algebra* on sparse grids. We subsume this step implicitly in our discussion. For details on this, see Schiekofer (1998) and Griebel (1998).



(a) $\boldsymbol{E}_j$

(b) $\boldsymbol{E}_j\,\boldsymbol{D}_j^+$

(c) $\boldsymbol{E}_j\,\boldsymbol{D}_j^+\,\boldsymbol{H}_j$

Figure 10: Decomposition of Sparse Finite-Difference Stencil

We illustrate this intuition in Figure 10, where we break the sparse finite-difference matrix $\boldsymbol{D}_j^{+,S}$ into its three constituent operations $\boldsymbol{E}_j$, $\boldsymbol{D}_j^+$, and $\boldsymbol{H}_j$, each applied at the focal node $(x, y) = (0.5, 0.8125)$ marked with a cross. The underlying grid is a regular sparse grid of level $\boldsymbol{l} = (4, 4)$. In panel (a), we see that $\boldsymbol{E}_j$ identifies all points with the same $x = 0.5$ value as the focal node that have right-neighbors that are closer than the right-neighbor of the focal node itself.[17] For example, grid point $(0.5, 0.75)$ has a right neighbor at $(0.625, 0.75)$. The distance between these two nodes

[17] Why do we unpack these matrices left-to-right? Our goal is to compute the approximate derivative at point $(0.5, 0.8125)$ of a function represented by its function values $\boldsymbol{V}$ on the grid. Applying the sparse finite-difference stencil to these function values, the matrix-vector product $\boldsymbol{D}_x^+\,\boldsymbol{V}$, results in a vector that represents this derivative across the grid. We are interested in its value at a specific focal node, whose index we denote by i for illustrate. That is, our object of interest is $(\boldsymbol{D}_x^+\,\boldsymbol{V})_i = \boldsymbol{D}_{x,[i,:]}^+\,\boldsymbol{V}$, which itself can be represented as the dot product of the ith row of $\boldsymbol{D}_x^+$ with the vector $\boldsymbol{V}$. Similarly, when splitting up the sparse finite-difference stencil into its constituent operations, evaluating the derivative at node i yields $(\boldsymbol{E}_x\,\boldsymbol{D}_x^+\,\boldsymbol{H}_x\,\boldsymbol{V})_i = \boldsymbol{E}_{x,[i,:]}\,\boldsymbol{D}_x^+\,\boldsymbol{H}_x\,\boldsymbol{V}$, i.e., the dot product of the ith row of the leftmost matrix $\boldsymbol{E}_x$ with the vector $\boldsymbol{D}_x^+\,\boldsymbol{H}_x\,\boldsymbol{V}$. It is precisely this ith row of $\boldsymbol{E}_j$ that we display in panel (a) of Figure 10.

is $2 \cdot h_4 = 2 \cdot 2^{-|\boldsymbol{l}|_\infty} = 0.125$. The distance between the focal node itself and its right neighbor, on the other hand, is 0.5. Intuitively, therefore, grid point $(0.625, 0.75)$ may contain more information about the first derivative of a function at the focal node than the latter's right neighbor $(1, 0.8125)$.

In the second step, displayed in panel (b), we apply the standard forward stencil $\boldsymbol{D}_x^+$ to all those grid points previously identified by $\boldsymbol{E}_j$. Applying the standard forward stencil in turn identifies the respective right-neighbors of these points, each of which is closer in the x-dimension to the focal node than its own right-neighbor. Finally, we apply $\boldsymbol{H}_j$. Panel (c) displays the composite sparse finite-difference forward stencil applied at the focal node. Instead of simply looking for the right-neighbor of the focal node, the sparse stencil draws on information from those nearby grid points in the off-dimension that have relatively closer right-neighbors. In $d > 1$ dimensions, the sparse stencil draws on information from nearby grid points in a d-dimensional ball.

Interpolation. By definition, the first-order partial derivative of $V(x)$ in dimension j is $\partial_{x_j} V(x) = \lim_{h \to 0} \frac{1}{h}(V(\ldots, x_j + h, \ldots) - V(x))$. Consider a sequence of grids that we can associate with this limit and a focal node i, at which we want to evaluate the partial derivative $\partial_{x_j} V(x)$. Suppose the node $\boldsymbol{x}_{i^+} = \boldsymbol{x}_i + h\boldsymbol{e}_j$ is always on the grid as we take $h \to 0$, i.e., the focal node $\boldsymbol{x}_i$ always has its right-neighbor of distance h on the grid. In that case, simply applying the standard forward stencil leads to a consistent discretization of $\partial_{x_j} V(x)$ at point $\boldsymbol{x}_i$ because $V(\boldsymbol{x}_{i^+}) - V(\boldsymbol{x}_i) = \mathcal{O}(h)$ by construction. On uniform grids, this is true for every grid point, and so the standard stencil is consistent. On regular sparse grids of level $\boldsymbol{l}$, $\boldsymbol{D}_j^{+,\mathrm{S}}$ also collapses to the standard forward stencil at those nodes whose right-neighbor of distance $h = 2^{-|\boldsymbol{l}|_\infty}$ is on the grid. Whenever this is not the case, the sparse finite-difference stencil draws on information from off-dimensional neighbors as we discussed above.

These observations suggest a necessary condition for consistency: Any candidate discretization $\tilde{\boldsymbol{D}}_j^+$ of the partial derivative ∂_{x_j} must — in order to be consistent — satisfy

$$\tilde{\boldsymbol{D}}_j^+ \boldsymbol{V} = \frac{V(\ldots, x_j + h, \ldots) - V(x)}{h} + \mathcal{O}(h) = \frac{1}{h} V(\ldots, x_j + h, \ldots) - \frac{1}{h}\boldsymbol{V} + \mathcal{O}(h)$$

where $h = 2^{-|\boldsymbol{l}|_\infty}$ indexes a sequence of regular sparse grids with $h \to 0$ as $\boldsymbol{l} \to \infty$. Intuitively, therefore, any consistent discretization must recover an approximation of the h-removed right-neighbor $V(\ldots, x_j + h, \ldots)$ with error of order $\mathcal{O}(h)$. In other words, we would like to interpolate the vector of function values $\boldsymbol{V}$ onto the right-neigbhor $\boldsymbol{x}_{i^+}$ for every node i and then construct the standard forward difference using this interpolant.[18]

We now show that sparse finite-difference stencils implement exactly this interpolation. Formally, define the operator that interpolates onto $\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}} + h\boldsymbol{e}_j$ via

$$I_{j,h}^+ \circ V(x) = V(\ldots, x_j + h, \ldots),$$

[18] This also requires that the interpolation error decays at $\mathcal{O}(h)$.

and its matrix discretization by $\boldsymbol{I}_{j,h}^{+}$.

Proposition 7. (**Sparse Finite Differences via Interpolation**) *The sparse finite-difference operators and matrices implement an interpolation stencil, given by*

$$\boldsymbol{E}_j \boldsymbol{D}_j^{+} \boldsymbol{H}_j = \frac{\boldsymbol{I}_{j,h}^{+} - \boldsymbol{I}}{h}, \tag{29}$$

where $\boldsymbol{I}$ is the identity matrix.

We present the proof of Proposition 7 in Appendix C.4. Analogously to Figure 10, we illustrate the decompositions of the sparse finite-difference and interpolation matrices in equation (29) in Appendix C.5.

Taylor expansions. Consider again the negative result of Section 3.1. In the context of the grid shown in Figure 10, the negative result obtains because there are grid points, such as our focal node $(0.5, 0.8125)$, whose distance from their right neighbor is not proportional to $h = 2^{-|I|_\infty} = 2^{-4} = 0.0625$. Indeed, the right-neighbor of the focal node is $(1, 0.8125)$, and the distance between these two points is 0.5.

The consistency of a finite-difference scheme is often established by examining its Taylor expansion. Consider the standard forward stencil in the x-dimension applied at the focal node of Figure 10, which we denote $(x_i, y_i) = (0.5, 0.8125)$. The implied derivative approximation is

$$\begin{aligned} D_x^{+} \circ f(x_i, y_i) &= \frac{f(1, y_i) - f(0.5, y_i)}{0.5} \\ &\approx \frac{1}{0.5}\Big(f(x_i, y_i) + \partial_x f(x, y) \mid_{x_i, y_i} (1 - x_i) + \partial_{xx} f(x, y) \mid_{x_i, y_i} (1 - x_i)^2 - f(x_i, y_i)\Big) \\ &\approx \partial_x f(x, y) \mid_{x_i, y_i} + \partial_{xx} f(x, y) \mid_{x_i, y_i} (1 - x_i) \end{aligned}$$

where the second line follows from a second-order Taylor expansion around the point (x_i, y_i), and the third line uses $1 - x_i = 0.5$. Now, crucially, suppose $\partial_{xx} f(x, y) \mid_{x_i, y_i}$ is not 0. Then, the residual error in our discretization of the partial derivative $\partial_x f(x, y) \mid_{x_i, y_i}$ with the standard finite-difference stencil $D_x^{+} \circ f(x_i, y_i)$ is of order $\mathcal{O}(1)$ — rather than $\mathcal{O}(h)$ as consistency would require — because $(1 - x_i) = 0.5$ is $\mathcal{O}(1)$ and does not decay with h.

We can perform a similar Taylor expansion for the sparse finite-difference stencil displayed in panel (c) of Figure 10. Notice that $\boldsymbol{D}_x^{+,\mathrm{S}}$ loads on three nodes on the right boundary — namely, $(1, y_i)$, $(1, y_i - h)$, and $(1, y_i + h)$ — whereas the standard stencil $\boldsymbol{D}_x^{+}$ only loads on the single point $(1, y_i)$, which leads to the problem we discuss above. To illustrate how the sparse stencil achieves

consistency, consider the second-order Taylor expansions of the three points on the right boundary

$$
\begin{aligned}
f(1, y_i) &\approx f(x_i, y_i) + \partial_x f(x, y) \mid_{x_i, y_i} (1 - x_i) + \partial_{xx} f(x, y) \mid_{x_i, y_i} (1 - x_i)^2 \\
f(1, y_i + h) &\approx f(1, y_i) + \partial_y f(x, y) \mid_{x_i, y_i} h + \partial_{yy} f(x, y) \mid_{x_i, y_i} h^2 + \partial_{xy} f(x, y) \mid_{x_i, y_i} (1 - x_i) h \\
f(1, y_i - h) &\approx f(1, y_i) - \partial_y f(x, y) \mid_{x_i, y_i} h + \partial_{yy} f(x, y) \mid_{x_i, y_i} h^2 - \partial_{xy} f(x, y) \mid_{x_i, y_i} (1 - x_i) h.
\end{aligned}
$$

Next, we add these terms according to the weights implied by the sparse stencil $\boldsymbol{D}_x^{+,\mathrm{S}}$. We obtain

$$
2 \cdot f(1, y_i) - 1 \cdot f(1, y_i + h) - 1 \cdot f(1, y_i - h) \approx -2 \cdot \partial_{yy} f(x, y) \mid_{x_i, y_i} h^2.
$$

Crucially, all terms of order $\mathcal{O}(1)$ dropped out. The only remaining term under a second-order Taylor expansion is $\partial_{yy} f(x, y) \mid_{x_i, y_i} h^2$, which now scales in h. In a mechanical sense, this is how the sparse finite-difference stencil fixes the $\mathcal{O}(1)$ discretization error.

In conclusion, the standard finite-difference matrices $\boldsymbol{D}_j^{\pm}$ and $\boldsymbol{D}_{jj}$ lead to consistent discretizations of partial derivatives in dimension j on uniform grids. To obtain consistency on sparse grids, we simply interact the standard stencils with an initial hierarchization step and a subsequent dehierarchization step. In practice, this requires pre- and post-multiplying the standard finite-difference matrices by two projection matrices whose construction is cheap and straightforward.

3.3 Value Function Iteration on Adaptive Sparse Grids

We have so far discussed continuous-time dynamic programming on uniform and regular sparse grids. In this section, we develop a value function iteration algorithm to solve the sparse finite-difference scheme (27) on adaptive sparse grids. That is, we take as our starting point a regular sparse grid $G_{\boldsymbol{l}_0}$ of a given initial level $\boldsymbol{l}_0$ and then refine the grid by adding (removing) grid points where local approximation errors remain large (become small). At the heart of this procedure is the observation that hierarchical surpluses encode residual, local approximation errors.

Grid adaptation. Equipped with the results of Sections 1 and 3.2, we can recursively construct the sequence of vectors $\{\boldsymbol{V}^n\}$ on a given regular sparse grid $G_{\boldsymbol{l}_0}$ that discretizes the state space $\bar{\mathcal{X}}$. We now introduce adaptive grid refinement as an iterative outer step. Iteration k of this outer fixed point is associated with a grid G^k, taking $G^0 = G_{\boldsymbol{l}_0}$, and solves an approximate solution $\{\boldsymbol{V}^{k,n}\} \to \boldsymbol{V}^k$. We then construct an adapted grid G^{k+1} by adding and removing grid points according to the residual, local approximation errors encoded in the hierarchical representation of $\boldsymbol{V}^k$.

Formally, we can associate the $J^k \times 1$ vector $\boldsymbol{V}^k$ with an interpolant $V^k(x)$ in the approximation space of piecewise linear interpolants induced by grid G^k. Our goal, then, is to adaptively construct a grid that, given a desired number of degrees of freedom (grid points), minimizes the approxi-

Algorithm 1 Value Function Iteration on Adaptive Sparse Grids

Choose an index $\boldsymbol{l}^0$ and construct associated regular sparse grid $G^0 = G_{\boldsymbol{l}_0}$
Construct $\boldsymbol{H}^0$ and sparse FD operators on G^0 ▷ Equations (7) and (26)
Initialize guess for the value function $\boldsymbol{V}^0$ on G^0
for $k \geq 0$ **do**
    for $n \geq 0$ **do**
        Compute $\boldsymbol{A}^{k,n}$ using sparse FD operators ▷ Equation (27)
        Solve linear system for $\boldsymbol{V}^{k,n+1}$ ▷ Equation (27)
        Continue if $|\boldsymbol{V}^{k,n+1} - \boldsymbol{V}^{k,n}| > \epsilon^{\text{VFI}}$
    end
    Compute hierarchical surplus $\boldsymbol{\alpha}^{k,\text{H}} = \boldsymbol{H}^k \boldsymbol{V}^k$
    If adaptation criterion met, adapt grid G^{k+1} ▷ Equations (11) and (12)
    Reconstruct $\boldsymbol{H}^{k+1}$ and sparse FD operators on G^{k+1} ▷ Equations (7) and (26)
end

mation error $\|V - V^k\|$, where V is the true solution of the stochastic neoclassical growth model.[19] To this end, we leverage the hierarchical representation of $\boldsymbol{V}^k$ and use the resulting hierarchical surpluses as our adaptation criterion for G^{k+1}.[20] Using the hierarchization matrix $\boldsymbol{H}^k$ associated with grid G^k, the hierarchical surpluses associated with $\boldsymbol{V}^k$ are given by $\boldsymbol{\alpha}^{k,\text{H}} = \boldsymbol{H}^k \boldsymbol{V}^k$, where $\boldsymbol{\alpha}^{k,\text{H}}$ is the $J^k \times 1$ vector of hierarchical surpluses. We adopt the same adaptation criterion as in equations (11) and (12), given thresholds $0 < \epsilon^{\text{keep}} < \epsilon^{\text{add}}$.

Whenever a particular node i in the grid satisfies the adaptation criterion, we add all neighboring grid points one hierarchical level lower. We denote these grid points the *children* of the *parent* node i. During the adaptation process, it is crucial that the grid contains all the parents of all of its nodes. If this is not the case and the grid has *holes*, we are no longer able to map nodal into hierarchical function representations.[21] In practice, we check to add any parent nodes back in if our adaptation algorithm deletes them.

Algorithm. Our proposed algorithm to solve dynamic programming problems in continuous time using adaptive sparse grids is summarized in Algorithm 1.

3.4 Monotonicity and Convergence

For finite difference discretizations, we typically want to prove the convergence of the proposed numerical discretization scheme. In the context of partial differential equations that admit classical

[19] Mechanically, the residual approximation error is large in regions of the domain $\mathcal{X}$ where V features relatively more concavity, i.e., where V is particularly different from its interpolant based on the piecewise linear basis functions operative in that region.

[20] Due to its multi-level structure, the size of the lowest-level hierarchical coefficient is a good measure of the function's concavity in the region. An adaptation criterion can therefore be chosen simply in terms of the hierarchical coefficients obtained in iteration k.

[21] See, e.g., Ruttscheidt (2018) for further discussion of this point.

solutions, the Lax-Friedrich's lemma postulates that convergence obtains when the underlying scheme is consistent and stable. In the context of viscosity solutions, on the other hand, the seminal work by Barles and Souganidis (1991) shows that we additionally require a monotonicity condition on the underlying scheme.

The main difficulty with proving general results on the convergence of sparse finite-difference schemes in the context of viscosity solutions is that the Barles and Souganidis (1991) monotonicity condition is generally not satisfied. In this context, Hemker (2000) points out that interpolation on sparse grids does not generally satisfy monotonicity. Garcke and Ruttscheidt (2019) show that even in the case of concave, monotonically increasing functions, interpolation on sparse grids 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 sparse grid 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 underlying value function.

In the present version of this paper, our focus is on numerically demonstrating the convergence of value function iteration algorithm 1 using adaptive sparse grids in the context of a wide range of economic applications. We pursue this strategy in Section 5. In ongoing work, we continue to explore the formal stability and monotonicity properties of sparse finite-difference methods.

4 Boundary Conditions

Our discussion thus far has been focused around solution methods for equation (18) on differently structured grids over the torus $\mathcal{X} = (0, \bar{k}) \times (\underline{z}, \bar{z})$. In this section, we extend our results to sparse grids over the compact state space $\bar{\mathcal{X}} = [0, \bar{k}] \times [\underline{z}, \bar{z}]$. For convenience, we introduce the notation $\partial\bar{\mathcal{X}} = \bar{\mathcal{X}} \setminus \mathcal{X}$ to denote the boundary region which we have thus far abstracted from.

Equation (18) is only formally defined on the interior $\mathcal{X}$, not on the boundary. In other words, equation (18) is not sufficient to characterize the behavior of $V(k, z)$ over $\bar{\mathcal{X}}$. Additional information is required about the solution V along the boundary $\partial\bar{\mathcal{X}}$. The most general form of boundary condition we consider in this paper on the d-dimensional domain $x \in \Omega \subset \mathbb{R}^d$ takes the form

$$\mathcal{B}(x, V, V_x, V_{xx}) = 0\,, \qquad \text{for } x \in \partial\bar{\mathcal{X}} \tag{30}$$

Simply put, equation (30) specifies extra conditions for the grid points along the boundary, which help us to fully pin down V on $\bar{\mathcal{X}}$. While these conditions could in general depend on any partial derivative of V, we restrict attention to the case where at most second-order mixed partial derivatives appear in boundary conditions. Most economic applications we have encountered

(a) Differential equation holds in the interior

(b) Differential equation holds on entire grid

Figure 11: Alternative Approaches to Implement Boundary Conditions

Note. Figure 11 illustrates two popular approaches to implement the boundary conditions of a differential equation. Panels (a) and (b) plot a grid that discretizes the one-dimensional domain $[0, 1]$. The first approach—panel (a)—assumes the differential equation only holds in the interior of the state space (red), using the boundary conditions to characterize the function values along the boundary (blue). The second approach—panel (b)—assumes the differential equation holds at the boundaries as well (now red), and the boundary conditions are used to characterize function values for a set of hypothetical exterior nodes just outside the boundary (blue).

satisfy this restriction.

In the case of the neoclassical growth model, for example, we require four boundary conditions to fully pin down $V(k, z)$ on $[0, \bar{k}] \times [\underline{z}, \bar{z}]$. In the k dimension, we impose two *state-constrained boundary conditions*, one each at 0 and $\bar{k}$, to ensure that households never decumulate and accumulate capital when they are at $k = 0$ and $k = \bar{k}$, respectively.[22] In the z dimension, we impose *reflecting boundary conditions* which similarly ensure that households at $\underline{z}$ and $\bar{z}$ never "leave" the region $[\underline{z}, \bar{z}]$.

4.1 Boundary Conditions on Uniform Grids

To start our discussion of boundary conditions, consider a uniform grid over $[0, \bar{k}] \times [\underline{z}, \bar{z}]$ with grid points $x_{ij} = (k_i, z_j)$ for $i \in \{1, \ldots, I\}$ and $j \in \{1, \ldots, J\}$.[23] There are two popular approaches to implement boundary conditions of the form (30) on such a grid. The first approach considers $\{x_{ij} \mid i \in \{1, I\} \text{ and } j \in \{1, J\}\}$ as the boundaries and treats the associated differential equation as defined only on $\{x_{ij} \mid 2 \leq i \leq I - 1 \text{ and } 2 \leq j \leq J - 1\}$. Under this approach, we use (27) to construct a linear system of $(I - 2) \times (J - 2)$ equations, corresponding only to the interior of the grid. We illustrate this approach in panel (a) of Figure 11 for the one-dimensional case. We consider a uniform grid over the interval $[0, 1]$ comprising 9 grid points. The points colored in red make up the interior of the grid, while the two points colored in blue are the boundary points. A finite-difference method following the first approach would use the boundary conditions (30) to characterize the function values on the two boundary points, while solving a system of 7 linear equations for the function values on the remaining interior points.

We follow the second approach, which treats the differential equation (18) as if it was defined on the entire grid, i.e., for all $\{x_{ij} \mid 1 \leq i \leq I \text{ and } 1 \leq j \leq J\}$. Under this approach, we work with a set of hypothetical *exterior nodes* that are placed just outside the boundaries of the state space $\bar{\mathcal{X}}$.

[22] See Achdou et al. (2021) for a detailed discussion of state-constrained boundary conditions.

[23] Formally, we define $k_i = \frac{(i-1)\Delta k - 0}{\bar{k} - 0}$ for $i \in \{1, \ldots, I\}$ and $z_j = \frac{(j-1)\Delta z - \underline{z}}{\bar{z} - \underline{z}}$ for $j \in \{1, \ldots, J\}$.

For the one-dimensional case, this is illustrated in panel (b) of Figure 11, where the two exterior nodes -0.125 and 1.125, colored in blue, are added. We denote these boundary points with indices $i \in \{0, I+1\}$ and $j \in \{0, J+1\}$. In other words, we think of the boundary conditions (30) as holding *just outside* the actual boundary of the state space.

If we discretize equation (18) in this way, we obtain a linear system of $I \times J$ equations in the $I \times J$ unknowns V_{ij}. And the discretized boundary conditions (30) inform the behavior of V at the hypothetical exterior nodes. To be more concrete, consider the grid points x_{1j} and the derivative $(D_k^- V)_{1j} = (V_{1j} - V_{0j})/\Delta k$ using the backward stencil. The problem is, of course, that we do not have sufficient information in the $I \times J$ system of linear equations to tell us anything about V_{0j}, which is our hypothetical exterior node. Discretizing the boundary condition (30) at the grid points x_{1j} yields

$$\mathcal{B}(x_{1j}, V_{1j}, V_{0j}) = 0 \tag{31}$$

for all j. The term V_{0j} shows up because it is used in the finite-difference discretization we consider for V_k. The equations (31) provide the additional information to pin down V_{0j}, which we then use in the $I \times J$ linear system (27) that discretizes the differential equation itself.

The following result is trivial but worthy of emphasis.

Lemma 8. *Under any linear finite-difference stencil, the boundary condition $\mathcal{B}(\cdot)$ defined in equation (30) is invertible. Its discretization can therefore be used to solve explicitly for the implied exterior nodes in all dimensions.*

Under Lemma 8, it follows that we can invert the discretized boundary condition to solve explicitly for V_{0j}, which we denote $V_{0j} = b(x_{1j}, V_{1j})$. Crucially, this allows us to redefine the standard backward finite-difference stencil at k_1 as $(D_k^- V)_{1j} = V_{1j}/\Delta k - b(x_{1j}, V_{1j})/\Delta k$, which is now pinned down in terms of V_{1j}. The same argument applies for other stencils and in the other dimensions. Therefore, we have successfully substituted out all exterior nodes and re-expressed all finite difference stencils in terms of only those grid points x_{ij} which are on the grid and, crucially, for whose corresponding value V_{ij} we have an equation in the linear system.

It is worth noting explicitly that while the exterior nodes $\{x_{0j}, x_{J+1\,j}, x_{i0}, x_{I+1\,0}\}$ serve an important role in our treatment of boundary conditions, they never explicitly appear in the implementation of the resulting linear system because we expressed them in terms of interior nodes using equation (30).

Lemma 9. *When $b(\cdot)$ is linear in the value function and all its derivatives, then we can compute boundary-adjusted finite-difference matrices outside of the value function iteration step using*

$$\partial_j V \approx \boldsymbol{D}_j^{\text{int}} \boldsymbol{V} + (\boldsymbol{D}_j^{\text{bound}} \boldsymbol{V} + \text{const}_j). \tag{32}$$

Intuitively, Lemma 9 establishes that, for a broad class of boundary conditions, we can solve for the implied function values on exterior ghost nodes in terms of the interior nodes, and then modify the finite-difference matrices accordingly. In the discretization of the partial derivative ∂_j on the compact state space $\bar{\mathcal{X}}$ in equation (32), $\boldsymbol{D}_j^{\text{int}}$ is the same standard finite-difference matrix we have used thus far in Sections 2 and 3, $\boldsymbol{D}_j$, except that we set the *rows* corresponding to nodes on the boundary to 0. The second term in equation (32), $\boldsymbol{D}_j^{\text{bound}}\boldsymbol{V} + \text{const}_j$, returns the derivative values on the boundary. In particular, a constant term const_j may be necessary here for some types of boundary conditions, e.g., the von-Neumann boundary condition $\partial_j V(x) = \kappa(x)$ for $x \in \partial\bar{\mathcal{X}}$. We discuss this further in Section 4.3.

Lemma 9 suggests a two-step procedure: First, we discretize all boundary conditions (30) and use them to derive a linear solution for the function values $\boldsymbol{V}$ on the exterior ghost nodes. Second, we solve the $I \times J$ linear system (27) using the boundary-adjusted finite-difference matrices (32).

The main limitation of Lemma 9 is when $b(\cdot)$ depends on prices or other endogenous variables that depend on the value function. When the conditions of Lemma 9 are not satisfied, then the finite-difference matrices must be re-computed in each iteration n of the algorithm as functions of the current value function guess $\boldsymbol{V}^n$. For expositional simplicity, we assume in the following that Lemma 9 is satisfied.

With these boundary-adjusted finite-difference matrices in hand, we can now extend the finite-difference scheme (21) to the entire grid over the compact state space $\bar{\mathcal{X}}$, even though equation (18) is only formally defined on the interior. This results in the following finite-difference scheme

$$\begin{aligned}\frac{1}{\Delta t}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n) + \rho\boldsymbol{V}^{n+1} =& u\Big(\boldsymbol{c}^n\Big) + \Big(\boldsymbol{F} - \boldsymbol{c}^n - \delta\boldsymbol{k}\Big)\Big(\boldsymbol{D}_k^{\text{int}}\boldsymbol{V}^{n+1} + \boldsymbol{D}_k^{\text{bound}}\boldsymbol{V}^{n+1} + \text{const}_k\Big) \\ &- \theta\boldsymbol{z}\Big(\boldsymbol{D}_z^{\text{int}}\boldsymbol{V}^{n+1} + \boldsymbol{D}_z^{\text{bound}}\boldsymbol{V}^{n+1} + \text{const}_z\Big) \\ &+ \frac{\sigma^2}{2}\Big(\boldsymbol{D}_{zz}^{\text{int}}\boldsymbol{V}^{n+1} + \boldsymbol{D}_{zz}^{\text{bound}}\boldsymbol{V}^{n+1} + \text{const}_{zz}\Big),\end{aligned} \tag{33}$$

where $\boldsymbol{D}_k^{\text{bound}}$, $\boldsymbol{D}_z^{\text{bound}}$, and $\boldsymbol{D}_{zz}^{\text{bound}}$, as well as const_k, const_z, and const_{zz}, depend on the types of boundary conditions used. For the neoclassical growth model, we use von-Neumann boundary conditions at $k \in \{0, \bar{k}\}$ that ensure households do not save (dissave) at the right (left) boundary of the capital grid, as well as reflecting boundaries at $z \in \{\underline{z}, \bar{z}\}$. For further discussion, see Section 4.3.

The local truncation error of this boundary-adjusted finite-difference scheme is now given by $\tau_{l,i}^n = \tau^n(\boldsymbol{x}_{l,i})$, with

$$\begin{aligned}\tau^n(k,z) = &\frac{1}{\Delta t}(V^{n+1}(k,z) - V^{n+1}(k,z)) + \rho V^{n+1}(k,z) - u\Big(c^n(k,z)\Big) \\ &- \Big(F(k,z) - c^n(k,z) - \delta k\Big)D_k^{\text{BC}} \circ V^{n+1}(k,z) + \theta z D_z^{\text{BC}} \circ V^{n+1}(k,z) - \frac{\sigma^2}{2}D_{zz}^{\text{BC}} \circ V^{n+1}(k,z),\end{aligned} \tag{34}$$

where D_k^{BC}, D_z^{BC}, and D_{zz}^{BC} denote the operator analogs of equation (32). As before, the LTE evaluates the finite-difference scheme by plugging in the true function values $V^{n+1}(k,z)$. Crucially and unlike our discussion in Section 2, we have now formally characterized the $J \times 1$ vector $\boldsymbol{\tau}^n = \{\tau_{l,i}^n\}$ over the entire grid, including the boundary nodes.

Proposition 10. (**Boundary-Adjusted Finite-Difference Scheme on Uniform Grids**) *Let G be a uniform grid with mesh size h over the compact state space $\bar{\mathcal{X}}$ including its boundary $\partial\bar{\mathcal{X}}$. Construct the sequence $\{\boldsymbol{V}^n\}$, where $\boldsymbol{V}^n \in \mathbb{R}^J$, recursively according to (33) using the boundary-adjusted standard finite-difference matrices defined in (32). Then the local truncation error defined in (34) converges uniformly, with $\tau_{l,i}^n \to 0$ as $\boldsymbol{l} \to \infty$ and $\Delta t \to 0$.*

The remainder of Proposition 4 extends to the boundary-adjusted scheme as well.[24] On uniform grid, the discretization scheme based on standard finite differences can therefore be extended to the entire grid over the compact state space, including its boundaries, by directly encoding the boundary conditions (30) in the finite-difference matrices.

4.2 Boundary Conditions on Sparse Grids

We now show that the strategy we outline above generalizes to sparse grids.

Definition 8. (**Boundary-Adjusted Sparse Finite-Difference Matrices**) *The first- and second-order sparse finite-difference matrices are defined as*

$$\boldsymbol{D}_j^{\pm,\text{S-bc}} = \boldsymbol{E}_j \left(\boldsymbol{D}_j^{\pm,\text{ int}} + \boldsymbol{D}_j^{\pm,\text{ bound}}\right) \boldsymbol{H}_j \quad \text{and} \quad \boldsymbol{D}_{jj}^{\text{S-bc}} = \boldsymbol{E}_j \left(\boldsymbol{D}_{jj}^{\text{int}} + \boldsymbol{D}_{jj}^{\text{bound}}\right) \boldsymbol{H}_j, \tag{35}$$

where $\boldsymbol{D}_\bullet^{int}$ and $\boldsymbol{D}_\bullet^{bound}$ are the boundary-adjusted standard finite-difference matrices introduced in (32), and $\boldsymbol{E}_\bullet$ and $\boldsymbol{H}_\bullet$ are hierarchization matrices.

With the sparse finite-difference matrices appropriately extended to the grid boundary, we can now write down a consistent finite-difference scheme for sparse grids. Let

$$\begin{aligned}\frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}^n}{\Delta} + \rho \boldsymbol{V}^{n+1} = u\left(\boldsymbol{c}^n\right) + \left(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\right)\left(\boldsymbol{D}_k^{\text{S-bc}} \boldsymbol{V}^{n+1} + \text{const}_k\right) \\ - \theta \boldsymbol{z}\left(\boldsymbol{D}_z^{\text{S-bc}} \boldsymbol{V}^{n+1} + \text{const}_z\right) + \frac{\sigma^2}{2}\left(\boldsymbol{D}_{zz}^{\text{S-bc}} \boldsymbol{V}^{n+1} + \text{const}_{zz}\right).\end{aligned} \tag{36}$$

We again define the associated local truncation error as $\tau_{l,i}^{n,\text{S-bc}}$. The following Proposition is the main result of this section, showing that the boundary-adjusted sparse finite-difference scheme (36) is consistent.

[24] We are working on extending the proof to the remaining conditions of Proposition 4 in ongoing work.

Proposition 11. **(Consistency of Boundary-Adjusted Sparse Finite-Difference Scheme)** *Let G be a regular sparse grid of level $\boldsymbol{l}$ over the compact state space $\bar{\mathcal{X}}$, including its boundary $\partial\bar{\mathcal{X}}$. Consider the sequences of function values $\{\boldsymbol{V}^n\}$ and LTEs $\{\boldsymbol{\tau}^{n,\,S\text{-}bc}\}$ defined by (36). Then:*

(i) The boundary-adjusted sparse finite-difference matrices $D_j^{\pm,\text{S-bc}}$ and $D_{jj}^{\text{S-bc}}$, together with the constant terms const_j, lead to consistent discretizations of first- and second-order partial derivatives in dimension j, including at the boundaries.

(ii) The boundary-adjusted sparse finite-difference scheme (36) is a consistent discretization of the HJB equation (18) over the state space $\bar{\mathcal{X}}$. That is, $\tau_{\boldsymbol{l},\boldsymbol{i}}^{n,\,\text{S-bc}} \to 0$ for all $\boldsymbol{i} \in I_{\boldsymbol{l}}^{\text{H}}$ and n as $\boldsymbol{l} \to \infty$ and $\Delta t \to 0$.

4.3 Types of Boundary Conditions

Representation (30) nests several types of boundary conditions encountered in economic applications. We briefly discuss the most prominent of these.

The *von-Neumann* boundary condition takes the form

$$\partial_x V(x) = \kappa(x)\,, \qquad \text{for } x \in \partial\bar{\mathcal{X}} \tag{37}$$

and specifies first-order partial derivatives along the boundary in terms of an exogenously given function $\kappa : \bar{\mathcal{X}} \to \mathbb{R}$. The most prominent example of a von-Neumann boundary condition in macroeconomics is the borrowing constraint (Huggett, 1993; Aiyagari, 1994).

The *reflecting* boundary condition is a special case of the von-Neumann condition, with $\partial_x V(x) = 0$. In economic terms, it signifies that the problem features no drift outside the boundary of the state space. In practice it is often used as a default boundary condition for exogenous processes such as earnings risk or productivity.

The *Dirichlet* boundary condition takes the form

$$V(x) = \kappa(x)\,, \qquad \text{for } x \in \partial\bar{\mathcal{X}}. \tag{38}$$

Unlike the von-Neumann condition, it directly specifies the level of the function $V(\cdot)$ along the boundary. Dirichlet conditions appear in many economic applications. In life-cycle models, for example, Dirichlet conditions are often used to pin down a terminal condition on lifetime utility at the time of death. When computing transition dynamics, Dirichlet conditions are used as initial or terminal conditions for the transition paths under consideration.

Finally, several economic applications feature boundary conditions that restrict the second-order mixed derivatives. In asset pricing and portfolio choice problems, for example, frequently feature boundary conditions of the form

$$\partial_{xx} V(x) = \kappa(x)\partial_x V(x)\,, \qquad \text{for } x \in \partial\bar{\mathcal{X}}. \tag{39}$$

In the portfolio choice context, condition (39) may be used at the high-wealth boundary to encode the asymptotically linear behavior of consumption for a class of utility functions (Achdou et al., 2021).

5 Use Cases and Applications

In this section, we demonstrate the power of adaptive sparse grid methods across a wide range of dynamic programming applications in economics.

5.1 High-Dimensional Dynamic Programming

Solving dynamic programming problems in high dimensions remains challenging due to the curse of dimensionality (Bellman, 1961). The potentially greatest promise of adaptive sparse grid methods in economics is their efficiency in high dimensions. Regular sparse grids reduce the complexity of a grid in d dimensions from $\mathcal{O}(n^d)$ to $\mathcal{O}(n \cdot \log(n)^{d-1})$, where n is the number of unique nodes per dimension (Zenger, 1991; Bungartz, 1992). Adaptive sparse grids can achieve significant further efficiency gains by refining grids adaptively based on the underlying economic application.

High-dimensional dynamic programming applications abound in economics. One rapidly growing area of research, for example, studies the implications of cross-sectional heterogeneity. Globally solving dynamic general equilibrium models with rich heterogeneity remains a serious challenge, however: Heterogeneous-agent models often feature a high-dimensional state space because, in general equilibrium, the entire cross-sectional distribution of agents becomes part of the aggregate state of the economy. Nevertheless, global solutions are necessary to study many economic applications such as uncertainty, asset pricing, and occasionally binding constraints that give rise to crisis regions.

We illustrate the power of adaptive sparse grids to tackle dynamic programming problems in high dimensions by computing a high-dimensional, global solution of the benchmark Krusell and Smith (1998) model. In particular, we employ the algorithm proposed by Schaab (2020) to solve the model with a 14-dimensional distribution representation. The power of adaptive sparse grids allows us to compute this global solution of the model and its simulation on a 17-dimensional state space in less than 5 minutes on a personal computer.

Model. There is a continuum of households with preferences over consumption, given by $\mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t) dt$, where ρ is a common discount rate and c_t the rate of consumption. Households face both uninsurable idiosyncratic risk in the form of unemployment spells and aggregate risk. Insurance markets are incomplete but households can trade capital, which they rent to firms. A household's budget constraint is given by $\dot{k}_t = (r_t^k - \delta)k_t + (1-\tau)w_t z_t + \tau^{\text{UI}}(z_t) - c_t$, subject to the short-sale constraint $k_t \geq 0$, where $r_t^k - \delta$ is the rental rate of capital net of depreciation, τ is a

labor income tax, and $\tau^{\text{UI}}(z_t)$ is unemployment insurance. We assume that z_t follows a two-state Poisson process.

A representative and perfectly competitive firm uses capital and labor to produce the final consumption good according to the aggregate production function $Y_t = e^{Z_t} K_t^{\alpha} N_t^{1-\alpha}$, where Z_t is aggregate TFP, and K_t and N_t denote the aggregate capital stock and labor, respectively. α denotes the income share of capital. Under perfect competition, factor prices are simply given by $r_t^k = \alpha \frac{Y_t}{K_t}$ and $w_t = (1-\alpha)\frac{Y_t}{L_t}$.

The role of the government is to provide unemployment insurance, which it funds via income taxation. We solve residually for the income tax rate τ so that the government budget is balanced. Since the process z_t is stationary, the unemployment rate in this model is constant by a law of large numbers argument, so the required labor tax rate is also time-invariant. The markets for capital, labor and the consumption good must clear at all times.

Recursive representation and state space. Aggregate uncertainty in this environment derives from the aggregate productivity process Z_t, which follows the mean-reverting process $dZ_t = -\theta Z_t dt + \sigma dB_t$, where B_t is standard Brownian motion. We denote by $g_t(k,z)$ the joint density of households over capital and earnings. In general equilibrium, households must forecast prices which, in turn, depend on the realization of future cross-sectional distributions. As a result, the aggregate state of our economy is (Z_t, g_t). The state space of the household's dynamic problem is then given by (k_t, z_t, Z_t, g_t).

To reduce the dimensionality of the dynamic programming problem, we follow Schaab (2020) and posit a high- but finite-dimensional approximation of the cross-sectional distribution, given by

$$g_t(k,z) \approx \hat{g}_t(k,z) = F(\boldsymbol{\alpha}_t)(k,z),$$

where F is a set of basis functions that are parametrized by $\boldsymbol{\alpha}_t \in \mathbb{R}^N$. As in Schaab (2020), we solve a sequence of *approximate economies*, in which households form expectations *as if* future cross-sectional distributions were given by $\hat{g}_t(k,z)$. Increasing N and choosing a flexible class of basis functions allows for an increasingly accurate representation $g_t(k,z) \approx F(\boldsymbol{\alpha}_t)(k,z)$. Crucially, the state variables of the household problem are now given by $(k_t, z_t, Z_t, \boldsymbol{\alpha}_t)$, implying a $3+N$-dimensional state space. We leverage adaptive sparse grids to solve the resulting high-dimensional dynamic programming problem.

Calibration. On the household side, we calibrate $\rho = 0.05$ and assume isoelastic preferences with $u(c) = \frac{1}{1-\gamma} c^{1-\gamma}$ and coefficient of relative risk aversion $\gamma = 2$. Households face uninsurable earnings risk encoded in the two-state Markov process $z_t \in \{z^{\text{E}}, z^{\text{U}}\}$. We model z_t using a Poisson process with arrival rates $\lambda(z)$. For simplicity, we set $z^{\text{E}} = 1.2$ and $z^{\text{U}} = 0.8$, as well as $\lambda(z^{\text{E}}) = \lambda(z^{\text{U}}) = 1/3$. Similarly, we set $\tau = \tau^{\text{UI}}(z) = 0$. For aggregate TFP risk, we set $\theta = 0.05$ and $\sigma = 0.007$. On the production side, we calibrate a capital share of $\alpha = 0.33$ and set the rate of capital

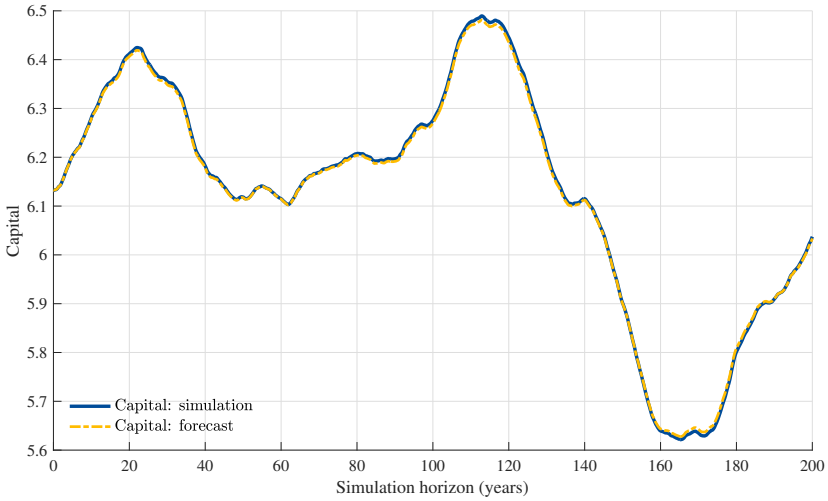


Figure 12: Adaptive Sparse Grids in High-Dimensions

Note. Figure 12 plots a global solution of the Krusell and Smith (1998) model using the projection algorithm of Schaab (2020), parameterizing the joint density over capital and earnings with 14 Chebyshev polynomials. The solid blue line corresponds to a simulated time path of aggregate capital, K_t^{sim}. The dashed yellow line corresopnds to an unconditional and out-of-sample forecast of aggregate capital by households, denoted K_t^{lom}, observing only the initial state of the economy and the sequence of realized TFP shocks. The distance between these two sequences is proportional to the Den Haan (2010) error metric ϵ_K^{DH}.

depreciation to $\delta = 0.05$.

While Schaab (2020) develops a non-parametric estimation algorithm to choose an efficient set of basis functions $F(\cdot)$, we simply use Chebyshev polynomials here for illustration. We set $N = 14$, so that the marginal distribution of capital holdings for each earnings type j, $g_t(k, z^j)$, is approximated by 7 Chebyshev polynomials. Time variation in the underlying cross-sectional distribution is captured by the time-varying basis function coefficients $\boldsymbol{\alpha}_t$.

Performance and Accuracy. Figure 12 presents the numerical solution of the model. We initialize the economy at the stationary equilibrium without aggregate uncertainty at time $t = 0$ and simulate the economy under a sequence of aggregate TFP shocks. The blue solid line corresponds to the time series of aggregate capital, denoted K_t^{sim}. The dashed yellow line, on the other hand, plots an unconditional and out-of-sample time-0 forecast by households, denoted K_t^{lom}. When making this forecast, households only observe the initial state of the economy and the realized sequence of aggregate TFP shocks.[25]

[25] The first simulation of the model, which yields K_t^{sim}, updates the cross-sectional household distribution g_t^{sim} at each time step by directly simulating the consumption and savings decisions of households at the micro level. This corresponds to a kind of Monte Carlo simulation. In discrete time, the Young (2010) algorithm is typically used for this, whereas in continuous time we can directly use the Kolmogorov forward equation. When computing households' unconditional forecast K_t^{lom}, on the other hand, we directly use $d\boldsymbol{\alpha}_t$ to compute $d\hat{g}_t$ to simulate the approximate distribution according to households' perceived law of motion.

To assess the accuracy of our solution method, we compute the Den Haan (2010) metric of our model solution, which is defined as $\epsilon_K^{\text{DH}} = 100 \times \max_t \log\{K_t^{\text{lom}} - K_t^{\text{sim}}\}$. We can interpret ϵ_K^{DH} as the largest percentage error in an unconditional forecast of capital given the realized draw of shocks $\{Z_t\}$. When ϵ_K^{DH} is small, household beliefs are, by construction, consistent with the true law of motion of the economy, which is the required condition for a rational expectations equilibrium. Simulating the model based on a numerical solution with 14 Chebyshev polynomials achieves an accuracy metric of $\epsilon_K^{\text{DH}} = 0.152$, which implies that the largest unconditional household forecast error is 0.15%. The entire solution and simulation of the model takes less than 5 minutes on a personal computer.[26]

5.2 Occasionally-Binding Constraints

Dynamic optimization problems in economics are often subject to constraints, such as collateral or borrowing constraints, due to which the value function becomes substantially more concave in one (often small) region of the state space. In other words, an agent's behavior is often substantially more sensitive to small changes in fundamentals when that agent is close to an occasionally-binding constraint. Sparse grid methods excel in such environments because grid points are added in the high-concavity region. We illustrate this in the Aiyagari (1994) model. Further details are presented in Appendix H.2.

We illustrate the power of adaptive sparse grids to handle occasionally-binding constraints in the context of the benchmark Aiyagari (1994) model with borrowing constraint.

Model. There is a continuum of households with preferences over consumption given by

$$V_0 = \mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t) dt \tag{40}$$

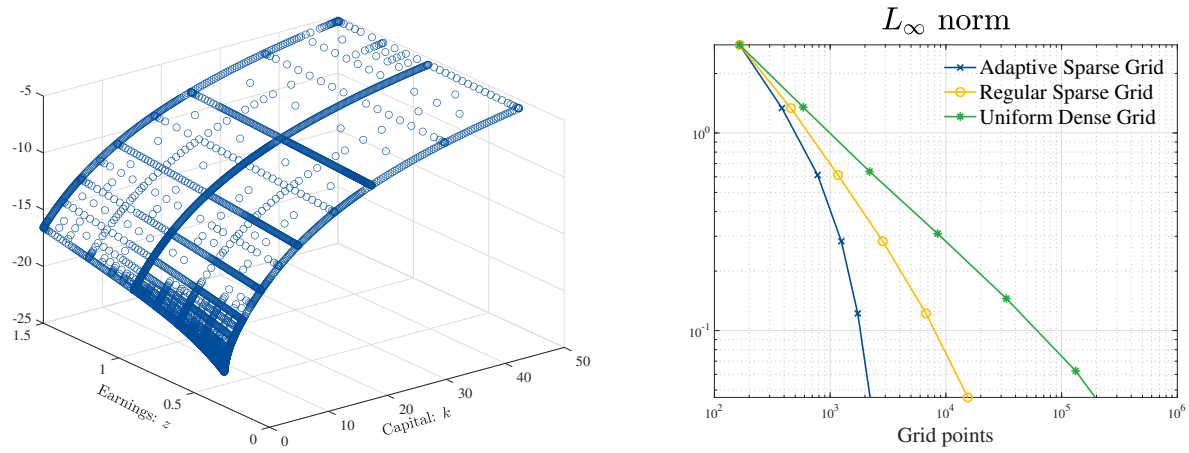
where c_t denotes the rate of consumption and ρ is a constant discount rate. Households can accumulate a stock of capital k_t and face the budget constraint

$$\dot{k}_t = (r_t - \delta)k_t + w_t e^{z_t} - c_t, \tag{41}$$

where r is an exogenously given, constant real interest rate, and z_t captures idiosyncratic earnings risk. Capital depreciates at rate δ. Household labor supply is implicitly taken as exogenous and normalized to 1, so that $w_t e^{z_t}$ corresponds to labor income, where w_t is the real wage rate. In particular, we assume that a household's idiosyncratic labor productivity follows a mean-reverting diffusion process, with

$$dz_t = -\theta z_t dt + \sigma dB_t, \tag{42}$$

[26] We use a 2019 16-inch MacBook Pro, with a 2.4 GHz 8-Core Intel i9 processor and 32 GB memory.



(a) Value Function

(b) Relative Errors and Convergence

Figure 13: Adaptive Sparse Grids and Occasionally-Binding Constraints

Note. Figure 13 presents the numerical solution of the Aiyagari (1994) model with an occasionally-binding borrowing constraint. Panel (a) displays the stationary value function $V(k,z)$ on an adaptive sparse grid that starts with a uniform grid of level $\boldsymbol{l}^0 = (5,2)$ and refines 8 times. Panel (b) plots the convergence rates of numerical errors in the $L_\infty^j = \max|V^{\text{fine}} - V^j|$ norm for a sample of 30,000 grid points, where V^{fine} is a solution of the value function on a uniform grid with 2.1 million grid points. $\{V^j\}$ corresponds to a sequence of numerical solutions on increasingly fine uniform, regular sparse, and adaptive sparse grids. We start with a uniform grid of level $\boldsymbol{l}^0 = (5,2)$ in each case and plot 5 successive refinement levels. For adaptive grid refinement, we use thresholds $\epsilon^{\text{add}} = 10^{-5}$ and $\epsilon^{\text{keep}} = 10^{-6}$.

where B_t is a standard Brownian motion that is uncorrelated across households. Finally, households face an ad-hoc borrowing constraint of the form

$$k_t \geq 0. \tag{43}$$

A representative and perfectly competitive firm operates the production technology $Y_t = K_t^\alpha L_t^{1-\alpha}$, where K_t and L_t denote the aggregate capital stock and labor, respectively. Factor prices are given by $r_t^k = \alpha \frac{Y_t}{K_t}$ and $w_t = (1-\alpha)\frac{Y_t}{L_t}$.

We compute the stationary equilibrium where all macroeconomic aggregates are constant. The markets for consumption goods and capital must clear at all times.

Recursive optimization problem. The household's problem is to choose consumption to maximize (40) subject to (41), (42), and (43). As in Section 2, this dynamic optimization problem admits a recursive representation, with capital and earnings as the two state variables. In the stationary equilibrium with constant interest and wage rates, i.e., $r_t = r$ and $w_t = w$, the value function solves the stationary Hamilton-Jacobi-Bellman equation

$$\rho V(k,z) = u(c(k,z)) + \Big(rk + we^z - c(k,z)\Big)\partial_k V(k,z) - \theta z \partial_z V(k,z) + \frac{\sigma^2}{2}\partial_{zz} V(k,z), \tag{44}$$

where the consumption policy function is defined by the first-order condition

$$u'(c(k,z)) = \partial_k V(k,z). \tag{45}$$

While equations (44) and (45) hold everywhere in the interior of the state space, a state constraint boundary condition characterizes the value function at the borrowing constraint (Achdou et al., 2021),

$$u'(r\underline{k} + we^z) \geq \partial_k V(\underline{k}, z)$$

for all z. This boundary inequality ensures that households never dissave at the borrowing constraint.

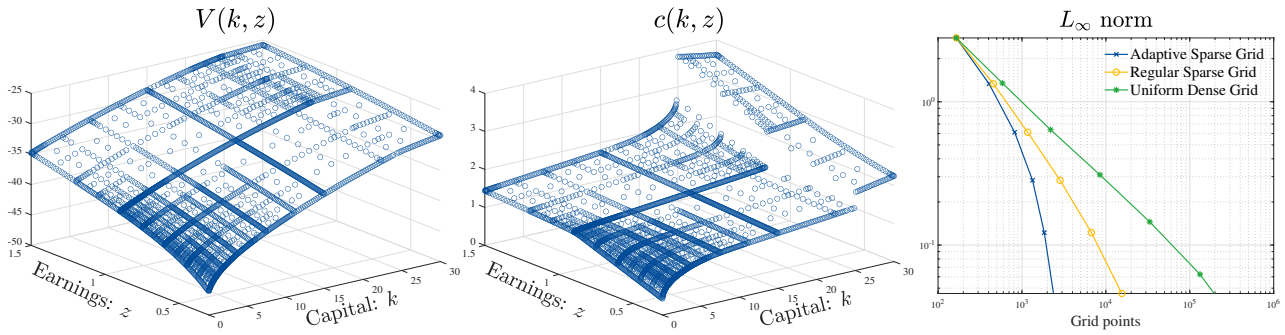
Calibration. We solve for the stationary household value function $V(k,z)$ on a grid over capital, $k \in [0, \bar{k}]$ and earnings $z \in [\underline{z}, \bar{z}]$, where we set $\bar{k} = 50$, $\underline{z} = 0.3$ and $\bar{z} = 1.5$. The household discount rate is $\rho = 0.02$. We adopt CRRA utility, with $u(c) = \frac{1}{1-\gamma}c^{1-\gamma}$, and set the coefficient of relative risk aversion to $\gamma = 2$. On the firm side, we set $\alpha = 0.33$ and $\delta = 0.05$. Finally, we calibrate the earnings process with $\theta = 0.25$ and $\sigma = 0.02$.

Numerical analysis. We numerically solve this variant of the Aiyagari (1994) model on sequences of uniform grids, regular sparse grids, and adaptive sparse grids. To assess the accuracy of the numerical solution in each case, we compare the resulting value function to a benchmark solution computed on a fine, uniform grid with over 2 million grid points.[27]

Figure 13 plots the household value function $V(k,z)$ on an adaptive sparse grid in panel (a). The grid adaptation procedure places grid points near the occasionally-binding borrowing constraint, especially at low earnings levels. Household behavior in this region is highly sensitive to marginal changes in wealth and earnings, and the value function consequently becomes increasingly non-linear near the constraint. On the other hand, the procedure removes grid points from other regions of the state space. This is particularly obvious in the high-wealth region, where the value function becomes increasingly linear as a function of both capital and earnings.

Panel (b) plots the L_∞ error convergence relative to the number of grid points used on the uniform, regular sparse, and adaptive sparse grids. The error decays slowly on the uniform grid and only slightly more quickly for regular sparse grids. Even in this two-dimensional example, the adaptive sparse grid registers substantial efficiency gains compared to alternative grid structures. Concretely, a numerical solution on an adaptive sparse grid with $2,840$ grid points is as accurate as a solution on a uniform grid with over $200,000$ grid points.

[27] As discussed in Achdou et al. (2021), numerical analysis in continuous time has no analog to the Euler condition error that is used in discrete time to evaluate the approximation error in dynamic programming applications. To assess the accuracy of our numerical solutions, we use as a reference the solution of the model on a very large, uniform grid with over 2 million grid points. This approach becomes inapplicable in higher dimensions, which in part motivates our focus on two-dimensional applications.



(a) Value Function (b) Consumption Policy Function (c) Relative Errors and Convergence

Figure 14: Adaptive Sparse Grids and Non-Convexities

Note. Figure 14 presents the numerical solution of the Aiyagari (1994) model with an occasionally-binding borrowing constraint and a non-convex capital income tax schedule. Panels (a) and (b) display the stationary value and consumption policy functions, $V(k,z)$ and $c(k,z)$ on an adaptive sparse grid that starts with a uniform grid of level $\boldsymbol{l}^0 = (5,2)$ and refines 6 times. The non-convex income tax schedule results in a discontinuity in $c(k,z)$ and a kink in $V(k,z)$ at $k = k^*$. Panel (c) plots the convergence rates of numerical errors in the $L_\infty^j = \max|V^{\text{fine}} - V^j|$ norm for a sample of 30,000 grid points, where V^{fine} is a solution of the value function on a uniform grid with 2.1 million grid points. $\{V^j\}$ corresponds to a sequence of numerical solutions on increasingly fine uniform, regular sparse, and adaptive sparse grids. We start with a uniform grid of level $\boldsymbol{l}^0 = (5,2)$ in each case and plot 5 successive refinement levels. For adaptive grid refinement, we use thresholds $\epsilon^{\text{add}} = 10^{-5}$ and $\epsilon^{\text{keep}} = 10^{-6}$.

5.3 Non-Linearities, Non-Convexities, and Kinks

Adaptive sparse grids are especially useful in the context of applications that exhibit non-convexities in agents' behavior. To illustrate this, we again consider a variant of the Aiyagari (1994) model and introduce a non-convex capital income taxation schedule.[28]

Model and calibration. The model is similar to that in Section 5.2, except that the household's budget constraint now takes the form

$$\dot{k}_t = \Big(1 - T(k_t)\Big) r k_t + e^{z_t} - c_t,$$

where $T(k_t)$ is a capital income tax schedule given by

$$T(k_t) = \begin{cases} 0 & \text{if } k \leq k^* \\ \kappa & \text{if } k > k^* \end{cases}$$

Wealthy households pay a constant marginal tax rate κ on financial income, while poorer households face no tax. For illustration, we set $\kappa = 0.02$ and $k^* = 27$ and otherwise adopt the same calibration also used above in Section 5.2.

[28] This application is also used in the review article on adaptive sparse grids by Brumm et al. (2022).

Numerical analysis. Panels (a) and (b) of Figure 14 again display a numerical solution of the value and consumption policy functions on an adaptive sparse grid. As before, the adaptation procedure adds grid points near the borrowing constraint. In the presence of a non-convex tax schedule, the consumption policy function $c(k, z)$ exhibits a jump at $k = k^*$ while the value function $V(k, z)$ features a kink. Relative to Figure 13, grid adaptation also adds grid points near k^*, where the value function now exhibits larger concavity. Panel (c) demonstrates the relative efficiency of adaptive sparse grids relative to both uniform and regular sparse grids in the context of this model with borrowing constraints and non-convexities.

5.4 Finite Horizons, Life-Cycle, and Overlapping Generations

Many economic applications feature finite horizons, including life-cycle and overlapping generations models. Such problems are typically solved by explicitly iterating over the time or age dimension, effectively solving a partial differential equation for every discretized time point. In practice, this often becomes the most computationally intensive step of the algorithm.

Adaptive sparse grids can substantially reduce the computational burden of life-cycle applications by simply adding age as an additional state variable or dimension. This approach has three key advantages: First, adaptive grid refinement efficiently places grid points in the age dimension where value and policy functions feature most non-linearity, typically near the boundaries. As we illustrate below, accurately characterizing agents' behavior in the retirement region often requires substantially more grid points than elsewhere.

Second, treating age as an additional dimension allows sparse grid methods to further reduce grid density by drawing on off-dimensional information. Under the standard approach, we construct a grid of J nodes in the $n-1$ dimensions excluding age and then explicitly iterate over the N points used to discretize the age dimension. This results in a "grid" of complexity $J \times N$ because the age dimension is effectively treated as a uniform grid. Even if we already use a sparse grid of size J for the $n-1$ dimensions excluding age, substantial efficiency gains can be realized by not iterating separately over age and instead discretizing all n dimensions jointly on a sparse grid.

Third, modeling age as just another dimension in our partial differential equation also allows us to us implicit and semi-implicit rather than explicit time marching schemes. In practice, the number of iterations required for VFI convergence under a semi-implicit scheme is small. For the model we show below, VFI convergence requires 7 iterations in $n-1$ dimensions excluding age and only 10 when adding age as an nth dimension. The standard approach instead requires first solving for the value function at one of the boundaries and then explicitly iterating N times, where N is the number of points used to discretize the age dimension.

We illustrate the power of adaptive sparse grids for finite-horizon problems in the context of a simple one-asset life-cycle model. Further details are presented in Appendix H.5.

Model. The model is again similar to the Aiyagari (1994) variant presented in Section 5.2, except that we introduce a life-cycle. We denote the age of a household by $a_t \in [\underline{a}, \bar{a}]$. We explicitly distinguish between t, which indexes the flow of time in the economy, and a_t, which denotes the age of a particular household at calendar time t. We interpret $\underline{a}$ as the start of working life and $\bar{a}$ as the time of death. Households that enter the economy at time 0 with $a_0 = \underline{a}$ now face the finite-horizon problem

$$V_0 = \mathbb{E}_0 \int_0^{\bar{a}-\underline{a}} e^{-\rho t} u(c_t) dt,$$

subject to the capital accumulation equation (41), the diffusion process for labor productivity (42), the borrowing constraint (43), as well as the law of motion for the new state variable age, which is trivially given by

$$\dot{a}_t = 1. \tag{46}$$

Crucially, the household consumption policy function now depends on both calendar time t and age a separately, that is, $c_t = c_t(a, k, z)$.

Stationary equilibrium and recursive representation. As before, we focus on the stationary equilibrium of this economy, where the production side is the same as in Section 5.2. All macroeconomic aggregates are constant and we can drop calendar time t from the problem.

We seek a recursive formulation of the household's finite-horizon problem with state variables (a, k, z). Having introduced age as an additional state variable raises the question of boundary conditions for the new boundaries $a \in \{\underline{a}, \bar{a}\}$. Since the evolution of age is given by (46), which is a simple advection force, we only require one new boundary condition. Following Achdou et al. (2021), we impose a terminal condition on the household's lifetime value at death, $a = \bar{a}$, given by

$$V(\bar{a}, k, z) = \epsilon_1 u(\epsilon_2 + k). \tag{47}$$

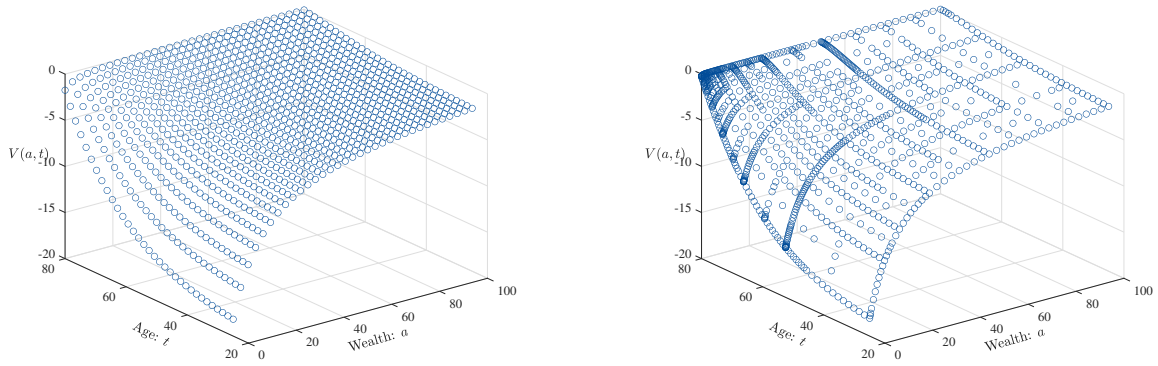
This represents a Dirichlet boundary condition (see Section 4).

In the interior of the state space, the household's lifetime value now solves the HJB equation

$$\begin{aligned} \rho V(a,k,z) = u(c(a,k,z)) + \partial_a V(a,k,z) + \Big(rk + we^z - c(a,k,z)\Big)\partial_k V(a,k,z) \\ - \theta z \partial_z V(a,k,z) + \frac{\sigma^2}{2}\partial_{zz} V(a,k,z), \end{aligned} \tag{48}$$

implying a first-order condition for consumption given by $u'(c(a,k,z)) = \partial_k V(a,k,z)$.

Calibration. We set $\underline{a} = 25$ as the start of working life and $\bar{a} = 80$ as the time of death. For the new terminal condition for the household's lifetime value at death, we set $\epsilon_1 = 10^{-8}$ and $\epsilon_2 = 10^{-5}$. Since visually illustrating grid structures is easiest in two dimensions, we model earnings risk as a two-state Markov chain, with $z_t \in \{z^{\mathrm{L}}, z^{\mathrm{H}}\}$ and Poisson transition rates $\lambda = 1/3$ out of both states.



(a) Uniform Grid

(b) Adaptive Sparse Grid

Figure 15: Adaptive Sparse Grids and Life-Cycle / OLG Economies

Note. Figure 14 presents the numerical solution of a life-cycle variant of the Aiyagari (1994) model. Panel (a) plots the stationary value function $V(a, k, z^L)$ for the low-earnings type on a uniform grid of level $\boldsymbol{l} = (5, 5)$. Panel (b) displays the solution on an adaptive sparse grid that starts with a uniform grid of lebel $\boldsymbol{l}^0 = (5, 5)$ and adapts 10 times, using thresholds $\epsilon^{\text{add}} = 5 \cdot 10^{-4}$ and $\epsilon^{\text{keep}} = 5 \cdot 10^{-6}$.

This allows us to cleanly illustrate the value function for a discrete earnings state as we do in Figure 15. The remainder of the calibration is as in Section 5.2.

Numerical analysis. In Figure 15, we plot the numerical solution of the stationary household value function for the low earnings type, $V(a, k, z^{\text{L}})$. Panel (a) plots the solution on a uniform grid while panel (b) exhibits an adaptive sparse grid featuring roughly the same number of grid points. It is again obvious how the grid adaptation procedure removes grid points from regions where the value function is approximately linear, while adding them to regions featuring substantial non-linearity. In this application, accuracy of the numerical solution requires increasing the grid density near the retirement boundary substantially. In fact, it is evident from panel (a) that a uniform age grid leads to substantialy approximation error given the terminal condition at retirement.

5.5 Free Boundary Problems

Adaptive sparse grids excel in the context of free boundary problems for two reasons. First, the value and policy functions of agents typically become increasingly non-linear close to the free boundary. Grid adaptation therefore allows us to place additional grid points close to the free boundary, while removing them from other regions where economic behavior is not as sensitive to changes in fundamentals. Second, once the free boundary is identified in the state space, the solution of the dynamic programming problem no longer depends on information from outside the free boundary. Grid adaptation allows us to drop most of the grid points outside the free boundary for additional efficiency gains.

We illustrate both of these features in the standard stopping time problem where a firm

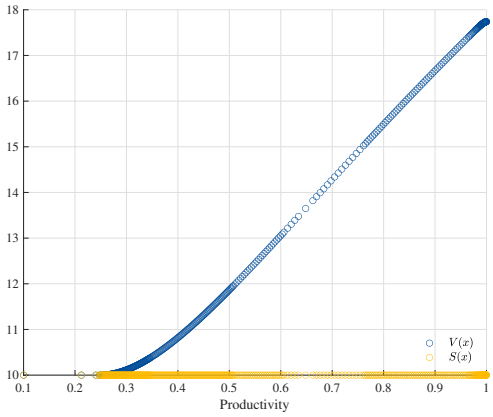


Figure 16: Adaptive Sparse Grids and Free Boundaries

Note. Figure 16 presents the numerical solution of the firm stopping time problem on an adaptive sparse grid that starts with a uniform grid of level $l = 5$ and adapts 8 times, using thresholds $\epsilon^{\text{add}} = 10^{-5}$ and $\epsilon^{\text{keep}} = 10^{-6}$. The blue dots correspond to the firm value function $V(x)$ for productivity levels $x \in [0.1, 1]$. The yellow dots correspond to the firm's scrap value $S(x) = 10$. The free boundary, at which the firm exits and shuts down its plant, takes the value $x^* = 0.23$.

optimally decides when to close its production plant. We follow the formulation of Achdou et al. (2021).

Model. Firm productivity x_t evolves according to the diffusion process

$$dx_t = \mu(x_t)dt + \sigma(x_t)dB_t, \tag{49}$$

where dB_t is a standard Brownian motion, and $\mu(\cdot)$ and $\sigma(\cdot)$ correspond to the potentially state-contingent drift and volatility of the process. The firm's lifetime value in state x_0 is defined by

$$V(x_0) = \max_{\tau} \mathbb{E}_0 \int_0^{\tau} e^{-\rho t} u(x_t)dt + e^{-\rho\tau} S(x_\tau), \tag{50}$$

where τ is the optimal stopping time of plant closure. While the plant is operating, the firm obtains cash flows $u(x_t)$ given productivity x_t, which it discounts at rate ρ. Finally, at the time of plant closure $t = \tau$, the firm realizes a scrap value $S(x_\tau)$.

The firm's problem is to maximize its lifetime value (50) by choosing optimally the time of plant closure τ, taking as given the evolution of productivity (49). This problem admits a recursive representation in the form of the Hamilton-Jacobi-Bellman variational inequality

$$\rho V(x) = \max\left\{u(x) + \mu(x)\partial_x V(x) + \frac{\sigma(x)^2}{2}\partial_{xx} V(x),\, \rho S(x)\right\}, \tag{51}$$

where $S(x)$ is the outside option of plant closure in state x.

Calibration. We set the firm's discount rate to $\rho = 0.05$ and adopt a CRRA functional form for utility, with $u(x) = \frac{1}{1-\gamma}x^{1-\gamma}$, with coefficient of relative risk aversion $\gamma = 0.5$. We set the firm's outside option, or plant scrap value, to $S(x) = 10$. Finally, we choose a constant productivity drift of $\mu(x) = -0.1$ and volatility $\sigma(x) = 0.01x$.

Numerical analysis. Figure 16 plots the numerical solution of the firm value function $V(x)$ on an adaptive sparse grid. The endogenous free boundary is at $x^* = 0.23$. Figure 16 illustrates two important features of the adaptive sparse grid solution of this optimal stopping time problem. First, the value function becomes increasingly non-linear close x^*, and the grid adaptation procedure consequently places a large number of grid points in this region. On the other hand, firm value $V(x)$ becomes increasingly linear away from the free boundary, allowing us to reduce grid density in other regions of the state space.

Second, the adaptive sparse grid removes nearly all grid points in the interval $[0.1, x^*)$, which results in substantial efficiency gains relative to a uniform grid. Once the free boundary is identified, solving the firm's stopping time problem requires no information about the economy outside the free boundary. That is, the Hamilton-Jacobi-Bellman equation (51) does not require solving for the firm value — or, indeed, the scrap value — in the interval $[0.1, x^*)$.

6 The `SparseEcon` Dynamic Programming Repository

An online repository accompanies this paper, providing pedagogical code and tutorials for many prominent dynamic programming applications across several fields of economics, including macroeconomics, asset pricing, industrial organization, and game theory. This repository is available at `https://github.com/schaab-lab/SparseEcon`. Our hope is that our repository will help make our method broadly accessible.

7 Conclusion

In this paper, we propose a new approach to dynamic programming in continuous time leveraging adaptive sparse grids. We develop a sparse finite-difference method and a value function iteration algorithm that are robust across a broad class of non-uniform grids. Our algorithm automatically adapts the grid and adds local resolution in regions of the state space where the value function approximation error remains large. We demonstrate the power and versatility of our approach across a wide range of applications in economics that feature high-dimensional state spaces, occasionally-binding constraints, life-cycles and overlapping generations, kinks and non-convexities, discrete choice, free boundaries, and dynamic games.

References

Achdou, Yves, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll. 2021. Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach.

Ahn, S, Greg Kaplan, Benjamin Moll, Thomas Winberry, and Christian Wolf. 2017. Micro Heterogeneity and Aggregate Consumption Dynamics. *NBER Macroeconomics Annual, forthcoming*.

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

Babenko, Konstantin Ivanovich. 1960. Approximation of periodic functions of many variables by trigonometric polynomials. In *Doklady Akademii Nauk*, volume 132, pp. 247–250. Russian Academy of Sciences.

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

Baszenski, G, F-J Delvos, and S Jester. 1992. Blending approximations with sine functions. In *Numerical Methods in Approximation Theory, Vol. 9*, pp. 1–19. Springer.

Bellman, Richard E. 1961. *Adaptive control processes*.

Brumm, Johannes, Christopher Krause, Andreas Schaab, and Simon Scheidegger. 2022. Sparse grids for dynamic economic models.

Brumm, Johannes and Simon Scheidegger. 2017. Using adaptive sparse grids to solve high-dimensional dynamic models. *Econometrica*, 85(5):1575–1612.

Bungartz, Hans-Joachim. 1992. *Dünne Gitter und deren Anwendung bei der adaptiven Lösung der dreidimensionalen Poisson-Gleichung*. Technische Universität München.

—. 1998. Finite elements of higher order on sparse grids. Ph.D. thesis, Technische Universität München.

Bungartz, Hans-Joachim and Michael Griebel. 1999. A note on the complexity of solving Poisson's equation for spaces of bounded mixed derivatives. *journal of complexity*, 15(2):167–199.

—. 2004. Sparse grids. *Acta numerica*, 13(1):147–269.

Daubechies, Ingrid. 1992. *Ten lectures on wavelets*. SIAM.

Delvos, F-J. 1982. d-variate Boolean interpolation. *Journal of Approximation Theory*, 34(2):99–114.

Den Haan, Wouter J. 2010. Comparison of solutions to the incomplete markets model with aggregate uncertainty. *Journal of Economic Dynamics and Control*, 34(1):4–27.

Deslauriers, Gilles and Serge Dubuc. 1989. Symmetric iterative interpolation processes. In *Constructive approximation*, pp. 49–68. Springer.

Ericson, Richard and Ariel Pakes. 1995. Markov-perfect industry dynamics: A framework for empirical work. *Review of Economic Studies*, 62(1):53–82.

Faber, Georg. 1909. Über stetige funktionen. *Mathematische Annalen*, 66(1):81–94.

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

Griebel, Michael. 1991. A parallelizable and vectorizable multi-level algorithm on sparse grids. In *Parallel algorithms for partial differential equations*.

—. 1998. Adaptive sparse grid multilevel methods for elliptic PDEs based on finite differences. *Computing*, 61(2):151–179.

Griebel, Michael, Michael Schneider, and Christoph Zenger. 1992. A combination technique for the solution of sparse grid problems. *Iterative Methods in Linear Algebra*, pp. 263–281.

Heinecke, Alexander, Stefanie Schraufstetter, and Hans-Joachim Bungartz. 2012. A highly parallel Black–Scholes solver based on adaptive sparse grids. *International Journal of Computer Mathematics*, 89(9):1212–1238.

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

Huggett, Mark. 1993. The risk-free rate in heterogeneous-agent incomplete-insurance economies. *Journal of economic Dynamics and Control*, 17(5):953–969.

Judd, Kenneth L, Lilia Maliar, Serguei Maliar, and Rafael Valero. 2014. Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain. *Journal of Economic Dynamics and Control*, 44:92–123.

Koster, Frank. 2002. Multiskalen-basierte Finite-Differenzen-Verfahren auf adaptiven dünnen Gittern.

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

Krusell, Per and Anthony A. Jr. Smith. 1998. Income and Wealth Heterogeneity in the Macroeconomy. *Journal of Political Economy*, 106(5):867–896.

LeVeque, Randall J. 2007. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM.

Merton, Robert C. 1973. An intertemporal capital asset pricing model. *Econometrica: Journal of the Econometric Society*, pp. 867–887.

Oswald, Peter. 2013. *Multilevel finite element approximation: Theory and applications*. Springer-Verlag.

Pflüger, Dirk Michael. 2010. Spatially adaptive sparse grids for high-dimensional problems. Ph.D. thesis, Technische Universität München.

Ruttscheidt, Steffen. 2018. Adaptive Sparse Grids for Solving Continuous Time Heterogeneous Agent Models.

Schaab, Andreas. 2020. Micro and Macro Uncertainty.

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

Schober, Peter. 2018. Solving dynamic portfolio choice models in discrete time using spatially adaptive sparse grids. In *Sparse Grids and Applications-Miami 2016*, pp. 135–173. Springer.

Smolyak, Sergei Abramovich. 1963. Quadrature and interpolation formulas for tensor products of certain classes of functions. In *Doklady Akademii Nauk*, volume 148, pp. 1042–1045. Russian Academy of Sciences.

Stokey, Nancy L and Robert E Lucas. 1989. *Recursive methods in economic dynamics*. Harvard University Press.

Young, Eric R. 2010. Solving the incomplete markets model with aggregate uncertainty using the Krusell–Smith algorithm and non-stochastic simulations. *Journal of Economic Dynamics and Control*, 34(1):36–41.

Yserentant, Harry. 1986. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412.

—. 1992. Hierarchical bases.

Zenger, Christoph. 1991. Sparse grids. In *Proceedings of the Research Workshop of the Israel Science Foundation on Multiscale Phenomenon, Modelling and Computation*, p. 86.