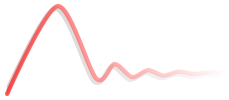

Dynare Working Papers Series
https://www.dynare.org/wp/

Scalable Global Solution Techniques for High-Dimensional Models in Dynare

Aryan Eftekhari
Michel Juillard
Normann Rion
Simon Scheidegger

Working Paper no. 86

November 2025

CEPREMAP
CENTRE POUR LA RECHERCHE ECONOMIQUE ET SES APPLICATIONS
48, boulevard Jourdan — 75014 Paris — France
https://www.cepremap.fr

Scalable Global Solution Techniques for High-Dimensional Models in Dynare

Aryan Eftekhari
Institute of Computing
USI Lugano
Switzerland
aryan.eftekhari@usi.ch

Michel Juillard
Banque de France
France
michel.juillard@mjui.fr

Normann Rion
CY Cergy Paris Université
CEPREMAP
France
normann@dynare.org

Simon Scheidegger*
Department of Economics
University of Lausanne
Switzerland
simon.scheidegger@unil.ch

November 17, 2025

Abstract

For over three decades, Dynare has been a cornerstone of dynamic stochastic modeling in economics, relying primarily on perturbation-based local solution methods. However, these techniques often falter in high-dimensional, non-linear models that demand more comprehensive approaches. This paper demonstrates that global solutions of economic models with substantial heterogeneity and frictions can be computed accurately and swiftly by augmenting Dynare with adaptive sparse grids (SGs) and high-dimensional model representation (HDMR). SGs mitigate the curse of dimensionality, as the number of grid points grows significantly slower than in traditional tensor-product Cartesian grids. Additionally, adaptivity focuses grid refinement on regions with steep gradients or non-differentiabilities, enhancing computational efficiency. Complementing SGs, HDMR tackles large state spaces by approximating policy functions with a hierarchical expansion of low-dimensional terms. Using a time iteration algorithm, we benchmark our approach on an international real business cycle model. Our results show that both SGs and HDMR alleviate the curse of dimensionality, enabling accurate solutions for at least 100-dimensional models on standard hardware in relatively short times. This advancement extends Dynare's capabilities beyond perturbation approaches, establishing a versatile platform for sophisticated non-linear models and paving the way for integrating the most recent global solution methods, such as those from machine learning.

JEL classification: C63, E30, F44.

Keywords: Adaptive Sparse Grids, High-dimensional Model Representation, Global Solution Methods, International Real Business Cycles, Time Iteration.

*We thank Lorenzo Bretscher, Johannes Brumm, Aurélien Eyquem, Galo Nuño, Tamás Simon, Dániel Sali, Gauthier Vermandel, and the participants of the "Celebrating Michel Juillard's Career and 30 Years of Dynare" conference for their valuable comments. This work is supported by a grant from the Swiss National Science Foundation under project ID "New Methods for Asset Pricing with Frictions."

1 Introduction

Motivation. Over the past three decades, Dynare (Juillard, 1994, Collard and Juillard, 2001, Adjemian et al., 2024) has become an indispensable tool for researchers and policymakers in macroeconomics and beyond. Its guiding principle, *"write your model almost as you would on paper, and Dynare will take care of the rest!"*, offers a user-friendly approach that has led to widespread adoption in academic and policy circles alike.

Dynare's intuitive framework and local solution methods have transformed the specification and analysis of dynamic stochastic models in economics. Yet, these methods struggle with problems involving substantial heterogeneity and non-linearities that demand global solutions,[2] such as those with financial frictions, the zero lower bound, rare disasters, and models with many agents (e.g., Krueger and Kubler, 2004, Brumm et al., 2015a, Fernández-Villaverde et al., 2015, Azinovic and Žemlička, 2023), or tipping points in climate-economic models (e.g., Cai and Lontzek, 2019, Kotlikoff et al., 2021). While perturbation methods are confined to a neighborhood of the steady state, *naive* global solution methods, such as basic Cartesian grid-based approaches requiring M^d points for M points per dimension, face exponential computational costs as dimensionality d grows. This challenge, driven, for instance, by the heterogeneity in a model, is commonly known as the *curse of dimensionality* (Bellman, 1961).

Goal and Contribution of this Article. Recognizing the limitations of local solution methods, this paper demonstrates that global solutions for economic models with substantial heterogeneity and frictions can be computed both accurately and fast by extending the Dynare environment—specifically, its `.mod`-files—to accommodate contemporary global solution techniques. We illustrate these enhancements by integrating two widely used and well-studied methods—(adaptive) sparse grids (SGs[3]; e.g., Brumm and Scheidegger, 2017) and high-dimensional model representation (HDMR; e.g., Eftekhari et al., 2017)—and by showcasing their generic applicability and scalability through the solution of an international real business cycle (IRBC) model (Haan et al., 2011, Kollmann et al., 2011) via the time iteration algorithm (Coleman, 1990).[4] The IRBC model is an ideal test case because its dimensionality grows linearly with the number of countries, making it well-suited for testing high-dimensional global solution methods.

General Problem Formulation and Global Solution Methods. This article examines dynamic economic models frequently characterized by *recursive equilibria* (Ljungqvist and Sargent, 2004). In these models, a (potentially high-dimensional) *state variable* $\mathbf{x} \in X \subset \mathbb{R}^d$ represents the current state of the economy, with d indicating the dimensionality of the state space. The evolution of the model is governed by a time-invariant *equilibrium function* $f : X \to Y \subset \mathbb{R}^m$, which is determined by solving a functional equation of the form

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

[2] We adopt the nomenclature from Brumm and Scheidegger (2017), referring to a "global solution" as one computed using equilibrium conditions at numerous points throughout the state space of a dynamic model, as opposed to a "local solution" which relies on a local approximation around the model's steady state, as achieved through perturbation methods.

[3] We use the term SGs interchangeably for both "regular" sparse grids with piecewise linear basis functions and "adaptive" sparse grids with linear basis functions (cf. Section 4.1).

[4] Two main challenges arise when applying time iteration to large-scale dynamic stochastic models: (i) each iteration step requires the global approximation of a high-dimensional, multivariate function, and (ii) each point in the chosen approximation scheme entails solving a system of non-linear equations. Together, these issues can significantly prolong the time to solution, underscoring the need for an efficient and highly scalable implementation of the algorithm.

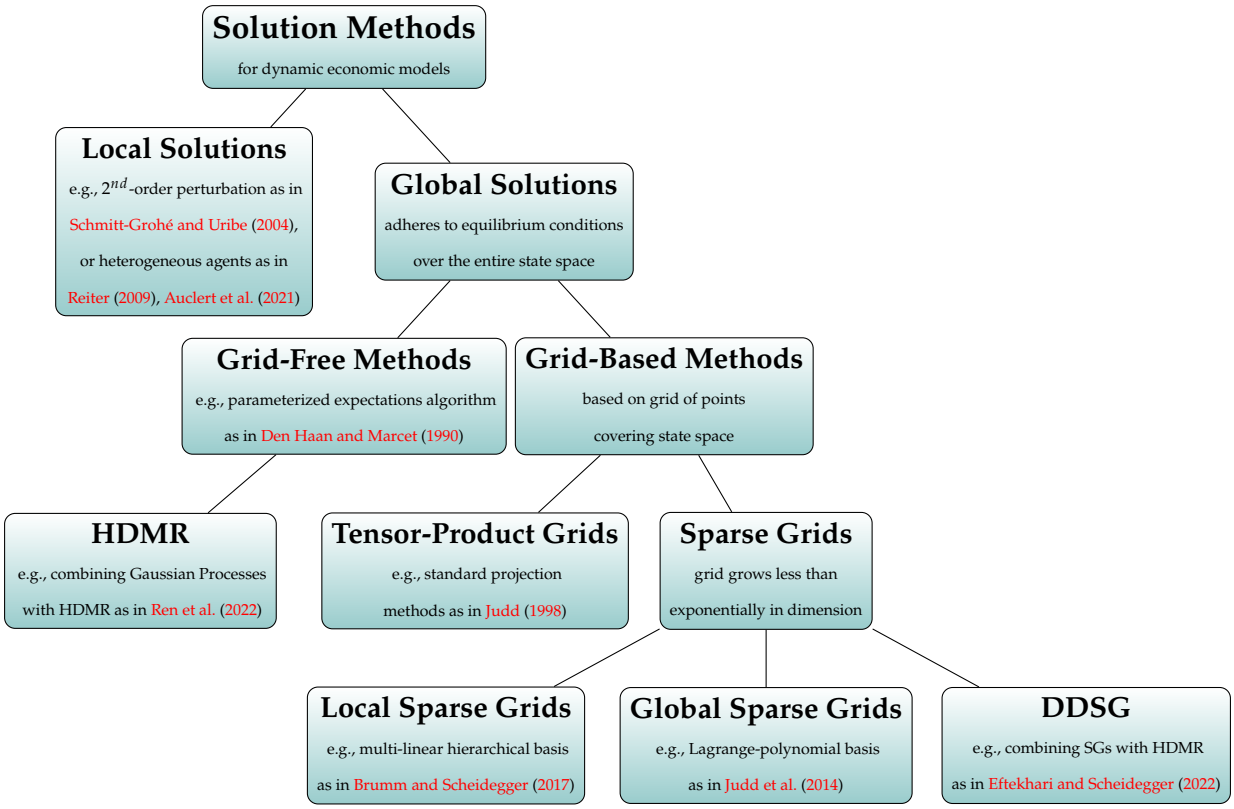

Figure 1: Taxonomy of solution methods for dynamic economic models, classified into several categories. Note that HDMR can be utilized in both grid-based and grid-free contexts. However, in this article, we focus exclusively on grid-based approaches, as the combination of SGs with HDMR (denoted as *dimension-decomposed sparse grids* (DDSG); cf. Section 4.2) offers significant numerical advantages.

where $\mathcal{E}$ may, for example, denote a Bellman equation in a discrete-time framework or the Hamilton-Jacobi-Bellman (HJB) equation in continuous time. Alternatively, $\mathcal{E}$ might capture the first-order equilibrium conditions in a discrete-time setting, with f representing a (possibly multi-dimensional) policy function, which is the primary focus of this article.

In each of these settings, the *curse of dimensionality* becomes an issue when the state space X is moderately high-dimensional and a global solution is required, particularly in the presence of significant non-linearities. In contrast to local solutions, which depend on equilibrium conditions and their derivatives at a specific point, global approaches demand that these conditions hold throughout the entire state space. Consequently, researchers have to turn to grid-based and grid-free approximation methods that mitigate the curse of dimensionality. Figure 1 provides a structured, albeit non-exhaustive, taxonomy of solution methods used in dynamic economic models, adapted and expanded from Brumm et al. (2022).

Grid-free methods have been proposed, notably by Den Haan and Marcet (1990), and have recently gained traction through advancements in machine learning.[5] Unlike many contem-

[5]See, for instance, Duffy and McNelis, 2001, Norets, 2012, Renner and Scheidegger, 2018, Maliar et al., 2021, Azinovic et al., 2022, Fernández-Villaverde et al., 2023, Han et al., 2021, Friedl et al., 2023, Gaegauf et al., 2023, Nuño et al., 2024, Valaitis and Villa, 2024, Duarte et al., 2024, Kase et al., 2022, Kübler et al., 2025, Payne et al., 2024, and Chen et al., 2025. For a recent review of deep learning applications in economics, refer to Fernández-Villaverde et al. (2024).

porary machine learning approaches, which often require extensive ad-hoc hyperparameter tuning, SGs benefit from well-understood convergence properties. SGs enable researchers to scale up models that are numerically formulated on a grid to higher dimensions without necessitating a complete overhaul of the solution technique. Additionally, they can be relatively easily parallelized, potentially reducing computation time significantly (Brumm et al., 2015b). Thus, SGs provide a straightforward means to augment current modeling and solution frameworks, thereby broadening the spectrum of addressable research questions. We have thus opted to initially employ SGs (in combination with HDRM) to extend Dynare beyond perturbation-based methods before transitioning to more recent, albeit currently less stable, global solution techniques.

Adaptive Sparse Grids. SG methods offer a highly efficient and structured approach to the computational challenges of high-dimensional state spaces in dynamic economic models, that is, to approximate non-linear, high-dimensional (policy) functions. By extending univariate interpolation formulas to the multivariate case through linear combinations of tensor products, as detailed in various techniques like the Smolyak algorithm, classical SG methods, combination techniques, and dimension- or spatially-adaptive methods (e.g., Bungartz and Griebel, 2004, and references therein), SGs alleviate the curse of dimensionality. Specifically, SG methods reduce the exponential growth of grid points with increasing dimensionality d, from $O(M^d)$ to $O(M \cdot (\log M)^{d-1})$, while maintaining nearly the same accuracy for sufficiently smooth functions (see Figure 2). This approach cuts grid points by several orders of magnitude compared to full Cartesian tensor-product grids, significantly mitigating the computational burden while marginally increasing interpolation errors. The primary distinction among SG techniques lies in their use of either local or global basis functions, which we categorize as local sparse grids and global sparse grids, respectively. Local SGs (LSG), utilizing hierarchical, multi-linear basis functions, are particularly effective for non-smooth functions exhibiting localized irregularities like kinks. On the other hand, Global SGs (GSG), often based on Lagrange characteristic polynomials (commonly known as the Smolyak method in economics), excel at approximating smooth functions (cf. Figure 1). In this article, we focus strictly on LSGs; for GSGs, please refer to the review by Brumm et al. (2022).

The Role of HDMR: When Sparse Grids Are Not Sufficient. SGs are highly effective for approximating functions in moderately high-dimensional spaces (e.g., when $d < 20$). However, their computational cost can become prohibitive as either the dimensionality or the complexity (i.e., the degree of non-linearity) of the underlying functions increases. For example, in highly non-linear dynamic economic models with many state variables, a high *a priori* grid resolution (i.e., refinement ℓ) produces an explosive number of grid points (see Figure 2). Moreover, numerical operations on SG data structures may become prohibitively expensive in high-dimensional settings (Murarașu et al., 2012). HDMR addresses this limitation by decomposing the target function into a sum of lower-dimensional components, each of which can be approximated very efficiently (see, e.g., Ma and Zabaras, 2010, Yang et al., 2012, and references therein). Evaluating the combined sum of these component functions is considerably less computationally demanding than evaluating a full SG in high-dimensional settings. When HDMR is combined with SGs, the resulting approach is often called *dimension-decomposed sparse grids* (DDSG; cf. Figure 1). This integration substantially reduces computational load and memory requirements while preserving approximation accuracy. As a result, DDSG is particularly well-suited for high-dimensional economic models characterized by significant heterogeneity and complex non-linearities, provided that a (possibly latent) additively separable structure exists and can be detected computationally.

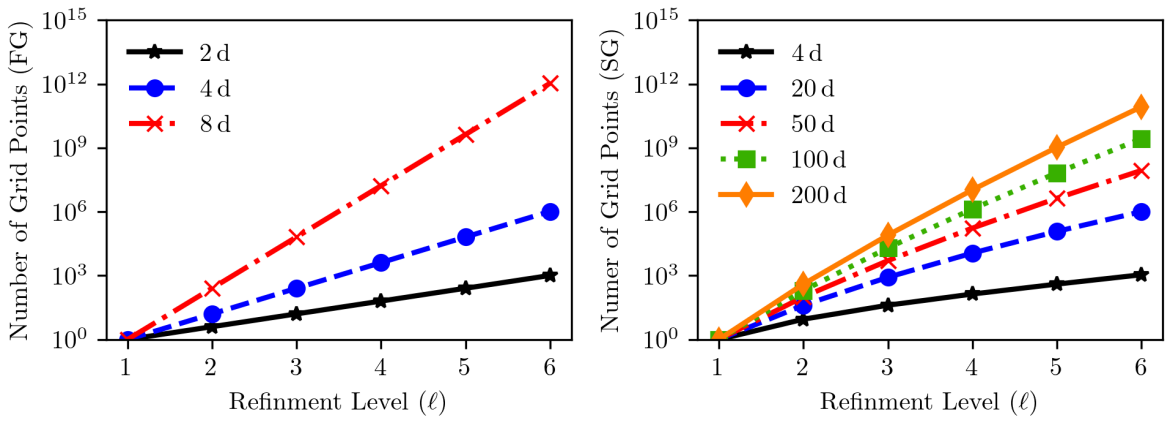

Figure 2: Left Panel: Number of grid points in a full Cartesian grid (FG) of dimensionality d with increasing grid resolution, denoted by the refinement level ℓ. In SGs, ℓ dictates the grid's resolution, producing approximately $M \propto 2^{\ell}$ points per dimension, where a higher ℓ resolves finer features as the spacing between points shrinks to $h \propto 2^{-\ell}$, resulting in a total of $M \propto 2^{\ell \cdot d}$ grid points across all dimensions. Right Panel: Number of grid points in an SG of increasing dimensionality. Notably, even a 200-dimensional SG with resolution $\ell = 6$ has orders of magnitude fewer points than an 8-dimensional FG at the same resolution level (cf. Section 4.1 below).

Preview of Results. We demonstrate that the SG approach for computing global solutions scales up to at least 16-dimensional IRBC models[6] and exhibits subexponential increases in runtime. Even when targeting a high level of accuracy for global solutions, our computation times remain relatively short on a standard laptop: 8-dimensional models are solved in seconds, while 16-dimensional ones require only about half an hour. Moreover, initializing the time iteration algorithm with Dynare's perturbation solution—rather than a naive guess—reduces convergence time by up to 20 times, highlighting the practical viability of the SG method for high-dimensional models. In addition, our DDSG approach, which builds upon standard SG methods, can provide additional efficiency gains, as demonstrated in the context of the IRBC model with its latent additively separable structure. Although the DDSG method incurs a modest overhead in low-dimensional settings, this cost diminishes rapidly with increasing dimensions; runtimes break even by eight dimensions, and at 16 dimensions, a single DDSG timestep is more than 13 times faster than its SG counterpart. These efficiency gains, coupled with an up to 80 times speedup achieved by employing Dynare's initialization within the DDSG framework, make our approach a powerful tool for tackling complex, high-dimensional economic models in which latent additively separable structures are present. As we demonstrate in our numerical experiments, the DDSG approach also exhibits subexponential increases in runtime, thereby enabling the computation of global solutions for models with at least 100 dimensions on a standard laptop within hours, whereas models with fewer than 20 dimensions exhibit runtimes of only seconds to minutes.

Organization of the Article. The remainder of this paper is structured as follows. Section 2 offers a brief overview of related literature. Section 3 then formally characterizes the models we aim to solve via a canonical time iteration algorithm. Section 4 reviews the mathematical underpinnings of SGs and HDMR, which serve to approximate and interpolate policy functions

[6]Recall that while we focus on an IRBC model here, the approach is broadly applicable to high-dimensional dynamic stochastic models. The IRBC model simply serves as a convenient test bed, because one can easily adjust the number of countries (and thus the number of states) to control the computational complexity.

within the time iteration algorithm. Section 5 details the required modifications to Dynare's `.mod` files to accommodate the global solution methods proposed. Section 6 then discusses illustrative performance results, and finally, Section 7 concludes. Furthermore, we provide supplementary code examples demonstrating our developments, available at https://nvls.co/Dynare/GlobalMethods/SparseGrids.

2 Related Literature

Building on Juillard, 1994, Collard and Juillard, 2001 and Adjemian et al., 2024, this article aims to extend the Dynare framework—currently relying on various local perturbation methods—by incorporating two particular types of global solution methods, namely SGs and HDMR. These methods have become an important and widely used tool in economics and finance for solving high-dimensional models, and are broadly applicable in both discrete and continuous time frameworks, as summarized in the recent review by Brumm et al. (2022).[7] In what follows, we present a brief, though by no means exhaustive, overview of the diverse applications within these fields.

The introduction of global sparse grids (GSGs; cf. Figure 1), specifically the Smolyak sparse grids, in economics was pioneered by Krueger and Kubler (2004, 2006), focusing on discrete-time overlapping generations (OLG) models. Fernández-Villaverde et al. (2015) further utilized these GSGs to explore non-linear dynamics in a New Keynesian model constrained by a zero lower bound on nominal interest rates. Enhancements to the Smolyak method for better economic model performance were made by Judd et al. (2014). Local sparse grids (LSGs; cf. Figure 1) entered economic analysis through the work of Brumm and Scheidegger (2017), who applied these grids to IRBC models with irreversible investment and menu-cost models. Moreover, they introduced adaptivity, which adds a second layer of sparsity, as grid points are added only where they are most needed. Brumm et al. (2017) and Brumm and Hußmann (2024) employed LSGs for solving calibrated OLG models with aggregate shocks, while Usui (2019) used them to analyze rare natural disasters and adaptation in a dynamic stochastic economy. Recent developments include the introduction of continuous-time adaptive LSG methods in macroeconomics by Garcke and Ruttscheidt (2019), particularly for heterogeneous agent models. Schaab (2020) utilized LSGs to solve HJB type equations, examining the interplay between micro and macro uncertainties in a heterogeneous agent New Keynesian model, accounting for aggregate risk, counter-cyclical unemployment, and monetary policy constraints. In the realm of finance, LSGs have been applied both in discrete and continuous time to tackle high-dimensional option-pricing problems, as demonstrated by Reisinger and Wittum (2007), Bungartz et al. (2012), Scheidegger and Treccani (2018), and to address dynamic portfolio choice models with transaction costs as per Schober et al. (2021). Further financial applications include the integration of likelihood functions on GSGs by Heiss and Winschel (2008), the embedding of GSGs in Bayesian estimation frameworks by Winschel and Krätzig (2010), and the application of LSGs in the context of the generalized method of moments by Gilch et al. (2021). Young and Ratto (2011) have used ideas from the HDRM literature in the context of splines to estimate linear models, whereas Eftekhari and Scheidegger (2022) combined SGs with HDMR to solve dynamic stochastic models containing up to 300 continuous states. Finally, numerous high-performance, user-friendly open-source SG implementations are available in various programming languages. Among the most popular are *SG++*[8], the sparse grids Matlab

[7] For an extensive review of numerical techniques for dynamic models, see the handbook chapters by Maliar and Maliar, 2014 and Cai and Judd, 2014.

[8] see https://sgpp.sparsegrids.org

kit[9], spinterp[10], and TASMANIAN[11]. In our numerical experiments within Dynare, we utilize TASMANIAN, as it is actively developed and maintained by Oak Ridge National Laboratory.

3 Solving Dynamic Models with Sparse Grids and HDMR

To demonstrate the capabilities of embedding SGs and HDMR within Dynare to compute global solutions to a very broad range of discrete-time dynamic stochastic models, we begin by formally characterizing the class of models we aim to solve. Section 3.1 presents the general structure common to many infinite-horizon, discrete-time dynamic stochastic economic models. Next, Section 3.2 introduces a specific benchmark example, namely the IRBC model (see, e.g., Haan et al., 2011). Finally, in Section 3.3, we demonstrate how to iteratively compute global solutions for such dynamic models via time iteration, using the IRBC model as a guiding example.

3.1 Abstract Model Formulation

Let $\mathbf{x}_t \in X \subset \mathbb{R}^d$ denote the state of the economy at time $t \in \mathbb{N}$. We define the actions of all agents in the economy through a policy function $p : X \to Y$, where Y represents the space of all possible policies. The evolution of the state $\mathbf{x}_t$ from period t to $t+1$ follows the state transition

$$\mathbf{x}_{t+1} \sim \mathcal{D}(\cdot \mid \mathbf{x}_t, p(\mathbf{x}_t)), \tag{2}$$

where the distribution $\mathcal{D}(\cdot)$ is model-specific. The optimal policy function $p(\cdot)$ is not known a priori and must satisfy the period-to-period equilibrium conditions $E(\cdot)$. Specifically, the policy is time-invariant, so

$$\mathbb{E}\Big[E\big(\mathbf{x}_t, \mathbf{x}_{t+1}, p(\mathbf{x}_t), p(\mathbf{x}_{t+1})\big) \,\Big|\, \mathbf{x}_t, p(\mathbf{x}_t)\Big] \;=\; 0 \quad \forall t, \tag{3}$$

where $\mathbb{E}[\cdot]$ is the expectation taken over the distribution in expression (2).

This time-invariant policy $p(\cdot)$ can be determined via time iteration, by iterating directly on condition (3). The time iteration algorithm (Coleman, 1990) computes a recursive equilibrium of a dynamic economic model by starting with an initial guess for the policy function and iteratively refining it based on the model's first-order conditions (e.g., Judd, 1998, Section 17.8).

The Need for Global Solution Methods. In many economic models, the equilibrium conditions E exhibit substantial non-linearity due to concave utility and production functions or large shocks; moreover, (financial) frictions often imply non-differentiability. As a result, the optimal policy satisfying Equation (3) is generally non-linear and not necessarily smooth, necessitating function-approximation techniques suited to (high-dimensional) non-linear environments. Global solution methods, such as SGs and HDRM, are thus preferred over perturbation approaches, like those in the standard Dynare implementation (Adjemian et al., 2024).

3.2 International Real Business Cycle Model

The IRBC model is a widely used benchmark for studying methods that solve high-dimensional dynamic stochastic models, because its dimensionality scales linearly with the number of

[9] see https://sites.google.com/view/sparse-grids-kit
[10] see https://people.math.sc.edu/Burkardt/m_src/spinterp/spinterp.html
[11] see https://tasmanian.ornl.gov

countries. In the following, we present the equations of the IRBC model, which formally correspond to the abstract formulation given in Equation (3). A detailed discussion of the model itself is beyond the scope of this paper; the interested reader is referred to Haan et al., 2011 and Brumm and Scheidegger, 2017 for derivations. Here, we focus on the explicit set of non-linear equations to be solved at many points in the state space in order to construct the optimal policy function $p(\cdot)$.

The IRBC model is dynamic and stochastic with a $(d = 2N)$-dimensional state space, where $N \in \mathbb{N}_+$ is the number of countries. The state variables are

$$\mathbf{x}_t = \left(a_t^1, \ldots, a_t^N,\ k_{t-1}^1, \ldots, k_{t-1}^N\right) \in \mathbb{R}^{2N}, \tag{4}$$

where a_t^n and k_t^n represent the productivity and the end-of-period capital stock of country $n \leqslant N$, respectively. The policy function $p : \mathbb{R}^{2N} \to \mathbb{R}^{N+1}$ maps the current state $\mathbf{x}_t$ into the next period,

$$p(\mathbf{x}_t) = \left(k_t^1, \ldots, k_t^N,\ \lambda_t\right), \tag{5}$$

where λ_t is the multiplier for the aggregate resource constraint. For optimality, the IRBC policy must satisfy the following $N + 1$ non-linear equations:

$$\begin{aligned} &\lambda_t \frac{\partial q_t^n\left(k_{t-1}^n, k_t^n\right)}{\partial k_t^n} - \beta \mathbb{E}_t\left[\lambda_{t+1} \frac{\partial\left(y^n(a_{t+1}^n, k_t^n) - q^n(k_t^n, k_{t+1}^n)\right)}{\partial k_t^n}\right] = 0 \quad \forall n, \\ &\sum_{n=1}^{N} y^n\left(a_t^n, k_{t-1}^n\right) - \left(\frac{\lambda_t}{\tau^n}\right)^{-\gamma^n} - q^n\left(k_{t-1}^n, k_t^n\right) = 0. \end{aligned} \tag{6}$$

The parameters τ^n and γ^n are model-specific, while $y^n(\cdot)$ and $q^n(\cdot)$ denote the production and convex adjustment-cost functions, respectively. The complete parameterization of the model is given in Brumm and Scheidegger (2017), Table 2.

3.3 Time Iteration

We next describe how the time iteration algorithm (Coleman, 1990) is implemented in Dynare to iteratively solve dynamic models expressed abstractly as expression (3), such as the IRBC model introduced in Section 3.2. Algorithm 1 illustrates how to solve the non-linear system (6) using an interpolation/approximation, sloppily denoted as $\mathcal{I}$, of the previous policy guess, denoted by $\mathcal{I}p_0(\cdot)$. Specifically, we approximate the terms $\left(k_{t+2}^1, \ldots, k_{t+2}^N, \lambda_{t+1}\right)$ by $\mathcal{I}p_0\left(\mathbf{x}_{t+1}\right)$ and subsequently solve for the $N + 1$ unknowns using a non-linear solver, such as `IPOPT` (Waechter and Biegler, 2006). Starting with an initial policy guess $\mathcal{I}p_{\text{guess}}$, which may be obtained from a linearized Dynare solution, the procedure iteratively refines the solution until it satisfies a specified tolerance, `tol`, measured, for instance, as mean squared error (cf. Sections 6.1 and 6.2 below). In each iteration, a high-dimensional, non-linear policy is updated using an appropriate approximation scheme, such as SGs or HDMR (as discussed in this paper). For further implementation details, see Brumm and Scheidegger (2017).

4 Numerical Function Approximation

Dynamic stochastic models, solved via time iteration (see Sections 3.2 and 3.3), necessitate repeated approximations and interpolations of high-dimensional, non-linear functions. In this section, we demonstrate that these tasks can be efficiently addressed using SGs or by combining SGs with HDMR, an approach we term, as mentioned before, *dimension-decomposed sparse grids*

Algorithm 1 Time Iteration Algorithm.

Require: $\mathcal{I}p_{\text{guess}}$, `tol`

$\mathcal{I}p \leftarrow \mathcal{I}p_{\text{guess}}$

repeat

  $\mathcal{I}p_0 \leftarrow \mathcal{I}p$

  construct $\mathcal{I}p$ by solving Eq. (5) given $\mathcal{I}p_0$

until $\|\mathcal{I}p - \mathcal{I}p_0\| <$ `tol`

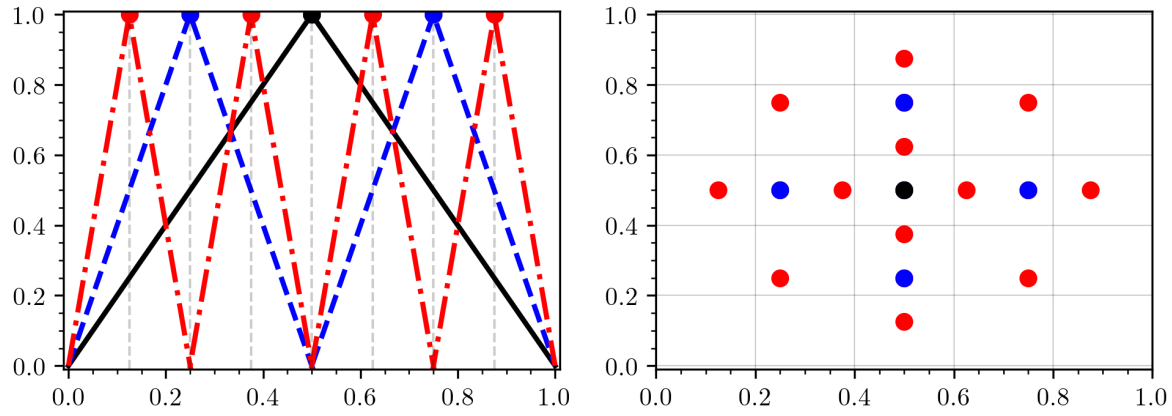

Figure 3: Left Panel: Hierarchical basis functions at refinement levels $\ell = 1$ (solid black), $\ell = 2$ (dashed blue), and $\ell = 3$ (dash-dotted red). Right Panel: A two-dimensional SG with corresponding refinement levels and colors.

(DDSG). Section 4.1 provides a concise overview of SG techniques (e.g., Bungartz and Griebel, 2004, Pflüger, 2010), Section 4.2 reviews HDMR and DDSG, and Section 4.3 presents analytical examples to build intuition. Throughout, we adopt the notation established in Eftekhari et al. (2017).

4.1 Adaptive Sparse Grids

We aim to approximate the policy function, where each policy is represented as $f : \Omega \to \mathbb{R}$, with $\Omega = \mathbb{R}^d$ and $\mathbf{x} \subset \Omega$ rescaled to the unit domain $[0, 1]^d$. In our case d the number of continuous state variables in the economic model of interest, a potentially large number.[12] In one dimension, the unit domain $[0, 1]$ can be discretized with grid spacing $h_l = 2^{-l}$, grid points at $x_{l,i} = i \cdot h_l$, where $i \in \{1, \ldots, 2^l\}$ and refinement level $l \in \mathbb{N}_+$. The univariate basis functions for this discretization are defined as

$$\phi_{l,i}(x) = \max\left(1 - \frac{1}{h_l}\left|x - x_{l,i}\right|, 0\right), \tag{7}$$

which have support $[x_{l,i} - h_l, x_{l,i} + h_l]$. In the left panel of Figure 3, we illustrate a one-dimensional hierarchical piecewise linear basis function. We extended the basis function to a d-dimensional domain by first introducing the multi-indices for the refinement level $\mathbf{l} = (l_1, \ldots, l_d) \in \mathbb{N}_+^d$ and grid index $\mathbf{i} = (i_1, \ldots, i_d)$ with the corresponding grid point $\mathbf{x}_{\mathbf{l},\mathbf{i}} =$

[12] For illustration and notational brevity, we assume zero boundary conditions. For a discretization scheme with nonzero boundary conditions—such as the Clenshaw-Curtis sparse grid setting employed in all our numerical experiments, see Stoyanov (2015) and Pflüger (2010).

$(x_{l_1,i_1}, \ldots, x_{l_d,i_d})$. The d-dimensional basis functions are defined as

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j). \tag{8}$$

We define the hierarchical index sets $\mathbf{I}_\mathbf{l}$, and the corresponding hierarchical subspace $W_\mathbf{l}$ as

$$\mathbf{I}_\mathbf{l} = \left\{\mathbf{i} : 0 < i_j < 2^{l_j},\ i_j \text{ odd},\ 1 \leqslant j \leqslant d\right\}, \quad \text{and} \quad W_\mathbf{l} = \operatorname{span}\left\{\phi_{\mathbf{l},\mathbf{i}} : \mathbf{i} \in \mathbf{I}_\mathbf{l}\right\}. \tag{9}$$

The restriction to odd indices i_j ensures that the supports of the basis functions are disjoint and collectively cover $[0,1]^d$. For the space of piecewise linear functions,

$$V_\ell = \bigoplus_{\|\mathbf{l}\|_\infty \leqslant \ell} W_\mathbf{l}, \tag{10}$$

we can construct a corresponding equidistant Cartesian grid, also called a *full grid*, with $M_\ell = 2^\ell$ number of grid points in each dimension, where ℓ denotes the *maximum refinement level*.

Although the L_2 interpolation error is of order $O(M_\ell^{-2})$, the total number of grid points is $O(M_\ell^d)$, effectively rendering this approach impractical for high-dimensional functions. SG mitigates the curse of dimensionality by retaining only the most significant hierarchical subspaces. Formally, the SG space is defined as

$$V_\ell^{\mathrm{SG}} = \bigoplus_{\|\mathbf{l}\|_1 \leq \ell+d-1} W_\mathbf{l}. \tag{11}$$

The SG interpolation of f at a point $\mathbf{x}$ with a maximum refinement level ℓ is defined as

$$\mathcal{I}_\ell f(\mathbf{x}) := \sum_{\|\mathbf{l}\|_1 \leq \ell+d-1} \sum_{\mathbf{i} \in \mathbf{I}_\mathbf{l}} \alpha_{\mathbf{l},\mathbf{i}}\, \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}), \tag{12}$$

where $\alpha_{\mathbf{l},\mathbf{i}} \in \mathbb{R}$ (hierarchical surpluses). In the right panel of Figure 3, we illustrate the grid points for a two-dimensional SG. For functions with bounded mixed second derivatives, the SG interpolation error is of the order $O(M_\ell^{-2}(\log M_\ell)^{d-1})$, while the number of grid points is $O(M_\ell(\log M_\ell)^{d-1})$, substantially less grid points than the full grid in high dimensions. For further details on SGs, including the error analysis and the computation of hierarchical surpluses $\alpha_{\mathbf{l},\mathbf{i}}$, we refer the reader to Bungartz and Griebel (2004) and references therein. Finally, quadrature on SGs, denoted as

$$Q_\ell f = \sum_{\|\mathbf{l}\|_1 \leq \ell+d-1} \sum_{\mathbf{i} \in \mathbf{I}_\mathbf{l}} \alpha_{\mathbf{l},\mathbf{i}} \int \phi_{\mathbf{l},\mathbf{i}}(\mathbf{x})\, d\mathbf{x}, \tag{13}$$

can be efficiently evaluated by integrating the basis functions defined in Equation (8).

If the function to be approximated has sharp local features, steep gradients, or non-differentiabilities—for example, in economic models with occasionally binding constraints—the assumed condition of bounded second-order mixed derivatives is not satisfied. As a result, significantly higher SG refinement levels (and thus more grid points) may be required, limiting the applicability of SG in many high-dimensional problems of interest. In such scenarios, an *adaptive* refinement procedure can be employed to preserve efficiency (e.g., Pflüger, 2010, Brumm and Scheidegger, 2017). In this adaptive approach, one monitors the magnitude of each hierarchical surplus $\alpha_{\mathbf{i},\mathbf{l}}$, which indicates the local irregularity of the function at $\mathbf{x}_{\mathbf{i},\mathbf{l}}$. Let $\epsilon_\gamma \in \mathbb{R}_+$

be a predefined refinement threshold, and define a refinement criterion g. If $g(\alpha_{\mathbf{i},\mathbf{l}}) \leqslant \epsilon_\gamma$, then the point $\mathbf{x}_{\mathbf{i},\mathbf{l}}$ is considered *insignificant* and no further refinement is performed at that location. Due to the hierarchical structure of the grid, excluding $\mathbf{x}_{\mathbf{i},\mathbf{l}}$ automatically excludes any descendant points at higher refinement levels. In many practical applications, the commonly used refinement criterion is

$$g(\alpha_{\mathbf{i},\mathbf{l}}) := |\alpha_{\mathbf{i},\mathbf{l}}|, \tag{14}$$

although alternative criteria may also be applied (e.g., Stoyanov, 2015).

SGs are, as mentioned above, highly effective for approximating moderately high-dimensional functions (say, $d < 20$); however, they can become computationally prohibitive as dimensionality or complexity increases (cf. Brumm and Scheidegger (2017)). To address this limitation, we now turn our attention to HDMR and DDSG.

4.2 Dimensional Decomposition

In this section, we follow the approach of Hooker (2007) and Rabitz and Aliş (1999) to illustrate the fundamental steps for constructing a dimensional decomposition (DD). As in the previous sections, we consider a scalar-valued function $f(\mathbf{x})$, where $\mathbf{x} \in [0,1]^d$.[13] Denote $\mathbf{u} \subseteq \mathcal{S} = \{1, 2, \ldots, d\}$ as a *component index*, and $f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}})$ as a *component function*, where $\mathbf{x}_{\mathbf{u}}$ is a vector that consists of the values x_i for $i \in \mathbf{u}$. The function $f(\mathbf{x})$ can be expressed as the expansion

$$f(\mathbf{x}) = \sum_{\mathbf{u}\subseteq\mathcal{S}} f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}). \tag{15}$$

This representation of the function is referred to as HDMR. The terms in the summation are classified by the *expansion order* $k := |\mathbf{u}|$, which corresponds to the dimension of the input vector $\mathbf{x}_{\mathbf{u}}$. Expressed explicitly, the function can be decomposed as

$$f(\mathbf{x}) = f_\emptyset + \sum_{1\leqslant i\leqslant d} f_i(x_i) + \sum_{1\leqslant i<j\leqslant d} f_{i,j}(x_i, x_j) + \ldots + f_{1,2,\ldots,d}(x_1, x_2, \ldots, x_d). \tag{16}$$

Here, $f_\emptyset$ is a constant term corresponding to the zeroth-order contribution; the functions f_i, with $1 \leq i \leq d$, are univariate (first-order) contributions; the functions $f_{i,j}$ represent bivariate (second-order) contributions; and so forth, culminating in the dth-order contribution $f_{1,2,\ldots,d}$. In its complete form, this decomposition is exact, since the final term captures all interactions among the input variables. The principal advantage of the DD approach becomes evident when the high-dimensional function f can be well approximated by truncating the expansion in expression (15) at a maximum order $\mathcal{K} \ll d$.

Among the various formulations of HDMR, *cut-HDMR* and *ANOVA-HDMR* are particularly prominent (e.g., Rabitz et al., 1999 and Li et al., 2001).[14] We focus specifically on cut-HDMR, as it more closely aligns with the goals of our application. In contrast to ANOVA-HDMR, which requires high-dimensional numerical integration, cut-HDMR relies solely on direct function evaluations (see also Eftekhari and Scheidegger (2022)).

Let $w(\mathbf{x}) = \prod_{i=1}^d w_i(x_i)$ be a product measure, with $w_i(x_i)$ having a unit volume. By sequentially ascending through the expansion orders, starting from the zeroth order, the optimally

[13] The presented method is not limited to scalar-valued functions but can be directly extended to vector-valued functions. The scalar formulation is used solely for notational simplicity.

[14] For a comprehensive discussion of alternative cut-HDMR variants, including RS-HDMR, mp-cut-HDMR, Multicut-HDMR, and lp-RS-HDMR, see Li and Rabitz (2012).

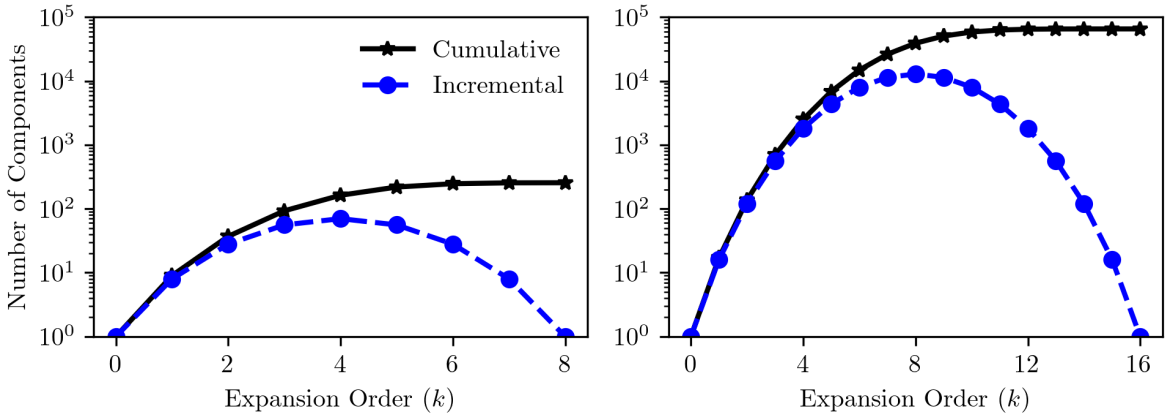

Figure 4: The number of DD component functions, both incremental and cumulative, as a function of expansion order for an 8-dimensional (left panel) and a 16-dimensional function (right panel). The number of component functions increases significantly with both expansion order and dimensionality.

and uniquely defined HDMR component function

$$f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) = \underset{g_{\mathbf{u}}}{\operatorname{argmin}} \int \left(\sum_{\mathbf{u} \subseteq \mathcal{S}} g_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) - f(\mathbf{x}) \right)^2 w(\mathbf{x}) d\mathbf{x}, \tag{17}$$
$$\text{subject to} \int g_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) w_i(\mathbf{x}_i) dx_i = 0, \quad \forall i \in \mathbf{u},$$

will only be dependent on lower-order component functions $f_{\mathbf{v}}(\mathbf{x}_{\mathbf{x}})$ for $\mathbf{v} \subset \mathbf{u}$. This is particularly important because the contrary would eliminate any reduction in dimensionality. This attribute is a result of the orthogonality condition imposed in Equation (17) (Rabitz and Aliş, 1999, Hooker, 2007). The cut-HDMR component functions are defined using the Dirac measure,

$$w(\mathbf{x})\, d\mathbf{x} = \prod_{i=1}^{d} \delta(x_i - \bar{x}_i)\, dx_i, \tag{18}$$

where $\delta(\cdot)$ denotes the Dirac delta function, and $\bar{\mathbf{x}} = (\bar{x}_1, \ldots, \bar{x}_d)$ is a reference point known as the *anchor point*. As shown by Sobol (2003), a suitable anchor point should satisfy

$$\bar{\mathbf{x}} \approx \underset{\mathbf{z}}{\operatorname{argmin}} \| f(\mathbf{z}) - \mathbb{E}[f(\mathbf{x})] \|_1, \tag{19}$$

and can be selected by sampling $\bar{\mathbf{x}}$ so that $f(\bar{\mathbf{x}})$ is close to the mean of the function. Evaluating expression (17), the cut-HDMR component functions are defined as

$$f_{\mathbf{u}}(\mathbf{x}_{\mathbf{u}}) = \sum_{\mathbf{v} \subseteq \mathbf{u}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} f(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}} \setminus \mathbf{x}_{\mathbf{v}}}, \quad \text{with } f_{\emptyset} = f(\bar{\mathbf{x}}). \tag{20}$$

We use the notation $\mathbf{x} = \bar{\mathbf{x}} \setminus \mathbf{x}_{\mathbf{v}}$ to refer to assigning $\mathbf{x}$ the values of $\bar{\mathbf{x}}$ but excluding the indices of $\mathbf{v}$. For example, given $\mathbf{x} = (x_1, x_2, x_3)$, then $\bar{\mathbf{x}} \setminus \mathbf{x}_{1,2} = (x_1, x_2, \bar{x}_3)$.
Performing the full expansion in expression (15) up to order $k = d$ is computationally infeasible for problems of nontrivial dimensionality, as it simply reconstructs the original d-dimensional function, thereby negating the intended efficiency gains from dimensional reduction. More-

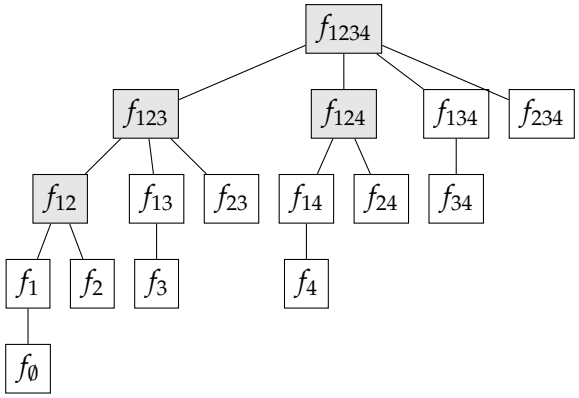

Figure 5: Visualization of the component functions summation in Equation (15) of a four-dimensional function with active dimension criteria, where the component index $\mathbf{u} = \{1, 2\}$ is deemed insignificant. For clarity, repeated cells are omitted. All component functions that are supersets of $\mathbf{u} = \{1, 2\}$ are excluded from the summation (shown in gray).

over, the number of component functions increases combinatorially with the expansion order k:

$$\sum_{j=0}^{k} \frac{d!}{(d-j)!j!}, \tag{21}$$

which poses significant computational challenges even for moderately sized problems (e.g., 8- or 16-dimensional), as illustrated in Figure 4. Therefore, truncating the expansion to a maximum order $\mathcal{K} \ll d$ is imperative; moreover, the optimal selection of $\mathcal{K}$ is inherently problem-dependent.

Truncating the HDMR Expansion. Two adaptive criteria have proven particularly effective for truncating the expansion in Equation (15), especially in economic applications (Eftekhari and Scheidegger, 2022). These criteria involve: (i) assessing the relative importance of individual component functions, and (ii) evaluating the incremental benefit of proceeding to higher expansion orders.

The first criterion, termed *active dimension selection*, evaluates the significance of each component function by comparing the norm of its integral to the norm of the cumulative integral of all previously computed lower-order component functions. Component functions whose integrals fall below a specified threshold are discarded, together with all supersets of their corresponding indices (see Figure 5 for a visual representation). The second criterion, known as the *expansion criterion*, examines convergence across expansion orders by quantifying the incremental changes in the integral values of the component functions. Expansion is terminated once these incremental changes fall below a predefined convergence threshold. Both criteria exploit the hierarchical structure of the expansion, thereby enhancing computational efficiency by reusing previously computed integrals.[15]

Dimensional Decomposition With Adaptive Sparse Grids. The abstract formulation of DD provided above has been described independently of the numerical methods for interpolation and quadrature. However, practical implementation requires efficient numerical schemes,

[15]Starting from lower-order terms, numerical integration is performed in lower-dimensional spaces, thus mitigating the computational challenges associated with high-dimensional integration.

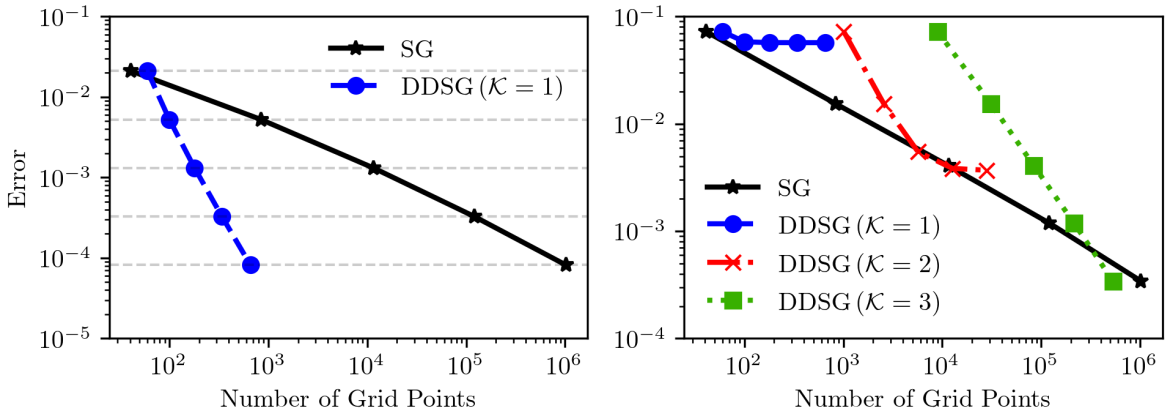

Figure 6: Interpolation error for SG and DDSG at different maximum expansion orders $\mathcal{K}$ for the test function given in Equation (23), with coefficients (left panel) $c = 1$ and (right panel) $c = 3$. The approximation error is measured as the relative average error over 1,000 samples of $\mathbf{x} \in [0,1]^d$, where $d = 20$.

particularly at higher DD expansion orders, due to the curse of dimensionality. SGs are particularly well suited for numerical methods for DD because of two key attributes: (i) SGs are effective in approximating high-dimensional functions with localized features, and (ii) their capability for efficient numerical integration (see Equation (15)). This combined approach is referred to as DDSG. Following the previously established exemplar of the d-dimensional function f, the DDSG function is expressed as

$$\begin{aligned} f(\mathbf{x}) &\approx \sum_{\substack{\mathbf{u}\subseteq\mathcal{S}\\ |\mathbf{u}|\leqslant\mathcal{K}}} \sum_{\mathbf{v}\subseteq\mathbf{u}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} \mathcal{I}_\ell f(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}\setminus\mathbf{x}_\mathbf{v}}, \\ &\approx \sum_{\substack{\mathbf{u}\subseteq\mathcal{S}\\ |\mathbf{u}|\leqslant\mathcal{K}}} \sum_{\mathbf{v}\subseteq\mathbf{u}} (-1)^{|\mathbf{u}|-|\mathbf{v}|} \sum_{\|\mathbf{k}\|_1\leq\ell+d-1} \sum_{\mathbf{i}\in\mathbf{I}_\mathbf{k}} \alpha_{\mathbf{i},\mathbf{k}}\, \phi_{\mathbf{i},\mathbf{k}}(\mathbf{x})|_{\mathbf{x}=\bar{\mathbf{x}}\setminus\mathbf{x}_\mathbf{v}} \end{aligned} \tag{22}$$

which represents a nested summation over a series of $|\mathbf{v}|$-dimensional (adaptive) SGs. The DDSG-based quadrature procedure adopts a similar structure (Eftekhari and Scheidegger, 2022). Note that we impose a strict maximum expansion order, which can be effectively combined with the adaptive criteria discussed above (e.g., active dimension selection and convergence criteria).

4.3 Analytical Examples

In this section, we present an analytical example to highlight the potential efficiencies and drawbacks of DDSG compared to SG. Specifically, we aim to demonstrate that, when a (latent) additively separable structure exists, DDSG can enhance the efficiency of SG by orders of magnitude, whereas in the absence of such a structure, SG outperforms DDSG.

We consider the following non-linear, analytical test function:

$$f(\mathbf{x}) = \left(\sum_{i=1}^{d} \sin(x_i) \right)^c, \tag{23}$$

where $c = \{1, 2, \ldots\}$. This function is additively separable, and DD provides an exact de-

composition with expansion order $r = c$. For any case where $\mathcal{K} < c$, the DDSG approach introduces an approximation error that cannot be mitigated by increasing the refinement levels of the SG (applied to the lower-dimensional component functions). In Figure 6, we compare the approximations generated by SG and DDSG for a 20-dimensional example, evaluating the average relative interpolation error using $1,000$ random samples over the domain $\mathbf{x} \in [0,1]^d$. Each marker in the plots represents a refinement level, ranging from $\ell = 1, \ldots, 5$. In the left panel of the figure the test function uses $c = 1$, meaning that DDSG with expansion order $\mathcal{K} = 1$ achieves the same interpolation error as SG but with substantially fewer grid points. In the right panel of the figure the test function has $c = 3$, where DDSG with expansion order $\mathcal{K} = 1$ performs poorly compared to SG, and the minimum error remains relatively high regardless of the refinement level. Increasing the expansion order reduces the error but significantly increases the number of grid points due to the growing number of component functions. Only when DDSG has an expansion order of $\mathcal{K} = c = 3$ does the decomposition become exact, allowing it to achieve the same error as SG.

Notice that for this test function, a simple log-transform can effectively render the function additively separable with a DD expansion order $\mathcal{K} = 1$, regardless of the value of c; a technique that could also be applied in the context of economic models. While this approach may not be universally applicable, an appropriate transformation can significantly reduce DDSG approximation errors for non-additively separable functions. Moreover, in real applications where the function is unknown, the effective expansion criterion outlined in Section 4.2 can be used to implicitly deduce the separability of the unknown function (see Eftekhari and Scheidegger (2022) for more details).

5 Dynare Syntax

This section details the required modifications to Dynare's `.mod` files to accommodate the proposed generic global solution methods.[16] Code Listing 5 presents the `.mod` file command for Dynare that characterize the IRBC model (cf. Section 3.2), beginning with the specification of variables shared across countries. The subsequent declaration of variables, shock innovations, and parameters follows standard Dynare syntax.

We then leverage Dynare's macro-language to programmatically specify country-specific model variables, shocks, equations, and initial conditions. The macro-language extends Dynare's basic syntax by introducing conditionals, loops, display, and error directives, among other commands.[17] The macro-processor expands these commands in the `.mod` file, generating a fully specified version that Dynare can process using its standard workflow. Since this macro expansion occurs as a pre-processing step, it ensures that the final file conforms to standard Dynare syntax. This approach simplifies the creation of `.mod` files by automatically handling repetitive variable definitions and equations, as demonstrated in the multi-country model considered in this study.

In our implementation, the `@#define` command sets the macro-variable `N`, representing the number of countries in the model. We then define country-specific variables and parameters using a macro loop statement (`@#for-@#endfor`) combined with expression substitution, where `@{j}` is replaced by the current value of the macro-variable `j` in the `.mod` file. Specifically, for illustrative purposes, we fix the number of countries at `N` = 2. The country-specific variables then include the capital levels (denoted by `k_1` and `k_2`) and log-productivity levels (denoted by `a_1`

[16] Notice that our code developments were implemented in Julia (i.e., `Dynare.jl`). However, the model parsing is independent of the programming language, so the tools developed can also be made available in Dynare environments such as `Matlab`.

[17] More details on Dynare's macro-language and macro-processor are available in Dynare's manual.

and a_2). Similarly, e_1 and e_2 represent the country-specific components of the productivity innovations. The country-specific parameters comprise the utility function curvatures (gamma_1 and gamma_2) and the welfare weights (t_1 and t_2).

The model block also leverages the macro-language to programmatically declare country-specific optimality conditions for investment in capital and productivity processes using macro loop statements. In contrast to standard Dynare models—where exogenous variables (varexo) represent pure stochastic innovations—our approach distinguishes between stochastic innovations and shock-driven state variables. This distinction is made explicit in the [preamble] block, which groups all equations governing the processes of exogenous variables. In our setup, the productivity processes (a_1 and a_2) are classified as exogenous because their evolution depends solely on their past values and stochastic innovations (e and e_j), unlike the conventional Dynare approach in which only the innovations are exogenous and the full autoregressive process is treated as endogenous. The dedicated [preamble] block isolates the exogenous dynamics, thereby enhancing computational efficiency, particularly in numerical methods such as sparse grids. Moreover, our implementation requires that shock innovations be set to 1, so users must manually scale the innovations in the [preamble] equations by their respective standard errors. The shocks block then assigns the standard errors of country-specific shock innovations using a macro loop. Finally, the initval block specifies initial values for simulation or serves as initial guesses for non-linear solvers, with a macro loop setting the values for the country-specific capital and log-productivity levels.

Thus far, declarations (variables, shocks, parameters, and model equations) largely adhere to standard Dynare syntax, aside from a few modifications in the definition of exogenous variables. The macro-language is employed primarily to automate and structure repetitive components, thereby ensuring scalability as the number of countries increases. We further exploit the ability of the .mod file to interpret native Julia code by incorporating the SG routines sparsegridapproximation and DDSGapproximation.

We leverage the capability of the .mod file to interpret native Julia code by incorporating the SG routines sparsegridapproximation and DDSGapproximation. The function sparsegridapproximation implements an adaptive sparse grid method that iteratively refines the approximation grid to solve the model. Its accuracy and computational cost are primarily determined by several key parameters. In particular, the parameter gridDepth sets the initial grid depth, thereby influencing the number of interpolation points. SG refinements are controlled by maxRef—which specifies the maximum number of refinement steps—and by surplThreshold, which defines the surplus error threshold that triggers refinement. The time iteration process employs a convergence criterion specified by tol_ti. Additionally, the parameter polUpdateWeight regulates the update of the policy function during iterations by assigning a weight to the newly computed policy relative to the previous iteration. Lower values of polUpdateWeight stabilize the updates, albeit at the potential cost of slower convergence.

The DDSGapproximation function extends the standard SG approach by employing the DDSG method, which restricts interactions among state variables to alleviate the curse of dimensionality. In addition to the parameters used in sparsegridapproximation, DDSGapproximation introduces k_max, which specifies the maximum order of interaction terms in the DDSG decomposition. This parameter is pivotal in balancing computational efficiency with approximation accuracy.

```
var lambda;
varexo e;
parameters kappa beta delta phi rho A sigE;

kappa = 0.36;
```

```
beta  = 0.99;
delta = 0.01;
phi   = 0.5;
rho   = 0.95;
sigE = 0.01;
A = (1 - beta*(1 - delta))/(kappa*beta);

a_eis = 0.25;
b_eis = 1;

@#define N=2
@#for j in 1:N
var k_@{j} a_@{j};
  varexo e_@{j};
  parameters gamma_@{j} t_@{j};
  gamma_@{j} = a_eis + (@{j} - 1)*(b_eis - a_eis)/(@{N}-1);
  t_@{j} = A^(1/gamma_@{j});
@#endfor

model;
  @#for j in 1:N
    lambda*(1 + phi*(k_@{j}/k_@{j}(-1) - 1))
      = beta*lambda(+1)*(exp(a_@{j}(+1))*kappa*A*k_@{j}^(kappa - 1)
        + 1 - delta + (phi/2)*(k_@{j}(+1)/k_@{j} - 1)*(k_@{j}(+1)/k_@{j} + 1));
      [preamble]
      a_@{j} = rho*a_@{j}(-1) + sigE*(e + e_@{j});
  @#endfor
    exp(a_1)*A*k_1(-1)^kappa
  @#for j in 2:N
    + exp(a_@{j})*A*k_@{j}(-1)^kappa
  @#endfor
  =
  (lambda/t_1)^(-gamma_1) + k_1 - (1 - delta)*k_1(-1) + (phi/2)*k_1(-1)*(k_1/k_1(-1) - 1)^2
  @#for j in 2:N
  + (lambda/t_@{j})^(-gamma_@{j}) + k_@{j} - (1 - delta)*k_@{j}(-1)
              + (phi/2)*k_@{j}(-1)*(k_@{j}/k_@{j}(-1) - 1)^2
  @#endfor
  ;
end;

initval;
@#for j in 1:N
    k_@{j} = 1;
    a_@{j} = 0;
  @#endfor
  lambda = 1;
end;

steady;

shocks;
  var e; stderr 1;
  @#for j in 1:N
    var e_@{j}; stderr 1;
  @#endfor
end;

@#for j in 1:N
  limits!("k_@{j}", min = 0.8, max = 1.2);
  limits!("a_@{j}", min = -0.8*sigE/(1 - rho), max = 0.8*sigE/(1 - rho));
@#endfor

(SG_grid, sgws) = sparsegridapproximation(gridDepth=3,maxRef=0);
(DDSG_grid, ddsgws) = DDSGapproximation(gridDepth=3,maxRef=0,k_max=1);
```

# Dimensions	SG Level	# of Points	Avg. Error	Max. error (99.9%)
4	3	137	-3.62	-2.61
4	5	1105	-4.21	-3.10
4	7	7,537	-4.64	-3.31

Table 1: Average (Avg.) and maximum (Max.) Euler equation residuals (i.e., errors) for the 4-dimensional IRBC model across increasing SG refinement levels (SG Level), with corresponding grid point counts. All errors are reported in $\log_{10}$ scale.

6 Results

We now evaluate the performance of SGs and DDSG within Dynare for computing global solutions to our benchmark IRBC model. Section 6.1 reports SG results, while Section 6.2 examines the DDSG performance.[18] These experiments illustrate that global solutions of high-dimensional stochastic models can be computed both accurately and swiftly in Dynare without any need to resort to high-performance computing and with only minimal modifications, thereby broadening the scope of applications available to the Dynare community. This section demonstrates that i) SG and DDSG operate reliably within Dynare, and ii) leveraging Dynare's perturbation solution as an initial guess for the time iteration algorithm significantly reduced the time to solution. For clarity of exhibition, we disable SG adaptivity throughout this section and employ a fixed SG with a specified refinement level. Discussions on adaptivity and model-specific hyperparameter tuning for SGs and DDSG are beyond the scope of this paper and are addressed in Brumm and Scheidegger (2017) and Eftekhari and Scheidegger (2022), respectively.

6.1 Solving the IRBC model with SGs and Dynare

To systematically evaluate how accuracy varies with grid refinement ℓ and problem dimensionality, we examine SG solutions for the IRBC model, following the performance metrics discussed in Juillard and Villemot (2011) and Brumm and Scheidegger (2017). The time iteration algorithm (Algorithm 1), initialized with Dynare's first-order perturbation solution, was executed until either an accuracy of $\texttt{tol} = 1 \cdot 10^{-7}$ was reached on the SG points or the error ceased to decrease, a condition known as "early stopping" in the machine learning literature. Table 1 presents Euler equation residuals for the four-dimensional (2-country) IRBC model, computed over a $10,000$-step simulated trajectory (discarding $1,000$ burn-in steps) from the stochastic steady state. As expected, both the maximum and average errors decrease consistently with higher SG refinement levels, remaining reasonably low even with modest grid point counts. We next increase the problem dimensionality from $d = 4$ to $d = 16$ while holding the grid level constant. Table 2 indicates that performance remains relatively consistent despite significant dimensional growth. Furthermore, the quality of the results reported here is at least on par with that achieved by other global solution methods (e.g., Juillard and Villemot, 2011, and references therein). Table 2 shows that global solutions for 4 and 8-dimensional models require only seconds to few minutes, whereas 16-dimensional cases demand about half an hour (cf. Table 3).[19] Although the increase in runtime is notable, it exhibits subexponential

[18] All tests presented in this section used a standard laptop with an Intel Core i9-12900H (14 cores, 20 threads, 2.5 GHz) and 64 GB of memory.

[19] Although a detailed discussion is beyond the scope of this paper, one could accelerate the time iteration algorithm by initially employing coarse SGs and progressively refining them to finer levels. For instance, Brumm et al. (2017) show that SGs can reduce computation time by an order of magnitude by using a level-2 grid for 200 iterations, followed by 80 iterations on a level-3 grid or higher.

# Dimensions	SG Level	# of Points	Avg. Error	Max. error (99.9%)	Time (s/step)
4	3	137	-3.62	-2.61	0.13
8	3	849	-3.78	-2.71	2.09
16	3	6,049	-4.05	-2.94	267.35

Table 2: Average (Avg.) and maximum (Max.) solution errors for 4- to 16-dimensional IRBC models, alongside the corresponding grid point counts for SGs at a fixed refinement level 3, reported on a $\log_{10}$ scale, with indicative average runtimes in seconds per time iteration step (s/step).

growth and can, if needed, further be reduced by orders of magnitude through the use of high-performance computing resources (Scheidegger et al., 2018).

Next, we demonstrate that a well-informed initial guess for the time iteration algorithm (Algorithm 1) significantly accelerates convergence (i.e., drastically reduces the time to solution). Table 3 shows that initializing the computations with Dynare's linear solution reduces iteration steps by factors of approximately 5 (37 vs. 188 steps) in the 4-dimensional model and nearly 20 times (5 vs. 94 steps) in the 16-dimensional IRBC, compared to a naive, constant initial guess ($\mathcal{I}\, p_{\text{guess}} = 1$ for all individual policies). Moreover, a good initial guess becomes increasingly important with rising dimensionality.

# Dimensions	SG Level	# of Points	# TI steps Dyn. Guess	# TI Steps Naive Guess
4	3	137	37	188
8	3	849	11	168
16	3	6,049	5	94

Table 3: The table reports the number of iteration steps required by Algorithm 1 to achieve the target tolerance (`tol`), as evaluated on SG points at refinement level 3, for models of increasing dimensionality. The columns labeled "# TI Steps Dyn. Guess" and "# TI Steps Naive Guess" indicate the number of steps for models initialized with the linear Dynare solution and with a naive, constant initial guess, respectively.

A severe drawback of SGs of all types, including Smolyak's method (Krueger and Kubler, 2004), is that the number of grid points grows very fast with the level of the approximation, as was shown in Figure 2. It is, therefore, often not practical to increase accuracy by simply going to the next level. For instance, in 50 dimensions, a refinement level-3 SG comprises approximately 5,000 points, whereas a level-4 grid contains roughly 170,000 points, resulting in a substantial increase in computational burden. Adaptive sparse grids can resolve this problem to some extent, as intermediate grid sizes can be reached by choosing the refinement threshold and maximum refinement level appropriately (cf. Brumm and Scheidegger (2017)). However, for very high-dimensional problems (say, $d > 20$), even they can fail. We, therefore, explore DDSG in Section 6.2, which addresses this limitation by exploiting latent, additively separable structures in the model.

6.2 Solving the IRBC model with DDSG and Dynare

Having assessed the accuracy of our global solutions to the IRBC model and its scaling with SG resolution, we now systematically investigate how the performance of the DDSG method scales with increasing problem dimensionality. In accordance with our findings in Section 6.1 (see Table 2), we fix the SG refinement level of each DDSG component function to 3, since a level 3 SG previously yielded accurate results. Given the relative simplicity of the IRBC model under

# Dimensions	# of Points	Avg. Error	Max. error (99.9%)	Time (s/step)
4	36	-3.74	-2.56	0.19
8	72	-3.91	-2.67	1.62
16	144	-3.97	-2.88	19.20
50	450	-3.91	-2.48	1849.41
100	900	-3.97	-2.45	7165.30

Table 4: Average (Avg.) and maximum (Max.) errors for IRBC models of dimensions 4 to 100, alongside grid point counts for DDSG expansions with order $\mathcal{K} = 1$, and the SG component functions fixed at a refinement level 3. Errors and counts are reported on a $\log_{10}$ scale, with indicative average runtimes per time step in seconds (s/step).

consideration, we conjecture that an additively separable structure is present and, accordingly, truncate the DDSG expansion at $\mathcal{K} = 1$.

Table 4 reports the Euler equation residuals for the IRBC model across dimensions ranging from four (2-country) to one hundred (50-country). These residuals are computed over a simulated trajectory of $10,000$ steps, with the first $1,000$ steps discarded as burn-in, starting from the stochastic steady state. Both the average and maximum errors remain consistently low across all dimensions, indicating robust performance. As discussed above, the quality of the results reported here is at least comparable to that achieved by other global solution methods for similar IRBC models, while operating on much larger state spaces (e.g., Juillard and Villemot, 2011 and references therein).

Furthermore, this table indicates that runtimes increase more gradually than in the pure SG case (see Table 2), confirming that evaluating N one-dimensional component functions is far more efficient for large N than constructing an N-dimensional SG when a latent additively separable structure exists. For the four-dimensional model, the DDSG solution requires approximately forty percent more time per timestep due to the additional overhead required to carry around multiple grid structures in the computer memory. However, this overhead diminishes in the eight-dimensional case, where runtimes are roughly equivalent (i.e., slightly favoring DDSG). In the sixteen-dimensional case, a single DDSG timestep is over thirteen times faster. This advantage will become increasingly pronounced for higher dimensions, implying that DDSG keeps models tractable in scenarios where SG methods become infeasible.[20]

Next, we show that employing a well-informed initial guess for the time iteration algorithm (Algorithm 1) significantly accelerates convergence within the DDSG framework relative to a naive, constant initial guess (i.e., $\mathcal{I}p_{\text{guess}} = 1$ for all individual policies), consistent with previous findings. To ensure comparability between the SG and DDSG methods, we examine the four-, eight-, and sixteen-dimensional cases. Table 5 reveals that initializing computations

# Dimensions	# of Points	# TI steps Dyn. Guess	# TI Steps Naive Guess
4	36	7	209
8	72	3	186
16	144	2	167

Table 5: Number of iteration steps required by Algorithm 1 to reach the target tolerance (`tol`), evaluated at DDSG points with $\mathcal{K} = 1$, and SG refinement level 3, for IRBC models of increasing dimensionality. The columns "# TI Steps Dyn. Guess" and "# TI Steps Naive Guess" report the steps for models initialized with Dynare's linear solution and a naive zero guess, respectively.

with Dynare's linear solution reduces the number of iterations by factors of approximately 5 (7

[20] Note that these figures are conservative, as our code remains unoptimized.

vs. 209 steps) in the four-dimensional model and 80 (2 vs. 167 steps) in the sixteen-dimensional IRBC, relative to a naive, constant initial guess. Moreover, the importance of an effective initial guess grows with increasing dimensionality. As observed in the SG case, the initial guess becomes more impactful in higher dimensions; however, in the DDSG framework, Dynare's solution proves even more effective than in the SG case (see Table 3).

7 Conclusion & Outlook

This study enhances Dynare (`Dynare.jl`) by integrating sparse grids and high-dimensional model representation into its native `.mod` files with minimal syntactic changes, enabling global solutions for high-dimensional, non-linear dynamic stochastic models. We implement these techniques in a generic manner, thereby substantially broadening the range of models that Dynare can tackle, and subsequently validate them using an international real business cycle model. Our numerical experiments demonstrate their scalability, handling up to at least 100 dimensions on standard laptops within minutes to hours. Our methods mitigate the curse of dimensionality, reduce the time to solution by leveraging Dynare's perturbation-based initial guesses (yielding speedups of up to 80 times), and maintain consistent accuracy as model dimensionality increases.

We conclude that these methodological additions significantly enhance Dynare's versatility, reinforcing its role as a pivotal tool for macroeconomic analysis. Looking forward, further advancements could be achieved by incorporating recent algorithms, such as those developed in the machine learning literature, thereby expanding Dynare's capacity to address increasingly intricate economic questions and preserving its continued relevance in the field.

References

Stéphane Adjemian, Michel Juillard, Fréderic Karamé, Willi Mutschler, Johannes Pfeifer, Marco Ratto, Normann Rion, and Sébastien Villemot. Dynare: Reference manual, version 6. Dynare Working Papers 80, CEPREMAP, 2024.

Adrien Auclert, Bence Bardóczy, Matthew Rognlie, and Ludwig Straub. Using the sequence-space jacobian to solve and estimate heterogeneous-agent models. *Econometrica*, 89(5): 2375–2408, 2021. doi: https://doi.org/10.3982/ECTA17434. URL https://onlinelibrary.wiley.com/doi/abs/10.3982/ECTA17434.

Marlon Azinovic and Jan Žemlička. Economics-inspired neural networks with stabilizing homotopies. *arXiv preprint arXiv:2303.14802*, 2023.

Marlon Azinovic, Luca Gaegauf, and Simon Scheidegger. Deep equilibrium nets. *International Economic Review*, 63(4):1471–1525, 2022.

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

Johannes Brumm and Jakob Hußmann. Public debt in calibrated olg models: Fiscal arithmetic versus welfare analysis. *Available at SSRN 4510125*, 2024.

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

Johannes Brumm, Michael Grill, Felix Kubler, and Karl Schmedders. Collateral requirements and asset prices. *International Economic Review*, 56(1):1–25, 2015a.

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

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

Johannes Brumm, Christopher Krause, Andreas Schaab, and Simon Scheidegger. Sparse grids for dynamic economic models. In *Oxford Research Encyclopedia of Economics and Finance*. Oxford University Press, 2022.

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

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

Yongyang Cai and Kenneth L Judd. Advances in numerical dynamic programming and new applications. *Handbook of computational economics*, 3, 2014.

Yongyang Cai and Thomas S. Lontzek. The social cost of carbon with economic and climate risks. *Journal of Political Economy*, 127(6):2684–2734, 2019. doi: 10.1086/701890. URL https://doi.org/10.1086/701890.

Hui Chen, Giovanni Gambarotta, Simon Scheidegger, and Yu Xu. A dynamic model of private asset allocation. *arXiv preprint arXiv:2503.01099*, 2025.

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

Fabrice Collard and Michel Juillard. Accuracy of stochastic perturbation methods: The case of asset pricing models. *Journal of Economic Dynamics and Control*, 25(6):979–999, 2001. ISSN 0165-1889. doi: https://doi.org/10.1016/S0165-1889(00)00064-6. URL https://www.sciencedirect.com/science/article/pii/S0165188900000646. Computing, economic dynamics, and finance.

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

Victor Duarte, Diogo Duarte, and Dejanir H Silva. Machine learning for continuous-time finance. *The Review of Financial Studies*, 37(11):3217–3271, 09 2024. ISSN 0893-9454. doi: 10.1093/rfs/hhae043. URL https://doi.org/10.1093/rfs/hhae043.

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

Aryan Eftekhari and Simon Scheidegger. High-dimensional dynamic stochastic model representation. *SIAM Journal on Scientific Computing*, 44(3):C210–C236, 2022. doi: 10.1137/21M1392231. URL https://doi.org/10.1137/21M1392231.

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

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

Jesús Fernández-Villaverde, Samuel Hurtado, and Galo Nuño. Financial frictions and the wealth distribution. *Econometrica*, 91(3):869–901, 2023.

Jesús Fernández-Villaverde, Galo Nuño, and Jesse Perla. Taming the curse of dimensionality: Quantitative economics with deep learning. Working Paper 33117, National Bureau of Economic Research, November 2024. URL http://www.nber.org/papers/w33117.

Aleksandra Friedl, Felix Kübler, Simon Scheidegger, and Takafumi Usui. Deep uncertainty quantification: With an application to integrated assessment models. Working paper, University of Lausanne, 2023.

Luca Gaegauf, Simon Scheidegger, and Fabio Trojani. A comprehensive machine learning framework for dynamic portfolio choice with transaction costs. *Swiss Finance Institute Research Paper*, (23-114), 2023.

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

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

Wouter J. Den Haan, Kenneth L. Judd, and Michel Juillard. Computational suite of models with heterogeneous agents ii: Multi-country real business cycle models. *Journal of Economic Dynamics and Control*, 35(2):175 – 177, 2011. ISSN 0165-1889. doi: 10.1016/j.jedc.2010.09.010. URL http://www.sciencedirect.com/science/article/pii/S0165188910002149. Computational Suite of Models with Heterogeneous Agents II: Multi-Country Real Business Cycle Models.

Jiequn Han, Yucheng Yang, and Weinan E. Deepham: A global solution method for heterogeneous agent models with aggregate shocks. *arXiv preprint arXiv:2112.14377*, 2021.

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

Giles Hooker. Generalized functional anova diagnostics for high-dimensional functions of dependent variables. *Journal of Computational and Graphical Statistics*, 16(3):709–732, 2007. doi: 10.1198/106186007X237892. URL https://doi.org/10.1198/106186007X237892.

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

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

Michel Juillard. Dynare-a program for the resolution of non-linear models with forward-looking variables. *Release Matlab*, 3, 1994.

Michel Juillard and Sébastien Villemot. Multi-country real business cycle models: Accuracy tests and test bench. *Journal of Economic Dynamics and Control*, 35(2):178–185, 2011. URL http://ideas.repec.org/a/eee/dyncon/v35y2011i2p178-185.html.

Hanno Kase, Leonardo Melosi, and Matthias Rottner. Estimating nonlinear heterogeneous agents models with neural networks. Discussion Paper 17391, CEPR, 2022.

Robert Kollmann, Serguei Maliar, Benjamin A Malin, and Paul Pichler. Comparison of solutions to the multi-country real business cycle model. *Journal of Economic Dynamics and Control*, 35 (2):186–202, 2011.

Laurence Kotlikoff, Felix Kubler, Andrey Polbin, and Simon Scheidegger. Pareto-Improving Carbon-Risk Taxation. *Economic Policy*, 02 2021. ISSN 0266-4658. doi: 10.1093/epolic/eiab008. URL https://doi.org/10.1093/epolic/eiab008. eiab008.

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

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

Felix Kübler, Simon Scheidegger, and Oliver Surbek. Using machine learning to compute constrained optimal carbon tax rules. *arXiv preprint arXiv:2507.01704*, 2025.

Genyuan Li and Herschel Rabitz. General formulation of HDMR component functions with independent and correlated variables. *Journal of Mathematical Chemistry*, 50(1):99–130, 2012. ISSN 02599791. doi: 10.1007/s10910-011-9898-0.

Genyuan Li, Carey Rosenthal, and Herschel Rabitz. High dimensional model representations. *The Journal of Physical Chemistry A*, 105(33):7765–7777, 2001. doi: 10.1021/jp010450t.

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

Xiang Ma and Nicholas Zabaras. An adaptive high-dimensional stochastic model representation technique for the solution of stochastic partial differential equations. *J. Comput. Phys.*, 229(10):3884–3915, 2010. ISSN 0021-9991. doi: 10.1016/j.jcp.2010.01.033. URL http://dx.doi.org/10.1016/j.jcp.2010.01.033.

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

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

Alin Murarașu, Gerrit Buse, Dirk Pflüger, Josef Weidendorfer, and Arndt Bode. Fastsg: A fast routines library for sparse grids. *Procedia Computer Science*, 9:354–363, 2012.

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

Galo Nuño, Philipp Johannes Renner, and Simon Scheidegger. Monetary policy with persistent supply shocks. Working Paper Series 11463, CESifo, November 2024. URL https://ssrn.com/abstract=5045497.

Jonathan Payne, Adam Rebei, and Yucheng Yang. Deep learning for search and matching models, 2024. Available at SSRN 4768566.

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

Herschel Rabitz and Ömer F. Aliş. General foundations of high-dimensional model representations. *Journal of Mathematical Chemistry*, 25(2/3):197–233, 1999. ISSN 02599791. doi: https://doi.org/10.1023/A:1019188517934.

Herschel Rabitz, Ömer F. Aliş, Jeffrey Shorter, and Kyurhee Shim. Efficient input–output model representations. *Computer Physics Communications*, 117(1):11–20, 1999. ISSN 0010-4655. doi: https://doi.org/10.1016/S0010-4655(98)00152-0.

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

Michael Reiter. Solving heterogeneous-agent models by projection and perturbation. *Journal of Economic Dynamics and Control*, 33(3):649 – 665, 2009. ISSN 0165-1889. doi: https://doi.org/10.1016/j.jedc.2008.08.010. URL http://www.sciencedirect.com/science/article/pii/S0165188908001528.

Owen Ren, Mohamed Ali Boussaidi, Dmitry Voytsekhovsky, Manabu Ihara, and Sergei Manzhos. Random sampling high dimensional model representation gaussian process regression (rs-hdmr-gpr) for representing multidimensional functions with machine-learned lower-dimensional terms allowing insight with a general method. *Computer Physics Communications*, 271:108220, 2022. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2021.108220. URL https://www.sciencedirect.com/science/article/pii/S0010465521003325.

Philipp Renner and Simon Scheidegger. Machine learning for dynamic incentive problems. *Available at SSRN 3462011*, 2018. Working paper.

Andreas Schaab. Micro and macro uncertainty. *Available at SSRN 4099000*, 2020. URL http://dx.doi.org/10.2139/ssrn.4099000.

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

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

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

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

I. M. Sobol. Theorems and examples on high dimensional model representation. *Reliability Engineering & System Safety*, 79(2):187–193, 2003. ISSN 09518320. doi: 10.1016/S0951-8320(02)00229-6. URL http://www.sciencedirect.com/science/article/pii/S0951832002002296.

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

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

Vytautas Valaitis and Alessandro T. Villa. A machine learning projection method for macro-finance models. *Quantitative Economics*, 15(1):145–173, 2024. doi: https://doi.org/10.3982/QE1403. URL https://onlinelibrary.wiley.com/doi/abs/10.3982/QE1403.

Andreas Waechter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1): 25–57, 2006. ISSN 0025-5610. doi: 10.1007/s10107-004-0559-y. URL http://dx.doi.org/10.1007/s10107-004-0559-y.

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

Xiu Yang, Minseok Choi, Guang Lin, and George Em Karniadakis. Adaptive anova decomposition of stochastic incompressible and compressible flows. *Journal of Computational Physics*, 231(4):1587 – 1614, 2012. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2011.10.028. URL http://www.sciencedirect.com/science/article/pii/S0021999111006280.

P. C. Young and M. Ratto. Statistical emulation of large linear dynamic models. *Technometrics*, 53(1):29–43, 2011. doi: 10.1198/TECH.2010.07151. URL https://doi.org/10.1198/TECH.2010.07151.