Tensor-Train Approximation for High-Dimensional Economic Models*

Johannes Brumm Jakob Hußmann

June 1, 2026

Abstract

We introduce a scalable and broadly applicable method for computing global solutions of dynamic stochastic models. Tensor train approximation (TTA) identifies latent low-rank structure in high-dimensional functions, thereby capturing complex non-linearities while mitigating the curse of dimensionality — the number of parameters grows only linearly in dimension. We show that our TTA approach can accommodate irregular ergodic sets, is compatible with the endogenous grid method, enables quasi-analytical expectations, and can solve continuous-time models via least-squares projection, all in high dimensions. We demonstrate TTA's scalability by solving multi-country models of growing dimension with only moderate increases in compute time and no loss of accuracy. We show TTA's versatility in heterogeneous-agent models with large aggregate shocks. To approximate the wealth distribution efficiently, we introduce a simulation-based moment-selection procedure. In a model with stochastic wealth taxation, using the first nine selected moments, instead of mean wealth only, reduces approximation errors tenfold.

Keywords: global solutions, tensor train decomposition, high-dimensional approximation, endogenous grid method, heterogeneous agents, wealth taxation.

JEL Classification Codes: C61, C63, C68, E21, D31, D52.

*Contact details: Johannes Brumm, Karlsruhe Institute of Technology, johannes.brumm@kit.edu. Jakob Hußmann, Karlsruhe Institute of Technology, jakob.hussmann@kit.edu. We thank Benjamin Born, Lukas Frank, Felix Kubler, Michael Reiter, Simon Scheidegger, Yu-Cheng Yang, and participants at EEA-ESEM 2024 in Rotterdam, the Conference on Deep Learning for Economics and Finance 2025 in Turin and the 2026 Meeting of the VfS Standing Field Committee Macroeconomics for helpful comments. We acknowledge financial support from the ERC project SOLG for Policy (101042908).

1 Introduction

Macroeconomic research has, over recent decades, increasingly focused on the role of heterogeneity, thereby dramatically raising the complexity of solving the respective models. As a result, most studies have relied on perturbation techniques for aggregate dynamics.[1] While relatively simple and light on computations, perturbation methods suffer from limited accuracy beyond the neighborhood of the expansion point. Global solution approaches do not, but face the curse of dimensionality. Recent advances in overcoming that challenge fall mainly into two categories, sparse grids and neural nets.[2] Yet, no single dominant technique has emerged due to inherent trade-offs. Sparse grids, while robust and effective in medium dimensions, require too many points in high dimensions and lack flexibility in fitting ergodic sets. Neural networks, in contrast, are flexible and scalable, yet prone to overfitting and fragile convergence behavior.

This paper introduces a novel approach for solving high-dimensional economic models which builds on the tensor train decomposition of Oseledets (2011). Tensor train approximation (TTA) enables the use of tensor product bases in high-dimensional approximation despite the exponential growth in the number of their basis coefficients. It does so by approximating the d-dimensional tensor of basis coefficients by d separate 3-dimensional tensor cores, thereby reducing parameter growth from exponential to linear. TTA exploits latent low-rank structure in the basis coefficients of the tensor product basis and can robustly and efficiently be implemented with an alternating least squares (ALS) scheme.

The resulting framework features three key strengths in solving dynamic stochastic models. First, it is grid-free and highly flexible, allowing for arbitrary ergodic sets and the application of the endogenous grid method (EGM) in high dimensions. Second, it supports analytic integration and differentiation, enabling fast, exact computation of expectations and the solution of continuous-time models through least-squares projection. Finally, TTA works, in principle, with any type of basis functions, which can thus be chosen to best fit policy or value functions in specific applications.

Compared to sparse grids, TTA offers much greater flexibility as it is not restricted to regular grids. Compared to neural networks, it often delivers comparable approximation accuracy with fewer effective degrees of freedom, which reduces the risk of overfitting, noisy solutions, and unstable convergence behavior. Compared to both

[1] For discrete-time linearizations, see Reiter (2009), Auclert et al. (2021), and Bayer and Luetticke (2020). In continuous time, based on Achdou et al. (2022), Ahn et al. (2018) provide linearizations of the Kolmogorov Forward Equation. A growing literature explores higher-order perturbations, including Bilal (2023), Bhandari et al. (2023), and Bayer et al. (2026).

[2] Brumm et al. (2022) and Fernández-Villaverde et al. (2024) review the literature on solving dynamic economic models using sparse grids and neural nets, respectively.

approaches, TTA has the potential to approximate economic policy functions more efficiently, in particular when these functions exhibit low-curvature regions, smoothness, or limited cross-dimensional interactions.

To demonstrate the flexibility, efficiency, and scalability of TTA in solving dynamic economic models, we apply it to the international real business cycle (IRBC) model as specified in Brumm and Scheidegger (2017). We first embed TTA in a standard time-iteration algorithm and show that this simple approach already scales favorably. We also show how accuracy is driven by the order of basis functions and by the ranks of the tensor cores, which govern interactions across dimensions. By increasing basis order and ranks, accuracy can be improved by several orders of magnitude. We then unleash TTAs' full potential by combining it with a high-dimensional endogenous grid method and by enabling the quasi-analytical evaluation of expectations. For the smooth version of the model we use a polynomial basis, which turns out to deliver high accuracy at modest polynomial order. For the non-smooth version, with occasionally binding irreversible-investment constraints, we employ a piecewise linear hierarchical basis. In both model versions compute time increases only by two orders of magnitude between the 11- and 51-dimensional cases, while the average Euler error remains well below 0.001% (0.01%) in the smooth (non-smooth) case.[3] The IRBC model thus highlights both the excellent scaling properties of TTA and the additional gains made possible by quasi-analytical computation of expectations and by the high-dimensional endogenous grid method.

To illustrate the broad applicability of TTA, we also apply it to heterogeneous-agent models in continuous time. To ensure that the wealth distribution — the primary source of multi-dimensionality in these models — meaningfully affects household decisions, we introduce large wealth-tax shocks. In the absence of these shocks, the mean suffices to summarize the distribution. However, after introducing a stochastic wealth tax of 10% or 20% occurring with a 2.5% probability per quarter, reasonable accuracy can only be maintained if additional moments are incorporated.[4] We do so by using a novel a posteriori model reduction procedure, based on proper orthogonal decomposition, and find that the moments identified this way substantially reduce the errors in the master equation. Going from the three-dimensional specification to the eleven-dimensional specification — from one distributional moment to nine moments — reduces the average error by one order of magnitude. The heterogeneous agent application demonstrates two key insights. First, TTA can effectively handle the workhorse

[3] By comparison, it is very hard to bring errors down to such levels with (adaptive) sparse grids, see Figures 8 and 11 in Brumm and Scheidegger (2017). Moreover, sparse grids require (even when only a level three grid is employed) an increase in computation time of more than four orders of magnitude for such an increase in dimensionality, see Figure 15 in that paper.

[4] In a similar setup with wealth taxation, Reiter (2009) already demonstrates the need to include higher-order moments.

models of modern macroeconomics, even in the presence of large aggregate shocks. Second, TTA enables solving continuous-time models via least-squares projection.

All in all, the different applications presented in this paper demonstrate that TTA is a scalable, flexible, and broadly applicable method to compute accurate global solutions of high-dimensional dynamic stochastic models. We believe that TTA adds a great new tool to macroeconomists' toolbox. It may emerge as the most efficient and practical approach for many applications, particularly those that require global approximation yet also feature latent low-rank structure.

Related Literature. This paper connects to two strands of the literature. The first is on global solution techniques for high-dimensional economic models, including heterogeneous agent models; the second is on tensor-decomposition methods in general, and on the tensor train decomposition in particular.

Recent advances in global solution methods for high-dimensional dynamic stochastic models fall mainly into two categories, sparse grids and neural nets. Sparse grids significantly mitigate the curse of dimensionality by reducing the exponential growth of grid sizes. Krueger and Kubler (2006) introduce sparse grids with polynomial basis functions to economics and Judd et al. (2014) extend this framework by incorporating anisotropic grids and adaptive domain selection. Brumm and Scheidegger (2017) further advance the method by employing hierarchical basis functions and enabling local adaptivity, while Schaab and Zhang (2022) apply adaptive sparse grids to continuous-time models.

A growing literature applies machine-learning techniques, and neural networks in particular, to approximate value and policy functions in dynamic economic models.[5] Maliar et al. (2021) and Azinovic et al. (2022) show that deep neural networks can be trained to solve Bellman equations and first-order equilibrium conditions in discrete-time models with high-dimensional state spaces. Fernández-Villaverde et al. (2020), Gopalakrishna (2021), and Sauzet (2021) develop neural-network and projection-based methods for solving continuous-time models. Deep-learning methods have enabled quantitatively rich applications across economics, including household finance (Gorodnichenko et al., 2022), asset pricing and intergenerational risk sharing (Azinovic-Yang & Žemlička, 2025b), macro-finance (Gopalakrishna et al., 2026), optimal monetary policy (Nuño et al., 2024), and climate economics (Folini et al., 2025).

When it comes to heterogeneous-agent models with aggregate risk, a key challenge is that the aggregate state includes the cross-sectional distribution of agents. Han et al. (2026) jointly approximate value functions and learn low-dimensional representations of the distribution. Gu et al. (2024) solve continuous-time heterogeneous-agent

[5] Although neural networks have dominated much of the recent machine-learning literature, Scheidegger and Bilionis (2019) and Eftekhari and Scheidegger (2022) are noteworthy exceptions.

economies by approximating the master equation with neural networks while retaining a full histogram of the distribution. Kase et al. (2025) use neural networks to solve and estimate nonlinear heterogeneous-agent New Keynesian models, while Payne et al. (2025) develop a deep-learning approach for search-and-matching models with heterogeneous agents and aggregate shocks.

A recent strand of the literature sidesteps the challenge of approximating the cross-sectional distribution of agents by approximating individual choices as functions of states other than the natural recursive state of the model. Azinovic-Yang and Žemlička (2025a) develop a deep-learning method for approximating rational-expectations equilibria using histories of aggregates. Yang et al. (2025) propose a structural deep-reinforcement-learning approach in which agents learn equilibrium dynamics from prices rather than from an explicit law of motion for the distribution.

Our work is also related to the earlier heterogeneous-agent literature on reducing the dimensionality of the distributional state. Krusell and Smith (1998) show that, in some heterogeneous-agent economies, a small set of aggregate moments can forecast equilibrium prices and aggregate dynamics with high accuracy. Ahn et al. (2018) and Reiter (2023) provide a systematic model-reduction approach for linearized heterogeneous-agent models. In contrast to these and other a priori approaches, we develop an a posteriori, simulation-based model-reduction method that identifies low-dimensional state variables from simulated distributional dynamics.

The second strand of the literature to which this paper contributes concerns tensor train (TT) methods for high-dimensional problems. Following the seminal work of Oseledets (2011), the TT decomposition has gained significant attention as a tool for representing high-dimensional tensors with linear scaling in dimensionality. Holtz et al. (2012) propose the efficient and numerically stable ALS algorithm for optimization problems in the TT-format that we employ. Grasedyck and Krämer (2019) study optimal rank selection for tensor decompositions, offering complementary strategies to enhance the convergence and stability of ALS-type methods. Gorodetsky et al. (2019) develop a continuous analogue of the TT decomposition, enabling the representation of multivariate functions rather than discrete arrays. Bigoni et al. (2016) introduce a spectral tensor train decomposition that combines the TT structure with spectral polynomial approximation, further improving accuracy in function approximation. Recent work has applied TT techniques to the solution of high-dimensional PDEs and control problems. Dektor et al. (2021) develop a rank-adaptive method that combines functional TT expansions with a dynamic algorithm that adjusts ranks during time integration. Dolgov et al. (2021) apply TT methods to solve high-dimensional Hamilton-Jacobi-Bellman equations, demonstrating their potential in optimal control problems. Richter et al. (2021, 2024) extend TT-based solvers to parabolic PDEs and compare their performance favorably to neural network-based approaches, emphasiz-

ing robustness and interpretability. Bachmayr (2023) offers a comprehensive review of low-rank tensor techniques, including TT formats, covering theoretical underpinnings, numerical algorithms, and applications to PDEs and high-dimensional optimization. Recent work by Dennis (2026) also applies tensor-train decomposition methods to dynamic economic models, yet in a manner that is complementary to our approach in several respects. Most importantly, his method is grid-based and requires more demanding local solves to optimize the tensor train parameters, which leads to less favorable scaling properties than in our framework.

2 Tensor Train Approximation

This section explains how to approximate multi-dimensional functions via TTA. First, we show how to write the coefficients of a tensor-product basis for approximating d-dimensional functions as an order d tensor. Second, we demonstrate how to alleviate the curse of dimensionality by approximating the latter with a so-called tensor train composed of d different order three tensors. Third, we provide an ALS algorithm for computing the coefficients of such a TTA efficiently and robustly. Finally, we point out that derivatives and integrals of TTAs can be computed analytically. In Appendix A.2, we show that the least-squares approximation problem solved by the TTA algorithm is a special case of a broader class of quadratic problems that can be minimized efficiently using ALS. In Section 4, we construct another special case of this class in the context of solving master equations.

2.1 Tensor Representation of Basis Coefficients

To approximate a multi-dimensional function, $f : \mathbb{R}^d \to \mathbb{R}$, $d > 1$, a natural approach is to write it as a sum of products of one-dimensional basis functions. Assuming, for ease of exposition, that all dimensions are treated equally, we allow for m one-dimensional basis functions, $\phi_1, \ldots, \phi_m$, where $\phi_i : \mathbb{R} \to \mathbb{R}$. Thus, to approximate f at $x \in \mathbb{R}^d$, denoted by $\mathcal{F}(x)$, we evaluate $m \cdot d$ basis functions $\phi_{i_1}, \ldots \phi_{i_d}$ with $i_k = 1, \ldots, m$ and $k = 1, \ldots, d$. In the applications below, we use polynomial basis functions and piecewise linear hierarchical basis functions. To map these basis functions to the corresponding scalar $\mathcal{F}(x)$, we define an order d tensor A containing all basis coefficients. An order d tensor is a multidimensional generalization of a matrix. The entry $A[i_1 \ldots i_d]$ indicates the weight by which the product of the respective basis functions contributes to the value of $\mathcal{F}(x)$:

$$\mathcal{F}(x) = \sum_{i_1=1}^{m} \ldots \sum_{i_d=1}^{m} A[i_1, \ldots, i_d] \cdot \phi_{i_1}(x_1) \cdot \ldots \cdot \phi_{i_d}(x_d). \tag{1}$$

As A contains weights for all combinations of bases $\phi_1, \ldots, \phi_m$ across d dimensions it consists of m^d entries and is thus clearly susceptible to the curse of dimensionality. Computing all its entries becomes rapidly infeasible as d increases. However, doing so might not be necessary for achieving an accurate approximation. Instead of using A in its entirety, we can employ a lower-dimensional object to approximate it. This is exactly what TTA does, as we explain next. Of course, such an approach is only promising if A possesses a latent low-rank structure. In economic applications this is often the case, as the applications in Sections 3 and 4 show.

2.2 Tensor Train Format

Following Oseledets (2011), we approximately factorize an order-d tensor A into a product of d order-three tensors W^i, each of size $r_{i-1} \times m \times r_i$ for $i = 1, \ldots, d$, where the intermediate ranks satisfy $r_i \geq 1$ and the boundary ranks are $r_0 = r_d = 1$. Accordingly, we suppress r_0 and r_d in the notation below. We call $\mathcal{T} = (W^1, \ldots, W^d)$ the tensor-train decomposition of A, and each W^i a tensor core, or wagon, of the train:

$$A \approx \left(W^1_{i_1 j_1} \cdot W^2_{j_1 i_2 j_2} \cdot \ldots \cdot W^{d-1}_{j_{d-2} i_{d-1} j_{d-1}} \cdot W^d_{j_{d-1} i_d} \right)_{i_1 \ldots i_d}. \tag{2}$$

The ranks govern the size of the cores and control the dependencies between them. TTA approximates each entry of A through a chain of contractions over the intermediate indices linking consecutive cores. To define how exactly it does so, we employ the so-called Einstein convention, summing over repeated indices and letting the free indices determine the output tensor's shape.[6]

Suppose $d = 3$, $m = 3$, $r = 2$, then we are contracting a 3×2 matrix with a $2 \times 3 \times 2$ tensor to obtain a $3 \times 3 \times 2$ tensor, which we then contract with a 2×3 tensor to obtain a $3 \times 3 \times 3$ tensor.[7] In this small example, the tensor train has 24 free parameters, just slightly less than the 27 of A. Yet with $d = 10$, $m = 3$, $r = 2$, the tensor train format already reduces the degrees of freedom from 59.049 to 108. In general, when all basis orders equal m as assumed above and the ranks satisfy $r_\ell \leq r$, the number of free

[6]We use the notation as follows: For all tensors involved in contraction we explicitly denote their dimension indices. If an index present in the original tensors is missing in the result, this implies summation over that index. For example the contraction of two consecutive cores is written as

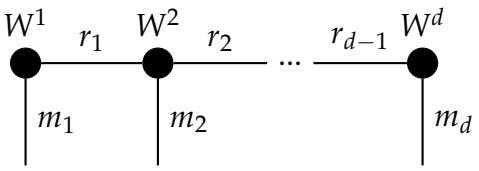
$$U[j_1, i_2, i_3, j_3] = \left(W^2_{j_1 i_2 j_2} \cdot W^3_{j_2 i_3 j_3} \right)_{j_1 i_2 i_3 j_3} [j_1, i_2, i_3, j_3] = \sum_{j_2=1}^{r_2} W^2_{j_1 i_2 j_2}[j_1, i_2, j_2] \cdot W^3_{j_2 i_3 j_3}[j_2, i_3, j_3].$$

Indices in brackets signal specific entries of the tensor, while indices in lower-subscripts denote all entries over this dimension of the tensor.

[7]Unless stated otherwise, scalar notation for m and r indicates that these quantities do not vary across dimensions, except for $r_0 = r_d = 1$. Note that basis order m need not be uniform across cores for the method to work. In some of the applications below, basis order will vary across dimensions.

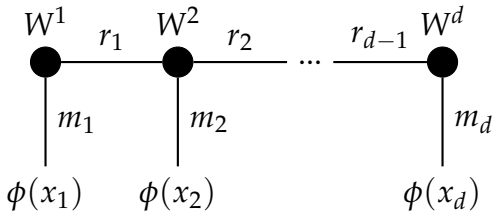
Figure 1: Tensor Train Formulas and Diagrammatic Notation

$$A \approx \left(W^1_{i_1 j_1} W^2_{j_1 i_2 j_2} \cdots W^d_{j_{d-1} i_d} \right)_{i_1 \dots i_d}$$



(a) TT-Decomposition of A

$$\mathcal{F}(x;\mathcal{T}) = \left(W^1_{i_1 j_1} \phi(x_1)_{i_1} \right)_{j_1} \cdot \ldots \cdot \left(W^d_{j_{d-1} i_d} \phi(x_d)_{i_d} \right)_{j_{d-1}}$$



(b) TT-Approximator $\mathcal{F}(x;\mathcal{T})$

Diagrammatic tensor-train notation of the coefficient tensor A in tensor-train form on the left and of the tensor-train approximator $F(x;\mathcal{T})$ on the right. The latter is obtained by contracting the same chain with the local basis vectors $\phi(x_k)$. Filled nodes denote tensors, with $W^1, \dots, W^d$ representing the TT cores. Edges denote tensor indices; connected edges indicate contractions. In particular, horizontal edges carry the TT ranks r_k, while downward edges correspond to indices of order m_k.

parameters of the TTA $\mathcal{T}$ is bounded by dmr^2 as compared to m^d for A. Thus, the TTA approach alleviates the curse of dimensionality: with fixed m and r, the number of parameters grows linearly instead of exponentially in d. Given a tensor train $\mathcal{T}$, we can express an approximator of f as follows:

$$\mathcal{F}(x;\mathcal{T}) = \left(W^1_{i_1 j_1} \cdot \phi(x_1)_{i_1} \right)_{j_1} \cdot \left(W^2_{j_1 i_2 j_2} \cdot \phi(x_2)_{i_2} \right)_{j_1 j_2} \cdot \ldots \cdot \left(W^d_{j_{d-1} i_d} \cdot \phi(x_d)_{i_d} \right)_{j_{d-1}}. \quad (3)$$

Thus, in each dimension ℓ we evaluate the basis functions, obtaining the vector $\phi(x_\ell) = (\phi_1(x_\ell), \dots, \phi_m(x_\ell))$, which we then contract with the core W^ℓ resulting in an order-two tensor, except in case of $\ell = 1$ or $\ell = d$ where we get an order-one tensor. These d tensors are then contracted successively with each other to finally obtain the scalar $\mathcal{F}(x;\mathcal{T})$. Evaluations of the approximator are of computational complexity $\mathcal{O}(mdr^2)$; thus, function evaluations scale efficiently in dimensions d. To provide intuition for TTA, we complement the notation-heavy formal equations with diagrammatic illustrations frequently used in the tensor-train literature. Figure 1 presents a diagrammatic representation of both the decomposition of A and the tensor-train approximator $\mathcal{F}(x;\mathcal{T})$; we return to the same visual style later to illustrate additional concepts.

2.3 Optimizing Tensor Trains — The TTA Algorithm

We now present an algorithm for TT-based function approximation. It computes a TT $\mathcal{T}$ such that $\mathcal{F}(x;\mathcal{T})$ fits the sample points $\{(x^n, y^n)\}_{n=1}^N \subset \mathbb{R}^d \times \mathbb{R}$ in a least squares sense. We first formulate the associated least-squares problem and then introduce the alternating least squares (ALS) scheme for solving it. We denote the resulting TTA with basis orders m and ranks r by $\mathcal{T} = \mathcal{A}\left(m, r, \{x^n, y^n\}_{n=1}^N\right)$. For notational convenience,

we later use the shorthand notation TTA $\left(\{x^n, y^n\}_{n=1}^N\right) \equiv \mathcal{F}\left(\cdot; \mathcal{A}\left(m, r, \{x^n, y^n\}_{n=1}^N\right)\right)$, omitting the parameters m and r for brevity.

For given basis order m and rank r, we fit the TTA approximator $\mathcal{F}(x; \mathcal{T})$ by choosing a tensor train $\mathcal{T} = (W^1, \ldots, W^d)$ to minimize the regularized least squares error on the sample set $\{(x^n, y^n)\}_{n=1}^N$:

$$\min_{W^1,\ldots,W^d} \frac{1}{N} \sum_{i=1}^{N} ||y^i - \mathcal{F}(x^i; W^1, \ldots, W^d)||_2^2 + \lambda \sum_{k=1}^{d} ||W^k||_F^2, \tag{4}$$

where λ is a regularization parameter, and $||\cdot||_F$ is the Frobenius norm. In the above problem the coefficients of different cores interact multiplicatively making the global problem non-convex. Following Holtz et al. (2012), we use an alternating least-squares (ALS) scheme to circumvent this problem. Fixing all cores except one, W^k, makes the mapping from W^k to $\mathcal{F}(x^i; W^1, \ldots, W^k, \ldots W^d)$ linear, so finding the optimal W^k reduces to a simple least-squares problem:

$$\min_{W^k} \frac{1}{N} \sum_{i=1}^{N} ||y^i - \mathcal{F}(x^i; W^1, \ldots, W^k, \ldots W^d)||_2^2 + \lambda ||W^k||_F^2 \tag{5}$$

Exploiting this insight, ALS sweeps over the cores, left to right, and back, solving the local subproblems until convergence. Figure 2 illustrates a single sweep of the ALS algorithm using our graphical notation.

In practice, the ALS algorithm requires two additional ingredients: vectorization and orthonormalization, which we only sketch here and describe in detail in Appendix A.1. Vectorization forms the local subproblems by fixing all cores except W^k and contracting the off-dimensions, so that the approximation depends only on W^k. This reduces the problem to an order-three tensor and, after reshaping, to a small linear system; the required contractions can be computed efficiently using left and right stacks. Each local solve is then followed by an orthonormalization step, in which a matricization of the updated core is decomposed into an orthonormal factor and a residual factor. The former becomes the new W^k, while the latter is absorbed into the next core, preserving the overall solution, maintaining orthonormality in the off-dimensions, and improving numerical stability.

Finally, we briefly comment on convergence of the ALS scheme. Rohwedder and Uschmajew (2013) provide local convergence results for ALS in general minimization problems. In particular, their Theorem 2.11 shows that, when the tensor-train rank is correctly specified and the Hessian of the objective is positive definite, ALS converges locally to a minimum at a linear rate. The positive-definiteness condition is satisfied for the minimization problems considered here, which reduce to least-squares fitting problems. The case of misspecified rank is less well understood theoretically and is

Figure 2: Visualization of an ALS Iteration

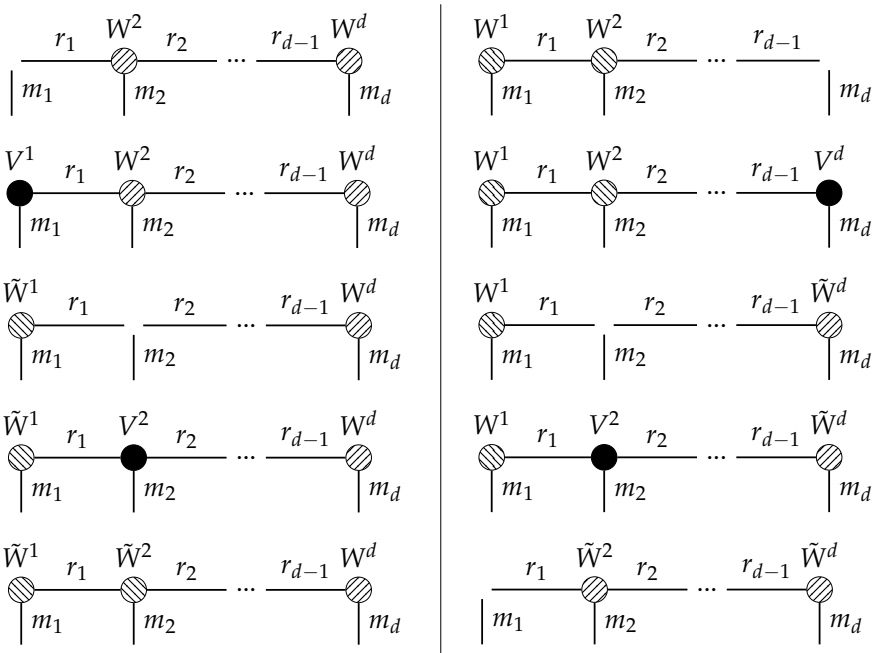


Illustration of one sweep of alternating least squares (ALS). A forward (left–to–right) half-sweep depicted on the left followed by a backward (right–to–left) half-sweep depicted on the right. Hollow nodes indicate the yet-unknown subproblem solution at the current position; filled nodes indicate the just-solved subproblem; an upward (downward) sloping texture marks a right- (left-) orthonormal core.

arguably more relevant in practice. In our numerical applications, however, ALS is robust and converges to solutions with low approximation error as we document in Sections 3 and 4 extensively.

2.4 Analytic Differentiation and Integration

As it will become extremely useful below, we now point out that function representations in TT-format admit efficient rules for differentiation and integration. Let $\mathcal{F}(x;\mathcal{T})$ be a TT approximator, and let $\mathcal{L}^k_{j_{k-1}}$ and $\mathcal{R}^k_{j_k}$ be the left and right stacks obtained by contracting all cores except the k-th with the local basis evaluations as formally defined in Appendix A.1. We can then take the derivative of $\mathcal{F}(x;\mathcal{T})$ with respect to x_k by replacing the basis vector in dimension k with its derivative, and contract the cores in the standard way:

$$\frac{\partial^n \mathcal{F}(x;\mathcal{T})}{\partial x_k^n} = \mathcal{L}^k_{j_{k-1}} \cdot W^k_{j_{k-1} i_k j_k} \cdot \left(\partial^n_{x_k} \phi(x_k)\right)_{i_k} \cdot \mathcal{R}^k_{j_k}. \tag{6}$$

To calculate the indefinite integral of $\mathcal{F}(x;\mathcal{T})$ with respect to x_k one simply replaces the basis vector in dimension k with its anti-derivative:

$$\int \mathcal{F}(x;\mathcal{T})\,\mathrm{d}x_k = \mathcal{L}^k_{j_{k-1}} \cdot W^k_{j_{k-1} i_k j_k} \cdot \Big(\int \phi(x_k)\,\mathrm{d}x_k\Big)_{i_k} \cdot \mathcal{R}^k_{j_k} \;+\; C(x_{-k}), \tag{7}$$

where $C(x_{-k})$ is the integration constant that may depend on all variables except x_k. Definite integrals are obtained analogously. The formulas in (6) and (7) show that, provided the basis functions admit analytic differentiation and integration, partial derivatives and one-dimensional integrals can be computed at the same cost as an ordinary function evaluation. This fact will prove extremely useful in two respects: First, it will allow us, in Section 3, to compute expectations in high-dimensional models both accurately and efficiently. Second, it will allow us, in Section 4, to solve partial differential equations as apparent in continuous-time heterogeneous-agent models.

3 Solving High-Dimensional Models with TTA

To demonstrate the accuracy and scalability of TTA, we apply it to the international real business cycle (IRBC) model as in Brumm and Scheidegger (2017). After briefly describing the model, we introduce our solution strategy in two stages. First, we insert TTA into a standard time iteration routine and show that it works out of the box as a drop-in replacement for existing approximation methods. Second, we exploit additional structure offered by TTA to improve performance: we show how the tensor-train representation enables quasi-analytic computation of the high-dimensional expectations in the Euler equations and how it can be combined with a high-dimensional version of the endogenous grid method (EGM). We then report accuracy and scaling results for this enhanced algorithm in both the smooth IRBC specification and a non-smooth variant with occasionally binding constraints.

3.1 IRBC Model

Physical Economy. The model features M countries, indexed by $j = 1, \ldots, M$, each utilizing its capital stock to produce output via Cobb-Douglas production with fixed labor supply. The output good can be used as either consumption, c_j, or investment, χ_j. Consumption generates utility through an additively separable utility function with discount factor β and per-period utility function of the CRRA type with risk aversion $1/\gamma_j$, varying across countries. Investment is subject to adjustment costs $k_j \cdot \psi \cdot g_j^2/2$, where $g_j \equiv k'_j/k_j - 1$ and $\psi > 0$. Capital depreciates at a rate $\delta > 0$. Countries' productivity is given by

$$\ln a'_j = \rho \ln a_j + \sigma(e'_j + z'), \tag{8}$$

where the country-specific shocks, e'_j, and the global shock, z', are assumed to be standard normal shocks that are independent both across time and from one another. We parameterize the model as in Brumm and Scheidegger (2017), except for the depreciation rate, which we set equal to 0.5% (instead of 1%) to make the irreversibility constraints in the non-smooth model more frequently binding in simulations. All parameters are reported in Table 4 in Appendix B. Details of the non-smooth model are described in Appendix B.2.

Complete Markets. Following Kollmann et al. (2011) and Brumm and Scheidegger (2017), we assume complete markets. This assumption implies that the decentralized competitive equilibrium allocation can be characterized as the solution to a social planner's problem. Subject to aggregate resource constraints, the planner maximizes the weighted sum of country-specific utilities, where each country's utility is weighted by τ_j, a welfare weight that depends on its initial capital stock.

Recursive Equilibrium Conditions. The equilibrium conditions in each period consist of the optimality conditions for investment in each country's capital,

$$\lambda \left[1 + \psi g_j\right] = \beta \mathbb{E}_t \left\{ \lambda' \left[a'_j A \zeta (k'_j)^{\zeta - 1} + 1 - \delta + \frac{\psi}{2} g'_j (g'_j + 2) \right] \right\}, \quad j = 1, \ldots, M, \tag{9}$$

and the aggregate resource constraint, given by

$$\sum_{j=1}^{M} \left(a_j A (k_j)^{\zeta} + k_j \left(1 - \delta - \frac{\psi}{2} g_j^2 \right) - k'_j - \left(\frac{\lambda}{\tau_j} \right)^{-\gamma_j} \right) = 0. \tag{10}$$

These equations jointly determine the capital choice of each country j, k'_j, and the Lagrange multiplier on the resource constraint, λ. For the derivation of these conditions, see Brumm and Scheidegger (2017).

3.2 Simple Solution Approach

This section presents a simple TTA-based solution strategy, intentionally leaving aside several more advanced capabilities that TTA makes possible. We introduce those features in the next section as powerful extensions of the baseline approach.

A Simple TTA-Algorithm. The recursive state of the economy

$$y = (k_1, \ldots, k_M, a_1, \ldots, a_M) \in Y \subset \mathbb{R}^{2M}, \tag{11}$$

consists of the country-specific capital stocks k_j, and productivities a_j. The policy we solve for consists of the capital choices k'_j and the Lagrange multiplier λ:

$$p : Y \to \mathbb{R}^{M+1}, p(y) = \left(k'_1(y), \ldots, k'_M(y), \lambda(y)\right). \tag{12}$$

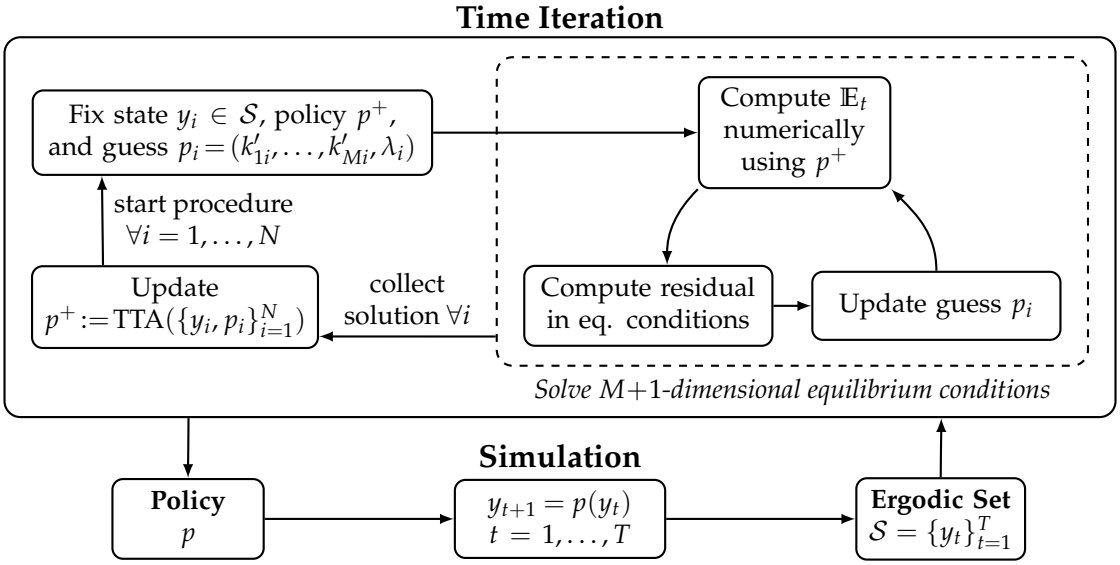
A recursive equilibrium consists of a policy function p and an ergodic set $\mathcal{E} \subset Y$, such that (i) policies satisfy the equilibrium conditions on the ergodic set and (ii) simulating the policies generates the ergodic set. Corresponding to conditions (i) and (ii), the algorithm consists of two loops, with the inner loop computing policies that approximately satisfy optimality conditions (9) and (10) on a set of sample states $\mathcal{S}$ using time iteration. The outer loop, once the inner loop converged, simulates the model to update the set $\mathcal{S}$ to get closer to the ergodic set. Figure 3 illustrates this algorithm and Appendix B.3 provides a detailed description. Note that this approach is only possible as approximation via TTA can operate on irregular datasets. In high dimensions it becomes increasingly inefficient to approximate equilibrium functions on hypercubes enveloping the ergodic set, as the ratio of volumes between ergodic set and hypercube declines drastically with dimension. Moreover, the corners of the hypercube represent extreme conditions with zero probability that unnecessarily require substantial non-linearity in the approximation.

To measure the accuracy of the solution, we follow the standard approach of computing unit-free (relative) Euler equation errors for each of the M countries, as well as an error for the aggregate resource constraint. These measures reflect how closely the numerical solution satisfies the model's equilibrium conditions. Details on the Euler equation errors are relegated to Appendix B.1.

The Role of Ranks and Bases. To study how approximation quality and computation time depend on the tensor-train rank and the basis order, we now vary those crucial parameters of TTA while holding the dimensionality of the problem fixed. We focus on the case with $M = 5$ countries, i.e., a ten-dimensional state vector. We vary the rank $r \in \{2, 3, 4, 5\}$ and the basis order $m \in \{2, 3, 4, 5\}$ independently, and set the sample size to $100\, mr^2d$ to keep the ratio of sample points to parameters approximately constant. Table 1 reports the average error, the 99th-percentile error, and computation time, with runtimes expressed relative to the benchmark specification $(r, m) = (3, 3)$.[8] Accuracy improves markedly as we move along the diagonal from simpler to richer specifications, that is, from $(r, m) = (2, 2)$ to $(5, 5)$. Along this sequence, both the average error and the 99th-percentile error decline steadily by about two orders of

[8] In our benchmark implementation, the simple solution requires 7,727 seconds on 51 Intel Xeon Platinum 8358 CPUs, whereas the sophisticated solution in Section 3.5 completes in 334 seconds. Note, however, that the two numbers are not directly comparable due to differing sample size and simulation length.

Figure 3: Simple Solution Algorithm



Visualization of the simple solution algorithm. Given a sample set of size $N \leq T$ (approximating the ergodic set) and a guess for the policy function, the algorithm iterates on candidate policy values until the equilibrium conditions are satisfied. Expectations are computed with a monomial quadrature rule, and the resulting residuals are used to update the policy guess. Once solutions at all sample points are obtained, the policy function is re-approximated using TTA. This procedure is repeated until the policy function converges. The converged candidate policy is then used to simulate the model for T periods and to update the sample set. The algorithm continues until both the policy function and the simulated sample set have converged.

magnitude in total. The largest improvement occurs when moving from $(2,2)$ to $(3,3)$, while subsequent increases deliver more moderate gains. More generally, increases in basis order m or rank r, holding the other fixed, tend to improve accuracy, although the pattern is not fully monotone throughout the table. Computation time rises with both m and r. Relative time increases from 0.14 at $(2,2)$ to 15.44 at $(5,5)$. The effect of the rank is stronger in comparison to the basis order, as the number of tensor-train parameters, and hence the cost of the underlying linear-algebra operations, grows roughly quadratically in r and only linearly in m.

Scaling with the Simple Solution. In the second exercise, we scale the state-space dimension from $d = 10$ to $d = 26$ and examine how computation time and solution accuracy change. Throughout, we fix the tensor train architecture at basis order $m = 3$ and rank $r = 3$. To keep the problem comparable across dimensions, we keep on setting the sample size to $100\, mr^2 d$. Table 2 reports computation time normalized by the five-country model ($d = 10$), along with the average and 99th-percentile errors. As dimension increases, runtimes grow only moderately — by a factor of 17.5 when going from $d = 10$ to $d = 26$ — and clearly sub-exponentially. At the same time, accuracy is stable and even improves slightly in both the mean and the upper tail.

Table 1: Approximation Error and Computation Time by Basis Order and Rank.

	$m = 2$	$m = 3$	$m = 4$	$m = 5$
$r = 2$	**-3.26 (-2.16)** **t = 0.14**	-4.32 (-3.27) $t = 0.38$	-4.36 (-3.26) $t = 0.77$	-4.29 (-3.20) $t = 1.11$
$r = 3$	-3.22 (-2.11) $t = 0.61$	**-4.59 (-3.49)** **t = 1.00**	-4.63 (-3.56) $t = 2.31$	-4.79 (-3.73) $t = 3.19$
$r = 4$	-3.24 (-2.13) $t = 1.28$	-4.73 (-3.64) $t = 2.69$	**-4.82 (-3.77)** **t = 5.16**	-4.93 (-3.88) $t = 7.69$
$r = 5$	-3.24 (-2.12) $t = 2.53$	-4.81 (-3.72) $t = 6.14$	-4.98 (-3.91) $t = 9.64$	**-5.06 (-4.02)** **t = 15.44**

This table reports the average error and the 99th-percentile error (in brackets) both in $\log_{10}$ units, together with computation times expressed relative to the benchmark specification $(r, m) = (3, 3)$, for different combinations of tensor-train rank $r \in \{2, 3, 4, 5\}$ and basis order $m \in \{2, 3, 4, 5\}$. All results are obtained from the model with $M = 5$ countries, implying a ten-dimensional state vector.

Table 2: Approximation Error and Computation Time by Dimensions.

d	**10**	**14**	**18**	**22**	**26**
Q99	-3.47	-3.55	-3.60	-3.66	-3.69
AVG	-4.56	-4.68	-4.73	-4.82	-4.86
Time	1.00	2.29	4.87	9.44	17.50

This table reports average (AVG) and 99th-percentile (Q99) errors in $\log_{10}$ units. It also reports compute times normalized by the $d = 10$ case. The basis order and tensor-train rank are fixed at $(m, r) = (3, 3)$.

The results for the simple solution are already very promising, although the approach does not make use of several refinements that can substantially reduce computation time and, in some cases, improve accuracy. Most importantly, evaluating expectations entails a trade-off between integration precision and computational effort: the monomial rule used here requires $2(M + 1)$ function evaluations per point yet delivers only modest accuracy. At the same time, the time-iteration procedure relies on Newton-type methods to frequently solve a nonlinear system of size $M + 1$ that becomes increasingly challenging as M grows. In the next subsections, we address both bottlenecks, which allows us to tackle substantially higher-dimensional specifications. Section 3.3 shows how expectations can be computed quasi-analytically, and Section 3.4 shows how time iteration can be replaced by multi-dimensional EGM — both with the help of TTA.

3.3 Quasi-Analytic Expectations with TTA

When solving high-dimensional dynamic stochastic models, not only function approximation but also the computation of expectations poses a formidable challenge. In case of the IRBC model, the $M + 1$ dimensional integral in equation (9) has to be computed. Previous papers solve the high-dimensional IRBC model either by resorting to compute-intense Monte-Carlo integration or by employing monomial rules, which exhibit polynomial growth in the number of dimensions (see Kollmann et al., 2011). Quadratically growing monomial rules achieve decent accuracy at the expense of becoming too compute intensive at medium scale already, while monomial rules that grow only linearly (as used in Section 3.2) lack accuracy.[9] Due to the stark trade-off between speed and accuracy in high-dimensional numerical integration, an approach that allowed for calculating expectations analytically would be very welcome. Such an approach would not only be computationally efficient, it would also achieve greater accuracy by integrating over the entire distribution rather than relying on a finite number of evaluation points. TTA allows for such an approach, as we now show.

Integration across the dimensions of a tensor train can be performed analytically, as detailed in Section 2, by substituting the basis functions of the relevant dimension with their anti-derivative. However, this property alone does not fully resolve the expectations problem, since expectations are not simply integrals over policy functions — as can be seen in equation (9). To address this complication, we proceed in three steps. First, instead of using the natural recursive state of the economy, capital stocks, k_j, and productivities, a_j, of all countries, we use the following slightly different specification for the state:

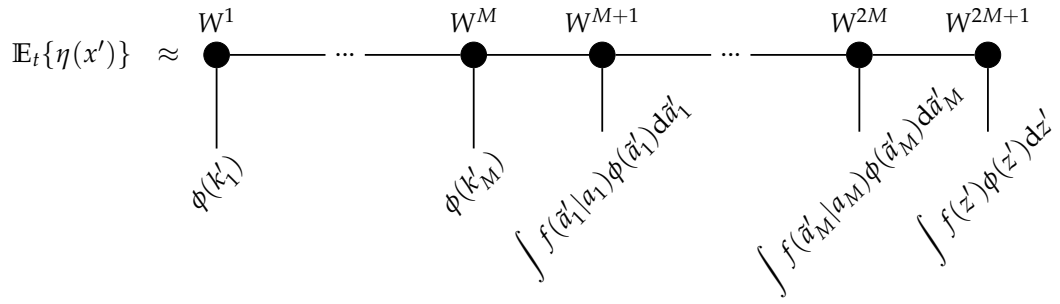
$$x = (k_1, \dots, k_M, \tilde{a}_1, \dots, \tilde{a}_M, z) \in X \subset \mathbb{R}^{2M+1}, \tag{13}$$

where $\tilde{a}_j = \ln a_j - \sigma z$ is the log-productivity of country j before the impact of the global shock, z, is taken into account. We adopt this definition because it makes the shock innovations conditionally independent, thereby allowing us to compute expectations quasi-analytically later on. As a second step, we define the term inside the expectations operator in country j's Euler equation, which we refer to as the expectations term,

$$\eta_j(x') = \lambda' \left[a'_j A \zeta (k'_j)^{\zeta - 1} + 1 - \delta + \frac{\psi}{2} g'_j (g'_j + 2) \right], \tag{14}$$

[9] Employing such a monomial rule in both the time iteration and the error evaluation step, Brumm and Scheidegger (2017) deliberately sidestep the challenge of computing accurate high-dimensional expectations to focus on high-dimensional function approximation.

Figure 4: Expectation Formation with Tensor Trains



The figure shows how the integrals in equation (15) can be evaluated using the tensor train decomposition. Since each integral involves only an isolated dimension of the tensor train, it can be computed directly at the corresponding core by replacing the relevant basis.

so that expectations can be computed as

$$\begin{aligned}\mathbb{E}_t\{\eta_j(x')\} = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(\tilde{a}'_1|\rho \ln a_1, \sigma) \cdot \ldots \cdot f(\tilde{a}'_M|\rho \ln a_M, \sigma) \cdot f(z'|0,1) \cdot \\ \eta_j(k'_1, \ldots, k'_M, \tilde{a}'_1, \ldots, \tilde{a}'_M, z')\, d(z', \tilde{a}'_1, \ldots, \tilde{a}'_M),\end{aligned} \tag{15}$$

where $f(\cdot|\mu, \varsigma)$ denotes the probability density function of a normal distribution with mean μ and standard deviation ς. In the third step, we perform a change of basis for the individual productivities $\tilde{a}'_1, \ldots, \tilde{a}'_M$ and the global shock z'. Specifically, we replace the polynomial basis functions $\phi(\vartheta)$ with a precomputed "expectations basis", given by the integral of the product of the density function and the polynomial basis functions, $\int f(\vartheta|\mu, \sigma)\phi(\vartheta)\mathrm{d}\vartheta$. [10] This transformation allows us to directly incorporate the probability distribution into the tensor train structure, streamlining the computation of expectations. For better intuition, Figure 4 illustrates this process using the graphical notation introduced in Section 2. Note that this quasi-analytic integration approach is only possible when integrating over random variables that are (conditionally) independent, which is the reason why we choose the state x as specified in equation (13) above.

3.4 Endogenous Grids in High Dimensions with TTA

There are two standard routes to enforcing the optimality conditions (9)–(10). Brumm and Scheidegger (2017) solve the resulting system with a numerical root finder and iterate on the policy function (time iteration)—the exact same approach used in Section 3.2. With TTA, we can take a more efficient path: a variant of the endogenous grid

[10] The expectations basis for z, which is normally distributed with mean zero, is straightforward. For $\tilde{a}'_j$, which is normally distributed with state-dependent mean $\rho \ln a_j$, we precompute the expectations basis on a fine grid over this mean and use (one-dimensional!) interpolation at the relevant a_j.

method of Carroll (2006).[11] In the IRBC model EGM replaces a joint $M+1$-dimensional numerical solve with a one-dimensional solve. Since this approach produces irregular (endogenous) grids it is not applicable for grid-based methods; TTA, however, is agnostic to the underlying sample, which makes it the natural companion for EGM.

Applied to the IRBC model, EGM works as follows. Fix a candidate policy η. For each exogenous state (z, a) fix the choice k', and compute the right-hand side of the Euler equation (9). With η given, the expectation is obtained quasi-analytically, according to the previous section. Given a solution candidate for λ we can rearrange the Euler equation (9) to obtain the endogenous state k analytically. Plug in the endogenous state in the aggregate resource constraint (10) and update the solution candidate λ until it solves (10). The procedure can be easily extended to accommodate the irreversibility constraint of the non-smooth model. Details can be found in Appendix B.2.

3.5 Sophisticated Solution Approach

We now present an algorithm that fully leverages TTA, combining quasi-analytical expectations with EGM to compute the recursive equilibrium of the IRBC model. The recursive state of the economy $x \in X$ consists of the country-specific capital stocks k_j, transformed productivities $\tilde{a}_j$, and the global shock z (see equation (13)). The policy we solve for consists of the expectation-terms η_j defined in equation (14):

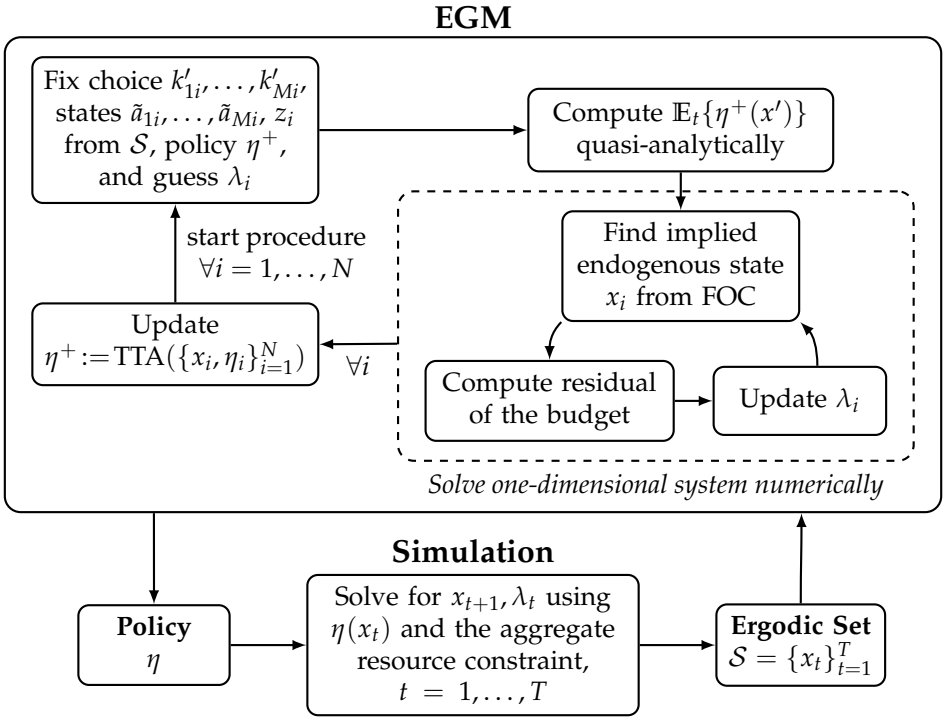
$$p : X \to \mathbb{R}^M, p(x) = (\eta_1(x), \ldots, \eta_M(x)) . \tag{16}$$

The expectations terms are functions of the other policy choices. To employ the quasi-analytical expectations approach explained above, it is essential to directly approximate these terms via TTA. An additional advantage is that η_j is typically smoother than the underlying choice variables, which makes it easier to approximate accurately. The mapping p determines only M policy components, while the equilibrium system involves $M+1$ unknowns. Hence, given $p(x)$, simulation requires solving for λ to enforce the aggregate resource constraint.[12] A recursive equilibrium is characterized by a policy function p and an ergodic set $\mathcal{E} \subset X$ such that (i) the equilibrium conditions hold on $\mathcal{E}$, and (ii) simulating the economy under p generates $\mathcal{E}$. As in Section 3.2, our algorithm proceeds in two nested loops. On the inner loop, we compute a policy that approximately satisfies the optimality conditions (9) and (10) on a finite set of sampled states $\mathcal{S}$, now using EGM. On the outer loop, we simulate the model under the current

[11] A series of papers extends the basic EGM approach to richer settings; see, for example, White (2015), Druedahl and Jørgensen (2017), and Bayer and Luetticke (2020).

[12] While this may seem unusual, it is central to the efficiency of the approach: we approximate a comparatively simple and smooth object (latent low rank), and recover the remaining choice via a one-dimensional market-clearing condition.

Figure 5: Sophisticated Solution Algorithm



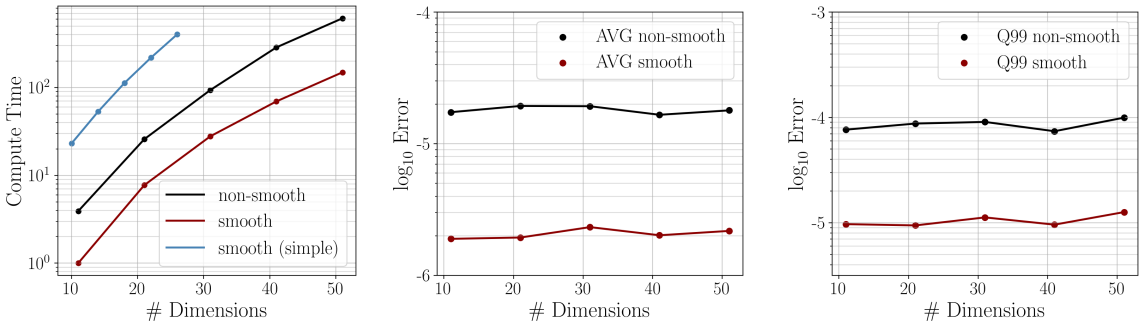
Schematic overview of the sophisticated solution algorithm. Starting from an initial TTA of the policy function η^+ and an initial training set of size $N \leq T$, the EGM step fixes choice variables and exogenous states, computes expectations quasi-analytically under the current policy guess, and solves for the implied endogenous state x by iterating on the multiplier λ until the resource constraint holds. The resulting policy is then re-approximated using TTA, and this procedure is repeated until η^+ converges. The candidate policy is subsequently used in simulation, where λ_t is again solved numerically to satisfy the resource constraint and to recover k_{t+1}. The T simulated observations are used to update the finite approximation of the ergodic set $\mathcal{S}$, and the full procedure is iterated until both the policy function and the sample set have converged.

policy and update $\mathcal{S}$ accordingly, iterating until $\mathcal{S}$ provides a close approximation to the ergodic set.

Figure 5 illustrates this algorithm and thereby highlights the differences to the simple algorithm displayed in Figure 3. A detailed description of the algorithm can be found in Appendix B.3. To assess accuracy, we adapt the Euler-equation error measure from Section 3.2 so that it is compatible with the η-policies. Details can be found in Appendix B.1.

A key advantage of our approach is that the expectations in the Euler equation are computed quasi-analytically. In contrast, approaches based on numerical quadrature—such as the monomial integration method used by Brumm and Scheidegger (2017) or in Section 3.2—introduce an additional source of approximation error due to the quadrature itself. By contrast, our method eliminates this source of error entirely, as expectations of the approximate policy function are evaluated quasi-analytically.

Figure 6: IRBC Model Scaling and Errors



The left panel reports computation time (normalized to the 11-d smooth model, which takes 334 seconds). The increase in computation time is modest: raising the dimensionality from 11 to 51 (5 to 25 countries) multiplies the time by two orders of magnitude. The middle panel displays the average error, which remains well below 0.001% and 0.01% in the smooth and the non-smooth model, respectively. The right panel displays the 99th percentile error, which is around one order of magnitude higher than the average error across both models and all dimensions. Accuracy remains approximately the same with increasing dimensions.

3.6 Accuracy and Scalability of TTA

We now scale the number of countries $M = 5, \ldots, 25$ in the smooth and non-smooth IRBC model, which implies dimensions $d = 2M + 1$, ranging from 11 to 51. Across all dimensions, we keep bases and ranks fixed, at $m = 3$ and $r = 3$ in the smooth model, and at $m = 5$ and $r = 3$ in the non-smooth model.[13] Note that, in the non-smooth model, we use a piecewise linear basis, as it handles kinks more robustly than polynomial bases. The sample size $N = |\mathcal{S}|$ is set to $50mr^2d$ such that the ratio of sample points to tensor train parameters remains approximately at 50. The left panel of Figure 6 reports normalized compute time as a function of the dimension, while the middle and right panel report average and 99th-percentile errors. The results demonstrate that compute time increases modestly and clearly sub-exponentially in the dimensionality of the problem.[14] Figure 6 shows that the normalized scaling of our method clearly outperforms the adaptive sparse-grid approach in Brumm and Scheidegger (2017): Increasing the dimensionality from $d = 11$ to $d = 51$ raises normalized computation time by only about two orders of magnitude, whereas sparse grids require more than four orders of magnitude for the same increase in dimension, despite producing less accurate results.[15] Computing the non-smooth model is not fundamentally more difficult than the smooth model, when accounting for the richer basis used to solve the non-

[13] In this section, we do not provide a systematic comparison across tensor-train ranks and basis orders. The reason is that the baseline specification already delivers very small errors, so increasing the number of degrees of freedom yields only marginal additional accuracy gains.

[14] Scaling is, however, more than linear. While evaluating the TTA at one state costs $\mathcal{O}(mr^2d)$, holding the sample-to-parameter ratio fixed requires the sample size to grow linearly in d. Moreover, each additional country adds a policy function that needs to be approximated separately.

[15] See Figure 15 in Brumm and Scheidegger (2017).

smooth model.[16] The average Euler error remains well below 0.001% (0.01%) in the smooth (non-smooth) model. In each case, the 99th-percentile error is approximately one order of magnitude higher than the average. The average error in the smooth model is roughly two orders of magnitude better than the simulation error reported in Figure 8 of Brumm and Scheidegger (2017) — despite being orders of magnitude faster to compute.

Relative to the simple approach in Section 3.2, the sophisticated solution method is more than one order of magnitude faster while achieving errors that are roughly one order of magnitude smaller — highlighting the effectiveness of the two extensions.

4 TTA for Heterogeneous Agents in Continuous Time

This section develops a tensor-train approach for solving heterogeneous agent models in continuous time. Section 4.1 presents the model and states the master equation characterizing equilibrium. Section 4.2 introduces an a posteriori model-reduction strategy for generating a discrete representation of the wealth distribution. Section 4.3 leverages that representation and the tensor train approach presented in Section 2 to build an algorithm for solving the master equation. Section 4.4 provides remaining algorithmic choices independent of the TTA approach. Section 4.5 defines an appropriate error metric and evaluates the method's performance, thereby demonstrating that the non-linearities induced by the stochastic wealth tax require a multi-dimensional representation of the wealth distribution to achieve accurate solutions.

4.1 Model with Stochastic Wealth Taxation

In this section, we consider a heterogeneous-agent model in the spirit of Krusell and Smith (1998), yet in continuous time and with stochastic wealth taxation.

Households. The economy is populated by a continuum of infinitely lived heterogeneous households differing in idiosyncratic labor productivity $\varepsilon_t \in \{\varepsilon_1, \varepsilon_2\}$, discount rate $\rho_t \in \{\rho_1, \rho_2\}$ and endogenous asset holdings a_t. The idiosyncratic productivity state switches from j to the other state $\jmath$ with Poisson rate $\lambda_j > 0$, the idiosyncratic discounting state switches from i to $\imath$ with Poisson rate $\omega_i > 0$. Each household solves

$$\mathbb{E}_0 \int_0^\infty e^{-\rho_t t}\, u(c_t)\, dt, \quad \text{with} \quad u(c_t) = \frac{c_t^{1-\gamma} - 1}{1-\gamma}, \tag{17}$$

[16] Note that in the non-smooth model the irreversibility constraint actually binds in simulations — the unconditional probability that a country's constraint is binding lies between 2 and 3 percent depending on the number of countries.

subject to

$$\dot{a}_t = w_t \varepsilon_t + r_t a_t - c_t, \quad a_t \geq \underline{a}, \tag{18}$$

with a borrowing limit $\underline{a}$.

Production. A representative firm operates a Cobb-Douglas technology with aggregate capital K_t and inelastically supplied labor $N_t \equiv 1$, so that output, the wage, and the interest rate are given by:

$$Y_t = \exp(z_t) K_t^{\alpha}, \quad w_t = (1-\alpha)\exp(z_t) K_t^{\alpha}, \quad r_t = \alpha \exp(z_t) K_t^{\alpha-1} - \delta.$$

The (log) total factor productivity z_t follows an Ornstein-Uhlenbeck process

$$dz_t = \kappa(\bar{z} - z_t)dt + \sigma_z dB_t, \tag{19}$$

with mean reversion $\kappa > 0$, long-run mean $\bar{z}$, and diffusion σ_z. Aggregate capital equals the first moment of the cross-sectional wealth distribution,

$$K_t = \int_{\underline{a}}^{\infty} a \sum_{i=1}^{2} \sum_{j=1}^{2} g_{ij}(a,t) da, \tag{20}$$

where $g_{ij}(a,t)$ denotes the density of agents with assets a, discounting type i, and productivity type j.

Wealth Tax. With Poisson intensity θ, a stochastic wealth-tax is levied. At such an event, the government collects a fraction $\tau \in [0,1)$ of each agent's assets and distributes the proceeds equally across agents, yielding post-tax asset holdings $\tilde{a}$ from pre-tax asset holdings a satisfying:

$$\tilde{a} = (1-\tau)a + \tau K_t. \tag{21}$$

HJB Equation. The value function for type ij, $V^{ij}(a,t)$, satisfies the HJB equation:

$$\begin{aligned}\rho_i V^{ij}(a,t) = \max_c \Big\{ u(c) + \partial_a V^{ij} \left[w_t \varepsilon_j + r_t a - c \right] \Big\} \\ + \lambda_j \left[V^{ij} - V^{ij} \right] + \omega_i \left[V^{tj} - V^{ij} \right] + \frac{1}{dt} \mathbb{E}_t \{ dV^{ij} \}\end{aligned}$$

The first-order condition implies

$$c_{ij}(a,t) = u'^{-1}\left(\partial_a V^{ij}(a,t) \right), \quad s_{ij}(a,t) = w_t \varepsilon_j + r_t a - c_{ij}(a,t), \tag{22}$$

where $s_{ij}(a,t)$ denotes the optimal savings function (drift of assets) for type ij.

Kolmogorov Forward Equation. Given optimal savings $s_{ij}(a,t)$, the law of motion for the density function of type ij is given by the Kolmogorov forward equation (KFE):

$$\begin{aligned}\partial_t g_{ij}(a,t) = &-\partial_a[s_{ij}(a,t)g_{ij}(a,t)] - \lambda_j g_{ij}(a,t) + \lambda_j g_{ij}(a,t) - \omega_i g_{ij}(a,t) + \omega_\iota g_{\iota j}(a,t) \\ &+ \left[\frac{1}{1-\tau} g_{ij}\left(\frac{a-\tau K_t}{1-\tau}, t\right) - g_{ij}(a,t)\right] dN_t,\end{aligned}$$

i.e., transport by savings, switching between idiosyncratic states, and a jump (push-forward) operator for the tax where N_t is the Poisson tax process. [17] The borrowing constraint is imposed as a state-constraint boundary at $a = \underline{a}$, which ensures nonnegative drift at the boundary, $s_j(\underline{a}) \geq 0$, and hence zero outward flux. For compactness, we write $\partial_t g = (\mathcal{A}^* g)$.

Distributional Approximation. To make the infinite-dimensional state tractable, we summarize the wealth distribution by M moments $\Gamma_t = (m_t^1, \ldots, m_t^M)$ with

$$m_t^k = \int_{\underline{a}}^{\infty} \sum_{i=1}^{2} \sum_{j=1}^{2} f_{ij}^k(a) g_{ij}(a,t) da \qquad k = 1, \ldots, M, \tag{23}$$

for a chosen set of functions f_{ij}^k. For example, taking $f_{ij}^1(a) = a$, for $i = 1,2$ and $j = 1,2$ yields the capital stock.

Recursive Equilibrium. Turning from sequential to recursive characterization of the model, policy and value functions now depend on the distribution over endogenous assets and exogenous (productivity and discounting) types — or a finite dimensional representation of it, as just introduced. More precisely, a recursive competitive equilibrium consists of functions $\{V^{ij}, g^{ij}, c^{ij}, s^{ij}\}(a, z, \Gamma)$ for $i = 1,2$, $j = 1,2$ and aggregates $\{r, w, Y, K, C\}(z, \Gamma)$ such that (i) households optimize given rational expectations, (ii) factor prices are competitive, (iii) aggregates are consistent with distributions[18] , and (iv) the asset distribution evolves according to the Kolmogorov forward equation given policies.

[17] The term consists of an outflow $-g_j(a,t)$, and an inflow from the pre-image of the post-tax mapping. The additional term $\frac{1}{1-\tau}$ comes from a mass conservation consideration.

[18] In this application, market clearing is straightforward because factor prices are pinned down by aggregate capital. Yang et al. (2025) argue that classic solution methods struggle when market clearing is non-trivial. Although we abstract from such cases, they could be incorporated naturally into our framework by treating prices as additional policy arguments and solving for (or interpolating) equilibrium prices in simulations, as in Yang et al. (2025). Efficiency would be preserved because only the price dimension would need to be solved for, while the remaining dimensions would be evaluated only once.

Table 3: Calibration of the Continuous-Time Heterogeneous-Agent Economy.

Parameter	Symbol	Value
Discount rate	$[\rho_1, \rho_2]$	$[0.0, 0.09]$
Risk aversion	γ	2.0
Capital share	α	0.36
Depreciation	δ	0.02
Borrowing limit	$\underline{a}$	0.0
Labor productivity	$[\underline{\varepsilon}, \bar{\varepsilon}]$	[0.15, 1.096]
Productivity switching rates	$[\lambda_1, \lambda_2]$	[0.4, 0.045]
Share of high discounting households	d	0.5
Discounting redraw	q	0.05
Discounting switching rates	$[\omega_1, \omega_2]$	$[(1-d)q, dq]$
TFP long-run mean	$\bar{z}$	0.0
TFP diffusion (OU)	σ_z	0.007
OU mean reversion	κ	0.05

Master Equation. The recursive-equilibrium value function satisfies the following master equation, which can be derived by combining the HJB and Kolmogorov forward equation:

$$\begin{aligned}\rho_i V^{ij}(a,z,\Gamma) = \max_c \Big\{ u(c) + \partial_a V^{ij}\left[w(z,\Gamma)\varepsilon_j + r(z,\Gamma)a - c\right]\Big\} \\ + \lambda_j\big(V^{ij} - V^{ij}\big) + \omega_i\left(V^{\imath j} - V^{ij}\right) \\ + \kappa(\bar{z}-z)\,\partial_z V^{ij} + \tfrac{1}{2}\sigma_z^2\,\partial_{zz}V^{ij} \\ + \left\langle \mu_\Gamma(z,\Gamma), \nabla_\Gamma V^{ij}\right\rangle + \theta\big(V^{ij}(\tilde{a},z,\tilde{\Gamma}) - V^{ij}(a,z,\Gamma)\big),\end{aligned}$$

where $\mu_\Gamma(z,\Gamma) := \dot{m}$ is the drift of the chosen moment vector induced by the KFE and $\tilde{\Gamma}$ denotes the post-tax moments implied by $\tilde{a}$. At the borrowing limit $a = \underline{a}$, the drift of assets must be non-negative implying the following boundary condition

$$\partial_a V^{ij}(\underline{a},\cdot) \geq u'\left(w(z,\Gamma)\varepsilon_j + r(z,\Gamma)\underline{a}\right). \tag{24}$$

Calibration. We follow Krusell and Smith (1998) for the idiosyncratic labor-productivity process, aggregate production technology, and risk aversion. In addition, we introduce two discount-factor types, which increases mass at the borrowing constraint and in the upper tail of the wealth distribution. The discount factors ρ_1 and ρ_2 are set to the values in Auclert et al. (2025). We assume on average half of households has patience ρ_1 or ρ_2, respectively, and with a probability of 5% per quarter patience is redrawn. The

continuous TFP process is calibrated to the empirical estimates in Christensen et al. (2024). Parameter values are reported in Table 3.

4.2 A Posteriori Model Reduction for Distributional Dynamics

We now propose a simple simulation-based method to identify moments that provide a low-dimensional representation of the wealth distribution. Let $g(t) \in \mathbb{R}^N$ denote a finite-dimensional approximation of the wealth distribution. In this paper, we use histogram bins for this purpose.[19] The objective is to find a small number of directions that capture the dominant variation in the simulated distributional dynamics.

We observe $g(t)$ at equally spaced times $t_1, \ldots, t_T$ with step size Δt, and collect the resulting snapshots in the matrices[20]

$$X_- \;=\; \big[g(t_1), \ldots, g(t_{T-1})\big] \in \mathbb{R}^{N\times(T-1)}, \qquad X_+ \;=\; \big[g(t_2), \ldots, g(t_T)\big] \in \mathbb{R}^{N\times(T-1)}.$$

The dimensionality-reduction step follows the logic of proper orthogonal decomposition (POD), or equivalently principal components analysis applied to the simulated distributional snapshots. We compute a rank-k truncated singular value decomposition of X_-,

$$X_- \;\approx\; U\,\Sigma\,V^\top, \qquad U \in \mathbb{R}^{N\times k}, \quad \Sigma \in \mathbb{R}^{k\times k}, \quad V \in \mathbb{R}^{(T-1)\times k},$$

where $\Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_k)$ contains the k largest singular values. The columns of U define the POD basis: they span the low-dimensional subspace that captures the dominant variation in the simulated wealth distributions. The rows of $V^\top$ describe the corresponding time paths.[21]

Projecting the distribution onto the basis yields the low-dimensional moment coordinates

$$m(t) \;=\; U^\top g(t) \in \mathbb{R}^k.$$

These moments are the variables we carry into the global solution algorithm to represent the distribution parsimoniously. To obtain an approximate law of motion for the

[19] More generally, this approach may be applicable to a wider class of state variables that evolve according to the linear law of motion discussed in this section, as well as the nonlinear law of motion considered in Appendix C.1.

[20] We obtain $g(t)$ by simulating an approximate solution, for example through histogram evolution.

[21] In principle, one could instead follow Ahn et al. (2018) and Reiter (2023) by choosing directions that are informative for other objects of interest, such as aggregates, prices, or the value function, rather than for the distributional snapshots themselves. This amounts to applying the SVD not to the snapshot matrix itself, but to an output-relevant object, such as a whitened cross-covariance or observability matrix that identifies the distributional moments most relevant for the targeted objects.

moments, we assume that the distributional dynamics are locally well approximated by a linear transition. In the original high-dimensional space, this corresponds to

$$g(t_{i+1}) \;\approx\; A g(t_i),$$

for some linear operator $A \in \mathbb{R}^{N\times N}$. Rather than estimating A directly, we estimate the induced transition in the moment coordinates,

$$m(t_{i+1}) \;\approx\; A_k\, m(t_i),$$

where the reduced transition matrix is given by

$$A_k \;=\; U^\top X_+ \, V \, \Sigma^{-1} \;\in\; \mathbb{R}^{k\times k}.$$

Thus, the basis construction is POD-based, while the reduced law of motion is estimated by least squares in the moment coordinates. [22]

As a heuristic for choosing how many moments to retain, we use the energy share of each singular value,

$$E_i = \frac{\sigma_i^2}{\sum_j \sigma_j^2}, \tag{25}$$

which measures the fraction of snapshot variation explained by the i-th mode. We retain enough modes to capture the relevant variation in the simulated distributional dynamics while keeping the state representation low-dimensional.

As a final remark, we emphasize that the approach is not limited to linear dynamics. In Appendix C.1, we discuss how related projection-based ideas can be combined with Koopman-theoretic approximations to represent nonlinear distributional dynamics.

4.3 Solving the Master Equation with Tensor Trains

We solve the master equation using value function iteration (VFI). In each VFI step, we recover optimal consumption c^* from the first-order condition, taking the current candidate value function V as given. Conditional on c^*, we update V by approximately solving the master equation with a least-squares projection method. The solver operates directly on tensor-train (TT) coefficients, which allows the high-dimensional update to be carried out efficiently (see Section 2).

[22] We make two practical adjustments. First, since the distributional dynamics also depend on productivity z_t, we include productivity in the forecasting rule. Second, due to its importance for factor prices, we use capital as the first moment. To ensure that the remaining moments are orthogonal to capital, the matrix X_- is projected onto the subspace orthogonal to the capital weight vector before performing the SVD.

A generic master-equation problem has two components: (i) an interior equation that must hold on the state space, and (ii) boundary conditions. We compute an approximate solution by casting the problem as constrained least squares, in the spirit of Section 2. Specifically, we (a) minimize the squared residual of the interior equation evaluated at a set of interior points, and (b) enforce the boundary conditions at a separate set of boundary points.

The boundary conditions typically take the form of inequality constraints. For computational convenience, we convert each inequality constraint into an equality constraint by introducing a nonnegative slack variable at each boundary point. We then solve the resulting equality-constrained problem using an augmented Lagrangian approach, which incorporates the constraints into the objective via multipliers and quadratic penalties. Given this structure, the augmented-Lagrangian subproblems admit closed-form updates for both the value-function coefficients and the slack variables. We obtain a solution by iterating over updates of (i) the value-function coefficients, (ii) the slack variables, and (iii) the Lagrange multipliers — that is, by applying the alternating direction method of multipliers (ADMM). One run of ADMM corresponds to one VFI update. Updating the value-function coefficients amounts to a quadratic minimization problem. In TT form, this step can be solved efficiently using alternating least squares (ALS), which preserves the low-rank structure throughout.

This section proceeds in four steps. First, we introduce a generic master-equation problem and its constrained least-squares formulation. Second, we derive the corresponding augmented-Lagrangian representation. Third, we map our application from the previous section into this general framework. For clarity, we present the full-tensor formulation in the main text; the local subproblems underlying the tensor-train implementation are deferred to Appendix C.2. Finally, we describe the ADMM iterations that implement one VFI update.

Generic Master Equation. Let $\mathcal{D}$ be the state domain with interior $\mathrm{int}(\mathcal{D})$ and boundary $\partial\mathcal{D}$. Consider master equations that act linearly on V in the interior and on the boundary. Write $\mathcal{M}$ for the interior operator and $\mathcal{C}$ for the boundary operator. We seek a TT approximation $V(\cdot, \mathcal{T})$ with cores $\mathcal{T} = \{W^1, \ldots, W^d\}$ satisfying

$$\mathcal{M}[V(\cdot;\mathcal{T})](x) = u(x), \quad \forall x \in \mathrm{int}(\mathcal{D}), \quad \mathcal{C}[V(\cdot;\mathcal{T})](x) \geq h(x), \quad \forall x \in \partial\mathcal{D}.$$

Global Minimization Problem. Given interior samples $\{x_i\}_{i=1}^N$ and boundary samples $\{\bar{x}_\ell\}_{\ell=1}^L$, we minimize interior residuals with regularization for numerical stability and impose boundary inequalities

$$\begin{aligned}\min_{W_1,\dots,W_d} \quad & \frac{1}{2}\sum_{i=1}^{N} ||\mathcal{M}[V(\cdot;\mathcal{T})](x_i) - u(x_i)||_2^2 + \frac{\eta}{2}\sum_{j=1}^{d} ||W_j||_F^2 \\ \text{s.t.} \quad & \mathcal{C}[V(\cdot;\mathcal{T})](\bar{x}_\ell) \geq h(\bar{x}_\ell), \qquad \ell = 1,\dots,L.\end{aligned}$$

Augmented Lagrangian. To cast the constraint minimization above into an unconstrained convex problem, we proceed in two steps: First, we introduce slack variables $t_\ell \geq 0$ at boundary samples $\{\bar{x}_\ell\}_{\ell=1}^L$. Second, we incorporate the constraints into the objective via quadratic penalties and multipliers μ_ℓ — an approach known as augmented Lagrangian:

$$\begin{aligned}\mathcal{L}(\mathcal{T}, t, \mu) =& \frac{1}{2}\sum_{i=1}^{N} ||\mathcal{M}[V(\cdot;\mathcal{T})](x_i) - u(x_i)||_2^2 + \frac{\eta}{2}\sum_{j=1}^{d} ||W_j||_F^2 \\ &+ \frac{\gamma}{2}\sum_{\ell=1}^{L} \left(\mathcal{C}[V(\cdot;\mathcal{T})](\bar{x}_\ell) - h(\bar{x}_\ell) - t_\ell\right)^2 \\ &+ \sum_{\ell=1}^{L} \mu_\ell \left(\mathcal{C}[V(\cdot;\mathcal{T})](\bar{x}_\ell) - h(\bar{x}_\ell) - t_\ell\right), \quad t_\ell \geq 0.\end{aligned}$$

Full Tensor Formulation. Given multipliers μ and slack variables t write the minimization problem over $\mathcal{T}$, as problem over the full tensor of value function coefficients $v = \mathrm{vec}(A) \in \mathbb{R}^K$ with $K = \prod_{j=1}^d m_j$ as[23]

$$\min_v \frac{1}{2}||Mv - u||_2^2 + \frac{\gamma}{2}||Cv - h - t||_2^2 + \mu^\top (Cv - h - t). \tag{26}$$

This problem falls into the class of quadratic minimization problems that can be solved efficiently using ALS. In Appendix C we show how to solve the minimization problem in TTA form, for ease of exposition we stick with the full tensor problem here. The component matrices M, C and the vectors u, h take the following form in the application outlined above:

$$\begin{aligned}M_{ij} = &\left(\rho_i + \frac{1}{\Delta} + \lambda_j + \omega_i + \theta\right) X^{V,ij} - \left(w(z,\Gamma)\varepsilon_j + r(z,\Gamma)a - c_{ij}\right) X^{\partial_a,ij} \\ &- \kappa(\bar{z} - z) X^{\partial_z,ij} - \frac{1}{2}\sigma_z^2 X^{\partial_z^2,ij} - \mu(z,\Gamma) X^{\partial_\Gamma,ij} - \lambda_j X^{V,ij} - \omega_i X^{V,ij} - \theta X^{\text{tax},ij},\end{aligned}$$

[23] With $M \in \mathbb{R}^{N\times K}$, $u \in \mathbb{R}^N$, $C \in \mathbb{R}^{L\times K}$, $h \in \mathbb{R}^L$, $\mu \in \mathbb{R}^L$, $t \in \mathbb{R}^L$. In practice we also add a small regularization term to each core W^k for numerical stability.

$$u_{ij} = u(c^{ij}) + \frac{1}{\Delta} V^{ij,\text{old}}, \quad C_{ij} = X^{\partial_a, ij}, \quad h_{ij} = u'(w(z,\Gamma)\varepsilon_j + r(z,\Gamma)\underline{a}).$$

where $X^{\square,ij}$ is the full matrix of basis entries of the value function, or its derivatives, and Δ is the VFI time step.

ADMM. We obtain a solution of the minimization problem by iterating over updates of the value-function coefficients, the slack variables and the multipliers, of the augmented Lagrangian — this approach is known as alternating directions method of multipliers (ADMM). One iteration consists of minimizing the augmented Lagrangian with respect to v

$$\mathcal{T}^{n+1} := \arg\min_{v} \frac{1}{2}||Mv - u||_2^2 + \frac{\gamma}{2}||Cv - h - t^n||_2^2 + \mu^{n\top}(Cv - h - t^n),$$

which admits a closed form least squares solution. Followed by updating the slack variables, by minimizing the augmented Lagrangian with respect to t, taking $\mathcal{T}$ and μ as given,

$$t^{n+1} = \max\left\{\mathcal{C}[V(\cdot;\mathcal{T}^{n+1})](\bar{x}) - h(\bar{x}) + \frac{\mu^n}{\gamma}, 0\right\}, \tag{27}$$

also with a closed form solution. Lastly, in the dual ascent step we update the multiplier, as is standard in ADMM

$$\mu^{n+1} = \mu^n + \gamma\left(\mathcal{C}[V(\cdot;\mathcal{T}^{n+1})](\bar{x}) - h(\bar{x}) - t^{n+1}\right). \tag{28}$$

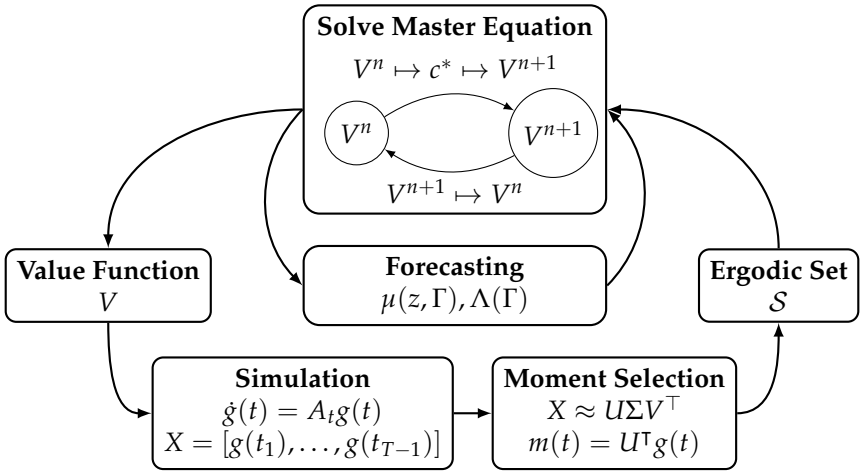
Once the ADMM algorithm converged, the value function iteration step is complete. We adapt the penalty γ to balance primal and dual progress.[24] After completing the VFI step, previous slacks, multipliers and penalties can be used to warm start subsequent iterations.

4.4 Overall Solution Algorithm

The solution algorithm consists of three nested routines aimed at constructing an approximation to the ergodic set, a set of informative moments of the distribution, forecasting rules, and the household value function. In the outermost loop, we approximate the ergodic set, that is, a collection of histogram snapshots of the cross-sectional distribution paired with realizations of the exogenous shocks. In the same loop, we also define a set of moments, represented by weight vectors that map each histogram snapshot into a scalar statistic. Given the approximation of the ergodic set, which also

[24] We choose the penalty parameter γ so that the primal residual, measuring constraint violation, and the dual residual, measuring variation in the implied dual variables across iterations, are of comparable magnitude.

Figure 7: Solution Algorithm for Heterogeneous Agent Model



The algorithm has three nested loops. The outer loop builds a training set of histogram snapshots of the cross-sectional distribution paired with exogenous shock realizations, and specifies a set of moments as weight vectors that map each histogram to scalar statistics. Given a training set and chosen moments, the middle loop estimates forecasting rules for the evolution of the aggregate state (the moments). In the inner loop, taking these moments and forecasting rules as given, the household problem is solved via value function iteration with TTA.

serves as a training set, and a choice of moments, solving the household problem requires forecasting rules for the evolution of the aggregate state summarized by these moments. These forecasting rules—one for the infinitesimal drift of the moments and one for their response to a tax event—are determined in the second loop. In the innermost loop, given the moments and forecasting rules, we solve the household problem by value function iteration (VFI). This step is taken as given, since it was described in the previous section. Figure 7 provides a graphical overview of the three nested procedures.

Training Set and Moments. We approximate the ergodic set by simulating the distribution forward under a sequence of exogenous shocks together with a candidate solution to the household value function V.[25] These simulated data from the training set $\mathcal{S}$ are used for moment selection, forecasting, and the solution of the household problem. First, we select moments $m(t)$ using the a posteriori model reduction procedure described above. Second, we map the simulated distribution snapshots into moment data (using $U^\intercal$), since these moments enter both the forecasting problem and the household problem. We also compute the time derivatives of the distribution $\dot{g}(t)$ and of the corresponding moments $\dot{m}(t)$.

[25]To construct an initial simulation set, we first solve the household problem under an ad hoc forecasting rule and simulate the model under this rough approximation. We keep the (long) sequence of exogenous shocks fixed across iterations, yet choose different sequences for error evaluation.

Forecasting Rules. To solve the household problem, agents must form beliefs about the evolution of the distribution. These beliefs comprise both the infinitesimal changes in moments and the effects of tax events on those moments. First, we estimate an approximate infinitesimal law of motion $\mu(z, \Gamma)$ by regressing the time derivatives of moments on the current moments and exogenous states. Second, we estimate a mapping $\Lambda(\Gamma)$ from the current moments to the moments immediately after a tax shock.

Overall Algorithm. Given these components, the algorithm proceeds as follows. On the innermost level, we solve the household problem using VFI and ADMM. After the value function has converged, we update $\dot{g}(t)$ and $\dot{m}(t)$ implied by the new household solution, and then revise the forecasting rule $\mu(z, \Gamma)$ accordingly. Once the value function and forecasting rules are jointly consistent, we simulate the model forward. Based on the resulting simulated sample, we (re-)select moments via a posteriori model reduction and update the training set. We iterate until the training set, the selected moments, the forecasting rules, and the household value function converge jointly. Algorithm 5 in Appendix C summarizes the full procedure.

4.5 Accuracy and Economic Effects under Different Tax Regimes

In this section, we show that the heterogeneous-agent model with large wealth-tax shocks requires a genuinely multidimensional representation of the wealth distribution, and that the tensor-train approximation (TTA) enables us to solve the resulting high-dimensional PDE accurately. We begin by introducing our error metric, then describe the setup of the exercise and go on to briefly discuss the economic impact of the different tax regimes. Finally, we carefully demonstrate the accuracy of our algorithm and show that the error decreases markedly as we enrich the state representation with additional moments.

Error Measure. We assess accuracy using a relative master equation error expressed in consumption units. Importantly, this measure combines the error in individual consumption decisions with the error in forecasting into a single metric. By contrast, much of the heterogeneous-agent literature evaluates these errors separately, giving rise to both practical and theoretical limitations: the resulting measures are hard to interpret, not easily comparable across models, and do not provide a clean decomposition of the overall error. Such a separation may even understate the true approximation error. Appendix C.3 provides details of the construction of our error measure.

Setup. To demonstrate that TTA can accurately solve multidimensional heterogeneous agent models, we solve the economy under three tax regimes: 0% (no tax), 10%,

and 20% wealth tax, each arriving with a quarterly probability of 2.5%. For each regime we vary the number of aggregate moments, M, used to summarize the cross-section. We approximate the value function (for each discrete idiosyncratic state) with TTA over the continuous state vector $(a, z, m_1, \ldots, m_M)$.[26] For assets, a, we use a polynomial basis of order 30, for each additional dimension we use a polynomial basis of order 3, while setting all TT ranks equal to 3.

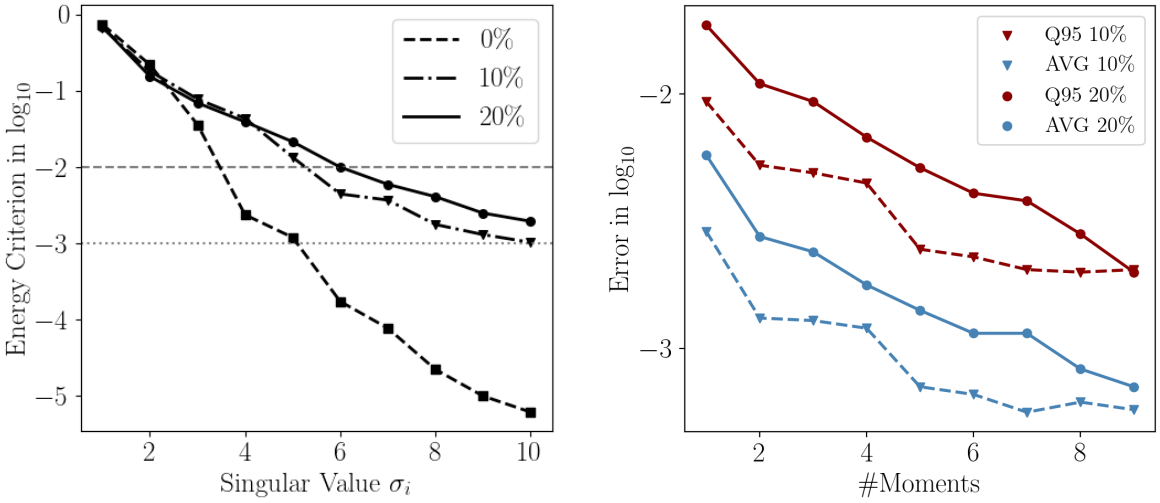
Economic Results. Relative to the zero-tax economy, the average capital stock in the 10% (20%) wealth-tax economy declines by 7.6% (14.3%), while the standard deviation of capital increases by 6.0% (18.4%). The response of aggregate consumption is qualitatively the same, yet less pronounced: average aggregate consumption falls by 1.8% (3.6%), and its standard deviation rises by 6.0% (15.7%). While the redistributive tax lowers aggregate consumption and increases its volatility, it also naturally mitigates inequality. The average wealth Gini coefficient falls by 2.0% (5.7%), while the consumption Gini declines more strongly, by 5.5% (10.9%). From an ex-ante welfare perspective, the model therefore features a trade-off between the level and volatility of aggregate outcomes, which are more favorable in the absence of the wealth tax, and redistribution across agents. In the calibrated model at hand the negative effects on aggregates outweigh the positive redistributive effect: the consumption-equivalent welfare measure declines by 1.3% (2.8%) under the stochastic wealth tax.

Accuracy Results. Figure 8 reports statistics from solving the model under different tax regimes and with varying number of aggregate moments. On the left-hand side of the figure, we report the energy shares E_i of the singular values from the model-reduction step. Without a wealth tax, most of the energy is concentrated in the first few singular values, while the remaining E_i are close to zero, indicating that higher moments add little to capturing aggregate dynamics. With a 10% or 20% wealth tax, the decay is notably flatter: higher moments contribute meaningfully and are therefore required to capture the dynamics. The change from 0% to 10% is much larger than from 10% to 20% — even a 10% tax, occurring on average every ten years, already reshapes the wealth distribution and raises the model's effective dimensionality, while the incremental effect of a larger tax is comparatively smaller.

On the right-hand side of Figure 8 we plot the mean error and the 95th-percentile error (both on $\log_{10}$ scale) against the number of moments used to summarize the distribution. For both positive tax levels and both error metrics, accuracy improves roughly by an order of magnitude as additional moments are included. In the 10% case, gains taper after the fifth moment, whereas the 20% case continues to benefit from additional

[26] All results use 10,000 sample points and the linear moment selection specification described in Section 4.2.

Figure 8: Heterogeneous Agent Model — Energy Shares and Errors.



The left panel shows the energy shares E_i of the i-th singular value (index reported on the horizontal axis) from the model-reduction step under wealth-tax rates of 0%, 10%, and 20%. Taxation flattens the singular-value decay, indicating a higher effective dimensionality than in the no-tax economy. The right panel plots the mean and 95th-percentile master equation errors against the number of aggregate moments employed in the solution — adding moments yields substantial accuracy gains.

moments — consistent with the flatter energy profile under the higher tax. At a low number of moments, the 10%-tax economy is more accurate than the 20%-tax economy, but the gap narrows as the number of moments grows.[27]

5 Conclusion

We develop a novel method for computing equilibria in dynamic economic models using tensor train approximation. By exploiting low-rank structure in equilibrium functions, the approach yields accurate approximations while remaining computationally tractable in high-dimensional state spaces. We believe that TTA can fill the gap between local polynomial approximation methods and global neural-net or sparse-grid solution approaches, combining flexibility with structure in a way that is particularly well suited to quantitative macroeconomic applications.

There are numerous promising macroeconomic applications for TTA. As Section 4 demonstrates, heterogeneous-agent models with aggregate risk are a natural setting for this method, with heterogeneous-agent New Keynesian models representing an immediate next step. Economic parameters could also be incorporated directly into the state space of the TTA, analogous to the neural-network approach in Kase et al.

[27] In the tax-free economy errors are substantially lower (-3.9 for the mean error and -3.5 at the 95th percentile), as the tax introduces discrete jumps and more distributional movement, while the TFP process alone generates relatively little variation.

(2025), thereby allowing estimation to be carried out using a single solution computed on an extended state space. Overlapping-generations models with large aggregate risk, such as those studied in Brumm and Hußmann (2025), as well as extensions with a finer generational structure, provide further examples where TTA could substantially outperform adaptive sparse grids and push the boundary of what can be done in quantitative macroeconomic modeling.

Beyond macroeconomic applications, tensor train approximation may also provide a useful tool for handling high-dimensional objects in econometrics. By exploiting low-rank structure across multiple dimensions, tensor methods could extend flexible econometric analysis to settings where the curse of dimensionality currently makes estimation computationally infeasible.

Future refinements of our TTA methodology may include the adaptive selection of ranks and basis functions while preserving simulation-based sampling. A natural starting point for such an extension would be the rank-selection procedure of Grasedyck and Krämer (2019). We also see considerable potential in developing more general state-space reduction methods that go beyond the moment-selection procedure introduced in this paper — for instance, akin to Scheidegger and Bilionis (2019). Such approaches appear particularly well suited to the off-grid sampling capabilities of TTA. Finally, tensor train methods remain an active area of research in applied mathematics and physics, suggesting substantial scope for further methodological advances that may also benefit quantitative economics.

APPENDIX

A Details TTA

This appendix collects technical details on the implementation of tensor train approximation. We first describe the vectorization and orthogonalization steps underlying the ALS algorithm and then present the resulting TTA routine in algorithmic form. Finally, we show that the same ALS structure applies to a broader class of quadratic problems, which clarifies why the computational approach used in the paper extends beyond the basic least-squares fitting problem.

A.1 Details TTA-Algorithm

In what follows, we discuss how the single-core subproblem (5) can be written in vectorized form and how the cores can be orthogonalized for numerical stability. Algorithm 1 presents the complete TTA algorithm.

Vectorization. The single-core subproblem (5) can be written as

$$\min_{v^k} \quad \frac{1}{N}||y - B^k v^k||_2^2 + \lambda||v^k||_2^2, \tag{29}$$

where $v^k = \text{vec}\left(W^k\right)$ represents the wagon to be optimized, and B^k contains all information about other wagons and about basis function evaluations at all sample points. B^k is constructed as follows: First, let $\mathcal{X}^k_{i_k n} = \phi_{i_k}\left(x^n_k\right)$ denote the basis values in dimension k for sample point n. Then define the left and right stacks, $\mathcal{L}^k$ and $\mathcal{R}^k$, starting with $\mathcal{L}^0 = \mathcal{R}^d = \vec{1} \in \mathbb{R}^N$ and recursively proceeding with

$$\mathcal{L}^k = \left(\mathcal{L}^{k-1}_{j_{k-2}n} \cdot W^{k-1}_{j_{k-2}i_{k-1}j_{k-1}} \cdot \mathcal{X}^{k-1}_{i_{k-1}n}\right)_{j_{k-1}n}, \quad \mathcal{R}^k = \left(W^{k+1}_{j_k i_{k+1} j_{k+1}} \cdot \mathcal{X}^{k+1}_{i_{k+1}n} \cdot \mathcal{R}^{k+1}_{j_{k+1}n}\right)_{j_k n}. \tag{30}$$

Finally, define

$$\mathcal{B}^k = \left(\mathcal{L}^k_{j_{k-1}n} \cdot \mathcal{X}^k_{i_k n} \cdot \mathcal{R}^k_{j_k n}\right)_{n j_{k-1} i_k j_k}, \tag{31}$$

and reshape it into a matrix $B^k \in \mathbb{R}^{N \times (r_{k-1} m_k r_k)}$.

Orthogonalization. To maintain numerical stability, Holtz et al. (2012) propose to keep a left-orthonormal prefix of cores and a right-orthonormal suffix during the ALS sweeps. For the k-th core W^k define its left and right unfoldings by grouping indices

$$\text{unf}_L(W^k) \in \mathbb{R}^{(r_{k-1}m_k)\times r_k}, \quad \text{unf}_R(W^k) \in \mathbb{R}^{(r_k m_k)\times r_{k-1}}. \tag{32}$$

We call W^k left-orthonormal and right-orthonormal, respectively, if it satisfies:

$$\mathrm{unf}_L(W^k)^\intercal \mathrm{unf}_L(W^k) = I, \quad \mathrm{unf}_R(W^k)^\intercal \mathrm{unf}_R(W^k) = I. \tag{33}$$

During a left-to-right sweep we enforce left-orthonormality via a (thin) QR decomposition,

$$\mathrm{unf}_L(W^k) = QR, \quad Q^\intercal Q = I, \tag{34}$$

denoting the outcome by $[Q, R] = \mathrm{qrd}(\mathrm{unf}_L(W^k))$. We then reshape Q, $\mathrm{res}(Q) \in \mathbb{R}^{r_{k-1} \times m_k \times r_k}$, and set $W^k := \mathrm{res}(Q)$. The triangular factor, R, is absorbed into the next core, $W^{k+1} := \mathrm{res}\left(\mathrm{unf}_R\left(W^{k+1}\right) \cdot R^\top\right)$, so that the overall tensor remains unchanged but the next subproblem is better conditioned. In the right-to-left sweep right-orthonormality is analogously enforced. Details of the procedure are given in Algorithm 1 and displayed in Figure 2.

A.2 Generic ALS Template

The TTA algorithm introduced in Section 2 is applicable to a wide range of quadratic problems that can be minimized efficiently using ALS. Let $A \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ be the full tensor $u = \mathrm{vec}(A) \in \mathbb{R}^M$ with $M = \prod_{k=1}^d m_k$. We consider functionals with constant global Hessian

$$\mathcal{J}(u) = \tfrac{1}{2}||Su - b||_2^2 + \langle Bu - c, e\rangle\,, \tag{35}$$

where $S \in \mathbb{R}^{N \times M}, b \in \mathbb{R}^N, B \in \mathbb{R}^{K \times M}, c \in \mathbb{R}^K$ and $e \in \mathbb{R}^K$. This class of problems covers the least-squares approximation problem (Algorithm 1), the augmented-Lagrangian objective in Section 4, and others; see Holtz et al. (2012). If a problem can be posed as (35) and all but one TT cores are frozen, each ALS update reduces to solving a small linear system, which makes this class particularly attractive for TTA.

Local Normal Equations. Let $W^k \in \mathbb{R}^{r_{k-1} \times m_k \times r_k}$ be the k-th core and $v = \mathrm{vec}(W^k) \in \mathbb{R}^{r_{k-1} m_k r_k}$. With all other cores frozen, the vectorized tensor can be written as

$$u = P_k v,$$

where $P_k \in \mathbb{R}^{M \times r_{k-1} m_k r_k}$ is the linear retraction operator that maps the local core into the global tensor. Then the local minimization $\min_v \mathcal{J}(P_k v)$ has the normal equations

$$\hat{S}_k^\top \hat{S}_k v = \hat{S}_k^\top b - \hat{B}_k^\top e, \quad \hat{S}_k := SP_k, \quad \hat{B}_k := BP_k \tag{36}$$

These are small linear systems of size $r_{k-1} m_k r_k$.

Algorithm 1 Tensor Train Approximation (TTA)

Require: Basis dimensions $m = (m_1, \ldots, m_d)$; ranks $r = (r_0, \ldots, r_d)$ with $r_0 = r_d = 1$; data points $\{x^n, y^n\}_{n=1}^N$ with $x^n \in \mathbb{R}^d$, $y^n \in \mathbb{R}$; initial cores $W^1, \ldots, W^d$ with $W^k \in \mathbb{R}^{r_{k-1} \times m_k \times r_k}$ and W^k right orthonormal for $k > 1$; regularization λ; tolerance ε.

Alternating Least Squares

Build initial stacks $\mathcal{L}^k, \mathcal{R}^k$ using equation (30).

Set $\tilde{\xi} = \infty$, $\epsilon = \infty$, $\tilde{y} = \vec{0}$.

while $\epsilon > \varepsilon$ **do** ▷ Macro-Iterations (sweeps)

for $k = 1, \ldots, d-1$ **do** ▷ First Half-Sweep

Compute B^k using equation (31).

Solve $v^k := \arg\min_v ||B^k v - y||_2^2 + \lambda ||v||_2^2$.

Compute $[Q, R] = \mathrm{qrd}(\mathrm{unf}_L(\mathrm{res}(v^k)))$.

Set $W^k := \mathrm{res}(Q)$, $W^{k+1} := \mathrm{res}\left(\mathrm{unf}_R\left(W^{k+1}\right) \cdot R^\top\right)$.

Update left stack $\mathcal{L}^{k+1}$ using equation (30).

end for

for $k = d, \ldots, 2$ **do** ▷ Second Half-Sweep

Compute B^k using equation (31).

Solve $v^k := \arg\min_v ||B^k v - y||_2^2 + \lambda ||v||_2^2$.

Compute $[Q, R] = \mathrm{qrd}(\mathrm{unf}_R(\mathrm{res}(v^k)))$.

Set $W^k := \mathrm{res}(Q)$, $W^{k-1} := \mathrm{res}\left(\mathrm{unf}_L\left(W^{k-1}\right) \cdot R^\top\right)$.

Update right stack $\mathcal{R}^{k-1}$ using equation (30).

end for

Set $y := B^k v^k$

Compute $\xi = ||y - \tilde{y}||_2$, set $\epsilon := |\tilde{\xi} - \xi|$, $\tilde{\xi} := \xi$, and $\tilde{y} := y$.

end while

Return Tensor train approximation $\mathcal{T} = \mathcal{A}\left(m, r, \{x^n, y^n\}_{n=1}^N\right) = \left(W^1, \ldots, W^d\right)$.

Efficient Implementation via Environments. Algorithm 1 introduced left / right stacks. They provide an efficient implementation of the reduced objects $\hat{S}_k$ and $\hat{B}_k$, so neither P_k nor S is ever formed explicitly.

Least-Squares Special Case. For the dataset $\{x^n, y^n\}_{n=1}^N$ with basis $\{\phi_{i_j}(\cdot)\}_{i_j=1}^{m_j}$ in each dimension j, the full-tensor least-squares problems corresponds to $B = 0, c = 0, e = 0$, and[28]

$$S = \begin{bmatrix} \left\{\prod_{j=1}^d \phi_{i_j}(x_j^1)\right\}_{i_1=1,\dots,i_d=1}^{m_1,\dots,m_d} \\ \vdots \\ \left\{\prod_{j=1}^d \phi_{i_j}(x_j^N)\right\}_{i_1=1,\dots,i_d=1}^{m_1,\dots,m_d} \end{bmatrix} \in \mathbb{R}^{N\times M}, \quad b = \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix} \in \mathbb{R}^N. \tag{37}$$

Let $\mathcal{L}^k \in \mathbb{R}^{N\times r_{k-1}}$ and $\mathcal{R}^k \in \mathbb{R}^{N\times r_k}$ denote the left/right stacks defined recursively from the cores and basis evaluations for all points in the dataset. Then $\hat{S}_k$ admits the compact factorization

$$\hat{S}_k = \mathcal{L}^k_{nj_{k-1}} \cdot \phi(x_k)_{ni_k} \cdot \mathcal{R}^k_{nj_k} \in \mathbb{R}^{N\times r_{k-1}m_k r_k}, \tag{38}$$

without ever building S or P_k. It is this structure that mitigates the curse of dimensionality in the ALS update steps. Algorithm 2 provides a generic formulation of ALS.

B Details IRBC Model

This appendix provides auxiliary material for the IRBC application. It defines the Euler equation errors used to assess accuracy, describes the non-smooth model variant with irreversible investment, and records the solution algorithms in algorithmic form.

B.1 Details Euler Equation Errors in the IRBC Model

This appendix defines the Euler equation errors used in Section 3 to assess the accuracy of our solution method. Because the simple and the sophisticated approaches (both applied to the smooth model) differ in their implementation, they require two corresponding formulations of the Euler equation error. The Euler equation error for the non-smooth model is reported in Appendix B.2.

[28] We have excluded the regularization term, which is only for numerical stability at the local cores.

Algorithm 2 Generic ALS Algorithm

Require: Basis dimensions $m = (m_1, \ldots, m_d)$; ranks $r = (r_0, \ldots, r_d)$ with $r_0 = r_d = 1$; initial right-orthonormal cores $W^1, \ldots, W^d$ with $W^k \in \mathbb{R}^{r_{k-1} \times m_k \times r_k}$; operator $\mathcal{J}$ of the form (35) with $S \in \mathbb{R}^{N \times M}, b \in \mathbb{R}^N, B \in \mathbb{R}^{K \times M}$, and $c \in \mathbb{R}^K, e \in \mathbb{R}^K$; tolerance ε.

Generic Alternating Least Squares

Set $\tilde{\xi} = \infty$ and $\epsilon = \infty$.

while $\epsilon > \varepsilon$ **do** ▷ Macro-Iterations (sweeps)

for $k = 1, \ldots, d-1$ **do** ▷ First Half-Sweep

Solve $v^k := \arg\min_v \mathcal{J}(P_k v)$ via (36) using $\hat{S}_k$ and $\hat{B}_k$.

Compute $[Q, R] = \mathrm{qrd}(\mathrm{unf}_L(\mathrm{res}(v^k)))$.

Set $W^k := \mathrm{res}(Q), W^{k+1} := \mathrm{res}\left(\mathrm{unf}_R\left(W^{k+1}\right) \cdot R^\top\right)$.

Update left stack $\mathcal{L}^{k+1}$ using equation (30).

end for

for $k = d, \ldots, 2$ **do** ▷ Second Half-Sweep

Solve $v^k := \arg\min_v \mathcal{J}(P_k v)$ via (36) using $\hat{S}_k$ and $\hat{B}_k$.

Compute $[Q, R] = \mathrm{qrd}(\mathrm{unf}_R(\mathrm{res}(v^k)))$.

Set $W^k := \mathrm{res}(Q), W^{k-1} := \mathrm{res}\left(\mathrm{unf}_L\left(W^{k-1}\right) \cdot R^\top\right)$.

Update right stack $\mathcal{R}^{k-1}$ using equation (30).

end for

Compute objective $\xi := \mathcal{J}(W^1, \ldots, W^d)$

Set $\epsilon := |\tilde{\xi} - \xi|$ and $\tilde{\xi} := \xi$.

end while

Return Tensor train approximation $\mathcal{T} = \mathcal{A}\left(m, r, \{x^n, y^n\}_{n=1}^N\right) = \left(W^1, \ldots, W^d\right)$.

Simple Solution. For each country j, we use the tensor train representation of the policy function k_j' and λ to determine choices and next periods state y', and evaluate next periods policy functions k_j'' and λ'. The Euler equation error for country j is then defined as:

$$EE^j = \left[\beta \mathbb{E}_t \left\{\lambda' \left[a_j' A \zeta (k_j')^{\zeta-1} + 1 - \delta + \frac{\psi}{2} g_j' (g_j' + 2)\right]\right\}\right] \cdot \left[\lambda(1 + \psi g_j)\right]^{-1} - 1 \quad (39)$$

where expectations are computed using the monomial integration rule[29]. For the aggregate resource constraint, we define the relative error as:

$$\frac{\sum_{j=1}^{M} \left[a_j A k_j^{\zeta} + k_j \left((1-\delta) - \frac{\psi}{2} g_j^2\right)\right] - k_j' - \left(\frac{\lambda}{\tau_j}\right)^{-\gamma_j}}{\sum_{j=1}^{M} \left[a_j A k_j^{\zeta} + k_j \left(-\frac{\psi}{2} g_j^2\right)\right]}. \quad (40)$$

These error measures are computed along a simulation path of 10,000 periods, using a sequence of shocks that is independent from the one used to construct the state set $\mathcal{S}$ in the solution algorithm. This allows us to assess out-of-sample accuracy.

Sophisticated Solution. For each country j, we use the tensor train representations of the policy function η_j, and determine the Lagrange multiplier λ from the resource constraint to infer the capital choice k_j' from equation (14). The Euler equation error for country j is then defined as:

$$EE^j = \beta \mathbb{E}_t \left\{\eta_j(x')\right\} \cdot \left[\lambda(1 + \psi \cdot g_j)\right]^{-1} - 1, \quad (41)$$

where expectations are computed using the same basis transformation employed during the model solution. This ensures consistency between the approximation and the evaluation of expectations. The error in the aggregate resource constraint is zero by construction. These error measures are computed along a simulation path of 10,000 periods, using a sequence of shocks that is independent from the one used to construct the state set $\mathcal{S}$ in the solution algorithm.

B.2 Details Non-Smooth IRBC Model

This appendix details the non-smooth variant of the IRBC model introduced in Section 3. In contrast to the smooth benchmark, investment is irreversible:

$$k_j' - (1-\delta) k_j \geq 0, \qquad j = 1, \ldots, M. \quad (42)$$

[29] Just like Brumm and Scheidegger (2017), we use a monomial integration rule with $2(M+1)$ integration nodes in total, which keeps the additional computational cost modest. Note, however, that this choice may come at the expense of accuracy.

The aggregate resource constraint (10) is unchanged. The Euler condition for capital now carries the KKT multiplier μ_j associated with (42):

$$\lambda[1+\psi g_j] - \mu_j = \beta \mathbb{E}_t \left\{ \lambda' \left[a'_j A \zeta (k'_j)^{\zeta-1} + 1 - \delta + \frac{\psi}{2} g'_j (g'_j + 2) \right] - (1-\delta)\mu'_j \right\}. \quad (43)$$

The multiplier satisfies the complementarity conditions

$$0 \leq \mu_j \perp (k'_j - k_j(1-\delta)) \geq 0. \quad (44)$$

For later shorthand, define the country-specific expectation kernel

$$\eta_j(x') = \lambda' \left[a'_j A \zeta (k'_j)^{\zeta-1} + 1 - \delta + \frac{\psi}{2} g'_j (g'_j + 2) \right] - (1-\delta)\mu'_j. \quad (45)$$

The map η_j is piecewise smooth and exhibits kinks along the locus where (44) switches between slack and binding.

Endogenous Grid Method. As in Section 3, we reduce the $(M+1)$-dimensional nonlinear system to a single scalar equation by exploiting EGM. The irreversibility constraint is handled by a simple case distinction. For each country j, fix next-period capital k'_j exogenously. Compute the expectations on the right-hand side of (43) via quasi-analytical integration $e_j \equiv \mathbb{E}_t\{\eta_j(x')\}$. First, compute the interior solution candidate. Set $\mu_j = 0$ and solve (43) for the implied current capital $\tilde{k}_j$. Then, check consistency between the assumed Lagrange multiplier and the implied current capital. If $k'_j \geq (1-\delta)\tilde{k}_j$, accept the interior solution $(\tilde{k}_j, \mu_j = 0)$. Otherwise, the constraint binds. Set $\tilde{k}_j = k'_j/(1-\delta)$, $\mu_j = \lambda[1+\psi g_j(\tilde{k}_j, k'_j)] - \beta e_j \geq 0$, which follows by rearranging (43). Given the set $\{\tilde{k}_j\}_{j=1}^M$ implied by the fixed $\{k'_j\}$, substitute into the aggregate budget constraint (10) and solve the resulting scalar equation for the multiplier λ. This completes one EGM update. The method retains all advantages of Section 3 while accommodating the kink induced by (42).

Error Measure. We follow Section 3 and report unit-free Euler errors. With irreversibility, we additionally penalize violations of (42). Define the percentage shortfall from the lower bound

$$IC^j \equiv 1 - \frac{k'_j}{k_j(1-\delta)}. \quad (46)$$

Table 4: Parameterization of IRBC model

Parameter	Symbol	Value
Discount Factor	β	0.99
EIS of country j	γ_j	$[0.35, 1]$
Capital share	ζ	0.36
Depreciation	δ	0.005
Std. of log-productivity	σ	0.01
Autocorrelation of log-productivity	ρ	0.95
Intensity of capital adjustment costs	ψ	0.5
Aggregate productivity	A	$(1-\beta(1-\delta))/(\zeta\beta)$
Welfare weight of country j	τ_j	A^{1/γ_j}

Let EE^j denote the standard (unit-free) Euler error as in Section 3, accounting for the amendments in (43), evaluated at points where the constraint is slack. We summarize the accuracy in the Euler equation for country j by

$$\max\left\{EE^j, IC^j, \min\left\{-EE^j, -IC^j\right\}\right\}.$$

By construction the error is non-negative, it coincides with $|EE^j|$ when the constraint is non-binding, and reduces to the (percentage) irreversibility violation when it binds. See Brumm and Scheidegger (2017) for a more detailed discussion.

B.3 IRBC Solution Algorithms

This section presents the formal solution algorithms discussed in Section 3.2, and Section 3.5.

C Details Heterogeneous Agent Model

This appendix reports implementation details for the heterogeneous-agent application. We discuss an extension of the model-reduction procedure, derive the local augmented-Lagrangian subproblems, define the reported error measure, and summarize the full solution approach in algorithmic form.

C.1 Model Reduction — POD in Koopman Observable Space

The model reduction approach in Section 4.2 uses POD to construct moments from simulated distributional snapshots. This appendix shows how the same logic can be

Algorithm 3 Solving the IRBC Model using Simple TTA

Require: Initial sample of states $\mathcal{S} = \{y_1, \ldots, y_N\} \subset Y$, initial guess for next period's policy $p^+ = (k'_1, \ldots, k'_M, \lambda)$ in TTA format, error thresholds ε_{TI} and ε_{Sim}, sequence of shocks $\{z_t\}_{t=1}^T$, $\{a_t\}_{t=1}^T$, initial simulation state y_1, quadrature rule $\mathcal{Q}$ for expectations.

Simulation & TI

while $\xi_{\text{Sim}} > \varepsilon_{\text{Sim}}$ **do** ▷ Simulation Loop

while $\xi_{\text{TI}} > \varepsilon_{\text{TI}}$ **do** ▷ TI Loop

for $k = 1, \ldots, N$ **do**

Initialize choices $\nu := (k'_1, \ldots, k'_M, \lambda) \leftarrow p^+(y_k)$

repeat

Using y_k, ν and p^+ compute RHS expectations in (9) via $\mathcal{Q}$.

Update $\nu = (k'_1, \ldots, k'_M, \lambda)$ using a non-linear solver on (9) and (10).

until residual norm at y_k is below tolerance.

Store optimal choice $\nu_k^*(y_k) := (k'_1(y_k), \ldots, k'_M(y_k), \lambda(y_k))$.

end for

Set $p_j := \mathcal{T}\left(m, r, \{y_k, \nu_k^*(y_k)\}_{k=1}^N\right)$, for $j = 1, \ldots, M+1$ using Algorithm 1.

Set $\xi_{\text{TI}} = ||p - p^+||$ and update $p^+ := p$.

end while

Set $\tilde{\mathcal{S}} = \{\tilde{y}_1, \ldots, \tilde{y}_N\} := \mathcal{S}$.

for $t = 1, \ldots, T$ **do** ▷ Update Simulation Path

Evaluate $p(y_t) = (k'_1, \ldots, k'_M, \lambda)$, set $y_{t+1} := (\vec{k}', \vec{a}_{t+1}, z)$.

end for

Compute $\xi_{\text{Sim}} = ||\tilde{\mathcal{S}} - \mathcal{S}|| = \frac{1}{N} \sum_{i=1}^N |\tilde{y}_i - y_i|$.

end while

Draw new shocks $\{z_t\}_{t=1}^T$ and $\{a_t\}_{t=1}^T$.

for $t = 1, \ldots, T$ **do** ▷ Compute Euler Errors

Evaluate $p(y_t) = (k'_1, \ldots, k'_M, \lambda)$, set $y_{t+1} := (\vec{k}', \vec{a}_{t+1}, z)$.

Compute Euler errors using (39) and (40).

end for

Algorithm 4 Solving the IRBC Model using TTA

Require: Initial sample of states $\mathcal{S} = \{x_1, \ldots, x_N\} \subset X$, initial guess for next period's expectations terms $\eta^+ = (\eta_1^+, \ldots, \eta_M^+)$ in TTA format, error thresholds ε_{EGM} and ε_{Sim}, sequence of shocks $\{z_t\}_{t=1}^T$, $\{\tilde{a}_t\}_{t=1}^T$, initial simulation state x_1.

Simulation & EGM

while $\xi_{\text{Sim}} > \varepsilon_{\text{Sim}}$ **do** ▷ Simulation Loop

  while $\xi_{\text{EGM}} > \varepsilon_{\text{EGM}}$ **do** ▷ EGM Loop

    for $k = 1, \ldots, N$ **do**

      Fix future capital choices $k'_1, \ldots, k'_M$ from $\mathcal{S}$.

      Initialize the multiplier $\lambda(x_k)$.

      repeat

        Given η^+, compute the expectations quasi-analytically using (15).

        Construct the endogenous state $x_k = (\vec{k}, \vec{\tilde{a}}_k, z_k)$ from (9).

        Update $\lambda(x_k)$ numerically to find the root of (10).

      until $\lambda(x_k)$ converges

      Compute $\eta(x_k) = (\eta_1(x_k), \ldots, \eta_M(x_k))$ via (14).

    end for

    Update $\eta_j := \mathcal{T}\left(m, r, \{x_k, \eta_j(x_k)\}_{k=1}^N\right)$, for $j = 1, \ldots, M$ using Algorithm 1.

    Set $\xi_{\text{EGM}} = ||\eta - \eta^+||$ and update $\eta^+ := \eta$.

  end while

  Set $\tilde{\mathcal{S}} = \{x_1, \ldots, x_N\} := \mathcal{S}$.

  for $t = 1, \ldots, T$ **do** ▷ Update Simulation Path

    Evaluate η at x_t, infer k', λ using equations (14, 10), set $x_{t+1} := (\vec{k}', \vec{\tilde{a}}_{t+1}, z)$.

  end for

  Compute $\xi_{\text{Sim}} = ||\tilde{\mathcal{S}} - \mathcal{S}|| = \frac{1}{N} \sum_{i=1}^N |\tilde{x}_i - x_i|$.

end while

Draw new shocks $\{z_t\}_{t=1}^T$ and $\{\tilde{a}_t\}_{t=1}^T$.

for $t = 1, \ldots, T$ **do** ▷ Compute Euler Errors

  Evaluate η at x_t, and infer k', λ using equations (14, 10). Set $x_{t+1} := (\vec{k}', \vec{\tilde{a}}_{t+1}, z)$.

  Compute Euler errors using (41).

end for

extended to nonlinear distributional dynamics by applying POD after lifting the distribution into a richer space of observables.

Suppose that the distribution evolves according to the nonlinear law of motion

$$g(t + \Delta t) = f(g(t)).$$

Although the map f may be nonlinear in the original state g, Koopman theory represents the dynamics linearly on functions of the state. For an observable ψ, the Koopman operator $\mathcal{K}$ is defined by

$$(\mathcal{K}\psi)(g) = \psi(f(g)).$$

Thus, while the distribution itself may follow nonlinear dynamics, suitably chosen observables of the distribution evolve under a linear operator.

To exploit this idea for model reduction, consider a dictionary of observables

$$\psi(g) = \left[\phi_1(g), \ldots, \phi_M(g)\right]^\top,$$

where the functions ϕ_j may include, for example, polynomial terms, interactions, or radial basis functions. Given simulated snapshots of the distribution, define the lifted snapshot matrices

$$\Psi_- = \left[\psi(g(t_1)), \ldots, \psi(g(t_{T-1}))\right], \qquad \Psi_+ = \left[\psi(g(t_2)), \ldots, \psi(g(t_T))\right].$$

We then apply POD to the lifted snapshots rather than to the original distributional snapshots. Specifically, compute a rank-k truncated SVD,

$$\Psi_- \approx U_\psi \Sigma_\psi V_\psi^\top.$$

The columns of U_ψ span the dominant directions of variation in the observable space. Projecting the lifted distribution onto this basis yields nonlinear moment coordinates

$$m_\psi(t) = U_\psi^\top \psi(g(t)) \in \mathbb{R}^k.$$

These moments generalize the moments in Section 4.2: instead of summarizing the distribution by dominant linear directions in $g(t)$, they summarize it by dominant directions in a nonlinear feature representation of $g(t)$.

To obtain an approximate law of motion for these moments, we estimate a linear transition in the reduced observable space,

$$m_\psi(t_{i+1}) \approx A_{\psi,k} m_\psi(t_i),$$

where

$$A_{\psi,k} = U_\psi^\top \Psi_+ V_\psi \Sigma_\psi^{-1}.$$

This is the projected finite-dimensional Koopman approximation associated with the chosen dictionary. In the special case in which the observable is simply the identity, $\psi(g) = g$, this construction reduces to the linear POD-based reduction described in Section 4.2. With a richer dictionary, the same logic can capture nonlinear distributional dynamics through a low-dimensional set of learned moments in observable space.

C.2 Local Augmented Lagrangian Problem

To apply Algorithm 2 efficiently to the augmented Lagrangian while working in TT format, we must never assemble the global functional $\mathcal{J}$ on the full tensor A nor form the retraction operator P_k. Section 2 showed how left/right contraction stacks — $\mathcal{L}$ and $\mathcal{R}$ — yield a reduced linear system that acts directly on the k-the core W^k. For the master equation in Section 4 we follow the same strategy, but we build separate stacks for each instance of the value function $V, V^{\text{tax}}, V^{\partial_a}, V^{\partial_z}, V^{\partial_{z^2}}, V^{\partial_\Gamma}$. From these we assemble the local projections M_k (interior operator) and C_k (boundary operator) at core k.

Prerequisites and Notation. Index coordinates by $d \in \{a, z, m_1, \ldots, m_D\}$, with polynomial basis order m_k along coordinates k. Let $\mathcal{S}$ be the sample set, $|\mathcal{S}| = N$. Precompute evaluation matrices

$$\mathcal{X}_k^{\square,ij} \in \mathbb{R}^{N \times m_k}, \quad \square \in \{V, \partial_a, \partial_z, \partial_{z^2}\partial_\Gamma, \text{tax}\}, \quad i, j \in \{1, 2\},$$

where $\mathcal{X}_k^{\square,ij}$ stores basis values and $\mathcal{X}_k^{\partial_a,ij}, \mathcal{X}_k^{\partial_z,ij}, \mathcal{X}_k^{\partial_{z^2},ij}$ store the corresponding differential operators applied in the basis. The matrix $\mathcal{X}_k^{\text{tax},ij}$ evaluates basis functions at post-tax variables $(\tilde{a}, \tilde{\Gamma})$. Represent V in TT form with cores $W^1, \ldots, W^d$ and TT ranks r_k (with $r_0 = r_d = 1$). For each label $\square$ define local left/right stacks by contracting all cores except W^k against the appropriate $\mathcal{X}_\ell^{\square,ij}$ for $\ell \neq k$

$$\mathcal{L}_k^{\square,ij} \in \mathbb{R}^{N \times r_{k-1}}, \quad \mathcal{R}_k^{\square,ij} \in \mathbb{R}^{r_k \times N}.$$

Form the local basis tensor by inserting the k-th dimension

$$\mathcal{B}_{no\ell p}^{\square} = \left(\left(\mathcal{L}_k^{\square,ij} \right)_{no} \cdot \left(\mathcal{X}_k^{\square,ij} \right)_{n\ell} \cdot \left(\mathcal{R}_k^{\square,ij} \right)_{pn} \right)_{no\ell p} \in \mathbb{R}^{N \times r_{k-1} \times m_k \times r_k}. \tag{47}$$

Finally, stack indices (o, ℓ, p) into columns to obtain the local design matrices

$$\boldsymbol{B}^{\square,ij} \in \mathbb{R}^{N \times K_k}, \quad K_k := r_{k-1} m_k r_k,$$

and identify the optimization variable with the vectorized core $v_k := \mathrm{vec}(W^k) \in \mathbb{R}^{K_k}$.

Local Augmented Lagrangian. Given interior and boundary projections $M_k \in \mathbb{R}^{N \times K_k}$ and $C_k \in \mathbb{R}^{M \times K_k}$, the localized subproblem at core k reads

$$\min_{v_k} \frac{1}{2} ||M_k v_k - u||_2^2 + \frac{\gamma}{2} ||C_k v_k - h - t||_2^2 + \mu^\top (C_k v_k - h - t), \tag{48}$$

where $u \in \mathbb{R}^N$ is the (value-independent) interior right-hand side, $h \in \mathbb{R}^L$ is the boundary target, $t \in \mathbb{R}^L$ are non-negativity slacks, $\mu \in \mathbb{R}^L$ are the current multipliers, and $\gamma > 0$ is the penalty.

Interior Operator. The interior projection is the full operator M with the global basis X replaced by the local basis $\boldsymbol{B}$.

$$\begin{aligned} M_k^{ij} = &\left(\rho_i + \frac{1}{\Delta} + \lambda_j + \omega_i + \theta \right) \boldsymbol{B}^{V,ij} - \left(w(z,\Gamma)\varepsilon_j + r(z,\Gamma)a - c_{ij} \right) \boldsymbol{B}^{\partial_a,ij} \\ &- \kappa(\bar{z} - z) \boldsymbol{B}^{\partial_z,ij} - \frac{1}{2}\sigma_z^2 \boldsymbol{B}^{\partial_{z^2},ij} - \mu(z,\Gamma) \boldsymbol{B}^{\partial_\Gamma,ij} - \lambda_j \boldsymbol{B}^{V,ij} - \omega_i \boldsymbol{B}^{V,ij} - \theta \boldsymbol{B}^{\mathrm{tax},ij}. \end{aligned}$$

Boundary Operator. For the no-borrowing boundary condition we have

$$C_k^{ij} = \boldsymbol{B}^{\partial_a,ij}, \quad \text{and } h \text{ as in the global problem.}$$

Reduced Normal Equations. Consistent with the general formulation in Appendix A.2 define the stacked system

$$\hat{S}_k = \begin{bmatrix} M_k \\ \sqrt{\gamma} C_k \end{bmatrix}, \quad \hat{B}_k = 0, \quad \hat{u}_k = \begin{bmatrix} u \\ \sqrt{\gamma}(h + t - \mu/\gamma) \end{bmatrix}.$$

To solve the minimization problem apply Algorithm 2.

C.3 Error Measure

We evaluate accuracy via a relative master equation error in consumption units. Given a candidate value function $V^{ij}(\cdot; \mathcal{T})$, obtain the optimal consumption from the FOC,

$$c_{ij}^*(a, z, \Gamma) = (u')^{-1} \left(\partial_a V^{ij}(a, z, \Gamma; \mathcal{T}) \right).$$

Compute the implied consumption $\tilde{c}^*_{ij}(a,z,\Gamma)$ from the master equation

$$\begin{aligned}\tilde{c}^*_{ij}(a,z,\Gamma) = u^{-1}\Big(&\rho_i V^{ij}(a,z,\Gamma;\mathcal{T}) - \partial_a V^{ij}(a,z,\Gamma;\mathcal{T})\left[w(z,\Gamma)\varepsilon_j + r(z,\Gamma)a - c^*_{ij}\right] \\ &- \lambda_j\big(V^{ij}(a,z,\Gamma;\mathcal{T}) - V^{ij}(a,z,\Gamma;\mathcal{T})\big) \\ &- \omega_i\Big(V^{ij}(a,z,\Gamma;\mathcal{T}) - V^{ij}(a,z,\Gamma;\mathcal{T})\Big) \\ &- \kappa(\bar{z}-z)\,\partial_z V^{ij}(a,z,\Gamma;\mathcal{T}) - \tfrac{1}{2}\sigma_z^2\,\partial_{zz}V^{ij}(a,z,\Gamma;\mathcal{T}) \\ &- \langle \mu_\Gamma(z,\Gamma), \nabla_\Gamma V^{ij}(a,z,\Gamma;\mathcal{T})\rangle - \theta\big(V^{ij}(\tilde{a},z,\tilde{\Gamma};\mathcal{T}) - V^{ij}(a,z,\Gamma;\mathcal{T})\Big)\end{aligned}$$

At the borrowing constraint, we also record the boundary violation in relative consumption units,

$$\mathcal{E}^B_{ij}(a,z,\Gamma) = \begin{cases} 0, & a > \underline{a} \\ \frac{\max\left\{(u')^{-1}\left(\partial_a V^{ij}(\underline{a},z,\Gamma;\mathcal{T})\right) - w(z,\Gamma)\varepsilon_j + r(z,\Gamma)\underline{a}, 0\right\}}{w(z,\Gamma)\varepsilon_j + r(z,\Gamma)\underline{a}}, & a = \underline{a} \end{cases}$$

Our pointwise error metric is then

$$\mathcal{E}_{ij}(a,z,\Gamma) = \max\left\{\left|\frac{\tilde{c}_{ij}(a,z,\Gamma)}{c^*_{ij}(a,z,\Gamma)} - 1\right|, \mathcal{E}^B_{ij}(a,z,\Gamma)\right\}.$$

We simulate a path of aggregate distributions $\{g(t)\}_{t=1}^T$ by drawing shocks and evolving the Kolmogorov forward equation on a fine wealth grid using the policies implied by $V^{ij}(\cdot;\mathcal{T})$. At each t, we draw 100 individuals from the joint distribution over (i,j) and assets $a \sim g_{ij}(t)$, evaluate $\mathcal{E}_{ij}(a,z,\Gamma_t)$ at those draws, and summarize the resulting error distribution over time. Note that, in computing the master equation error, both the drift of the aggregate moments and their jump in response to the tax shock are taken from the "true" transition in the simulation rather than from the lower-dimensional forecast used to obtain the Markovian solution of the value function. This ensures that the error induced by forecasting is captured by the error metric.

C.4 Heterogeneous Agent Solution Algorithm

Algorithm 5 provides a detailed description of the solution method described in Section 4.

Algorithm 5 Solving the Master Equation using TTA

Require: Asset grid $\mathring{a} \in \mathbb{R}^n$; initial distributions $\mathcal{G} = \{g_i\}_{i=1}^N \subset \mathbb{R}^{4n}$, where each g_i stacks the four idiosyncratic-state histogram over $\mathring{a}$; moment generators $\{f_k\}_{k=1}^M$, with $f_k \in \mathbb{R}^{4n}$. For each i, define the moment vector $m_i \in \mathbb{R}^M$ by $[m_i]_k = \langle g_i, f_k \rangle$. Let $\mathcal{S} = \{m_i\}_{i=1}^N \subset \mathbb{R}^M$; moment law of motion $\mu(z, \Gamma) : (z, m) \mapsto \dot{m}$, mapping today's (z, m) to drifts; tax-event moment forecast $\Lambda(\Gamma) : m \mapsto \tilde{m}$, mapping pre-shock into after-shock moments; state sample $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^{M+2}$ with $x_i = (a_i, z_i, m_i)$ and $\{a_i\}_{i=1}^N \subset \mathbb{R}$; initial value function (TT) for each idiosyncratic state an initial guess $V^{ij}(\cdot; \mathcal{T}_{ij})$; tolerance thresholds $\varepsilon_{\text{ADMM}}$, ε_{VFI}, ε_{FC}, $\varepsilon_{\text{Sim}} > 0$ aggregate shocks $\{z_t\}_{t=1}^T$ and tax indicators $\{\mathbb{1}_t^{\text{tax}}\}_{t=1}^T$.

Simulation, VFI & ADMM

Evaluate $\tilde{v}_{ij} := V^{ij}(X)$.

Initialize ADMM penalty $\gamma > 0$, multiplier μ and slacks $t \geq 0$.

Set residuals $\xi_{\text{ADMM}}, \xi_{\text{VFI}}, \xi_{\text{FC}}, \xi_{\text{Sim}} := \infty$.

while $\xi_{\text{Sim}} > \varepsilon_{\text{Sim}}$ **do** ▷ Simulation Loop

  while $\xi_{FC} > \varepsilon_{\text{FC}}$ **do** ▷ Forecast Loop

    Set $v_{ij} := \tilde{v}_{ij}$.

    while $\xi_{\text{VFI}} > \varepsilon_{\text{VFI}}$ **do** ▷ VFI Loop

      compute $c_{ij}^* = (u')^{-1}\left(\partial_a V^{ij}(X; \mathcal{T}_{ij})\right)$ from FOC (22).

      while $\xi_{\text{ADMM}} > \varepsilon_{\text{ADMM}}$ **do** ▷ ADMM Loop

        Minimize the augmented Lagrangian (26) using ALS (2).

        Update slack and multiplier using equations (27) and (28).

        Adjust the penalty γ to balance primal and dual residuals.

        Set $\xi_{\text{ADMM}} := ||t_{\text{new}} - t_{\text{old}}||_2^2$

      end while

      Update $V^{ij} = V(\cdot; \mathcal{T}_{ij})$ and evaluate $v'_{ij} := V^{ij}(X)$.

      Set $\xi_{\text{VFI}} := ||v'_{ij} - v_{ij}||$ and $v_{ij} := v'_{ij}$.

    end while

    On $\mathcal{G}$ compute $\{\dot{g}_i\}_{i=1}^N$ using $V^{ij}(\cdot; \mathcal{T}_{ij})$, then compute $\{\dot{m}_i\}_{i=1}^N$.

    Fit the updated law of motion $\tilde{\mu}(z, \Gamma) : (z, m) \mapsto \dot{m}$, and set $\mu(z, \Gamma) := \tilde{\mu}(z, \Gamma)$

    Set $\xi_{\text{FC}} = ||v'_{ij} - \tilde{v}_{ij}||$, $\tilde{v}_{ij} := v'_{ij}$.

  end while

  Simulate $\tilde{\mathcal{G}} := \{\tilde{g}_i\}_{i=1}^N$ on the grid $\mathring{a}$ using $V^{ij}(\cdot; \mathcal{T}_{ij})$, starting at g_1.

  Select moments $\{f_k\}_{k=1}^M$ using a posteriori model reduction (4.2).

  Fit updated tax-shock forecast $\tilde{\Lambda}(\Gamma) := m \mapsto \tilde{m}$, and set $\Lambda(\Gamma) := \tilde{\Lambda}(\Gamma)$.

  Set $\xi_{\text{Sim}} := ||\langle \tilde{g}, f \rangle - \langle g, f \rangle||$, and $\mathcal{G} := \tilde{\mathcal{G}}$.

end while

Compute error measure.

References

Achdou, Y., Han, J., Lasry, J.-M., Lions, P.-L., & Moll, B. (2022). Income and wealth distribution in macroeconomics: A continuous-time approach. *The Review of Economic Studies*, *89*(1), 45–86.

Ahn, S., Kaplan, G., Moll, B., Winberry, T., & Wolf, C. (2018). When inequality matters for macro and macro matters for inequality. *NBER Macroeconomics Annual*, *32*(1), 1–75.

Auclert, A., Bardóczy, B., Rognlie, M., & Straub, L. (2021). Using the sequence-space Jacobian to solve and estimate heterogeneous-agent models. *Econometrica*, *89*(5), 2375–2408.

Auclert, A., Rognlie, M., & Straub, L. (2025). Fiscal and monetary policy with heterogeneous agents. *Annual Review of Economics*, *17*, 539–562.

Azinovic, M., Gaegauf, L., & Scheidegger, S. (2022). Deep equilibrium nets. *International Economic Review*, *63*(4), 1471–1525.

Azinovic-Yang, M., & Žemlička, J. (2025a). Deep learning in the sequence space. *arXiv preprint arXiv:2509.13623*.

Azinovic-Yang, M., & Žemlička, J. (2025b). Intergenerational consequences of rare disasters. *Working Paper. Available at SSRN 4386477*.

Bachmayr, M. (2023). Low-rank tensor methods for partial differential equations. *Acta Numerica*, *32*, 1–121.

Bayer, C., & Luetticke, R. (2020). Solving discrete time heterogeneous agent models with aggregate risk and many idiosyncratic states by perturbation. *Quantitative Economics*, *11*(4), 1253–1288.

Bayer, C., Luetticke, R., Weiss, M., & Winkelmann, Y. (2026). An endogenous gridpoint method for distributional dynamics. *Journal of Monetary Economics*, *158*, 103895.

Bhandari, A., Bourany, T., Evans, D., & Golosov, M. (2023). A perturbational approach for approximating heterogeneous agent models (Working Paper No. 31744). National Bureau of Economic Research.

Bigoni, D., Engsig-Karup, A. P., & Marzouk, Y. M. (2016). Spectral tensor-train decomposition. *SIAM Journal on Scientific Computing*, *38*(4), A2405–A2439.

Bilal, A. (2023). Solving heterogeneous agent models with the master equation (Working Paper No. 31103). National Bureau of Economic Research.

Brumm, J., & Hußmann, J. (2025). Public debt in calibrated OLG models: Fiscal arithmetic versus welfare analysis. *accepted at Journal of Monetary Economics*.

Brumm, J., Krause, C., Schaab, A., & Scheidegger, S. (2022). Sparse grids for dynamic economic models. In *Oxford Research Encyclopedia of Economics and Finance*.

Brumm, J., & Scheidegger, S. (2017). Using adaptive sparse grids to solve high-dimensional dynamic models. *Econometrica*, *85*(5), 1575–1612.

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

Christensen, B. J., Neri, L., & Parra-Alvarez, J. C. (2024). Estimation of continuous-time linear DSGE models from discrete-time measurements. *Journal of Econometrics*, *244*(2), 105871.

Dektor, A., Rodgers, A., & Venturi, D. (2021). Rank-adaptive tensor methods for high-dimensional nonlinear PDEs. *Journal of Scientific Computing*, *88*(2), 36.

Dennis, R. (2026). Value function iteration without the curse of dimensionality. *CAMA Working Paper Series No. 12/2026*.

Dolgov, S., Kalise, D., & Kunisch, K. K. (2021). Tensor decomposition methods for high-dimensional Hamilton–Jacobi–Bellman equations. *SIAM Journal on Scientific Computing*, *43*(3), A1625–A1650.

Druedahl, J., & Jørgensen, T. H. (2017). A general endogenous grid method for multi-dimensional models with non-convexities and constraints. *Journal of Economic Dynamics and Control*, *74*, 87–107.

Eftekhari, A., & Scheidegger, S. (2022). High-dimensional dynamic stochastic model representation. *SIAM Journal on Scientific Computing*, *44*(3), C210–C236.

Fernández-Villaverde, J., Nuno, G., Sorg-Langhans, G., & Vogler, M. (2020). Solving high-dimensional dynamic programming problems using deep learning. *Working Paper*.

Fernández-Villaverde, J., Nuño, G., & Perla, J. (2024). Taming the curse of dimensionality: Quantitative economics with deep learning (Working Paper No. 33117). National Bureau of Economic Research.

Folini, D., Friedl, A., Kubler, F., & Scheidegger, S. (2025). The climate in climate economics. *The Review of Economic Studies*, *92*(1), 299–338.

Gopalakrishna, G. (2021). Aliens and continuous time economies. *Swiss Finance Institute Research Paper*, (21-34).

Gopalakrishna, G., Gu, Z., & Payne, J. (2026). Institutional asset pricing with segmentation and household heterogeneity. *Available at SSRN 6162106*.

Gorodetsky, A., Karaman, S., & Marzouk, Y. (2019). A continuous analogue of the tensor-train decomposition. *Computer Methods in Applied Mechanics and Engineering*, *347*, 59–84.

Gorodnichenko, Y., Maliar, L., Maliar, S., & Naubert, C. (2022). Household savings and monetary policy under individual and aggregate stochastic volatility. *CEPR Discussion Paper No. 15614*.

Grasedyck, L., & Krämer, S. (2019). Stable ALS approximation in the TT-format for rank-adaptive tensor completion. *Numerische Mathematik*, *143*(4), 855–904.

Gu, Z., Laurière, M., Merkel, S., & Payne, J. (2024). Global solutions to master equations for continuous time heterogeneous agent macroeconomic models. *arXiv preprint arXiv:2406.13726*.

Han, J., Yang, Y., & E, W. (2026). DeepHAM: A global solution method for heterogeneous agent models with aggregate shocks. *Quantitative Economics*, *17*(2), 297–341.

Holtz, S., Rohwedder, T., & Schneider, R. (2012). The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, *34*(2), A683–A713.

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

Kase, H., Melosi, L., & Rottner, M. (2025). Estimating nonlinear heterogeneous agent models with neural networks. *BIS Working Papers No. 1241*.

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

Krueger, D., & Kubler, F. (2006). Pareto-improving social security reform when financial markets are incomplete!? *American Economic Review*, *96*(3), 737–755.

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

Maliar, L., Maliar, S., & Winant, P. (2021). Deep learning for solving dynamic economic models. *Journal of Monetary Economics*, *122*, 76–101.

Nuño, G., Renner, P., & Scheidegger, S. (2024). Monetary policy with persistent supply shocks. *CESifo Working Paper Series No. 11463*.

Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, *33*(5), 2295–2317.

Payne, J., Rebei, A., & Yang, Y. (2025). Deep learning for search and matching models. *Working Paper. Available at SSRN 5123878*.

Reiter, M. (2009). Solving heterogeneous-agent models by projection and perturbation. *Journal of Economic Dynamics and Control*, *33*(3), 649–665.

Reiter, M. (2023). State reduction and second-order perturbations of heterogeneous agent models. *IHS Working Paper Series No. 49*.

Richter, L., Sallandt, L., & Nüsken, N. (2021). Solving high-dimensional parabolic PDEs using the tensor train format. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th International Conference on Machine Learning* (Vol. 139).

Richter, L., Sallandt, L., & Nüsken, N. (2024). From continuous-time formulations to discretization schemes: Tensor trains and robust regression for BSDEs and parabolic PDEs. *Journal of Machine Learning Research*, *25*(248), 1–40.

Rohwedder, T., & Uschmajew, A. (2013). On local convergence of alternating schemes for optimization of convex problems in the tensor train format. *SIAM Journal on Numerical Analysis*, *51*(2), 1134–1162.

Sauzet, M. (2021). Projection methods via neural networks for continuous-time models. *Working Paper. Available at SSRN 3981838*.

Schaab, A., & Zhang, A. (2022). Dynamic programming in continuous time with adaptive sparse grids. *Working Paper. Available at SSRN 4125702*.

Scheidegger, S., & Bilionis, I. (2019). Machine learning for high-dimensional dynamic stochastic economies. *Journal of Computational Science*, *33*, 68–82.

White, M. N. (2015). The method of endogenous gridpoints in theory and practice. *Journal of Economic Dynamics and Control*, *60*, 26–41.

Yang, Y., Wang, C., Schaab, A., & Moll, B. (2025). Structural reinforcement learning for heterogeneous agent macroeconomics. *arXiv preprint arXiv:2512.18892*.