NBER WORKING PAPER SERIES

A CLUSTER-GRID PROJECTION METHOD: SOLVING PROBLEMS WITH HIGH DIMENSIONALITY

Kenneth L. Judd
Lilia Maliar
Serguei Maliar

Working Paper 15965
http://www.nber.org/papers/w15965

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
May 2010

Lilia Maliar and Serguei Maliar acknowledge support from the Hoover Institution at Stanford University, the Ivie, the Ministerio de Ciencia e Innovación and FEDER funds under the project SEJ-2007-62656 and the Generalitat Valenciana under the grants BEST/2010/142 and BEST/2010/141, respectively. The views expressed herein are those of the authors and do not necessarily reflect the views of the National Bureau of Economic Research.

A Cluster-Grid Projection Method: Solving Problems with High Dimensionality
Kenneth L. Judd, Lilia Maliar, and Serguei Maliar
NBER Working Paper No. 15965
May 2010
JEL No. C02,C63

ABSTRACT

We develop a projection method that can solve dynamic economic models with a large number of state variables. A distinctive feature of our method is that it operates on the ergodic set realized in equilibrium: we simulate a model, distinguish clusters on simulated series and use the clusters' centers as a grid for projections. Making the grid endogenous to the model allows us to avoid costs associated with finding a solution in areas of state space that are never visited in equilibrium. On a standard desktop computer, we calculate linear and quadratic solutions to a multi-country growth model with up to 400 and 80 state variables, respectively. Our solutions are global, and their accuracy does not rapidly decline away from steady state.

Kenneth L. Judd
Hoover Institution
Stanford University
Stanford, CA 94305-6010
and NBER
kennethjudd@mac.com

Serguei Maliar
Department of Economics
University of Alicante
Campus San Vicente del Raspeig
Ap. Correos 99, 03080 Alicante, Spain
maliars@merlin.fae.ua.es

Lilia Maliar
Department of Economics
University of Alicante
Campus San Vicente del Raspeig
Ap. Correos 99, 03080 Alicante, Spain
maliarl@merlin.fae.ua.es

1 Introduction

In this paper, we focus on solving dynamic economic models that have a large finite number of heterogeneous agents (consumers, producers, sectors, countries, members of a group, etc.). Some of the heterogeneity can be in the form of random variables. Models of this kind arise in many fields of economics (macroeconomics, game theory, industrial organization, international trade among others). We consider a typical example which is a multi-country stochastic growth model with a state space composed of capital stocks and technology levels of all heterogeneous countries. Computing a numerical solution to such a model can be costly. In particular, the cost grows exponentially (curse of dimensionality) if an algorithm relies on a tensor-product rule either in constructing a grid of points used for solution approximation or in constructing nodes used for numerical integration; for example, this is the case of the Galerkin projection method studied in Judd (1992).

We present a simple projection method that can deliver sufficiently accurate solutions to models with hundreds of heterogeneous agents on a standard desktop computer. A distinctive feature of our method is that it operates on the ergodic set realized in equilibrium. Making the domain endogenous to the model allows us to avoid costs associated with finding a solution in areas of state space that are never visited in equilibrium. In Figure 1a, we illustrate the advantage of this strategy using an example of the standard one-country neoclassical growth model with a closed-form solution (see Section 4 for a description of this model). Standard projection methods compute a solution in a square area, while one typically needs a solution in an ellipsoid area (the ergodic set).[1] The higher is the dimensionality of the problem, the larger is the gain from focusing on the ergodic set.[2]

To construct a grid of points representing the ergodic set of the model in

[1] Having a control over the domain on which a problem is solved is particularly useful for applications in which off-equilibrium behavior is of interest, e.g., dynamic games or developing economies. In such applications, our approach allows us to extend the domain to include other, non-ergodic-set areas of state space which are relevant for a given application. For example, if an economy starts with a low initial capital stock, we can compute a solution both in the ergodic set and below the steady state (but not above the steady state).

[2] The ratio of the volume of a hypersphere (representing the ergodic set) to that of a hypercube (representing standard exogenous grids that contain the ergodic set) declines rapidly with dimension. For example, with 2 state variables, this ratio is equal to 0.79, whereas with 100 state variables, it is equal to $2 \cdot 10^{-70}$.

question, we proceed in three steps: (i) guess a decision rule to the model and simulate a time series solution; (ii) distinguish clusters on the simulated series; (iii) compute the centers of the constructed clusters and use them as a grid of points for projections. To distinguish clusters, we use either hierarchal or $\mathcal{K}$-means algorithms from the field of cluster analysis; for a review of the literature on data clustering see, e.g., Everitt, Landau and Leese (2001). With the help of clustering algorithms, we are able to replace a large number of closely-located simulated points with a given (small) number of uniformly distributed "representative" points. The clustering algorithms we use are relatively inexpensive even in high-dimensional applications. For example, it takes us about a minute to construct a grid of 300 clusters using a panel of 10000 observations for an economy with 200 heterogeneous agents (400 state variables).

We design our cluster-grid projection method to make it feasible for high-dimensional applications: First, we parameterize the decision rules using additively-separable polynomial functions which enable us to implement the approximation step with fast and numerically stable linear regression methods. Second, we evaluate the conditional expectations using low-cost numerical integration formulas that are particularly suitable for high-dimensional applications. Third, we solve for the polynomial coefficients using a fixed-point iteration procedure whose cost does not considerably increase with the dimensionality. Finally, we propose a cheap way of initializing the cluster-grid method, which is to apply the cluster-grid method itself to an arbitrary set of points and to use the obtained solution for constructing the ergodic set.[3]

Nevertheless, the above design does not preclude the cost of the cluster-grid method from growing with dimension: First, the approximation step becomes more expensive (because the number of polynomial terms in the approximating polynomial function increases and so does the number of grid points necessary for identifying the polynomial coefficients), and second, the integration step becomes more expensive (because the number of nodes for computing the conditional expectations in each grid point increases). To make our method cost-efficient, we identify the combinations of the approximation and integration strategies that match each other in terms of the over-

[3] An alternative way of initializing the cluster-grid method is to infer the ergodic set from a solution delivered by other methods such as log-linearization and stochastic simulation methods.

all accuracy of solutions (we consider approximating polynomial functions of different degrees, and we explore a variety of alternative integration methods such as the Gauss-Hermite quadrature tensor-product rule, non-product monomial rules and Monte Carlo simulation).

We first analyze the role of the approximation and integration strategies in the method's accuracy in the context of a one-country model. (As a measure of accuracy, we use unit-free Euler equation errors in the ergodic set). We find that accurate solutions require both a sufficiently flexible approximating function and a sufficiently accurate integration method. If an integration rule is not sufficiently accurate, the maximum accuracy is achieved under some low-degree polynomial, and similarly, if an approximating function is not sufficiently flexible, a more accurate integration rule does not help increase accuracy. In particular, in our benchmark model, we have the following results: Under an accurate ten-node Gauss-Hermite integration rule, increasing a degree of the approximating polynomial from one to five decreases the errors from a size of 10^{-3} to that of 10^{-8}; under a low accurate Monte Carlo integration method, the minimum errors of size 10^{-4} are achieved under the second-degree polynomial; and all the integration methods considered lead to virtually the same errors of size 10^{-3} under a rigid first-degree polynomial.

We next study the version of the model with N countries. Under $N = 2$, we approximate the decision rules by polynomials of the first, second and third degrees; under $4 \leq N \leq 40$, we consider polynomials of the first and second degrees; and finally, under $N \geq 40$, we use polynomials of the first degree only. We consider five alternative integration strategies such as the Gauss-Hermite product rule with 3^N and 2^N nodes, the monomial formulas with $2N^2 + 1$ and $2N$ nodes and the Gauss-Hermite rule with one node.[4] All these integration strategies are feasible for the two-country model with the polynomial approximating functions up to degree 3. Under the second-degree polynomial, the above mentioned five integration strategies are feasible for models with up to $N = 6$, $N = 8$, $N = 12$, $N = 20$ and $N = 40$ countries, respectively. Under the first-degree polynomial, monomial formulas with $2N$ nodes and the Gauss-Hermite rule with one node are feasible for the models with up to $N = 100$ and $N = 200$ countries, respectively. The running time ranges from 30 seconds to 24 hours depending on the number of countries, as

[4]We do not consider the Monte Carlo integration approach in the multi-country case because it is not competitive in the context of our projection method. Under polynomials of degrees higher than one, it typically restricts the overall accuracy of solutions even with a long series of 10000 observations.

well as on the specific approximation and integration strategies considered. The solutions are sufficiently accurate: under the first-, second- and third-degree polynomial approximations and accurate integration methods, the maximum Euler equation errors in the ergodic set are typically smaller than 0.1%, 0.01% and 0.001%, respectively.

For the multi-country models, we observe the same regularities as for the one-country model. Our key finding is that under low-degree polynomials, a specific integration method used plays only a minor role in the overall accuracy of solutions. Specifically, under the first-degree polynomial approximation, all the integration methods considered lead to virtually the same accuracy. Under the second-degree polynomials, the Gauss-Hermite product rule with 3^N and 2^N nodes and the two monomial formulas considered lead to Euler equation errors which are identical up to the fourth digit, and the one-node Gauss-Hermite rule increases the Euler equations errors only by $5-10\%$ relative to more accurate integration formulas (an acceptable cost given that this formula allows us to advance from the 20-country to 40-country models). These regularities are robust to variations in the model's parameters such as the volatility and persistence of shocks and the degrees of agents' risk-aversion. Finally, we find that both accuracy and numerical stability of the cluster-grid method increase if the number of grid points is not exactly equal to the number of terms in the approximating polynomial function (this case is referred to as *collocation*) but is somewhat larger.

The rest of the paper is as follows: In Section 2, we discuss a relation of our numerical method to the literature. In Section 3, we describe the construction of our endogenous cluster grid. In Section 4, we formulate the model. In Section 5, we design a projection method. In Section 6, we discuss a variety of computational strategies that help us reduce cost in high-dimensional applications. In Section 7, we describe the methodology of our numerical study and present the numerical results. In Section 8, we provide final comments.

2 Relation to the literature

The cluster-grid method, developed in this paper, is similar to stochastic simulation methods of Fair and Taylor (1984), Den Haan and Marcet (1990), Rust (1996), Pakes and McGuire (2001), and Judd, Maliar and Maliar (2009) in that it computes a solution on the ergodic set. However, we differ from the

above literature in two respects: first, we use a cluster-grid representation of the ergodic set, which is more efficient than a set of original closely-located simulated points, and second, we use numerical integration methods that are unrelated to the estimated density function and are more accurate than the Monte Carlo integration method.

Furthermore, our algorithm is similar to Smolyak's sparse grid method, developed in Krueger and Kubler (2004), in that both methods use non-product rules for ameliorating costs in high-dimensional applications. However, we differ from Smolyak's method in placement of grid points: our grid is endogenous while Smolyak's grid is exogenous. If a polynomial approximation and an integration formula are the same in the two methods, the cluster-grid method would typically have an advantage (disadvantage) in accuracy over Smolyak's method inside (outside) the ergodic set. This is because we fit a polynomial directly in the ergodic set, while Smolyak's algorithm fits a polynomial in a larger hypercube domain and faces a trade-off between the fit inside and outside the ergodic set.

Finally, our projection method is comparable to perturbation methods in its ability to solve models with a large number of agents.[5] However, our solutions are aimed to be accurate on the ergodic set, whereas solutions produced by perturbation methods are accurate in a neighborhood of steady state, which is much smaller than the ergodic set. As a consequence, accuracy of our global solutions does not decline rapidly away from steady state as does accuracy of local solutions obtained by perturbation methods.[6]

Large-scale heterogeneous-agent models are studied, for example, using perturbation and projection methods in Gaspar and Judd (1997) and using a stochastic simulation version of the parameterized expectations algorithm in Den Haan (1996).[7] A distinctive feature of the present paper is that we focus on much higher dimensions than those considered in the literature, namely, we compute global solutions to models with up to 400 state vari-

[5] Perturbation methods are studied in, e.g., Judd and Guu (1993), Gaspar and Judd (1997), Collard and Juillard (2001), and Kim, Kim, Schaumburg and Sims (2008). A collection of perturbation routines "DYNARE" is publicly available and can be adopted to individual applications; see http://www.dynare.org.

[6] Accuracy of perturbation methods is assessed in, e.g., Judd and Guu (1993).

[7] There is a variety of numerical methods for solving dynamic economic models; see Taylor and Uhlig (1990), Gaspar and Judd (1997), Judd (1998), Marimon and Scott (1999), Santos (1999), Christiano and Fisher (2000), Aruoba, Fernandez-Villaverde and Rubio-Ramirez (2006). However, most of the existing methods are not feasible for models with a large number of state variables due to their high computational expense.

ables. Furthermore, we carry out a systematic comparison of alternative approximation and integration strategies differing in accuracy and cost. Finally, we investigate the properties of solutions in a wide range of the model's parameters including the volatility and persistence of shocks and the degrees of a consumer's risk aversion.[8]

3 Cluster grid

Because we want our projection method to operate on the ergodic set, we should first construct a grid of points that covers the ergodic set. The simplest possible grid of this kind can be obtained by simulation. We can use either all or some of the simulated points as a grid for the projection method. A more efficient approach advocated in this paper is to replace the simulated points with a relatively small number of appropriately chosen "representative" points. In this section, we describe how to implement this approach using clustering algorithms. Such algorithms assign a set of observations into groups called *clusters* so that observations within each cluster are more similar to one another than observations belonging to different clusters. We then represent observations in each cluster with one point computed as the average of all observations in the given cluster. We call the constructed representative points a *cluster grid*.

3.1 Distance measure and data normalization

Any clustering algorithm requires measuring distance between observations. As a measure of distance between observations i and j, denoted d_{ij}, we use the Euclidean (or L_2 norm) distance

$$d_{ij} = \left[\sum_{\ell=1}^{L} \left(x_i^\ell - x_j^\ell \right)^2 \right]^{1/2}, \tag{1}$$

where x_i^ℓ is an i-th observation (object) on a variable $\ell \in \{1, ..., L\}$.

[8] A companion paper by Maliar, Maliar and Judd (2010) implements a further test of the cluster-grid method using a collection of 30 multi-country real-business cycle models with up to 20 state variables. These models are proposed by Den Haan, Judd and Juillard (2010) in the context of a JEDC project. Using a testsuite developed by Juillard and Villemot (2010), Kollmann, Maliar, Malin and Pichler (2010) compare the performance of six different numerical methods contributed to the JEDC project.

Before constructing clusters, we need to represent the ergodic set in a form which is suitable for clustering under the Euclidean distance. As an example, in Figure 1a, we plot the ergodic set of the standard one-country neoclassical growth model with a closed-form solution using 10000 simulated points (see Section 3 for a description of this model). As is seen from the figure, two state variables, which are capital and technology shock, have different ranges of values, and are highly correlated. Both measurement units of variables and correlation between variables affect the distance measures and hence, the outcome of a clustering procedure.

To transform variables into comparable measurement units, it is sufficient to normalize variables to zero mean and unit variance; the normalized ergodic set is shown in Figure 1b. To remove the correlation between variables, we can use a principal components (PCs) transformation. To be specific, let $X = \left(\mathbf{x}^1, ..., \mathbf{x}^L\right) \in \mathbb{R}^{T \times L}$ be a matrix of L variables which are normalized to zero mean and unit variance, where $\mathbb{R} \subseteq (-\infty, \infty)$ and $x^\ell \in \mathbb{R}^{T \times 1}$, $\ell = 1, ..., L$. Consider the eigenvalue decomposition $X'X = V \Lambda V'$, where $\Lambda \in \mathbb{R}^{L \times L}$ is a diagonal matrix with diagonal entries $\lambda^1 \geq \lambda^2 \geq ... \geq \lambda^L \geq 0$ being eigenvalues, and $V \in \mathbb{R}^{L \times L}$ is an orthogonal matrix of eigenvectors. Perform the following linear transformation of X: $Z \equiv XV$, where $Z \in \mathbb{R}^{T \times L}$.[9] The variables $\mathbf{z}^1, ..., \mathbf{z}^L$ are called *principal components* of X, and are orthogonal (uncorrelated), $\left(\mathbf{z}^\ell\right)' \mathbf{z}^\ell = \lambda^\ell$ and $\left(\mathbf{z}^j\right)' \mathbf{z}^\ell = 0$ for any $j \neq \ell$. As is seen from Figure 1c, switching to PCs rotates the original ergodic set so that the axes of the ellipse become parallel to the coordinate axes (a higher dimensional analogue of an ellipse is a hyperellipsoid). Since the obtained PCs have different ranges of values, we complete the transformation by normalizing PCs to unit variance. Figure 1d plots the resulting ergodic set that has the shape of a circle (a hypersphere in multi-dimensional space).

Figures 1a-1d give us an idea of how much we can save on cost in the two-dimensional case if we solve the model on the ergodic set instead of the standard squared domain containing the ergodic set [10] Our saving increases

[9] The PCs transformation can be equivalently defined using a singular value decomposition instead of the eigenvalue decomposition considered; see, e.g., Hastie, Tibshirani and Friedman (2009).

[10] In principle, we can solve models on domains, which are smaller than the ergodic set, namely, we can consider ranges of values for the individual state variables, which are smaller than the ergodic ranges. However, accuracy may decline rapidly when we deviate from the domain on which the problem is solved; this happens with the solutions obtained by perturbation methods.

with the dimensionality of the problem: for a model with p state variables, the ratio of a hypershere's volume Ω_p^s (with the hypershere representing the ergodic set) to a hypercube's volume Ω_p^c (with the hypercube representing the standard hypercube domain containing the ergodic set) can be estimated by:

$$\frac{\Omega_p^s}{\Omega_p^c} = \begin{cases} \frac{(\pi/2)^{\frac{p-1}{2}}}{1 \cdot 3 \cdot \ldots \cdot p} \text{ for } p = 1, 3, 5... \\ \\ \frac{(\pi/2)^{\frac{p}{2}}}{2 \cdot 4 \cdot \ldots \cdot p} \text{ for } p = 2, 4, 6... \end{cases} . \tag{2}$$

The ratio (2) declines very rapidly with p. For dimensions two, three, four, five, ten, thirty and one hundred, this ratio is 0.79, 0.52, 0.31, 0.16, $3 \cdot 10^{-3}$, $2 \cdot 10^{-14}$ and $2 \cdot 10^{-70}$, respectively.

3.2 Clustering algorithms

There exists a vast variety of clustering techniques in the field of cluster analysis; see, for example, Romesburg (1984), Everitt et al. (2001) for reviews. In this section, we describe two commonly used clustering algorithms, namely, hierarchical and $\mathcal{K}$-means, and we illustrate the construction of clusters by way of examples.

3.2.1 Hierarchical algorithm

A hierarchical algorithm creates a multi-level hierarchy of clusters in the form of a tree. The root corresponds to a single cluster that includes all observations and the leaves correspond to individual observations. Given a hierarchical tree, one can choose the most appropriate level of clustering for a given application. We consider an agglomerative type of hierarchical algorithm which begins from individual observations (leaves) and agglomerates them into larger clusters. The algorithm proceeds in the following steps: find the pairwise distances between the objects (observations) in the data; merge the closest two clusters into a single cluster and continue to group the newly created clusters into larger clusters until there is a single cluster that contains all objects; finally, cut off the obtained hierarchical tree at some point to obtain clusters.

In Section 3.1, we described how to measure a distance between individual observations. We shall now discuss how to measure a distance between groups of observations (clusters). The inter-cluster distance between clusters

A and B, denoted d_{AB}, can be computed from the set of pairwise distances between observations, d_{ij}, where one member of the pair i is in A and the other j is in B. The hierarchical algorithm can compute d_{AB} using alternative linkage algorithms. Single linkage (or the nearest neighbor) clustering uses the shortest distance between objects in the two clusters, $d_{AB} = \min_{i \in A,\ j \in B} d_{ij}$. Complete linkage (or the furthest neighbor) clustering uses the maximum distance between objects in the two clusters: $d_{AB} = \max_{i \in A,\ j \in B} d_{ij}$. Group average linkage clustering uses the average distance between all pairs of objects in the two clusters: $d_{AB} = \frac{1}{h_A h_B} \sum_{i \in A} \sum_{j \in B} d_{ij}$, where h_A and h_B is the number of objects in clusters A and B, respectively. Finally, Ward's linkage clustering uses the increase in the *sum of squared error*, SSE, or variance,

$$SSE_A \equiv \sum_{i \in A} \sum_{\ell=1}^{L} \left(x_i^\ell - \frac{1}{h_A} \sum_{i \in A} x_i^\ell \right)^2 ,$$

as a result of merging two clusters into a single cluster, $d_{AB} = SSE_{AB} - [SSE_A + SSE_B]$, where AB is a cluster obtained after merging A and B.

The following example illustrates the construction of clusters by the agglomerative hierarchical algorithm under single linkage clustering.

A numerical example Consider sample data that contains five observations for two variables, x_i^1 and x_i^2,

	Variable	
Observation i	x_i^1	x_i^2
1	1	0.5
2	2	3
3	0.5	0.5
4	3	1.6
5	3	1

These data, as well as the steps of the clustering construction, are shown in Figure 2. The Euclidean distance d_{ij} between two observations (objects) i and j with variables values (x_i^1, x_i^2) and $\left(x_j^1, x_j^2\right)$ is given by $d_{ij} = \left[\left(x_i^1 - x_j^1\right)^2 + \left(x_i^2 - x_j^2\right)^2 \right]^{1/2}$. Let us compute a matrix D_1 of the inter-

individual distances, in which each entry ij corresponds to the distance d_{ij},

$$D_1 = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ 1 & 0 & 2.7 & 0.5 & 2.3 & 2.1 \\ 2 & 2.7 & 0 & 2.9 & 1.7 & 2.2 \\ 3 & 0.5 & 2.9 & 0 & 2.7 & 2.5 \\ 4 & 2.3 & 1.7 & 2.7 & 0 & 0.6 \\ 5 & 2.1 & 2.2 & 2.5 & 0.6 & 0 \end{array}$$

The smallest non-zero distance for the five observations in D_1 is $d_{13} = 0.5$, so that we merge objects 1 and 3 into one cluster and call the obtained cluster "object 6". The distances for the four objects, namely, 6, 2, 4, and 5, are shown in a matrix D_2,

$$D_2 = \begin{array}{c|cccc} & 6 & 2 & 4 & 5 \\ 6 & 0 & 2.7 & 2.3 & 2.1 \\ 2 & 2.7 & 0 & 1.7 & 2.2 \\ 4 & 2.3 & 1.7 & 0 & 0.6 \\ 5 & 2.1 & 2.2 & 0.6 & 0 \end{array}$$

where, $d_{62} = \min\{d_{12}, d_{32}\}$, $d_{64} = \min\{d_{14}, d_{34}\}$, $d_{65} = \min\{d_{15}, d_{35}\}$. Given that d_{45} is the smallest non-zero entry in D_2, objects 4 and 5 should be merged into a new object (cluster) 7. The distances for three objects 6, 7 and 2 are given in D_3,

$$D_3 = \begin{array}{c|ccc} & 6 & 7 & 2 \\ 6 & 0 & 2.1 & 2.7 \\ 7 & 2.1 & 0 & 1.7 \\ 2 & 2.7 & 1.7 & 0 \end{array}$$

where $d_{67} = \min\{d_{14}, d_{15}, d_{34}, d_{35}\}$, $d_{62} = \min\{d_{12}, d_{32}\}$, $d_{72} = \min\{d_{42}, d_{52}\}$. The smallest non-zero distance in D_3 is $d_{72} = 1.7$. Hence, objects 2 and 7 should be merged into object 8. The only two objects left unmerged are 6 and 8, so that the last step is to merge those two to obtain object 9 containing all the observations. In sum, after progressive merging of clusters, we obtain

the following hierarchical tree:

Cluster created	Clusters merged		Shortest distance
6	1	3	0.5
7	4	5	0.6
8	2	7	1.7
9	6	8	2.1

For example, if we want to group the observations into three clusters, we obtain the clusters: $\{1, 3\}$; $\{4, 5\}$; $\{2\}$. ‖

If clusters are well-defined, then all the above linkage clustering algorithms produce similar cluster trees; otherwise, they can produce different trees. Single linkage tends to suffer from a phenomenon called *chaining* in which well separated clusters are joined together if there are outliers in between them. Complete linkage tends to deliver compact clusters with small diameters, however, it can find clusters in which observations are much closer to some observations of other clusters than to some observations of their own cluster. Group average linkage tends to merge clusters with a small variance and is relatively robust, however, its results are not invariant to monotone transformations to d_{ij}. Finally, Ward's linkage tends to produce spherical clusters of the same size; it has been observed to work well in many applications, however, it can deliver a spherical structure where such a structure does not exist. For a detailed discussion of the strengths and weaknesses of the above linkage clustering algorithms, see, e.g., Everitt et al. (2001).

The agglomerative hierarchical algorithm is relatively expensive in terms of computational time and memory. Another weakness is that a decision about combining two clusters is final and cannot be undone on a later stage. These weaknesses of the hierarchical algorithm are addressed by partitional class of algorithms.

3.2.2 $\mathcal{K}$-means algorithm

A well-known member of the partitional class is a $\mathcal{K}$-means clustering algorithm: it obtains a single partition of data instead of a cluster tree generated by a hierarchical algorithm. The algorithm starts with $\mathcal{K}$ random clusters, and then moves objects between those clusters with the goal to minimize variability within clusters and to maximize variability between clusters. The

basic $\mathcal{K}$-means algorithm proceeds as follows: choose the number of clusters, $\mathcal{K}$; generate randomly $\mathcal{K}$ clusters and determine the centers of the clusters; given a current set of centers, assign each observation to the nearest cluster center and re-compute the centers of the new clusters; iterate on the last step until convergence.[11] A frequently used criterion function in the $\mathcal{K}$-means algorithm is the SSE function,

$$SSE = \sum_{\kappa=1}^{\mathcal{K}} \sum_{i \in \kappa} \sum_{\ell=1}^{L} \left(x_i^{\ell} - \overline{x}^{\ell,\kappa} \right)^2,$$

where x_i^{ℓ} is the ℓ-th coordinate of a point in the κ-th cluster, and $\overline{x}^{\ell,\kappa}$ is the ℓ-th coordinate of a center of the κ-th cluster; see Hastie et al. (2009, p. 510). The center minimizing SSE of a cluster is given by the mean of the points in the cluster, $\overline{x}^{\ell,\kappa} \equiv \frac{1}{h_\kappa} \sum_{i \in \kappa} x_i^{\ell}$, with h_κ being the number of objects in the κ-th cluster.

The $\mathcal{K}$-means algorithm is relatively efficient and computationally cheap which makes it suitable for large data sets. However, it has an important shortcoming, namely, it can give different results with each run. This is because the $\mathcal{K}$-means algorithm is sensitive to initial random assignments of observations into clusters and can converge to a suboptimal local minimum.

3.2.3 Clusters for the model with a closed-form solution

We use the hierarchical algorithm with Ward's linkage clustering to construct clusters for the ergodic set shown in Figures 1a-1d. The results under the $\mathcal{K}$-means algorithm are similar (not reported). In Figures 3a-3d, we plot clusters obtained for the normalized ergodic set shown in Figure 1b (the circles around clusters' centers are drawn for expository convenience and show approximately a set of points that each cluster represents). When the number of clusters is small (see Figures 3a and 3b), all clusters are situated along the direction of the largest principle component, while the direction of the smallest principle component is not represented. When the number of clusters increases (see Figures 3c and 3d), the clusters are distributed over the ergodic set more uniformly. In Figures 4a-4d, we construct clusters on the normalized PCs of the ergodic set shown in Figure 1d. In this case, both

[11] Note that for this clustering algorithm, we need not compute a distance between any two observations but the one between an observation and a cluster center.

PC directions are represented equally well, and the clusters cover the ergodic set uniformly even if the number of clusters is small. In Figure 4d, we draw attention to a cluster which is considerably separated from the rest of the clusters (see the upper right part of the figure). The ability of the clustering algorithm to identify clusters outside of high-probability areas of state space is especially valuable in applications in which ergodic sets do not have a regular ellipsoid form or are possibly composed of disjoined sets. Given that we obtain a more uniform coverage of the ergodic set when using PCs than when using the original data, in the rest of the paper, we apply the clustering algorithm to PCs.

4 The model

In this section, we describe the model that we use for a presentation of our projection method. There is a finite number of countries, N, and each country is populated by a representative consumer. A social planner maximizes a weighted sum of expected lifetime utility functions of the countries' representative consumers,

$$\max_{\left\{\left\{c_t^n, k_{t+1}^n\right\}_{n=1}^N\right\}_{t=0}^\infty} E_0 \sum_{n=1}^N v^n \left(\sum_{t=0}^\infty \delta^t u^n \left(c_t^n \right) \right) \tag{3}$$

subject to the aggregate resource constraint,

$$\sum_{n=1}^N c_t^n + \sum_{n=1}^N k_{t+1}^n = \sum_{n=1}^N k_t^n \left(1-d\right) + \sum_{n=1}^N \theta_t^n A f^n \left(k_t^n \right), \tag{4}$$

where E_t is the operator of conditional expectation; c_t^n, k_t^n, θ_t^n and v^n are, respectively, consumption, capital, technology shock and welfare weight of a country $n \in \{1, ..., N\}$; $\delta \in (0,1)$ is the discount factor; $d \in (0,1]$ is the depreciation rate; A is the level of technology. Initial condition $(\mathbf{k}_0, \boldsymbol{\theta}_0)$ is given, where $\mathbf{k}_0 \equiv \left(k_0^1, ..., k_0^N\right)$ and $\boldsymbol{\theta}_0 \equiv \left(\theta_0^1, ..., \theta_0^N\right)$. The utility and production functions, u^n and f^n, respectively, are increasing, concave and continuously differentiable. The process for technology shocks of country n is given by

$$\ln \theta_t^n = \rho \ln \theta_{t-1}^n + \varepsilon_t^n, \tag{5}$$

where $\rho \in (-1,1)$ is the autocorrelation coefficient; and $\varepsilon_t^n \equiv \varepsilon_t + \xi_t^n$ with $\varepsilon_t \sim \mathcal{N}(0,\sigma^2)$ being identical for all countries and with $\xi_t^n \sim \mathcal{N}(0,\sigma^2)$ being country-specific.

We restrict our attention to the case in which the countries are characterized by identical preferences, $u^n = u$, and identical production technologies, $f^n = f$, for all n. The former implies that the planner assigns identical weights, $v^n = 1$, and consequently, identical consumption $c_t^n = C_t/N \equiv c_t$ to all agents, where C_t denotes aggregate consumption. If an interior solution exists, it satisfies N Euler equations,

$$u'(c_t) = \delta E_t \left\{ u'(c_{t+1}) \left[1 - d + \theta_{t+1}^n A f'\left(k_{t+1}^n\right) \right] \right\}, \tag{6}$$

where u' and f' denote the first-order derivatives of u and f, respectively. Thus, the planner's solution is determined by process for shocks (5), resource constraint (4) and the set of Euler equations (6).

The standard representative-agent neoclassical growth model can be obtained as a one-country version of heterogeneous-agent economy $(3)-(5)$ by assuming that $N = 1$ and that $\varepsilon_t = 0$ for all t. Under the assumptions of a logarithmic utility function, $u(c) = \ln(c)$, full depreciation of capital, $d = 1$, and the Cobb-Douglas production function, $f(k) = k^\alpha$, the representative-agent model admits a closed-form solution $k_{t+1} = \alpha\beta\theta_t A k_t^\alpha$ (here and further in the text, we omit a country's superscript in the one-country case). This closed-form solution was used for simulating times series and for constructing the ergodic set shown in Figures 1, 3 and 4.

5 A projection method

Our objective is to find N decision rules for capital, $k_{t+1}^n = K^n(\mathbf{k}_t, \boldsymbol{\theta}_t)$, where $\mathbf{k}_t \equiv \left(k_t^1, ..., k_t^N\right)$ and $\boldsymbol{\theta}_t \equiv \left(\theta_t^1, ..., \theta_t^N\right)$ are t-period vectors of state variables. Given that the countries are identical in their fundamentals (preferences and technology), the optimal decision rules are identical for all countries. Even though we could use this fact to simplify the solution procedure, we will not do so and will compute a separate decision rule for each country, thus, treating the countries as completely heterogeneous. This strategy allows us to evaluate the cost of finding solutions in general multi-dimensional settings with heterogeneous preferences and technology.

A projection method approximates a solution in a finite number of specified points; see Chapters 11 and 17 in Judd (1998) for a detailed review of

the projection methods. To solve the model described in Section 3, we parameterize the capital decision rules of each country, $k_{t+1}^n = K^n(\mathbf{k}_t, \boldsymbol{\theta}_t)$ with a flexible functional form $\Psi^n(\mathbf{k}_t, \boldsymbol{\theta}_t; \boldsymbol{\beta}^n)$ depending on a vector of coefficients $\boldsymbol{\beta}^n$. We then re-write Euler equation (6) as

$$k_{t+1}^n = E_t\left\{\delta\frac{u'(c_{t+1})}{u'(c_t)}\left[1-d+\theta_{t+1}^n A f'\left(k_{t+1}^n\right)\right]k_{t+1}^n\right\} \simeq \Psi^n(\mathbf{k}_t, \boldsymbol{\theta}_t; \boldsymbol{\beta}^n), \quad (7)$$

For each country $n \in \{1, ..., N\}$, we need to compute a vector $\boldsymbol{\beta}^n$ so that $\Psi^n(\mathbf{k}_t, \boldsymbol{\theta}_t; \boldsymbol{\beta}^n)$ is the best possible approximation of $K^n(\mathbf{k}_t, \boldsymbol{\theta}_t)$ on some domain given the functional form Ψ^n. Note that using N assumed decision rules $k_{t+1}^n = \Psi^n(\mathbf{k}_t, \boldsymbol{\theta}_t; \boldsymbol{\beta}^n)$, resource constraint (4) and process for shock (5), we can express the variable inside the conditional expectation in (7) as a function of the current-period state variables, $\mathbf{k}_t$ and $\boldsymbol{\theta}_t$, and the next-period errors, $\boldsymbol{\varepsilon}_{t+1} \equiv \left(\varepsilon_{t+1}^1, ..., \varepsilon_{t+1}^N\right)$,

$$\delta\frac{u'(c_{t+1})}{u'(c_t)}\left[1-d+(\theta_t^n)^\rho \exp\left(\varepsilon_{t+1}^n\right) A f'\left(k_{t+1}^n\right)\right]k_{t+1}^n \equiv \Gamma^n(\mathbf{k}_t, \boldsymbol{\theta}_t, \boldsymbol{\varepsilon}_{t+1}), \quad (8)$$

where

$$\begin{aligned} c_t &= \frac{1}{N}\sum_{n=1}^N \left[k_t^n(1-d) + \theta_t^n A f(k_t^n) - k_{t+1}^n\right], \\ c_{t+1} &= \frac{1}{N}\sum_{n=1}^N \left[k_{t+1}^n(1-d) + (\theta_t^n)^\rho \exp\left(\varepsilon_{t+1}^n\right) A f\left(k_{t+1}^n\right) - k_{t+2}^n\right], \end{aligned}$$

and $k_{t+2}^n = \Psi^n(\mathbf{k}_{t+1}, \boldsymbol{\theta}_{t+1}; \boldsymbol{\beta}^n)$ with $\mathbf{k}_{t+1} = \left(\Psi^1\left(\mathbf{k}_t, \boldsymbol{\theta}_t; \boldsymbol{\beta}^1\right), ..., \Psi^N\left(\mathbf{k}_t, \boldsymbol{\theta}_t; \boldsymbol{\beta}^N\right)\right)$ and $\boldsymbol{\theta}_{t+1} = \left[\left(\theta_t^1\right)^\rho \exp\left(\varepsilon_{t+1}^1\right), ..., \left(\theta_t^N\right)^\rho \exp\left(\varepsilon_{t+1}^N\right)\right]$.

We can compute the capital decision rules as follows:

- Step 1. Choose a grid of points of state variables $\{\mathbf{k}_i, \boldsymbol{\theta}_i\}_{i=1}^I$.
- Step 2. Consider a country n. Fix some vectors of the coefficients $\boldsymbol{\beta}^n$ and find the next period capital stock in all grid points using the assumed decision rule for capital, i.e., $\widetilde{k}_i^n \equiv \Psi^n(\mathbf{k}_i, \boldsymbol{\theta}_i; \boldsymbol{\beta}^n)$ for all $i = 1, ..., I$.

- Step 3. Using some numerical integration method, approximate the conditional expectations in each of I grid points and call the resulting capital stock $\widehat{k}_i^n$, i.e.,

$$\widehat{k}_i^n \equiv E\left\{\Gamma^n\left(\mathbf{k}_i, \boldsymbol{\theta}_i, \boldsymbol{\varepsilon}\right)\right\}. \tag{9}$$

  Here, the expectation is computed with respect to a vector of normally distributed random variables $\boldsymbol{\varepsilon} \equiv \left(\varepsilon^1, ..., \varepsilon^N\right)$.

- Step 4. Run a least-squares (LS) regression of response variables $\widehat{k}_i^n$ on the functional form $\Psi^n\left(\mathbf{k}_i, \boldsymbol{\theta}_i; \boldsymbol{\beta}^n\right)$,

$$\widehat{\boldsymbol{\beta}}^n \equiv \arg\min_{\boldsymbol{\beta}^n} \sum_{i=1}^{I} \left(\widehat{k}_i^n - \Psi^n\left(\mathbf{k}_i, \boldsymbol{\theta}_i; \boldsymbol{\beta}^n\right)\right)^2. \tag{10}$$

  Repeat Steps 2-4 for all N countries.

- Step 5. Compute the coefficients for the next iteration $\left\{\widehat{\widehat{\boldsymbol{\beta}}}^n\right\}_{n=1}^{N}$ as

$$\widehat{\widehat{\boldsymbol{\beta}}}^n = (1-\mu)\,\boldsymbol{\beta}^n + \mu\widehat{\boldsymbol{\beta}}^n, \tag{11}$$

  where $\mu \in (0, 1]$ is a dampening parameter.

- Step 6. Check the convergence criterion, which is the average relative difference between capital values on the grid before and after the iteration, $\widetilde{k}_i^n$ and $\widehat{k}_i^n$, respectively, i.e.,

$$\frac{1}{I \cdot N} \sum_{i=1}^{I} \sum_{n=1}^{N} \left| \frac{\widetilde{k}_i^n - \widehat{k}_i^n}{\widetilde{k}_i^n} \right| < 10^{-\vartheta}, \tag{12}$$

  where $\vartheta > 0$. If criterion (12) is not satisfied, go to Step 2.

The dampening procedure we use in (11) is called *fixed-point iteration*. Fixed-point iterations can be unstable; see Judd (1998, p.557) for a discussion. Setting dampening parameter μ to a small value stabilizes fixed-point iteration though it reduces the speed of convergence.

Projection methods are also referred to in the literature as *weighted residuals methods* because they minimize a weighted sum of residuals in specified points. Consider the residuals on the grid, $\widehat{k}_i^n - \Psi^n\left(\mathbf{k}_i, \boldsymbol{\theta}_i; \widehat{\boldsymbol{\beta}}^n\right)$, $i = 1, ..., I$, obtained in a country's n LS solution to (10). If the number of grid points I is the same as the number of coefficients in $\boldsymbol{\beta}^n$ (collocation), the system in (10) has I equations and I unknowns, and the LS regression leads to zero residuals in all grid points $i = 1, ..., I$ (assuming that the LS problem is well-conditioned). If the number of grid points I is larger than the number of coefficients in $\boldsymbol{\beta}^n$, the system in (10) has more equations than unknowns (i.e., it is overdetermined), and the LS regression minimizes a squared sum of equally weighted residuals on the given grid.

6 Strategies for high-dimensional problems

In the description of our projection method, we have not specified how to choose an approximating function for the capital decision rule, a grid of points on which the solution is computed and a numerical integration method for approximating the conditional expectations. These three choices play a determinant role in the accuracy and speed of our method in the context of high-dimensional problems, and they are discussed in Sections 5.1, 5.2 and 5.3, respectively.

6.1 Approximation step

To approximate the decision rules, we use a complete set of ordinary polynomials.[12] For a one-country model, the polynomial function of degree two is given by

$$\Psi\left(k_t, \theta_t; \beta\right) = \beta_0 + \beta_1 k_t + \beta_2 \theta_t + \beta_3 k_t^2 + \beta_4 k_t \theta_t + \beta_5 \theta_t^2. \tag{13}$$

The polynomial functions of degrees one, two, three, four and five have 3, 6, 10, 15 and 21 coefficients, respectively.

[12] In some numerical methods, the choice of a polynomial representation is related to the choice of grid points. For example, the Galerkin method relies on Chebyshev polynomials and uses zeros of such polynomials as an optimal grid; see Judd (1992). Our grids are not related to any specific polynomial family. It is an open question which family of approximating functions is the most appropriate for our endogenous grids.

For an N-country model, the number of coefficients in the m-th degree polynomial, $P_m(N)$, that approximates the capital decision rule of each country is given by

$$\begin{aligned} P_1(N) &= 1+2N, \\ P_2(N) &= 1+2N+N(2N+1), \\ P_3(N) &= 1+2N+N(2N+1)+4N^2+\frac{N(2N-1)(2N-2)}{3}. \end{aligned}$$

We illustrate how the number of polynomial coefficients increases with N in Table 1 (panel 1). For example, when $N = 100$, the number of terms in the first-, second- and third-degree polynomials is equal to 201, 20301, and 1373701, respectively. Since the decision rules should be computed for each country, the total number of polynomial coefficients in the $N-$country case is $N \cdot P_m(N)$.

Given that the number of polynomial terms and, hence, the number of polynomial coefficients grows rapidly with the dimensionality, it is important to design a procedure for finding fixed-point values of the coefficients so that its cost does not considerably increase with the number of polynomial terms (coefficients). We achieve this objective by making two choices. First, we employ a linearly additive polynomial representation like the one in (13); this allows us to perform the regression in Step 4 using fast and reliable linear approximation methods whose cost is low even in high-dimensional applications. Second, in Step 5, we use fixed-point iteration (11) whose cost does not considerably increase with the dimensionality of the problem (even though fixed-point iteration is generically slow when a small μ is required for convergence).[13]

6.2 Three alternative grids

We now describe three alternative grids to be evaluated for our projection method: the standard tensor-product grid, a grid composed from selected

[13]There are other iterative schemes for finding fixed-point coefficients such as time iteration and (quasi-)Newton methods; see Judd (1998, p. 553-558) and Judd (1998, p. 103-119), respectively. Time iteration can be more stable than fixed-point iteration, however, it requires solving costly nonlinear equations for finding future values of variables. (Quasi-)Newton methods can be faster, especially for low-dimensional problems, but can become expensive when the dimensionality increases. Gaspar and Judd (1997) argue that fixed-point iteration (referred to in their paper as successive approximation) has advantages over time iteration and dominates Newton methods for large-scale models.

simulated points and a grid constructed by clustering simulated data.

6.2.1 Tensor-product grid

In low-dimensional problems, we can construct a grid as a tensor (or Cartesian) product of state variables discretized on a relevant area of state space. For example, in a one-country version of our model, a tensor-product grid can be obtained by dividing some intervals for capital, $\left[\underline{k}, \overline{k}\right]$, and shock, $\left[\underline{\theta}, \overline{\theta}\right]$, into I_k and I_θ points, respectively, and by considering all possible pairs from $\left[\underline{k}, \overline{k}\right] \times \left[\underline{\theta}, \overline{\theta}\right]$ (the limits $\underline{k}$, $\overline{k}$, $\underline{\theta}$, $\overline{\theta}$ can be estimated by simulation). A version of our projection method that uses a tensor-product grid is referred to in the paper as *tensor-product-grid algorithm* (TPGA). The TPGA is not feasible in high-dimensional applications since the number of grid points and, hence, computational expense grows exponentially with the dimensionality of state space.

6.2.2 Simulation grid

As is argued in Section 3, we can construct a grid that covers the ergodic set using a set of points obtained by simulation. On one hand, simulated series should be long enough to accurately approximate the ergodic set. On the other hand, long simulated series have many points that are very close to each other. The redundancy in grid points increases cost without increasing accuracy which makes the projection method cost-inefficient.

A more efficient grid can be constructed by using simulated points which are separated by a sufficient time interval. To be specific, if we have a simulation of a length T, and we want to select I points to be used as a grid, we can take points separated by a time interval $\frac{T}{I}$ (rounded to the nearest smaller integer if necessary). Our grid points are therefore the simulated points taken in periods $t = \frac{T}{I}$, $\frac{2T}{I}$, ..., $\frac{(I-1)T}{I}$, T. In expected terms, points separated in time are more distant from each other and cover the ergodic set more uniformly than the original subsequently following simulated points. We call a version of the projection method using a simulation grid a *simulation-grid algorithm* (SGA).

6.2.3 Cluster grid

To construct a cluster grid, we use the hierarchical algorithm with Ward's linkage clustering as described in Section 3.2. An important practical ques-

tion is: How costly is the clustering process, namely, how does the cost of constructing clusters depend on the dimensionality of the problem, N, the number of observations, T, and the number of clusters to be constructed, $\mathcal{K}$? In Table 1 (panel 2), we report time necessary for constructing $\mathcal{K} = 3$, 30, 300 clusters under three alternative lengths of simulation, $T = 1000$, 3000, 10000 with the number of countries ranging from $N = 1$ to $N = 200$ (we use time series solutions of our benchmark model with partial depreciation of capital; see Section 7.1 for a description of our benchmark parameterization of the models).

As is seen from the table, time for constructing clusters depends primarily on the length of simulated series, T. An increase in T by one order of magnitude (from 1000 to 10000) raises the clustering time by about two orders of magnitude. In turn, an increase in N by two orders of magnitude from $N = 1$ to $N = 100$ countries only triplicates the clustering time. The largest time for constructing clusters is observed in the model with $N = 200$ and $T = 10000$ and is around one minute, which is not much time especially taking into account that we use a standard desktop computer. Given that clusters should be created only once (or sometimes twice, see Section 7.1) on the initial step of the projection method, we do not explore possibilities for reducing time for constructing clusters.

6.3 Numerical integration

Each iteration of our projection method requires computing the conditional expectation (9) for all grid points and all countries. Let us fix a country n and a grid point $\mathbf{k}_i$, $\boldsymbol{\theta}_i$, and let us denote the resulting function inside expectation (9) by $g(\boldsymbol{\varepsilon}) \equiv \Gamma^n(\mathbf{k}_i, \boldsymbol{\theta}_i, \boldsymbol{\varepsilon})$. In our model, $\boldsymbol{\varepsilon}$ follows a multivariate normal distribution, $\boldsymbol{\varepsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\boldsymbol{\mu}$ is an $N \times 1$ vector of means and Σ is an $N \times N$ variance-covariance matrix. To compute $E\{g(\boldsymbol{\varepsilon})\}$, we should evaluate a definite integral $\int_{\mathbb{R}^N} g(\boldsymbol{\varepsilon}) w(\boldsymbol{\varepsilon}) d\boldsymbol{\varepsilon}$ where the weighting function $w(\boldsymbol{\varepsilon})$ is a density function of the multivariate normal distribution,

$$w(\boldsymbol{\varepsilon}) = (2\pi)^{-N/2} \det(\Sigma)^{-1/2} \exp\left[-\frac{1}{2}(\boldsymbol{\varepsilon} - \boldsymbol{\mu})' \Sigma^{-1} (\boldsymbol{\varepsilon} - \boldsymbol{\mu})\right], \tag{14}$$

with $\det(\Sigma)$ denoting the determinant of Σ. For the model studied in the paper, $\boldsymbol{\mu} = (0, ..., 0)'$, and Σ has the diagonal terms equal to $2\sigma^2$ and the off-diagonal terms equal to σ^2 (the off-diagonal terms of Σ are non-zero because individual and aggregate shocks are correlated by assumption).

Numerical integration consists in approximating integrals by a weighted sum of values that the integrand, g, takes in a finite set of nodes, i.e.,

$$\int_{\mathbb{R}^N} g(\boldsymbol{\varepsilon})\, w(\boldsymbol{\varepsilon})\, d\boldsymbol{\varepsilon} \approx \sum_{j=1}^{J} \omega_j g(\boldsymbol{\varepsilon}_j), \tag{15}$$

where $\{\boldsymbol{\varepsilon}_j\}_{j=1}^J$ and $\{\omega_j\}_{j=1}^J$ are integration nodes and integration weights, respectively. Integration formulas differ in their choices of integration nodes and weights. Typically, there is a trade off between accuracy and cost: integration formulas with more nodes (and thus, a higher cost) lead to more accurate approximations.

The existing numerical integration formulas are constructed under the assumption of uncorrelated variables with zero mean and unit variance. Therefore, we should re-write the integral in (15) in terms of such uncorrelated variables prior to numerical integration. Given that Σ is symmetric and positive-definite, it has a Cholesky decomposition, $\Sigma = \Omega\Omega'$, where Ω is a lower triangular matrix with strictly positive diagonal entries. The Cholesky decomposition of Σ allows us to transform correlated variables in vector $\boldsymbol{\varepsilon}$ into uncorrelated ones in $\mathbf{y}$ with the following linear change of variables:

$$\mathbf{y} = \frac{\Omega^{-1}(\boldsymbol{\varepsilon} - \boldsymbol{\mu})}{\sqrt{2}}. \tag{16}$$

Note that $d\boldsymbol{\varepsilon} = \left(\sqrt{2}\right)^N \det(\Omega)\, d\mathbf{y}$. Using (16) and taking into account that $\Sigma^{-1} = (\Omega^{-1})'\,\Omega^{-1}$ and that $\det(\Sigma) = [\det(\Omega)]^2$, we obtain

$$E\{g(\boldsymbol{\varepsilon})\} = \pi^{-N/2} \int_{\mathbb{R}^N} g\left(\sqrt{2}\Omega\mathbf{y} + \boldsymbol{\mu}\right) \exp(-\mathbf{y}'\mathbf{y})\, d\mathbf{y}. \tag{17}$$

All subsequent numerical integration formulas are derived on the basis of (17) under $\boldsymbol{\mu} = (0, ..., 0)'$.

6.3.1 Gauss-Hermite quadrature

In a one-dimensional case, $N = 1$, definite integrals can be accurately approximated with the Gaussian quadrature approach. Under this approach, nodes and weights are chosen so that the approximation is exact if the function g is a polynomial of a certain degree. For the integral in (17), the Gaussian

quadrature approach leads to the Gauss-Hermite quadrature rule,

$$E\left\{g\left(\varepsilon\right)\right\} = \pi^{-1/2}\int_{-\infty}^{\infty} g\left(\sqrt{2}\sigma y\right)\exp\left(-y^2\right)dy$$
$$\approx \pi^{-1/2}\sum_{j=1}^{J}\omega_j g\left(\sqrt{2}\sigma y_j\right), \quad (18)$$

where the values of the Gauss-Hermite quadrature nodes $\{y_j\}_{j=1}^{J}$ and weights $\{\omega_j\}_{j=1}^{J}$ are provided in tables; see, e.g., Judd (1998, p. 262).

We can extend one-dimensional Gauss-Hermite quadrature rule to the case of multi-dimensional integral in (17) by way of a tensor-product rule:

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\} = \pi^{-N/2}\int_{\mathbb{R}^N} g\left(\sqrt{2}\Omega\mathbf{y}\right)\exp\left(-\mathbf{y}'\mathbf{y}\right)d\mathbf{y} \approx$$
$$\approx \pi^{-N/2}\sum_{j_1=1}^{J_1}...\sum_{j_N=1}^{J_N}\omega_{j_1}^1\omega_{j_2}^2\cdots\omega_{j_N}^N \cdot g\left(\sqrt{2}\Omega\cdot\left(y_{j_1}^1,...,y_{j_N}^N\right)'\right), \quad (19)$$

where $\left\{\omega_{j_n}^n\right\}_{j_n=1}^{J_n}$ and $\left\{y_{j_n}^n\right\}_{j_n=1}^{J_n}$ are, respectively, weights and nodes in a dimension n obtained from one-dimensional Gauss-Hermite quadrature rule (note that in general, the number of nodes in one dimension, J_n, can differ across dimensions). The total number of nodes is given by the product $J_1J_2\cdots J_N$. Assuming that $J_n = J$ for all dimensions, the total number of nodes, J^N, grows exponentially with the dimensionality N.

In the paper, we consider three different versions of the multi-dimensional Gauss-Hermite product rule, denoted $Q\left(1\right)$, $Q\left(2\right)$ and $Q\left(3\right)$, which have 1, 2 and 3 nodes in each dimension, respectively. The total number of nodes under $Q\left(1\right)$, $Q\left(2\right)$ and $Q\left(3\right)$ is equal to 1, 2^N and 3^N, respectively, and is reported in Table 1 (panel 3) for N ranging from 1 to 200. As is seen from the table, $Q\left(2\right)$ and $Q\left(3\right)$ are not feasible for high dimensions if there are more than one node in each dimension. In contrast, $Q\left(1\right)$ is the cheapest possible integration formula as there is one node independently of the dimensionality of the problem.

6.3.2 Monomial rules

The Gauss-Hermite product rule constructs multi-dimensional nodes from one-dimensional nodes, which makes the number of nodes grow exponentially with dimension. In contrast, monomial integration rules construct

multi-dimensional nodes directly in a multi-dimensional space. To be specific, monomial rules yield a finite number of N-dimensional nodes and the corresponding weights so that the approximation is exact if the function g is an N-dimensional complete polynomial of a certain degree. Typically, the number of nodes under monomial rules grows only polynomially with the dimensionality of the problem.

Below, we describe three monomial formulas for approximating a multi-dimensional integral (17) which, for the sake of notational convenience, we label $M1$, $M2$ and $M3$. These formulas come from Stroud (1971, p. 315-329) and Judd (1998, p. 275), and are adopted by us to the case of correlated variables.

The first formula, $M1$, has $2N$ nodes:

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\}=\frac{1}{2N}\sum_{n=1}^{N}g\left(\pm R\mathbf{e}^{n}\right), \tag{20}$$

where $R\equiv\sqrt{N}\Omega$, and $\mathbf{e}^{n}$ is an $N\times1$ vector whose n-th element is equal to one and the remaining elements are equal to zero, i.e., $\mathbf{e}^{n}\equiv\left(0,...,1,...,0\right)'$.

The second formula, $M2$, has $2N^{2}+1$ nodes:

$$\begin{aligned}E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\}&=\frac{2}{2+N}g\left(0,...,0\right)\\&+\frac{4-N}{2\left(2+N\right)^{2}}\sum_{n=1}^{N}\left[g\left(R\mathbf{e}^{n}\right)+g\left(-R\mathbf{e}^{n}\right)\right]+\frac{1}{\left(N+2\right)^{2}}\sum_{n=1}^{N-1}\sum_{j=n+1}^{N}g\left(\pm S\mathbf{e}^{n}\pm S\mathbf{e}^{j}\right),\end{aligned} \tag{21}$$

where $R\equiv\sqrt{2+N}\Omega$ and $S\equiv\sqrt{\frac{2+N}{2}}\Omega$.

The third formula, $M3$, has 2^{N} nodes:

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\}=\frac{1}{2^{N}}\sum g\left(\pm\Omega\mathbf{e}^{1},...,\pm\Omega\mathbf{e}^{N}\right). \tag{22}$$

The number of nodes under monomial formulas $M1$ and $M2$ grows polynomially, namely, it grows linearly under $M1$ and it grows quadratically under $M2$. Monomial formula $M3$ happened to coincide with Gauss-Hermite product formula $Q\left(2\right)$, so that the number of nodes grows exponentially. In Table 1 (panel 3), we show how the number of nodes under formulas $M1$, $M2$ and $M3$ (equivalently, $Q\left(2\right)$) increases in N.

6.3.3 An example of integration formulas for $N = 2$

In this section, we illustrate the integration formulas described in Sections 6.3.1 and 6.3.2 using an example of two-dimensional case, $N = 2$. We assume that variables ε^1 and ε^2 are uncorrelated, have zero mean and unit variance. Integral (17) is then given by

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\} = \frac{1}{\pi}\int_{\mathbb{R}^2} g\left(\sqrt{2}y^1, \sqrt{2}y^2\right) \exp\left[-\left(y^1\right)^2 - \left(y^2\right)^2\right] dy^1 dy^2.$$

(a) Gauss-Hermite product rule (19) with 3 nodes in each dimension (i.e., $Q\left(3\right)$) uses one-dimensional nodes and weights given by $y_1^n = 0$, $y_2^n = \sqrt{\frac{3}{2}}$, $y_3^n = -\sqrt{\frac{3}{2}}$ and $\omega_1^n = \frac{2\sqrt{\pi}}{3}$, $\omega_2^n = \omega_3^n = \frac{\sqrt{\pi}}{6}$ for each $n = 1, 2$:

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\} = \frac{1}{\pi}\sum_{j_1=1}^{3}\sum_{j_2=1}^{3}\omega_{j_1}^1\omega_{j_2}^2 g\left(\sqrt{2}y_{j_1}^1, \sqrt{2}y_{j_2}^2\right) =$$
$$\frac{4}{9}g\left(0,0\right) + \frac{1}{9}g\left(0,\sqrt{3}\right) + \frac{1}{9}g\left(0,-\sqrt{3}\right) +$$
$$\frac{1}{9}g\left(\sqrt{3},0\right) + \frac{1}{36}g\left(\sqrt{3},-\sqrt{3}\right) + \frac{1}{36}g\left(\sqrt{3},-\sqrt{3}\right) +$$
$$\frac{1}{9}g\left(-\sqrt{3},0\right) + \frac{1}{36}g\left(-\sqrt{3},\sqrt{3}\right) + \frac{1}{36}g\left(-\sqrt{3},-\sqrt{3}\right)\Big].$$

(b) Gauss-Hermite product rule (19) with 1 node in each dimension (i.e., $Q\left(1\right)$) uses one-dimensional nodes and weights given by $y_1^n = 0$ and $\omega_1^n = \sqrt{\pi}$ for each $n = 1, 2$:

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\} = \frac{1}{\pi}\sum_{j_1=1}^{1}\sum_{j_2=1}^{1}\omega_{j_1}^1\omega_{j_2}^2 g\left(\sqrt{2}y_{j_1}^1, \sqrt{2}y_{j_2}^2\right) = g\left(0,0\right).$$

(c) Monomial formula (20) with 4 nodes (i.e., $M1$) is

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\} = \frac{1}{4}\left[g\left(\sqrt{2},0\right) + g\left(-\sqrt{2},0\right) + g\left(0,\sqrt{2}\right) + g\left(0,-\sqrt{2}\right)\right].$$

(d) Monomial formula (21) with 9 nodes (i.e., $M2$) is

$$E\left\{g\left(\boldsymbol{\varepsilon}\right)\right\} = \frac{1}{2}g\left(0,0\right) + \frac{1}{16}\left[g\left(2,0\right) + g\left(-2,0\right) + g\left(0,2\right) + g\left(0,-2\right)\right] +$$
$$+\frac{1}{16}\left[g\left(\sqrt{2},\sqrt{2}\right) + g\left(\sqrt{2},-\sqrt{2}\right) + g\left(-\sqrt{2},\sqrt{2}\right) + g\left(-\sqrt{2},\sqrt{2}\right)\right].$$

(e) Monomial formula (22) with 4 nodes (i.e., $M3$) coincides with Gauss-Hermite product rule (19) with 2 nodes in each dimension (i.e., $Q(2)$) and is given by

$$E\{g(\boldsymbol{\varepsilon})\} = \frac{1}{4}\left[g(1,1) + g(-1,1) + g(1,-1) + g(-1,-1)\right].$$

6.4 Cost-efficient projection method

For a given degree of the approximating polynomial function and a specific integration formula, adding an extra dimension (country) makes both the approximation step and integration step more expensive. The approximation step becomes more expensive because we have more polynomial coefficients, and we need more grid points for identifying such coefficients. The integration step becomes more expensive because we have more integration nodes to evaluate integrad on (except for the one-node Gauss-Hermite rule).[14] As is evident from Table 1, high-degree polynomials and certain integration formulas are excessively costly in high dimensions. When the number of state variables becomes large, we should use either low-degree polynomials or cheap integration formulas or both. Moreover, to make the method cost-efficient, we need to coordinate properly the approximation and integration strategies, specifically, we shall identify combinations of approximating polynomial functions and integration methods that match each other in terms of the overall accuracy of solutions.

7 Numerical analysis

In this section, we describe the methodology of our numerical study and assess the performance of our projection method in the context of the one-country and multi-country models.

7.1 Methodology

We solve the one-country version of the model (3) − (5) under three different grid constructions, described in Sections 6.2.1, 6.2.2 and 6.2.3, that lead

[14] In addition, introducing more countries to the model leads to extra Euler equations to solve, and hence, to extra LS regressions to run and extra vectors of coefficients to iterate on. Furthermore, operating with large data sets can slow down computations or can lead to a memory congestion.

to three different versions of our projection method, the tensor-product-grid algorithm (TPGA), simulated-grid algorithm (SGA) and cluster-grid algorithm (CGA), respectively. In the multi-country case, we use only the CGA, which is our main algorithm among the three.

Model parameters We parameterize the model $(3)-(5)$ as is typically done in the macroeconomic literature. We assume the constant relative risk aversion (CRRA) utility function, $u\left(c_t\right)=\frac{c_t^{1-\gamma}-1}{1-\gamma}$ with $\gamma\in(0,\infty)$, and the Cobb-Douglas production function, $f\left(k_t\right)=k_t^{\alpha}$ with $\alpha\in(0,1)$. We set the share of capital in production and the discount factor at $\alpha=0.36$ and $\delta=0.99$, respectively. We explore a wide range of the model's parameters including three values of the risk aversion coefficient $\gamma\in\{0.2,1,5\}$; two values of the depreciation rate $d\in\{0.025,1\}$, and two values for the autocorrelation coefficient and the standard deviation in the process for shocks (5), $\rho=\{0.95,0.99\}$ and $\sigma=\{0.01,0.03\}$, respectively. In both the one-country and multi-country models, our benchmark parameterization is $\gamma=1$, $d=0.025$, $\rho=0.95$ and $\sigma=0.01$. We normalize steady state capital to one by setting $A=\frac{1/\delta-(1-d)}{\alpha}$.

Clustering algorithm We construct a cluster grid using the hierarchical algorithm with Ward's linkage clustering, as described in Section 3.2.1. We prefer the hierarchical algorithm over the $\mathcal{K}$-means algorithm because the results of hierarchical algorithms are reproducible while those of the $\mathcal{K}$-means algorithm might depend on a random assignment of observations into clusters on the initial step (both algorithms are relatively inexpensive in our applications). Overall, we find that the appropriate distance measure between objects and the appropriate change of variables are far more important for the outcome of clustering than a specific clustering algorithm itself.

Approximation and integration strategies In the one-country model, we parameterize the capital decision rule using polynomials of up to the fifth-degree and we compute conditional expectation using Gauss-Hermite product rule with 1, 2 and 10 nodes (i.e., $Q\left(1\right)$, $Q\left(2\right)$ and $Q\left(10\right)$, respectively), and Monte Carlo simulation with 1000 and 10000 observations. In the multi-country case, we consider polynomials of up to the third degree, and we perform integration using the Gauss-Hermite product rule with 1, 2^N and 3^N

nodes (i.e., $Q(1)$, $Q(2)$ and $Q(3)$, respectively), and the monomial formulas $M1$ and $M2$ with $2N$ and $2N^2+1$ nodes, respectively.

Initial guess Our projection algorithm requires a knowledge of the ergodic set, however, the ergodic set is unknown before the model is solved. Therefore, to initialize the algorithm, we first compute a low-accuracy solution, and we then construct the ergodic set by simulation (our simulated panel consists of $T = 10000$ observations). In the one-country case, we compute a low-accuracy solution on an equally-spaced grid of 10 points in the interval $\pm 1\%$ of steady state (we use this solution for deducing the limits of the domain $\left[\underline{k}, \overline{k}\right] \times \left[\underline{\theta}, \overline{\theta}\right]$ under the TPGA, for selecting a subset of simulated points under the SGA and for constructing clusters under the CGA). In the multi-country model, we initialize the CGA using the CGA itself as follows: we parameterize the capital decision rule using an arbitrary initial guess $k_{t+1}^n = 0.95k_t^n + 0.05\theta_t^n$ for all n (this guess matches the steady state level of capital equal to one), simulate the model, construct the clusters and solve the model on the constructed cluster grid; we next use the obtained solution for simulating the model again and for constructing the ergodic set (thus, we compute clusters twice). As an initial guess for polynomial approximations of degrees higher than one, we use solutions obtained under the polynomial approximations of the previous degree.

Approximation step and convergence parameters Under our projection methods, LS problem (10) is typically well-conditioned and the approximation step, Step 4, can be implemented with ordinary least-squares (OLS) method. We solve the LS problem using a Matlab's back-slash operator, which gives the same solution as the OLS method for well-conditioned problems**.**

We set the dampening parameter in (11) at $\mu = 0.1$ for all experiments except for the model with high risk aversion in which we use a smaller value for $\mu = 0.03$ to enhance convergence. In the one-country model, high-degree polynomials lead to very accurate solutions, and we use a tight convergence criterion $\vartheta = 11$ in (12). In the multi-country models, we consider only low-degree polynomials which lead to less accurate solutions than polynomials of high degrees, and we use a less tight criterion of $\vartheta = 8$.

Accuracy test We measure accuracy of the solutions by the size of Euler equation errors in the ergodic set. We rewrite FOCs (6) in a unit-free form to obtain Euler equation error $E^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)$ at τ,

$$\mathcal{E}^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau) \equiv E_\tau\left[\frac{c_{\tau+1}^{-\gamma}}{c_\tau^{-\gamma}}\left(1-d+\alpha\theta_{\tau+1}\left(k_{\tau+1}^n\right)^{\alpha-1}\right)\right]-1. \tag{23}$$

In the true solution, $\mathcal{E}^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)=0$ for all $(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)$, so that accuracy can be measured by how much $\mathcal{E}^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)$ differs from zero. We draw a new sequence of shocks of length $T=10000$, and we use the decision rules to simulate the time series $\{\mathbf{k}_\tau, c_\tau\}_{\tau=0}^{10000}$. We select 1000 observations separated by 10 periods, $\tau=0,\ 10,\ 20,\ ....$ For each selected point $(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)$, we compute $\mathcal{E}^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)$ by evaluating the conditional expectation in (23) with a given integration rule. We then find the average absolute and maximum absolute Euler equation errors across countries and time, $\mathcal{E}_{mean}=\frac{1}{N\cdot T}\sum_{n=1}^{N}\sum_{\tau=0,10,20,...}|\mathcal{E}^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)|$, and $\mathcal{E}_{\max}=\max\left(\{|\mathcal{E}^n(\mathbf{k}_\tau, \boldsymbol{\theta}_\tau)|\}_{\tau=0,10,20,...}\right)$, respectively.

Running time We report running time, denoted CPU, in seconds. For the CGA, it includes the time necessary for clustering. For the one-country model, running time for the first-degree polynomial approximation includes time for constructing an initial guess. For the multi-country models, the running time for the first-degree approximation does not include the time for finding the initial guess; if the latter time was included, the running time would approximately double. (Recall that our initial guess is a first-degree approximation computed by the CGA on an arbitrary grid of points).

Hardware and software We ran the computational experiments on a desktop computer ASUS with Intel(R) Core(TM)2 Quad CPU Q9400 (2.66 GHz), RAM 4MB. Our programs are written in Matlab, version 7.6.0.324 (R2008a).

7.2 The one-country model

In this section, we focus on the one-country model. For a wide range of the model's parameters, we explore how accuracy and cost of our projection method depend on the degree of the approximating polynomial function, the number and placement of grid points and the specific integration procedure

used. To evaluate accuracy of solutions (i.e., to compute the Euler equation errors in (23)), we use a very accurate ten-node Gauss-Hermite rule.

Benchmark model In Table 2, we present the solutions to our benchmark model obtained by the three projection algorithms, the TPGA, SGA and CGA, differing in their grid constructions. (Recall that our benchmark parameterization is $d = 0.025$, $\gamma = 1$, $\rho = 0.95$ and $\sigma = 0.01$). For each algorithm, we first consider the minimum number of grid points necessary for identifying the coefficients (collocation), $\mathcal{K}_c$ (see case $\mathcal{K} = \mathcal{K}_c$ in Table 2), and we then increase the number of grid points by 20%, 100% and 400% relative to $\mathcal{K}_c$ (see cases $\mathcal{K} = 1.2\mathcal{K}_c$, $\mathcal{K} = 2\mathcal{K}_c$ and $\mathcal{K} = 5\mathcal{K}_c$, respectively, in Table 2). To isolate the effect of a grid choice on accuracy, we use an accurate ten-node Gauss-Hermite rule in the solution procedure.

Three tendencies can be observed from the table. First, a degree of the approximating polynomial plays a determinant role in accuracy: the maximum errors decrease with a polynomial degree from errors of size 10^{-3} for the first-degree polynomial to those of size 10^{-8} for the fifth-degree polynomial. Second, accuracy of the three methods can be significantly lower under collocation than under more "generous" grids; even a moderate 20% increase in the number of grid points has a sizable effect on accuracy. Third, the SGA and CGA visibly dominate in accuracy the TPGA independently of the number of grid points. This happens because the SGA and CGA fit a polynomial only in a relevant part of the state space (the ergodic set), whereas the TPGA fits the same polynomial in a larger squared domain and thus, faces a trade off between the fit inside and outside the ergodic set. Finally, the CGA dominates in accuracy the SGA when the number of grid points is $\mathcal{K} = \mathcal{K}_c$, $\mathcal{K} = 1.2\mathcal{K}_c$ and $\mathcal{K} = 2\mathcal{K}_c$, and both of them have a comparable performance when the number of grid points becomes large $\mathcal{K} = 5\mathcal{K}_c$ (in the last case, many randomly drawn simulated points under the SGA cover the ergodic set as efficiently as do clusters under the CGA).

Sensitivity results In our sensitivity experiments, we vary one of the model's parameters holding the rest of the parameters fixed to the benchmark values. Specifically, we consider the cases of low risk aversion $\gamma = 1/5$, high risk aversion $\gamma = 5$, high persistence of shocks $\rho = 0.99$ and high volatility of shocks $\sigma = 0.03$. The results for these four sensitivity experiments are presented in Table 3. The number of grid points is the same for the TPGA,

SGA and CGA and is equal to $\mathcal{K} = 36$. In some experiments, accuracy is lower than in the benchmark model because of either a larger size of the domain or a greater curvature of the decision rules. Nevertheless, the accuracy ranking of the three methods is typically the same as in the benchmark case. In all the cases (except the one for low risk version $\gamma = 1/5$), the CGA dominates in accuracy the SGA, which in turn dominates the TPGA. Typically, the difference in accuracy between the CGA and TPGA is around one order of magnitude; it is surprising to see such a sharp contrast in the methods' performance given that there are only two state variables. In the case of low risk aversion $\gamma = 1/5$, the SGA and CGA dominate in accuracy the TPGA for low-degree polynomials but the accuracy ranking is mixed for high-degree polynomials (in this case, the capital decision rule is close to linear so that focusing on a smaller endogenous domain does not increase accuracy). Based on the experiments reported in Tables 2 and 3, we select the CGA as our preferred algorithm and concentrate on it in the rest of the paper.

Role of integration method in accuracy In Table 4, we explore how a specific integration method affects the overall accuracy of the CGA by considering five alternative integration methods such as Gauss-Hermite product rule with 1, 2 and 10 nodes (referred to as $Q(1)$, $Q(2)$ and $Q(10)$, respectively) and the Monte Carlo integration method with 1000 and 10000 nodes. We analyze four different cases: the model with a closed-form solution ($d = 1$), the benchmark model ($d = 0.025$), the model with highly persistent shocks ($\rho = 0.99$), and the model with highly volatile shocks ($\sigma = 0.03$) holding the remaining parameters fixed to the benchmark values. In all the experiments considered, the grid was constructed using $\mathcal{K} = 36$ clusters.

We have three findings in the table. First, formulas $Q(2)$ and $Q(10)$ deliver virtually the same solutions under the polynomial approximations of up to degree three; a small difference in their performance becomes visible only when we go to polynomials of degrees higher than three. Second, the one-node formula, $Q(1)$, delivers sufficiently accurate solutions under polynomials of up to degree two (even if we have highly volatile or highly persistent shocks), however, a low accuracy of integration restricts the overall accuracy under high-degree polynomials (the model with a closed-form solution is an exception here). Third, the Monte Carlo integration method is less accurate than quadrature integration even if as many as 10000 simulated nodes are used, and it restricts the overall accuracy under high-degree polynomi-

als (again, the model with a closed-form solution is an exceptional case). A comparison of the 1000-node and 10000-node cases shows that accuracy of the Monte Carlo integration method increases with the simulation length, however, at a relatively low rate. We will not consider the Monte Carlo integration method in the multi-country context since better alternatives are available (in our experiments, even the one-node Gauss-Hermite rule, $Q(1)$, typically delivers more accurate solutions than the Monte Carlo integration method with 10000 nodes).[15]

7.3 The multi-country model

In this section, we present the results for the multi-country version of the model obtained under the cluster-grid method.

Role of integration rule in accuracy of solution and test When we approach high dimensions, the accurate Gauss-Hermite product rule becomes unfeasible not only in the solution but also in testing procedures. Therefore, as a first step, we need to determine how reliable different integration formulas are for the purpose of accuracy testing (it might happen that low accuracy of integration in the testing procedure contaminates the results of the test). This issue did not arise in the one-country case since a very accurate integration method, $Q(10)$, was feasible for testing.

In Table 5, we report the results for the two-country and six-country models obtained under the benchmark parameterization. We use a grid composed of 300 cluster-grid points. In both the solution and testing procedures, we use five alternative integration methods, namely, $Q(3)$, $Q(2)$, $M2$, $M1$, and $Q(1)$.[16] Common sense suggests that an integration method used for testing should be as good as or better than the one used for finding a solution. We therefore do not consider those cases in which an integration method used in the testing procedure is inferior to the one used in the solution procedure (these cases are replaced by dashes in the table).

[15] The standard Monte Carlo integration method considered in the context of our cluster-grid algorithm requires summing up T observations in each grid point. Stochastic simulation approaches, described in Judd et al. (2009), rely on a more efficient integration method combining Monte Carlo simulation and a regression step, namely, T observations are used for inferring the conditional expectations in T points simultaneously.

[16] Compared to the one-country case, we substitute $Q(10)$ by $Q(3)$ as $Q(10)$ is too computationally demanding for the multi-country case.

The main findings in the table are as follows. Under the most accurate integration method in the testing procedure, $Q(3)$, the solutions produced by four integration methods, $Q(3)$, $Q(2)$, $M2$ and $M1$, are virtually identical in terms of the Euler equation errors (typically, up to the fifth digit); the solution delivered by $Q(1)$ is similar to that of the other methods for the first-degree polynomial approximation but is about 10% less accurate for the second-degree polynomial approximation. For any integration method used in the solution procedure, the four integration methods $Q(3)$, $Q(2)$, $M2$ and $M1$ are adequate for the purpose of testing, as they lead to very similar test results (again differing only in the fifth digit). An exception is $Q(1)$ applied for testing the $Q(1)$ solution, which under the second-degree polynomial approximation understates the errors relative to more accurate integration methods.

Role of the number of clusters in accuracy of solutions We next carry out a comparison of accuracy depending on the number of clusters for the models with $N = 2$, $N = 4$ and $N = 6$ (see Table 6). We use $Q(2)$ both in the solutions and testing procedures. As in the one-country model, we start from collocation, $\mathcal{K} = \mathcal{K}_c$, and then augment the number of clusters to $\mathcal{K} = 1.2\mathcal{K}_c$, $\mathcal{K} = 2\mathcal{K}_c$ and $\mathcal{K} = 5\mathcal{K}_c$. We find that collocation performs poorly in the context of our cluster-grid method not only in terms of accuracy but also in terms of numerical stability (our method failed to compute the third-degree polynomial approximations for the models with $N = 2$ and the second-degree polynomial approximations for the models with $N = 4$ and $N = 6$). However, even a moderate increase in the number of clusters, for example, by 20 %, i.e., $\mathcal{K} = 1.2\mathcal{K}_c$, is sufficient for restoring the numerical stability. Overall, we find that accuracy of solutions typically increases in the number of clusters, as was seen in the one-country case.

Coordinating approximation and integration strategies Table 7 is the core of the paper. In this table, we compute solutions to our benchmark model by ranging the number of countries from 2 to 200. To enhance accuracy and numerical stability of the cluster-grid method, we keep the number of clusters larger than the number of coefficients (see the table for the number of clusters in each experiment). In the solution procedure, we use four alternative integration formulas, $Q(2)$, $M2$, $M1$ and $Q(1)$. Under the second-degree polynomial, $Q(2)$, $M2$, $M1$ and $Q(1)$ were feasible for models

with up to $N = 8$, $N = 12$, $N = 20$ and $N = 40$ countries, respectively. To test accuracy of solutions, we use $Q(2)$ for N up to 12, use $M2$ for N from 12 to 20, and use $M1$ for N larger than 20.

As far as accuracy is concerned, the tendencies are the same as those previously seen in Table 5 for the two-country and six-country models. For the second- and third-degree polynomial approximation, the Euler equation errors obtained under $Q(2)$, $M2$ and $M1$ are nearly identical, and the errors obtained under $Q(1)$ are about 10% larger than under more accurate integration formulas (an acceptable reduction in accuracy given that this formula allows us to advance from the 20-country to 40-country models). Overall, our second-degree polynomial approximations are sufficiently accurate: the maximum error is always less than 0.01% in the ergodic set (even under $Q(1)$). Furthermore, if an integration method is sufficiently accurate, the third-degree polynomial approximation in the two-country model produces maximum errors which are lower than 0.001%.

Unlike accuracy, computational cost of our projection method depends dramatically on a specific integration formula used. For example, for $N = 8$, the running time for computing the second-degree approximations under $M2$ and $Q(1)$ is, respectively, 3774 and 109 seconds, i.e., it differs by up to a factor of 35, while for $N = 12$, the running time under $M2$ and $Q(1)$ is, respectively, 69025 and 226 seconds, i.e., it differs by up to a factor of 300. The running time increases rapidly with dimension under $Q(2)$ and $M2$, and it increases slower with dimension under $M1$ and $Q(1)$. In particular, under $M1$, we can compute a second-degree approximation to the model with $N = 20$ for about 4 hours and 40 minutes (16895 seconds), and under $Q(1)$, we can compute such an approximation with $N = 40$ for about 24 hours and 25 minutes (87748 seconds).

After we reach $N = 40$ countries, the second-degree polynomial approximation becomes expensive even under the cheapest integration formula $Q(1)$. We thus restrict attention to the first-degree polynomial: we use $M1$ and $Q(1)$ for N up to 100, and we use $Q(1)$ for N up to 200. As is observed from the table, our linear solutions are still sufficiently accurate: the maximum error never reaches 0.1%. Another class of numerical methods in the literature that allows us to solve models with a very large number of state variables is perturbation. However, as was pointed out in Section 3, perturbation methods compute solutions only in one point of state space (steady state), and accuracy of their local solutions reduces rapidly away from that point. In contrast, the cluster-grid method delivers global solutions that are

sufficiently accurate in the whole ergodic set. As far as the cost is concerned, under $M1$, we compute a linear solution to the model with $N = 100$ for about 10 hours and 46 minutes (38782 seconds), and under $Q(1)$, we can compute a linear solution with $N = 200$ for about 1 hour and 45 minutes (6316 seconds), respectively. The latter running time is small for such a large number of countries, which indicates that we can easily go beyond $N = 200$.

Sensitivity experiments We finally explore the robustness of our results obtained for the benchmark multi-country model by varying one of the model's parameters at a time, holding the rest of the parameters fixed to the benchmark values. We illustrate the observed regularities in Table 8 using an example of the model with $N = 6$. To evaluate the accuracy of solutions, we use Gauss-Hermite product formula $Q(2)$. We find it more difficult to enforce convergence in the experiment with high risk aversion, $\gamma = 5$, than in other sensitivity experiments; hence, we reduce the dampening parameter μ in (11) which leads to an increase in computational time. Overall, our sensitivity experiments in the multi-country case show the same regularities as those seen in Table 3 for the one-country case. All the integration formulas considered lead to very similar results except for $Q(1)$ which yields somewhat lower accuracy.

8 Conclusion

An ergodic set is a tiny fraction of a hypercube domain, which is normally examined by numerical methods. The cluster-grid method presented in this paper provides a simple and inexpensive way of solving a dynamic economic model on the ergodic set directly. For a given polynomial degree and a given integration method, a parsimonious representation of the ergodic set allows us to achieve essentially the same accuracy in high-dimensional models as we do in low-dimensional models. Our method can be especially useful for applications that require controlling the domain on which the problem is solved (development economics, dynamic games, etc).

Given that accuracy of the cluster-grid method depends primarily on the flexibility of the approximating function used, we can potentially improve the method's performance using other families of approximating functions. One possibility is to extend a complete polynomial of a given degree to include some high-degree polynomial terms. If a relatively few terms are added, it

could be that cost does not grow much but accuracy does. Another possibility is to explore whether non-polynomial families of approximating functions can lead to the same (a higher) accuracy with a smaller (the same) number of basis function.

Our assessment of accuracy and cost shows that a proper coordination between the approximation and integration strategies is needed for constructing efficient numerical methods. An example of such a coordination is a combination of a flexible second-degree polynomial with a cheap one-node Gauss-Hermite rule (as opposed to a combination of a rigid first-degree polynomial with expensive integration formulas). Overall, in our applications, low-cost integration formulas (i.e., the monomial formula with $2N$ nodes and the one-node Gauss-Hermite rule) have enough accuracy to be compatible with accuracy of solutions attainable under low-degree polynomial approximations (high-degree polynomials are not feasible with high dimensions anyway). Our results are suggestive for other economic applications.

Finally, we shall reiterate that our solutions are computed using a standard desktop computer. The speed of the cluster-grid method can be considerably increased using more powerful hardware. Furthermore, the cluster-grid approach is a natural candidate for parallelizing since we can compute integrals in different grid points independently of each other using different machines (processors). This will enable us to solve even more ambitious applications than those studied in the paper.

References

[1] Aruoba, S., J. Fernandez-Villaverde, and J. Rubio-Ramirez, (2006). Comparing solution methods for dynamic equilibrium economies. Journal of Economic Dynamics and Control 30, 2477-2508.

[2] Christiano, L. and D. Fisher, (2000). Algorithms for solving dynamic models with occasionally binding constraints. Journal of Economic Dynamics and Control 24, 1179-1232.

[3] Collard, F., and M. Juillard, (2001). Accuracy of stochastic perturbation methods: the case of asset pricing models, Journal of Economic Dynamics and Control, 25, 979-999.

[4] Den Haan, W., (1996). Heterogeneity, aggregate uncertainty, and the short-term interest rate. Journal of Business and Economic Statistics, 14, 399-411.

[5] Den Haan W., K. Judd and M. Juillard, (2010). Computational suite of models with heterogeneous agents: multi-country real business cycle models, Manuscript.

[6] Den Haan, W. and A. Marcet, (1990). Solving the stochastic growth model by parameterized expectations. Journal of Business and Economic Statistics 8, 31-34.

[7] Everitt, B., S. Landau, and M. Leese, (2001). Cluster Analysis. Arnold: London.

[8] Fair, R. and J. Taylor, (1983). Solution and maximum likelihood estimation of dynamic nonlinear rational expectation models. Econometrica 51, 1169-1185.

[9] Gaspar, J. and K. Judd, (1997). Solving large-scale rational-expectations models. Macroeconomic Dynamics 1, 45-75.

[10] Hastie, T., Tibshirani, R. and J. Friedman, (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York.

[11] Judd, K., (1992). Projection methods for solving aggregate growth models. Journal of Economic Theory 58, 410-452.

[12] Judd, K., (1998). Numerical Methods in Economics. Cambridge, MA: MIT Press.

[13] Judd, K. and S. Guu, (1993). Perturbation solution methods for economic growth models, in: H. Varian, (Eds.), Economic and Financial Modeling with Mathematica, Springer Verlag, pp. 80-103.

[14] Judd, K., L. Maliar and S. Maliar, (2009). Numerically stable stochastic simulation approaches for solving dynamic models. NBER Working Paper 15296.

[15] Juillard, M. and S. Villemot, (2010). Multi-country real business cycle models: accuracy tests and testing bench. Manuscript.

[16] Kim, J., S. Kim, E. Schaumburg and C. Sims, (2008). Calculating and using second-order accurate solutions of discrete time dynamic equilibrium models. Journal of Economic Dynamics and Control 32, 3397-3414.

[17] Kollmann, R., S. Maliar, B. Malin and P. Pichler, (2010). Comparison of solutions to the multi-country real business cycle model. Manuscript.

[18] Krueger, D. and F. Kubler, (2004). Computing equilibrium in OLG models with production. Journal of Economic Dynamics and Control 28, 1411-1436.

[19] Maliar, S., L. Maliar, and K. Judd, (2010). Solving the multi-country real business cycle model using ergodic set methods. Manuscript.

[20] Marimon, R. and A. Scott, (1999). Computational Methods for Study of Dynamic Economies, Oxford University Press, New York.

[21] Pakes, A. and P. McGuire, (2001). Stochastic algorithms, symmetric Markov perfect equilibria, and the 'curse' of dimensionality. Econometrica 69, 1261-1281.

[22] Romesburg, C., (1984). Cluster Analysis for Researchers. Lifetime Learning Publications: Belmont, California.

[23] Rust, J., (1996). Numerical dynamic programming in economics, in: H. Amman, D. Kendrick and J. Rust (Eds.), Handbook of Computational Economics, Amsterdam: Elsevier Science, pp. 619-722.

[24] Santos, M., (1999). Numerical solution of dynamic economic models, in: J. Taylor and M. Woodford (Eds.), Handbook of Macroeconomics, Amsterdam: Elsevier Science, pp. 312-382.

[25] Stroud A., (1971). Approximate Integration of Multiple Integrals. Prentice Hall: Englewood Cliffs, New Jersey.

[26] Taylor, J. and H. Uhlig, (1990). Solving nonlinear stochastic growth models: a comparison of alternative solution methods. Journal of Business and Economic Statistics 8, 1-17.

Figure 1a. Ergodic distribution.

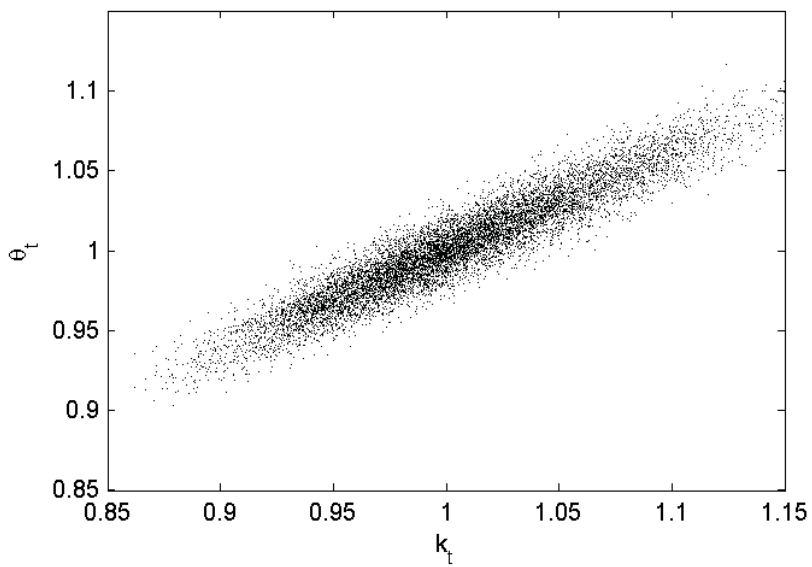

Figure 1b. Normalized ergodic distribution.

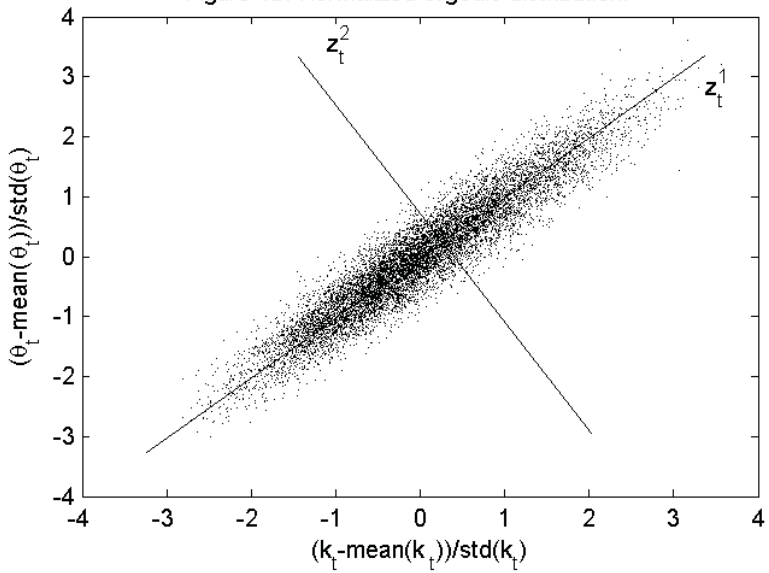

Figure 1c. PCs of ergodic distribution.

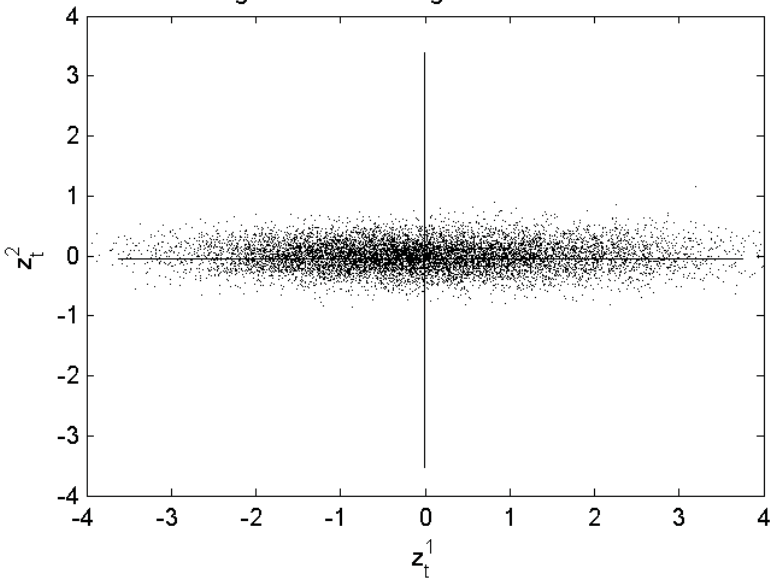

Figure 1d. Normalized PCs of ergodic distribution.

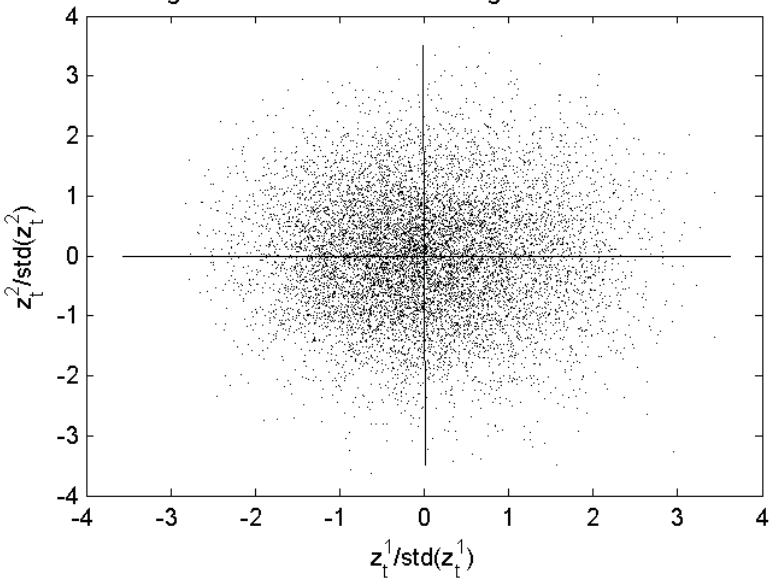

Figure 2. Agglomerative hierarchical clustering algorithm: an example.

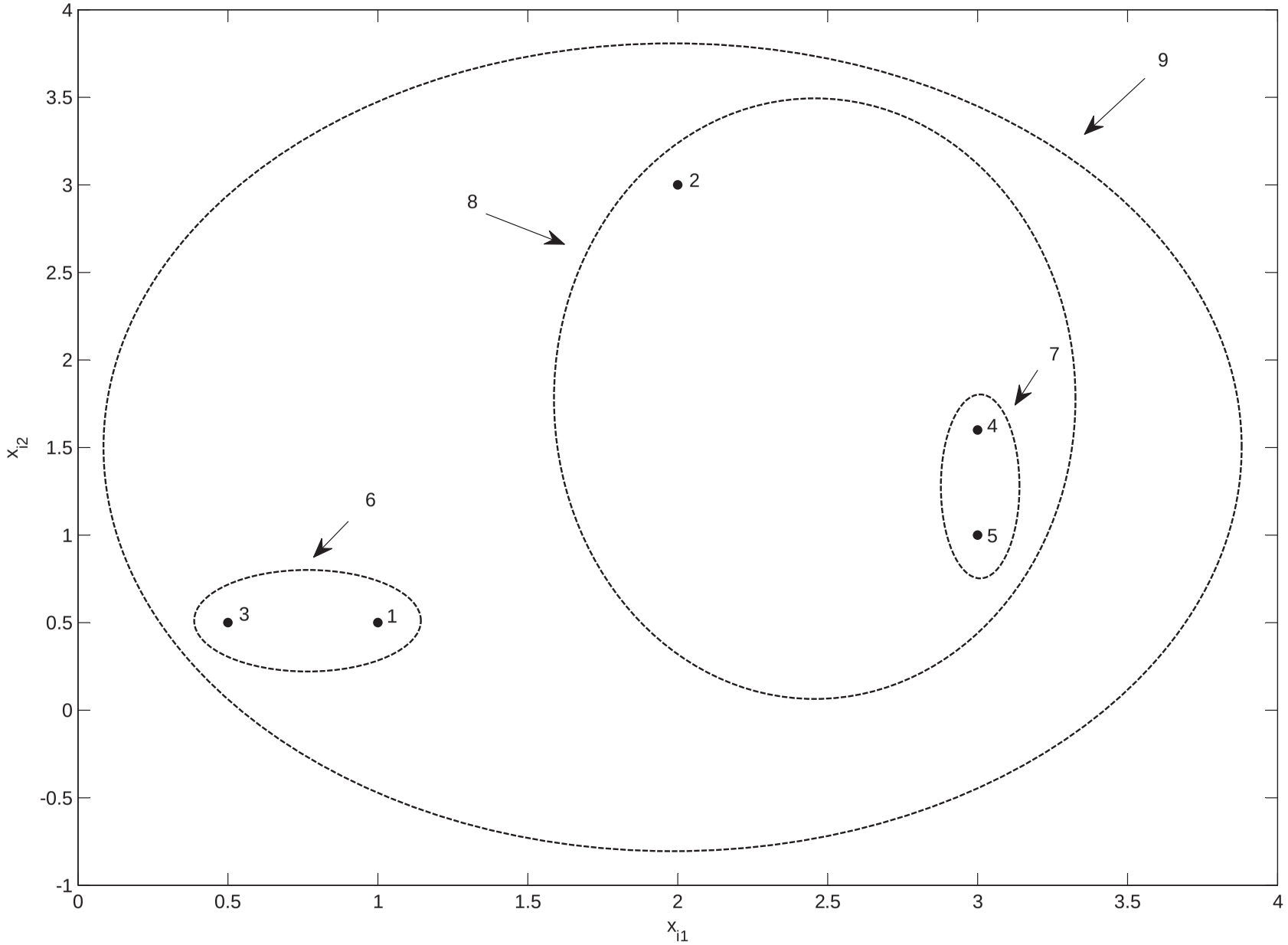

Figure 3a. Normalized ergodic distribution: 3 clusters.

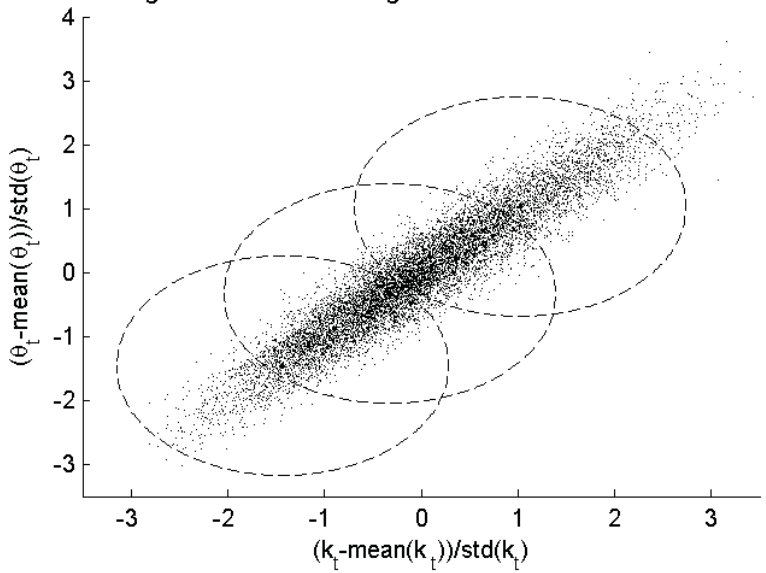

Figure 3b. Normalized ergodic distribution: 10 clusters.

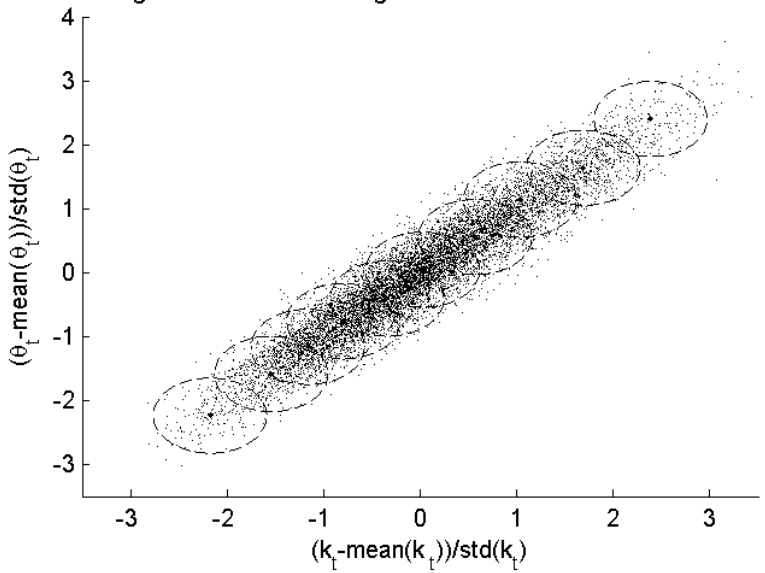

Figure 3c. Normalized ergodic distribution: 30 clusters.

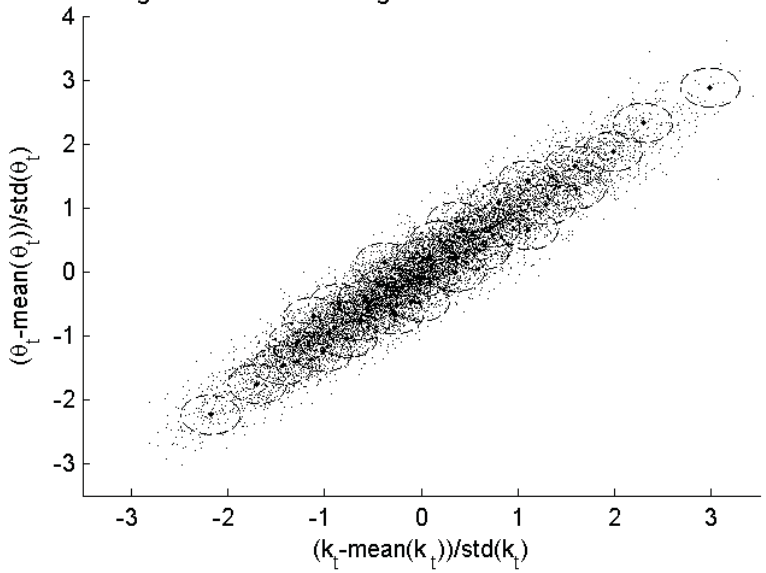

Figure 3d. Normalized ergodic distribution: 100 clusters.

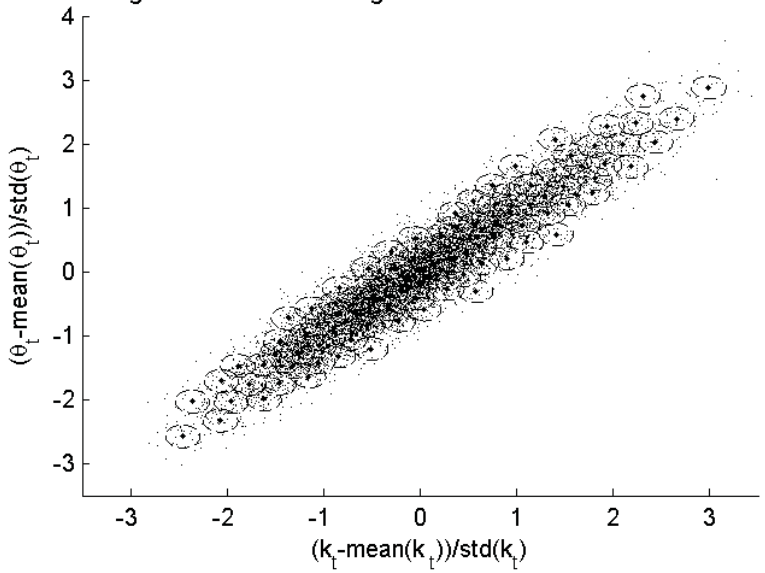

Figure 4a. Normalized PCs of ergodic distribution: 3 clusters.

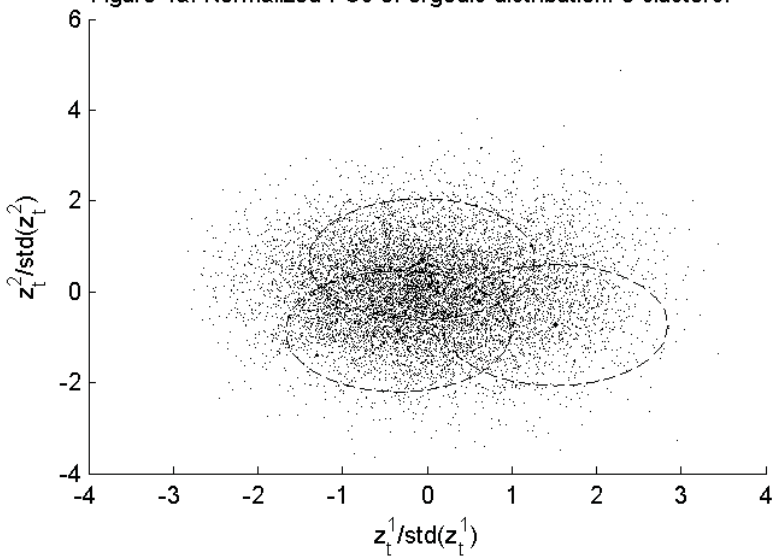

Figure 4b. Normalized PCs of ergodic distribution: 10 clusters.

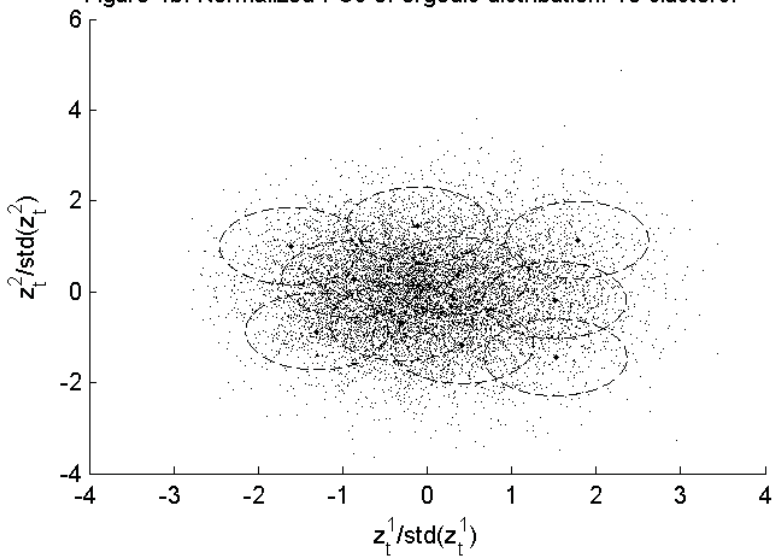

Figure 4c. Normalized PCs of ergodic distribution: 30 clusters.

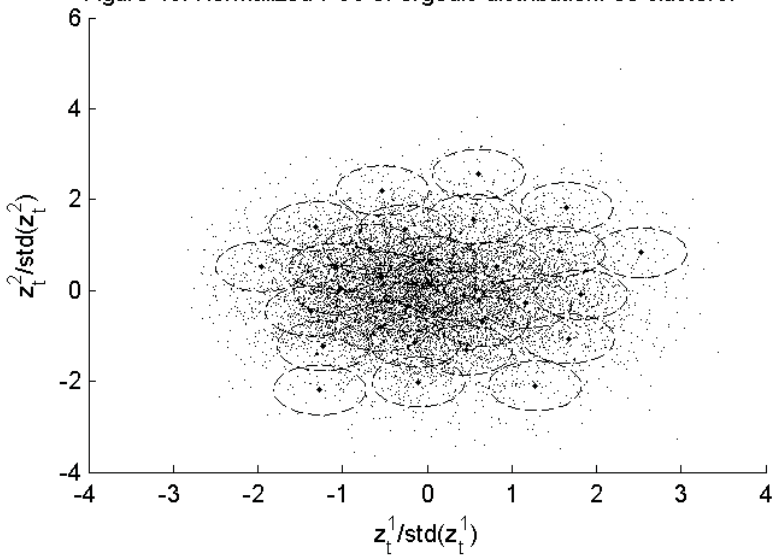

Figure 4d. Normalized PCs of ergodic distribution: 100 clusters.

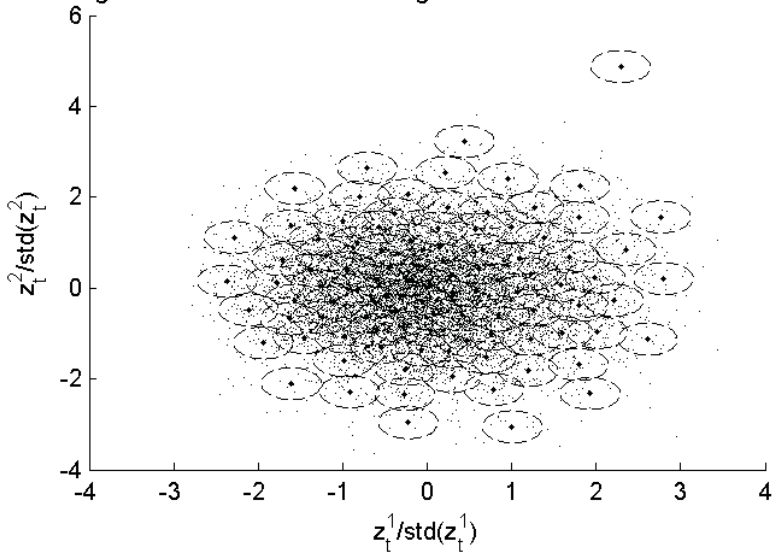

Table 1. Number of polynomial coefficients, time for constructing clusters and number of integration nodes depending on dimension *N*.

		N=1	*N*=2	*N*=6	*N*=10	*N*=20	*N*=40	*N*=60	*N*=80	*N*=100	*N*=200
Number of coefficients in the m-th degree polynomial, $P_m(N)$											
1st degree	$P_1(N)$	3	5	13	21	41	81	121	161	201	401
2nd degree	$P_2(N)$	6	15	91	231	861	3321	7381	13041	20301	80601
3rd degree	$P_3(N)$	10	35	455	1771	12341	91881	3(+5)	7(+5)	2(+6)	1(+7)
Time for constructing *K* clusters from simulated series of length *T* (in seconds)											
T=1000	*K*=3	0.07	0.08	0.09	0.11	0.12	0.18	0.24	0.29	0.39	1.18
	K=30	0.08	0.08	0.09	0.12	0.13	0.24	0.29	0.45	0.54	1.35
	K=300	0.09	0.10	0.12	0.15	0.31	0.83	1.15	1.16	1.85	3.23
T=3000	*K*=3	0.83	0.84	0.83	0.84	1.04	1.42	1.78	1.98	2.77	4.50
	K=30	0.80	0.81	0.85	0.86	1.13	1.54	1.96	2.21	3.08	5.15
	K=300	0.83	0.86	1.00	1.08	1.92	3.11	4.40	5.00	7.06	11.73
T=10000	*K*=3	8.93	8.87	9.18	9.51	12.05	17.36	21.56	22.11	31.45	42.08
	K=30	8.99	8.94	9.28	9.77	12.46	17.81	22.19	23.48	31.87	44.23
	K=300	9.05	9.33	10.08	10.91	14.78	22.40	30.04	32.92	43.76	66.37
Number of nodes in an integration rule											
Q(3)	3^N	3	9	729	59049	4(+9)	1(+19)	4(+28)	2(+38)	5(+47)	2(+95)
Q(2)	2^N	2	4	64	1024	1(+6)	1(+12)	1(+18)	1(+24)	1(+30)	2(+60)
M2	$1+2N^2$	3	9	73	201	801	3201	7201	12801	20001	80001
M1	$2N$	2	4	12	20	40	80	120	160	200	400
Q(1)	1	1	1	1	1	1	1	1	1	1	1

Remark: Notation *x(y)* means $x \cdot 10^y$.

Table 2. Effect of the number of clusters on accuracy and speed under three projection algorithms: the benchmark one-country model.

Polynomial degree	$K=K_c$			$K=1.2K_c$			$K=2K_c$			$K=5K_c$		
	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*
Tensor-product-grid algorithm (TPGA)												
1st degree	-3.43	-3.25	1.91	-3.43	-3.25	1.92	-3.59	-3.36	2.00	-3.67	-3.40	2.13
2nd degree	-5.05	-4.62	0.17	-5.05	-4.62	0.18	-5.12	-4.70	0.27	-5.19	-4.78	0.59
3rd degree	-6.12	-5.89	0.30	-6.12	-5.89	0.30	-6.20	-5.94	0.44	-6.33	-6.03	1.06
4th degree	-5.25	-4.76	0.41	-7.53	-7.25	0.46	-7.56	-7.29	0.66	-7.62	-7.37	1.40
5th degree	-8.48	-7.94	3.55	-8.82	-8.60	1.93	-8.81	-8.60	2.29	-8.85	-8.60	3.68
Simulation-grid algorithm (SGA)												
1st degree	-4.21	-3.53	1.89	-3.89	-3.36	1.90	-4.19	-3.50	1.93	-4.32	-3.70	2.05
2nd degree	-5.98	-5.30	0.12	-6.01	-5.15	0.13	-5.91	-5.02	0.21	-5.97	-5.21	0.49
3rd degree	-7.26	-6.31	0.18	-7.23	-6.36	0.22	-7.40	-6.68	0.35	-7.39	-6.45	0.83
4th degree	-6.89	-5.72	0.29	-8.21	-6.89	0.34	-8.41	-7.28	0.55	-8.85	-7.54	1.31
5th degree	-9.00	-7.65	1.62	-9.55	-8.57	1.70	-9.42	-8.00	2.06	-9.57	-8.56	3.34
Cluster-grid algorithm (CGA)												
1st degree	-4.26	-3.60	11.00	-4.28	-3.60	11.16	-4.30	-3.56	11.17	-4.32	-3.62	11.26
2nd degree	-5.69	-4.83	9.17	-6.15	-5.40	9.10	-6.06	-5.26	9.22	-6.08	-5.54	9.58
3rd degree	-7.44	-6.40	9.13	-7.41	-6.63	9.23	-7.60	-6.82	9.33	-7.39	-6.59	9.80
4th degree	-7.98	-6.88	9.30	-8.82	-7.62	9.47	-8.94	-7.98	9.58	-8.81	-8.03	10.34
5th degree	-9.60	-8.83	10.86	-9.62	-9.01	9.86	-9.57	-8.89	11.34	-9.61	-8.93	12.45

Remark: K is the number of grid points; K_c is the minimum possible number of grid points (collocation); $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units); and *CPU* is computational time in seconds.

Table 3. Accuracy and speed under three projection algorithms: sensitivity results for the one-country model.

Polynomial degree	$\gamma = 1/5$			$\gamma = 5$			$\rho = 0.99$			$\sigma = 0.03$		
	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*
Tensor-product-grid algorithm (TPGA)												
1st degree	-3.76	-3.63	5.64	-2.97	-2.52	1.42	-3.15	-2.90	2.83	-2.74	-2.41	2.92
2nd degree	-5.74	-5.33	1.77	-4.20	-3.71	0.31	-4.45	-3.96	0.75	-3.74	-3.32	0.84
3rd degree	-6.82	-6.67	1.04	-5.08	-4.79	0.25	-4.92	-4.68	0.56	-4.32	-3.97	0.52
4th degree	-8.36	-8.03	0.96	-6.11	-5.83	0.24	-5.86	-5.56	0.57	-5.15	-4.81	0.64
5th degree	-9.89	-9.37	1.18	-7.21	-6.91	0.67	-6.83	-6.50	1.31	-6.00	-5.60	1.49
Simulation-grid algorithm (SGA)												
1st degree	-4.81	-3.99	5.00	-3.30	-2.63	1.35	-3.89	-3.13	2.60	-3.35	-2.65	2.50
2nd degree	-6.56	-5.59	1.25	-4.75	-3.83	0.28	-5.27	-4.27	0.57	-4.65	-3.91	0.63
3rd degree	-8.00	-6.91	0.82	-5.89	-4.75	0.19	-6.27	-5.15	0.33	-5.47	-4.50	0.51
4th degree	-9.35	-8.25	0.42	-6.87	-5.62	0.14	-7.14	-5.91	0.24	-6.37	-4.94	0.42
5th degree	-9.60	-8.57	0.02	-7.90	-6.61	0.45	-7.78	-6.49	0.51	-6.72	-5.08	0.83
Cluster-grid algorithm (CGA)												
1st degree	-4.82	-4.02	14.52	-3.37	-2.73	10.36	-3.91	-3.28	11.69	-3.39	-2.74	11.67
2nd degree	-6.55	-5.83	1.11	-4.89	-4.13	0.30	-5.43	-4.71	0.55	-4.68	-4.01	0.58
3rd degree	-8.03	-6.97	0.57	-6.06	-5.27	0.20	-6.57	-5.57	0.36	-5.74	-4.92	0.43
4th degree	-9.40	-8.02	0.29	-7.10	-5.93	0.13	-7.42	-6.21	0.25	-6.54	-5.51	0.29
5th degree	-9.99	-8.90	0.16	-8.29	-7.22	0.41	-8.14	-6.77	0.53	-7.26	-6.02	1.09

Remark: the number of grid points is equal to $K = 36$; $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units); and *CPU* is computational time in seconds.

Table 4. Role of the integration methods in accuracy and speed under the CGA: the one-country model.

Polynomial degree	Gauss-Hermite quadrature integration									Monte Carlo integration					
	Q(1)			*Q(2)*			*Q(10)*			*1000* nodes			*10000* nodes		
	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*
Model with a closed-form solution, $d = 1$															
1st degree	-3.45	-2.75	10.81	-3.45	-2.75	10.92	-3.45	-2.75	11.13	-3.45	-2.76	40.33	-3.45	-2.75	186.11
2nd degree	-5.43	-4.52	0.23	-5.43	-4.53	0.25	-5.43	-4.53	0.30	-5.41	-4.44	6.17	-5.43	-4.53	35.99
3rd degree	-6.89	-5.89	0.16	-6.90	-5.89	0.19	-6.90	-5.89	0.22	-6.89	-5.92	4.44	-6.90	-5.89	27.77
4th degree	-8.17	-7.24	0.12	-8.15	-7.25	0.13	-8.17	-7.23	0.16	-8.20	-7.27	2.93	-8.17	-7.23	19.98
5th degree	-9.41	-8.37	0.06	-8.52	-8.15	0.25	-9.42	-8.38	0.23	-9.45	-8.35	1.48	-9.42	-8.38	10.45
Benchmark model, $d = 0.025$															
1st degree	-4.31	-3.67	10.92	-4.32	-3.68	11.24	-4.32	-3.68	11.59	-3.86	-3.43	58.76	-4.28	-3.58	259.97
2nd degree	-6.08	-5.42	0.42	-6.12	-5.46	0.26	-6.12	-5.46	0.30	-3.91	-3.90	9.91	-4.56	-4.53	52.24
3rd degree	-7.14	-6.63	0.18	-7.58	-6.92	0.22	-7.58	-6.93	0.26	-3.91	-3.90	4.46	-4.56	-4.55	24.94
4th degree	-7.19	-6.76	0.12	-8.49	-7.97	0.12	-8.91	-7.87	0.14	-3.91	-3.90	3.05	-4.56	-4.55	18.95
5th degree	-7.19	-6.75	0.07	-8.53	-8.37	0.21	-9.99	-8.85	0.24	-3.91	-3.90	1.77	-4.56	-4.55	11.54
Model with highly persistent shocks, $\rho = 0.99$															
1st degree	-3.91	-3.26	10.95	-3.91	-3.28	10.96	-3.91	-3.28	11.37	-3.65	-3.10	65.07	-3.92	-3.22	199.19
2nd degree	-5.19	-4.65	0.46	-5.43	-4.71	0.44	-5.43	-4.71	0.56	-3.68	-3.65	11.53	-4.33	-4.20	42.37
3rd degree	-5.23	-5.09	0.34	-6.57	-5.57	0.29	-6.57	-5.57	0.35	-3.68	-3.67	7.40	-4.33	-4.29	30.50
4th degree	-5.23	-5.19	0.18	-7.42	-6.21	0.20	-7.42	-6.21	0.25	-3.68	-3.67	4.91	-4.33	-4.32	22.85
5th degree	-5.23	-5.19	0.16	-8.06	-6.78	0.49	-8.14	-6.77	0.64	-3.68	-3.67	4.18	-4.33	-4.32	20.06
Model with highly volatile shocks, $\sigma = 0.03$															
1st degree	-3.39	-2.68	10.87	-3.39	-2.74	11.05	-3.39	-2.74	11.50	-3.32	-2.70	67.10	-3.39	-2.69	212.86
2nd degree	-4.64	-4.12	0.46	-4.68	-4.00	0.46	-4.68	-4.01	0.54	-3.44	-3.37	11.62	-4.10	-3.87	48.83
3rd degree	-5.48	-4.91	0.32	-5.74	-4.93	0.32	-5.74	-4.92	0.42	-3.43	-3.40	8.35	-4.08	-4.02	41.72
4th degree	-5.79	-5.28	0.28	-6.58	-5.56	0.17	-6.54	-5.51	0.27	-3.43	-3.41	5.66	-4.08	-4.05	28.64
5th degree	-5.82	-5.35	0.24	-7.26	-6.06	0.69	-7.26	-6.02	0.82	-3.43	-3.41	6.02	-4.08	-4.06	24.48

Remark: $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units); and *CPU* is computational time in seconds.

Table 5. Five integration rules in the solution and testing procedures: the benchmark multi-country model.

Integration rule in the testing procedure	Polynomial degree	Integration rule in the solution procedure									
		Q(3)		*Q(2)*		*M2*		*M1*		*Q(1)*	
		$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$
N=2											
Q(3)	1st degree	-4.089475	-3.187685	-4.089467	-3.187689	-4.089459	-3.187693	-4.089459	-3.187693	-4.068690	-3.194483
	2nd degree	-5.448083	-4.509820	-5.448012	-4.509746	-5.447928	-4.509657	-5.447934	-4.509664	-5.057745	-4.414187
Q(2)	1st degree	-	-	-4.089467	-3.187689	-4.089458	-3.187693	-4.089459	-3.187693	-4.068690	-3.194483
	2nd degree	-	-	-5.448005	-4.509738	-5.447921	-4.509649	-5.447927	-4.509656	-5.057723	-4.414181
M2	1st degree	-	-	-	-	-4.089459	-3.187693	-4.089459	-3.187693	-4.068690	-3.194483
	2nd degree	-	-	-	-	-5.447928	-4.509657	-5.447934	-4.509664	-5.057745	-4.414187
M1	1st degree	-	-	-	-	-	-	-4.089459	-3.187693	-4.068690	-3.194483
	2nd degree	-	-	-	-	-	-	-5.447928	-4.509657	-5.057724	-4.414182
Q(1)	1st degree	-	-	-	-	-	-	-	-	-4.091253	-3.186045
	2nd degree	-	-	-	-	-	-	-	-	-5.448464	-4.510172
N=6											
Q(3)	1st degree	-4.176011	-3.212384	-4.176006	-3.212397	-4.176001	-3.212410	-4.176001	-3.212410	-4.162796	-3.216565
	2nd degree	-5.507474	-4.377885	-5.507450	-4.377705	-5.507413	-4.377522	-5.507413	-4.377526	-4.927777	-4.286892
Q(2)	1st degree	-	-	-4.176006	-3.212397	-4.176001	-3.212410	-4.176001	-3.212410	-4.162796	-3.216565
	2nd degree	-	-	-5.507449	-4.377703	-5.507412	-4.377520	-5.507413	-4.377523	-4.927771	-4.286890
M2	1st degree	-	-	-	-	-4.176001	-3.212410	-4.176001	-3.212410	-4.162796	-3.216565
	2nd degree	-	-	-	-	-5.507413	-4.377522	-5.507413	-4.377526	-4.927777	-4.286892
M1	1st degree	-	-	-	-	-	-	-4.176001	-3.212410	-4.162795	-3.216565
	2nd degree	-	-	-	-	-	-	-5.507413	-4.377522	-4.927764	-4.286889
Q(1)	1st degree	-	-	-	-	-	-	-	-	-4.179199	-3.206127
	2nd degree	-	-	-	-	-	-	-	-	-5.516762	-4.389750

Remark: the number of clusters is equal to $K = 300$; $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units).

Table 6. Effect of the number of clusters on accuracy and speed: the benchmark multi-country model.

Polynomial degree	$K=K_c$			$K=1.2K_c$			$K=2K_c$			$K=5K_c$		
	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*
N=2												
1st degree	-4.08	-3.11	9.78	-4.10	-3.12	10.06	-4.10	-3.13	10.30	-4.03	-3.13	25.29
2nd degree	-5.31	-4.23	14.32	-5.33	-4.23	29.72	-5.49	-4.39	34.68	-5.44	-4.43	54.22
3rd degree	failed to converge			-6.31	-4.80	59.40	-6.48	-5.20	155.38	-6.47	-5.31	171.80
N=6												
1st degree	-3.95	-3.02	13.33	-4.01	-3.16	13.14	-4.12	-3.12	27.91	-4.14	-3.19	32.00
2nd degree	failed to converge			-5.27	-4.35	130.86	-5.47	-4.43	254.65	-5.52	-4.43	467.36
N=10												
1st degree	-3.80	-2.84	41.27	-3.87	-3.10	38.67	-4.05	-3.24	46.26	-4.17	-3.21	70.15
2nd degree	failed to converge			-5.19	-4.28	781.18	-5.47	-4.49	1150.20	-5.58	-4.53	2899.11

Remark: K_c is the minimum possible number of clusters (collocation); $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units); and *CPU* is computational time in seconds.

Table 7. Accuracy and speed under four integration rules: the benchmark multi-country model.

	Polynom. degree *m*	$P_m(N)$	*K*	*Q(2)*			*M2*			*M1*			*Q(1)*		
				$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*
N=2	1st degree	5	300	-4.09	-3.19	38	-4.09	-3.19	53	-4.09	-3.19	44	-4.07	-3.19	45
	2nd degree	15		-5.45	-4.51	108	-5.45	-4.51	150	-5.45	-4.51	114	-5.06	-4.41	85
	3rd degree	35		-6.51	-5.29	237	-6.51	-5.29	398	-6.51	-5.29	212	-5.17	-4.92	121
N=4	1st degree	9	300	-4.13	-3.15	63	-4.13	-3.15	120	-4.13	-3.15	50	-4.11	-3.16	39
	2nd degree	45		-5.47	-4.32	287	-5.47	-4.32	517	-5.47	-4.32	206	-4.95	-4.23	90
N=6	1st degree	13	300	-4.18	-3.21	222	-4.18	-3.21	232	-4.18	-3.21	68	-4.16	-3.22	42
	2nd degree	91		-5.51	-4.38	1282	-5.51	-4.38	1440	-5.51	-4.38	301	-4.93	-4.29	97
N=8	1st degree	17	300	-4.20	-3.25	947	-4.20	-3.25	468	-4.20	-3.25	114	-4.18	-3.26	44
	2nd degree	153		-5.49	-4.51	9511	-5.49	-4.51	3774	-5.49	-4.51	422	-4.91	-4.34	109
N=10	1st degree	21	400	-	-	-	-4.20	-3.24	1090	-4.20	-3.24	182	-4.18	-3.25	59
	2nd degree	231		-	-	-	-5.46	-4.50	12503	-5.46	-4.50	970	-4.90	-4.33	191
N=12	1st degree	25	400	-	-	-	-4.21	-3.28	1403	-4.21	-3.28	233	-4.19	-3.29	63
	2nd degree	325		-	-	-	-5.23	-4.30	69025	-5.23	-4.30	1307	-4.88	-4.34	226
N=16	1st degree	33	1000	-	-	-	-	-	-	-4.22	-3.29	843	-4.19	-3.29	175
	2nd degree	561		-	-	-	-	-	-	-5.44	-4.38	6790	-4.88	-4.27	1058
N=20	1st degree	41	1000	-	-	-	-	-	-	-4.21	-3.29	1238	-4.17	-3.28	184
	2nd degree	861		-	-	-	-	-	-	-5.08	-4.17	16895	-4.83	-4.10	1911
N=30	1st degree	61	4000	-	-	-	-	-	-	-4.23	-3.31	13985	-4.19	-3.29	3529
	2nd degree	1891		-	-	-	-	-	-	-	-	-	-4.86	-4.54	36304
N=40	1st degree	81	4000	-	-	-	-	-	-	-4.23	-3.31	19043	-4.19	-3.29	5321
	2nd degree	3321		-	-	-	-	-	-	-	-	-	-4.86	-4.48	87748
N=60	1st degree	121	1000	-	-	-	-	-	-	-4.13	-3.27	8836	-4.09	-3.25	805
N=100	1st degree	201	1000	-	-	-	-	-	-	-4.09	-3.24	38782	-4.06	-3.23	2174
N=150	1st degree	301	1000	-	-	-	-	-	-	-	-	-	-4.01	-3.22	4204
N=200	1st degree	401	1000	-	-	-	-	-	-	-	-	-	-3.97	-3.20	6316

Remark: $P_m(N)$ is the number of polynomial coefficients in the m-th degree polynomial; *K* is the number of clusters; $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units); and *CPU* is computational time in seconds.

Table 8. Accuracy and speed under four integration rules: sensitivity results for the multi-country model with $N = 6$.

Polynom. degree m	$P_m(N)$	K	$Q(2)$			$M2$			$M1$			$Q(1)$		
			$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	CPU	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	CPU	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	CPU	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	CPU
Model with low risk aversion, $\gamma = 1/5$														
1st degree	13	300	-4.60	-3.55	249	-4.60	-3.55	275	-4.60	-3.55	72	-4.57	-3.53	38
2nd degree	91		-6.05	-4.89	1193	-6.05	-4.89	1510	-6.05	-4.89	279	-5.21	-4.72	87
Model with high risk aversion, $\gamma = 5$														
1st degree	13	300	-3.31	-2.35	2024	-3.31	-2.35	1953	-3.31	-2.35	525	-3.31	-2.37	240
2nd degree	91		-4.39	-3.35	7707	-4.39	-3.35	8901	-4.39	-3.35	1891	-4.11	-3.51	639
Model with highly persistent shock, $\rho = 0.99$														
1st degree	13	300	-3.77	-2.37	305	-3.77	-2.37	337	-3.77	-2.37	105	-3.76	-2.37	60
2nd degree	91		-4.72	-3.43	1643	-4.72	-3.43	1937	-4.72	-3.43	401	-4.55	-3.38	129
Model with highly volatile shocks, $\sigma = 0.03$														
1st degree	13	300	-3.16	-1.84	333	-3.16	-1.84	362	-3.16	-1.84	108	-3.14	-1.86	63
2nd degree	91		-3.93	-2.58	1800	-3.93	-2.58	2062	-3.93	-2.58	422	-3.74	-2.60	160

Remark: $P_m(N)$ is the number of polynomial coefficients in the m-th degree polynomial; K is the number of clusters; $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum Euler equation errors (in log10 units); and CPU is computational time in seconds.