Journal of Economic Dynamics & Control 35 (2011) 207–228

Contents lists available at ScienceDirect

Journal of Economic Dynamics & Control

journal homepage: www.elsevier.com/locate/jedc

Solving the multi-country real business cycle model using ergodic set methods

Serguei Maliar [a,b,*], Lilia Maliar [a,b], Kenneth Judd [b]

[a] University of Alicante, Spain
[b] Hoover Institution at Stanford University, United States

ARTICLE INFO

Article history:
Received 10 May 2010
Accepted 26 August 2010
Available online 1 October 2010

JEL classification:
C63

Keywords:
Heterogeneous agents
Numerical methods
Stochastic simulation
Parameterized expectations algorithm
Projection
Perturbation

ABSTRACT

We use the stochastic simulation algorithm, described in Judd et al. (2009), and the cluster-grid algorithm, developed in Judd et al. (2010a), to solve a collection of multi-country real business cycle models. The following ingredients help us reduce the cost in high-dimensional problems: an endogenous grid enclosing the ergodic set, linear approximation methods, fixed-point iteration and efficient integration methods, such as non-product monomial rules and Monte Carlo integration combined with regression. We show that high accuracy in intratemporal choice is crucial for the overall accuracy of solutions and offer two approaches, precomputation and iteration-on-allocation, that can solve for intratemporal choice both accurately and quickly. We also implement a hybrid solution algorithm that combines the perturbation and accurate intratemporal-choice methods.

1. Introduction

In the present paper, we show how to apply two ergodic-set algorithms for solving a collection of multi-country real business cycle models proposed by Den Haan, Judd and Juillard (this issue). One of these algorithms is the stochastic simulation algorithm (SSA) described in Judd, Maliar and Maliar (2009, 2010b).[1,2] The other is the projection cluster-grid algorithm (CGA) developed in Judd, Maliar and Maliar (2010a). The models studied in the current JEDC project include up to 10 countries (i.e., 20 state variables) and feature heterogeneity in fundamentals (preferences and technology) and endogenous labor-leisure choice, as well as complete markets, adjustment costs, continuously valued state variables, and non-additively separable preferences and technology.

SSA and CGA build on strategies that allow to reduce the cost of finding global solutions in high-dimensional applications. The first and most important distinctive feature of these two methods is that they operate on endogenous domains which enclose the ergodic set: SSA computes a solution on a set of simulated points, whereas CGA does so on a grid of points constructed by clustering simulated data (the center of each cluster represents one grid point). Focusing on

* Corresponding author at: T24, Hoover Institution, 434 Galvez Mall, Stanford University, Stanford, CA 94305-6010, USA.
E-mail address: maliars@stanford.edu (S. Maliar).

[1] Judd et al. (2009) present SSA in the context of a one-country model. In a more recent version of the paper, Judd et al. (2010b) extend the results to include the case of a multi-country model similar to Model 1 of the current JEDC project.

[2] SSA is similar to the simulation-based parameterized expectations algorithm (PEA) by Marcet (1988) and Den Haan and Marcet (1990) in how it uses stochastic simulation to compute an ergodic distribution, its support and the associated policy functions. SSA differs from the simulation-based PEA in that it relies on a mixture of techniques that ensures numerical stability; see Judd et al. (2009) for a discussion.

0165-1889/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.jedc.2010.09.014

the ergodic set allows us to avoid the costs associated with computing solutions in those areas of the state space that are never visited in equilibrium. Second, to approximate policy functions, SSA and CGA use polynomials with additively separable terms, estimate the polynomial coefficients using numerically stable linear approximation methods and update the coefficients along iterations using a fixed-point iteration method. These choices ensure that the cost of approximating the policy functions does not increase significantly with the dimensionality of the problem (in particular, because we iterate on policy functions of all countries simultaneously rather than country by country). Finally, to evaluate conditional expectations, SSA and CGA rely on integration methods that are particularly suitable for high-dimensional applications. Namely, SSA combines Monte Carlo integration and regression in a manner that makes it possible to approximate expectations (integrals) in all simulated points at once, whereas CGA performs numerical integration using low-cost non-product monomial rules and the product Gauss–Hermite rule with small numbers of nodes in each dimension (including the rule with one node).

The models considered in Judd et al. (2010a, 2010b) include up to 200 countries and thus, are more challenging in the dimensional aspect than those studied in the JEDC project. However, the models of the JEDC project are more challenging in another aspect, namely, in that solving for consumption and labor of heterogeneous countries is a non-trivial task. Let us separate the intertemporal choice (capital) and intratemporal choice (consumption and labor). The intertemporal choice is concerned with dynamics and is characterized by the capital policy functions defined in terms of state variables. Such functions are also the laws of motion for capital and allow to compute an entire capital path without solving for consumption and labor. The intratemporal-choice problem is static: Given a capital path, in each period of time, we must solve a system of static optimality conditions (including a resource constraint) with respect to consumption and labor. In the models of the JEDC project, this system cannot be solved analytically. Solving this system numerically a large number of times (in each time period, iteration, grid point, integration node) is costly, especially when the number of countries is large. Moreover, as we show in the present paper, the intratemporal choice must be computed with a high degree of accuracy; otherwise, the overall accuracy of solutions is low.

In the present paper, we describe two novel intratemporal-choice approaches that can find the consumption and labor allocations both accurately and quickly. Our first approach, called iteration-on-allocation, relies on a numerical solver that implements fixed-point iteration directly on the intratemporal-choice variables. (The policy functions for the intratemporal-choice variables are never constructed explicitly.) This approach allows us to achieve effectively zero errors in all intratemporal-choice conditions (including the resource constraint) so that the only source of errors for us is Euler-equation errors.[3] The iteration-on-allocation solver does not require derivatives (Jacobian and Hessian), and its cost does not increase significantly with dimension. Moreover, it can work with vectors and matrices, making it fast in vectorized applications.

Our second approach, called precomputation, constructs the intratemporal-choice functions on an appropriately chosen grid of points outside the main iterative cycle and uses the precomputed functions to interpolate the intratemporal choice inside the main iterative cycle as if a closed-form solution was available.[4] Like iteration-on-allocation, this approach can work with vectors and matrices and attains high accuracy in the examples considered.

The iteration-on-allocation and precomputation approaches can be vectorized because of the separation of the intertemporal and intratemporal choices. Given the laws of motion for capital, we construct a capital path without solving for the intratemporal-choice variables (consumption and labor). Then, given a capital path, we compute all the consumption and labor allocations at once rather than one by one.

The accuracy and speed of the SSA and CGA methods under our baseline implementation are assessed in Kollmann, Maliar, Malin and Pichler (this issue-b). In the present paper, we report a few additional experiments that show how the performance of the CGA method depends on the specific integration method, approximating polynomial function and intratemporal-choice approach.

First, we find that CGA can compute solutions of essentially the same accuracy as those submitted for the comparison in Kollmann et al. (this issue-b) but at a significantly lower cost. Our baseline integration method (used to compute solutions submitted to the comparison in Kollmann et al., this issue-b) is an accurate but expensive two-step procedure that combines a cheap non-product monomial rule and an expensive product Gauss–Hermite rule. It turns out that such a high-accuracy integration method is not needed for the models of the JEDC project since cheaper and less accurate integration methods also produce accurate solutions. For example, it took us 35 h to solve an asymmetric 10-country model using our two-step integration procedure (see Table 3 in Kollmann et al., this issue-b). In the present paper, we solve the same model in 7 min using only the first step of our two-step integration procedure without a visible accuracy loss (solution errors are identical up to the fourth digit). Moreover, using the one-node Gauss–Hermite rule advocated in Judd et al. (2010a), we solve this model in 2 min with a modest accuracy loss (maximum error increases by 5%).

Second, we find that the third-degree polynomial delivers solutions that are almost an order of magnitude more accurate than those produced by our baseline second-degree polynomial (used to generate the results for the comparison in Kollmann et al., this issue-b). We also find that the Smolyak polynomial, used in the Smolyak collocation algorithm by

[3] This approach was originally proposed in the context of PEA in Maliar and Maliar (2004) and was later implemented for SSA and CGA in the present paper.

[4] Maliar and Maliar (2005) introduce the precomputation approach in the context of the standard neoclassical growth model for computing labor-leisure choice outside the main iterative cycle. Maliar and Maliar (2007) implement this approach in the context of the current JEDC project.

Malin, Krueger and Kubler (this issue), allows CGA to achieve nearly the same accuracy as does the third-degree polynomial. However, under the Smolyak polynomial, the cost grows less rapidly with dimension than under the third-degree polynomial (independently of dimension, the Smolyak polynomial has only four times more terms than the second-degree polynomial).

Third, our approach in which consumption and labor are approximated by functions of both the current-period state variables and the current-period capital choices (i.e., the next-period values of the endogenous state variables) delivers much better accuracy than the standard intratemporal-choice approach that approximates policy functions for consumption and labor by functions of state variables only. To be more specific, we approximate the consumption policy function(s) by a polynomial of the same (second) degree as that used to approximate the capital policy functions, and we obtain substantially larger approximation errors in the intratemporal-choice conditions than in the intertemporal-choice conditions (Euler equations).[5] For comparison, we also solve the same model using the precomputation approach. Neither our baseline iteration-on-allocation approach (used to compute the solutions reported in Kollmann et al., this issue-b) nor our precomputation approach (implemented in the present paper) restrict the overall accuracy of solutions.

Finally, we propose a way to increase the accuracy of the solution methods that do not accurately compute the intratemporal choice in their own solution procedures. We specifically take the capital policy functions delivered by such a method and replace its low-accuracy solution for consumption and labor with a high-accuracy solution (computed by the iteration-on-allocation and precomputation approaches). In our examples, this replacement increases the overall accuracy of solutions by an order of magnitude. We apply this idea for constructing a hybrid solution algorithm that combines the perturbation method (a cheap way to compute capital policy functions) and our intratemporal-choice methods (a cheap way to accurately compute consumption and labor allocations).

The rest of the paper is as follows: Section 2 presents the model and derives the first-order conditions. Section 3 describes how SSA and CGA address challenges of high-dimensional problems. Section 4 develops two approaches for computing the intratemporal choice. Section 5 outlines the steps of SSA and CGA. Section 6 describes the baseline and alternative implementations of SSA and CGA. Section 7 presents the numerical results for CGA and constructs a hybrid of the perturbation and accurate intratemporal-choice methods. Finally, Section 8 concludes.

2. The model

We consider a model with a finite number of countries, N, in which each country is populated by a representative consumer. A social planner maximizes a weighted sum of the expected lifetime utilities of the countries' representative consumers subject to the aggregate resource constraint, i.e.,

$$\max_{\{c_t^j,\ell_t^j,k_{t+1}^j\}_{t=0,\ldots,\infty}^{j=1,\ldots,N}} E_0\sum_{j=1}^{N}\tau^j\left(\sum_{t=0}^{\infty}\beta^t u^j(c_t^j,\ell_t^j)\right) \tag{1}$$

subject to

$$\sum_{j=1}^{N}c_t^j=\sum_{j=1}^{N}\left[a_t^j f^j(k_t^j,\ell_t^j)-\frac{\phi}{2}k_t^j\left(\frac{k_{t+1}^j}{k_t^j}-1\right)^2+k_t^j-k_{t+1}^j\right], \tag{2}$$

where E_t is the operator of conditional expectation; c_t^j, ℓ_t^j, k_t^j, a_t^j, u^j, f^j and τ^j are, respectively, consumption, labor, capital, productivity level, utility function, production function and welfare weight of a country $j\in\{1,\ldots,N\}$; β is the discount factor; and ϕ is the adjustment-cost parameter. Initial condition $(\boldsymbol{k}_0,\boldsymbol{a}_0)$ is given, where $\boldsymbol{k}_0\equiv(k_0^1,\ldots,k_0^N)$ and $\boldsymbol{a}_0\equiv(a_0^1,\ldots,a_0^N)$. The process for productivity levels in country j is given by

$$\ln a_t^j=\rho\ln a_{t-1}^j+\sigma\varepsilon_t^j, \tag{3}$$

where $\varepsilon_t^j\equiv e_t+e_t^j$ with e_t and e_t^j being common and country-specific productivity shocks, respectively, $e_t,e_t^j\sim\mathcal{N}(0,1)$; ρ is the autocorrelation coefficient of the productivity level; and σ determines the standard deviation of the productivity level.

An interior solution to the social planner's problem (1)–(3) satisfies first-order conditions (FOCs) of the form

$$u_c^j(c_t^j,\ell_t^j)\tau^j=u_c^{j'}(c_t^{j'},\ell_t^{j'})\tau^{j'}, \tag{4}$$

$$u_\ell^j(c_t^j,\ell_t^j)=-u_c^j(c_t^j,\ell_t^j)a_t^j f_\ell^j(k_t^j,\ell_t^j), \tag{5}$$

$$u_c^j(c_t^j,\ell_t^j)\omega_t^j=\beta E_t\{u_c^j(c_{t+1}^j,\ell_{t+1}^j)[\pi_{t+1}^j+a_{t+1}^j f_k^j(k_{t+1}^j,\ell_{t+1}^j)]\}, \tag{6}$$

[5] The importance of accuracy in intratemporal choice for the overall accuracy of solutions is also seen from Table 6 of the comparison paper by Kollmann et al. (this issue-b).

where $j, j' \in \{1,\ldots,N\}$, and ω_t^j and π_t^j are defined as

$$\omega_t^j \equiv 1+\phi\left(\frac{k_{t+1}^j}{k_t^j}-1\right),$$

$$\pi_t^j \equiv 1+\frac{\phi}{2}\left(\frac{k_{t+1}^j}{k_t^j}-1\right)\left(\frac{k_{t+1}^j}{k_t^j}+1\right).$$

Here, and further on, notation of type F_{x_m} stands for the first-order partial derivative of a function $F(\ldots,x_m,\ldots)$ with respect to a variable x_m.

In the project, eight models are considered. Models 1–4 have the same types of preferences and technology as do Models 5–8, respectively, however, the former models assume identical preferences and technology parameters for all countries, while the latter models have different parameters across countries. Models 1 and 5 do not have endogenous labor-leisure choice, while the other models do. A description of the models studied is provided in Juillard and Villemot (this issue).

Intertemporal versus intratemporal choices: Let us make a distinction between intertemporal and intratemporal choices. Consider a capital policy function that determines a country's j end-of-period capital stock, k_{t+1}^j, as a function of the current state variables, $\boldsymbol{k}_t$ and $\boldsymbol{a}_t$,

$$k_{t+1}^j = K^j(\boldsymbol{k}_t,\boldsymbol{a}_t), \tag{7}$$

where $\boldsymbol{k}_t \equiv (k_t^1,\ldots,k_t^N)$ and $\boldsymbol{a}_t \equiv (a_t^1,\ldots,a_t^N)$. We call k_{t+1}^j an *intertemporal-choice* variable because it captures dynamic aspects of the planner's choice. A capital policy function is an equilibrium law of motion for capital.

For each period t, given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$ and $\boldsymbol{k}_{t+1}$, we must solve a system of $2N$ static optimality conditions (i.e., one resource constraint (2), $N-1$ conditions (4) and N conditions (5)) with respect to $\boldsymbol{c}_t \equiv (c_t^1,\ldots,c_t^N)$ and $\boldsymbol{\ell}_t \equiv (\ell_t^1,\ldots,\ell_t^N)$. A solution to this system is given by solution manifolds for consumption and labor:

$$c_t^j = \Phi^j(\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1}) \quad \text{and} \quad \ell_t^j = \Theta^j(\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1}),\ j=1,\ldots,N. \tag{8}$$

We refer to consumption $\boldsymbol{c}_t$ and labor $\boldsymbol{\ell}_t$ as *intratemporal-choice* variables because under our representation, such variables are determined within period t if the state, $(\boldsymbol{k}_t,\boldsymbol{a}_t)$, and the intertemporal choice, $\boldsymbol{k}_{t+1}$, are given. For Model 1, the intratemporal choice can be expressed analytically, while for Models 2–8, it must be approximated numerically.

3. Addressing challenges of high dimensions

The high-dimensional models described in Den Haan et al. (this issue) pose four challenges for numerical methods designed to find a global solution: (i) a large size of the domain on which the solution is computed, (ii) a high cost of finding the polynomial coefficients in the approximating polynomial functions, (iii) a large number of integration nodes for evaluating the conditional expectation functions, and (iv) a high cost of solving for the intratemporal choice.

Judd et al. (2010a, 2010b) show how to address the first three challenges in the context of CGA and SSA, respectively. The problems solved in Judd et al. (2010a, 2010b) are of higher dimensionality (they include up to 200 countries) but simpler in the structure of the intratemporal choice (which can be characterized analytically) than those studied in the current JEDC project. The strategies used by Judd et al. (2010a, 2010b) to address challenges (i)–(iii) are discussed in Sections 3.1–3.3, respectively, and the coordination of these strategies is described in Section 3.4. The last challenge, (iv), which is concerned with the intratemporal choice in high-dimensional problems, is not studied in Judd et al. (2010a, 2010b). In the present paper, solving for the intratemporal choice accurately proved to be crucial for the overall accuracy of solutions. We address the intratemporal-choice challenge separately, in Section 4.

3.1. Multi-dimensional domain

To make a numerical method suitable for high-dimensional applications, we must restrict attention to a relatively small set of grid points in the multi-dimensional space.[6] Both SSA and CGA achieve this goal by focusing on the ergodic set of points realized in equilibrium. In Fig. 1a, we show the ergodic set constructed from a simulated series of 10,000 observations, which are produced by the standard representative-agent neoclassical stochastic growth model. SSA computes the solution on the given set of simulated points (the number of simulated points is controlled by the researcher and need not necessarily increase with the number of countries, N). CGA chooses a more efficient representation of the ergodic set; namely, it replaces a large number of closely situated simulated points with relatively few representative points constructed by grouping similar points into clusters (the number of representative points is again controlled by the researcher). To be specific, CGA transforms correlated variables into uncorrelated principal components (denoted PC_t^1 and PC_t^2) using principal components (PCs) decomposition (see Fig. 1b); normalizes the principal components to zero mean and

[6] The literature commonly considers a multi-dimensional hypercube domain composed of the tensor product of discretized state variables. In this case, the total number of grid points grows exponentially with the dimensionality of the state space (the curse of dimensionality).

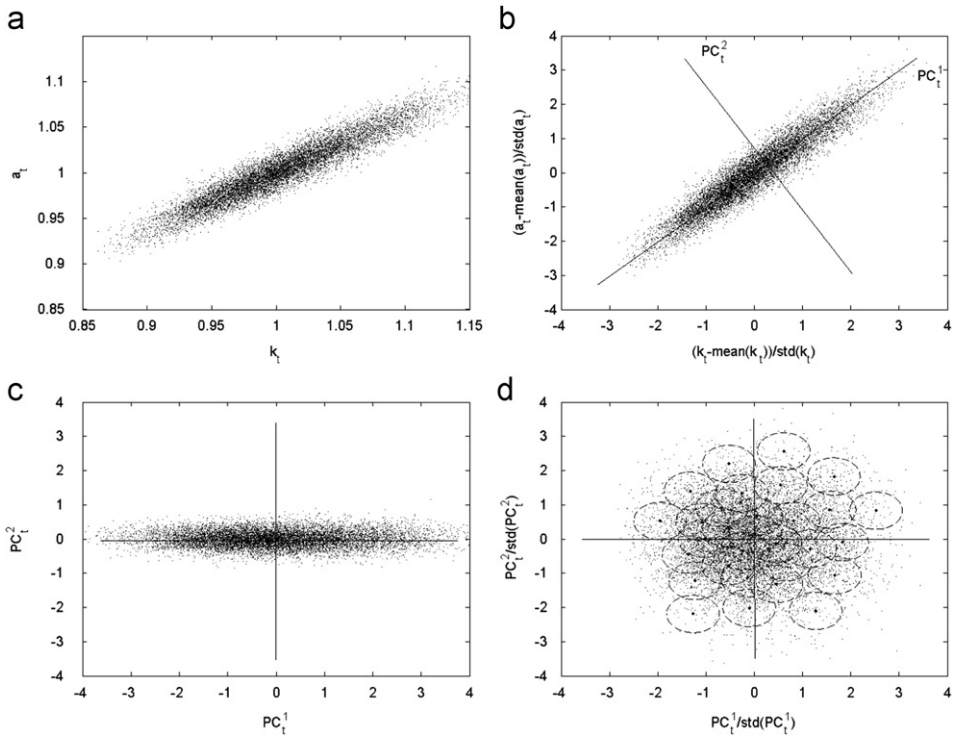

Fig. 1. (a) Ergodic distribution. (b) Normalized ergodic distribution. (c) PCs of ergodic distribution. (d) Normalized PCs of ergodic distribution and 30 clusters. *Note*: (a–d) show simulated series produced by the standard representative-agent neoclassical growth model. k_t and a_t are the capital stock and productivity level, respectively, and PCs are the corresponding principal components.

unit variance (see Fig. 1c); and constructs I clusters applying a clustering algorithm. It subsequently uses the centers of the constructed clusters as a grid for projections; see Judd et al. (2010a) for details. (Note that CGA does not compute different solutions in each cluster, but a global solution on the entire cluster grid.) Making the domain endogenous to the model allows SSA and CGA to compute a solution only in the relevant area of the state space (an ellipsoid area shown in Fig. 1a). This eliminates an enormously large number of grid points that are never visited in equilibrium. For example, for a model with 100 state variables, a hypersphere is only about a 2×10^{-70} fraction of a multi-dimensional hypercube which encloses the hypersphere; see Judd et al. (2010a) for a further discussion.

3.2. Multi-dimensional polynomials

SSA and CGA parameterize policy functions by an additively separable complete polynomial. For example, parameterizing the end-of-period capital stock of a country j by a first-degree polynomial yields

$$k_{t+1}^j = \nu_0^j + \nu_1^j k_t^1 + \cdots + \nu_N^j k_t^N + \nu_{N+1}^j a_t^1 + \cdots + \nu_{2N}^j a_t^N, \tag{9}$$

where $(\nu_0^j,\nu_1^j,\ldots,\nu_N^j,\nu_{N+1}^j,\ldots,\nu_{2N}^j)' \equiv \boldsymbol{\nu}^j \in \mathbb{R}^{(2N+1)\times 1}$ is the country's j vector of the polynomial coefficients. The number of polynomial terms in the first-, second- and third-degree complete polynomials grows with the dimensionality of the problem linearly, quadratically and cubically, respectively; see Table 1 in Judd et al. (2010a). When the dimensionality is large, high-degree polynomials are costly.

The assumption of additively separable polynomials allows us to estimate the polynomial coefficients using fast and numerically stable linear approximation methods, such as the least-squares methods using SVD and QR factorization, Tikhonov regularization, least-absolute deviation methods and the principal components method; see Judd et al. (2009). Moreover, it allows us to estimate the polynomial coefficients for all N countries at once rather than country by country.

To update the polynomial coefficients along iterations, SSA and CGA use fixed-point iteration, which is a simple derivative-free iteration method whose cost does not significantly increase with the dimensionality of the problem; see Judd (1998, pp. 555–557).[7] Fixed-point iteration computes the coefficients for the next iteration as a weighted average of

[7] Fixed-point iteration is simpler to implement than alternative iterative schemes such as time iteration (that involves finding a solution to a system non-linear equation; see Judd, 1998, pp. 553–555) or Newton methods (that require to compute Jacobian and Hessian matrices and to use optimization

the coefficients at the beginning and at the end of the previous iteration. It works for vectors and matrices and allows to iterate on policy functions of all countries simultaneously. A shortcoming of fixed-point iteration is that it does not guarantee convergence. However, a slow updating is typically sufficient to ensure convergence; see Section 4.2 for a discussion. In the models of the JEDC project, fixed-point iteration was always numerically stable.

3.3. Multi-dimensional integration

SSA and CGA require the calculation of integrals that represent conditional expectation functions in the Euler equations. SSA employs Monte Carlo integration combined with regression, as is used in Den Haan and Marcet (1990). This integration procedure makes it possible to infer expectations (to compute integrals) simultaneously in all simulated points. If the length of simulations T is held fixed, the cost of this integration procedure does not grow substantially with the dimensionality of the problem (though accuracy may decrease as more polynomial coefficients must be identified); see Judd et al. (2010b) for the corresponding results. In particular, Judd et al. (2010b) find that under T=10,000, first- and second-degree complete polynomials are feasible for a model (similar to Model 1 of the present project) with up to N=200 and up to N=30 countries, respectively.

CGA is a projection method and relies on deterministic methods of integration. The choice of an integration method depends on the dimensionality of the problem. One-dimensional integrals can be computed accurately using the Gaussian quadrature approach (as is done, for example, in Judd's, 1992, Galerkin algorithm). For a given weighting function $w(\varepsilon)$, Gaussian quadrature approximates an integral (conditional expectation) by $E[G(\varepsilon)]=\int_{\mathbb{R}} G(\varepsilon)w(\varepsilon)\,d\varepsilon \approx \sum_{h=1}^{H} w_h G(\varepsilon_h)$ for some nodes $\{\varepsilon_h\}_{h=1,\ldots,H}$ and weights $\{w_h\}_{h=1,\ldots,H}$. One can extend the Gaussian quadrature approach to multi-dimensional integration problems using a product rule. However, product rules are not feasible in high-dimensional problems due to the curse of dimensionality: the total number of integration nodes H^N increases exponentially with dimension. To reduce the cost of numerical integration in economic applications of high dimensionality, Judd (1998) proposes to use non-product monomial integration formulas; see Judd (1998, pp. 271, 331). A large collection of such formulas is available in Stroud (1971).

Judd et al. (2010a) elaborate the monomial formulas for a heterogeneous-agent model similar to those studied in the present paper, illustrate the use of such formulas by way of examples and provide an exhaustive comparison of the performance of the CGA method under different integration strategies. Such strategies include the product Gauss–Hermite rule with 1, 2 and 3 nodes in each dimension (referred to as $Q(1)$, $Q(2)$ and $Q(3)$, respectively) and non-product monomial rules with $2N$ and $2N^2+1$ nodes (referred to as $M1$ and $M2$, respectively). Using second-degree polynomials, Judd et al. (2010a) find that the integration formulas $Q(3)$, $Q(2)$, $M2$, $M1$ and $Q(1)$ are feasible for the models with up to N=6, 8, 12, 20 and 40 countries, respectively. Using first-degree polynomials, Judd et al. (2010a) find that the formulas $M1$ and $Q(1)$ are feasible for the models with up to N=100 and 200 countries, respectively.

3.4. Coordinating the approximation, integration and intratemporal-choice strategies

As is argued in Judd et al. (2010a), making a numerical method cost-efficient requires proper coordination between the approximation and integration strategies. For example, if a polynomial approximation of a given degree can deliver accuracy of no more than 10^{-4}, it would be inefficient to compute integrals with accuracy of 10^{-10} (doing so would increase costs without increasing the overall accuracy of the solutions). It is therefore important to identify combinations of the approximation and integration strategies that are well matched in terms of accuracy.

Judd et al. (2010a) identifies the following regularities: For a first-degree polynomial, all integration methods lead to the same level of accuracy, including the one-node Gauss–Hermite quadrature rule. For a second-degree polynomial, the two- and three-node Gauss–Hermite rule and the monomial formulas lead to the same level of accuracy (up to the fourth digit), while the one-node Gauss–Hermite rule leads to Euler-equation errors that are 5–10% larger than those calculated with more accurate integration methods. Judd et al. (2010a) give an example of coordination between the approximation and integration strategies that consists in combining the second-degree polynomial and the one-node Gauss–Hermite integration rule. This combination makes it possible to increase the number of countries, N, from 20 to 40 at a cost of a small decrease in accuracy.

In the presence of endogenous labor-leisure choice, the approximation and integration strategies should be properly coordinated not only with each other, but also with the intratemporal-choice strategy. The numerical results in Section 7.1 show that insufficient accuracy in intratemporal choice can drastically reduce the overall accuracy of the solutions; the importance of accuracy in intratemporal choice is also seen in Table 6 of the comparison paper by Kollmann et al. (this issue-b).

(*footnote continued*)
methods; see Judd, 1998, pp. 103–119). Also, Gaspar and Judd (1997) argue that fixed-point iteration has a lower computational cost than time iteration and Newton methods for problems of medium and high dimensionality.

4. Intratemporal choice

In Section 4.1, we discuss intratemporal-choice approaches that currently exist in the literature. In Sections 4.2 and 4.3, we describe two novel approaches, iteration-on-allocation and precomputation, that allow us to solve for the intratemporal choice both accurately and quickly. Finally, in Section 4.4, we show that combining iteration-on-allocation and precomputation can produce additional gains in speed.

4.1. Standard intratemporal-choice approaches

Existing literature provides two approaches to computing the intratemporal choice. First, given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$ and $\boldsymbol{k}_{t+1}$, one can solve a system of $2N$ equations, (2), (4) and (5), with respect to $2N$ unknowns, $\boldsymbol{c}_t$ and $\boldsymbol{\ell}_t$, using a standard Newton method. The cost of this approach can be prohibitive because we must find a numerical solution to the $2N$-dimensional system of equations a large number of times (in each time period, grid point, integration node) within an iterative cycle.

Second, one can treat the intratemporal choice in the same way as the intertemporal choice, i.e., one can compute the policy functions for the intratemporal-choice variables, $c_t^j = C^j(\boldsymbol{k}_t,\boldsymbol{a}_t)$ and $\ell_t^j = L^j(\boldsymbol{k}_t,\boldsymbol{a}_t)$ satisfying the corresponding optimality conditions (2), (4) and (5) inside the main iterative cycle. (In contrast to the intratemporal-choice manifolds $\Phi^j(\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1})$ and $\Theta^j(\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1})$ in (8) defined for any $\boldsymbol{k}_{t+1}$, the policy functions $C^j(\boldsymbol{k}_t,\boldsymbol{a}_t)$ and $L^j(\boldsymbol{k}_t,\boldsymbol{a}_t)$ do not include $\boldsymbol{k}_{t+1}$ as an argument because such functions are defined for the equilibrium intertemporal choice, $\boldsymbol{k}_{t+1}$, determined by the capital policy functions (7)). In our experiments, this approach does not produce sufficiently accurate results; see Section 7.1. Moreover, simultaneous iterations on policy functions for the intertemporal- and intratemporal-choice variables reduce both the numerical stability and the speed of convergence.

4.2. Iteration-on-allocation

The first intratemporal-choice approach we use relies on a numerical solver, (*fixed-point*) *iteration-on-allocation*, proposed by Maliar and Maliar (2004). This method's name emphasizes that fixed-point iteration is applied to the intratemporal-choice allocations and distinguishes it from a different fixed-point iteration procedure, described in Section 5, that is applied to the coefficients of an approximating polynomial function.

The iteration-on-allocation approach proceeds as follows:

- *Step* 1. Re-write conditions (2), (4) and (5) to define a mapping g that explicitly and uniquely maps a set of values $\boldsymbol{z}_t = (\boldsymbol{c}_t,\boldsymbol{\ell}_t)$ into a new set of values $\tilde{\boldsymbol{z}}_t = (\tilde{\boldsymbol{c}}_t,\tilde{\boldsymbol{\ell}}_t)$ (this is possible to do for all of the eight models studied in the current JEDC project):
$$\tilde{\boldsymbol{z}}_t = g(\boldsymbol{z}_t). \tag{10}$$
- *Step* 2. Use some initial guess on $\boldsymbol{z}_t$ and calculate $\tilde{\boldsymbol{z}}_t$ via mapping (10).
- *Step* 3. Use partial updating (damping) to compute an input for the next iteration as $(1-\varsigma)\boldsymbol{z}_t+\varsigma\tilde{\boldsymbol{z}}_t$, where $\varsigma \in (0,1)$ is a damping parameter.

  Iterate until a fixed point, $\boldsymbol{z}_t = g(\boldsymbol{z}_t)$, is found with a given degree of accuracy, i.e.,
$$\frac{1}{\varsigma \cdot T}\sum_{t=1}^{T}\left\|\frac{\tilde{\boldsymbol{z}}_t-\boldsymbol{z}_t}{\boldsymbol{z}_t}\right\| < 10^{-\theta}, \tag{11}$$

where $\theta > 0$, and $\|\cdot\|$ is some vector norm.

On the initial iteration, we can assume that $\boldsymbol{z}_t$ is equal to its steady-state value. Typically, we need not iterate on all $2N$ unknown elements of $\boldsymbol{c}_t$ and $\boldsymbol{\ell}_t$ since there are explicit closed-form expressions relating these variables, and fixing one or a few of them allows to analytically find the values of all the intratemporal-choice variables. As an example, we describe how to construct a mapping of type (10) for Model 5; the mappings for Models 6–8 are given in Appendix A.

Example 1 (*Model 5*). There is no labor-leisure choice, so condition (5) is absent. The remaining intratemporal-choice conditions (4) and (2), written in a form suitable for iteration-on-allocation, respectively, are

$$\tilde{c}_t^j = [(c_t^1)^{-1/\gamma^1}\tau^1/\tau^j]^{-\gamma^j}, \quad j = 2,\ldots,N, \tag{12}$$

$$\tilde{c}_t^1 = \sum_{j=1}^{N}\left[k_t^j + a_t^j A(k_t^j)^{\alpha} - \frac{\phi}{2}k_t^j\left(\frac{k_{t+1}^j}{k_t^j}-1\right)^2 - k_{t+1}^j\right] - \sum_{j=2}^{N}\tilde{c}_t^j, \tag{13}$$

where $\{\gamma^j\}^{j=1,\ldots,N}$ are the utility-function parameters, and A is a normalizing constant. For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$ and $\boldsymbol{k}_{t+1}$, Eqs. (12) and (13) define a mapping $\tilde{c}_t^1 = g(c_t^1)$. We iterate on consumption of the first country, c_t^1, as follows: assume some value for c_t^1; compute $\{\tilde{c}_t^j\}^{j=2,\ldots,N}$ from (12); obtain $\tilde{c}_t^1$ from (13); if $c_t^1 \neq \tilde{c}_t^1$, compute the input for the next iteration as $(1-\varsigma)c_t^1+\varsigma\tilde{c}_t^1$. Iterate until convergence.

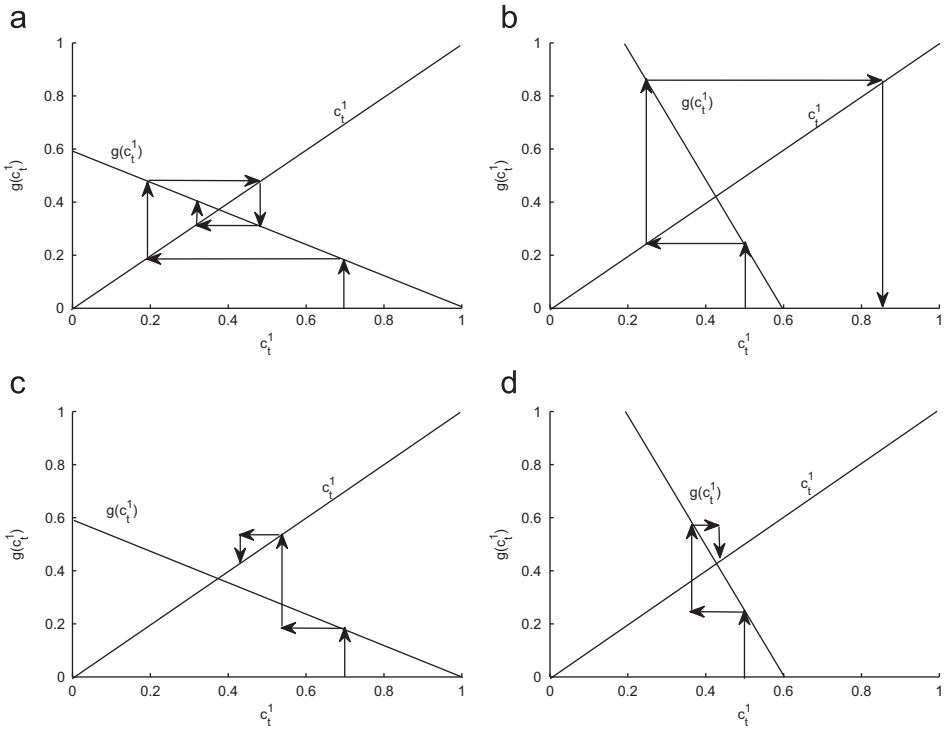

Fig. 2. (a) Iteration-on-allocation under $-1<g'<0$: convergence without damping. (b) Iteration-on-allocation under $g'<-1$: divergence without damping. (c) Iteration-on-allocation under $-1<g'<0$: convergence with damping. (d) Iteration-on-allocation under $g'<-1$: convergence with damping. *Note*: (a–d) illustrate possible outcomes of the iteration-on-allocation method; c_t^1 is consumption of the first country; g is the mapping used in the iteration-on-allocation method and g' is the first derivative of g.

Iteration-on-allocation has two advantages over standard Newton-type methods. First, iteration-on-allocation does not require to compute derivatives (such as Jacobian or Hessian) but instead performs a straightforward summation; as a result, its cost does not increase considerably with the dimensionality of the problem. Second, iteration-on-allocation can operate on a time series or on all grid points simultaneously while Newton-type methods compute a solution point by point and are more difficult to vectorize.

Convergence of fixed-point iteration is in general not guaranteed; for formal results about convergence of fixed-point iteration, see Judd (1998, pp. 165–166). However, damping can often help achieve convergence. In particular, by choosing an appropriate value of the damping parameter, ς, we were able to achieve convergence in all eight models of the current JEDC project. Below, we discuss the issue of convergence using Model 5 as an example.

Example 2 (*Model 5*). Conditions (12) and (13) together imply

$$c_t^1 = g(c_t^1) \equiv c_t - \sum_{j=2}^{N}\left[\frac{\tau^1}{\tau^j}(c_t^1)^{-1/\gamma^1}\right]^{-\gamma^j}, \tag{14}$$

where c_t is the aggregate consumption that is given. Note that if $\{\gamma^j\}^{j=1,\ldots,N}$ are of the same sign, then $g'(c_t^1)<0$. There is a unique fixed point $(c_t^1)^*$ satisfying $(c_t^1)^* = g((c_t^1)^*)$ (at this point, $g(c_t^1)$ crosses the 45° line). However, applying g iteratively to some initial guess c_t^1, i.e., $g(\ldots g(g(c_t^1)))$ does not guarantee the convergence to this fixed point. Depending on whether g' is larger than, smaller than or equal to minus one, the result will be convergence, divergence or cycling, respectively. (Note that the slope of g depends on the model's parameters and welfare weights, as well as on the specific way in which g is constructed.) Figs. 2a and b show, respectively, the cases of convergence and divergence of fixed-point iteration. Figs. 2c and d illustrate fixed-point iteration with damping $(1-\varsigma)c_t^1+\varsigma g(c_t^1)$. In particular, Fig. 2d demonstrates that a sufficiently small damping parameter ς can help restore convergence.

4.3. Precomputation

The second intratemporal-choice approach we use is precomputation, which consists of constructing intratemporal-choice manifolds defined in (8) outside the main iterative cycle using either analytical derivations or numerical

computations or a combination of both. This approach was originally proposed by Maliar and Maliar (2005) for constructing a labor manifold outside the main iterative cycle in the standard neoclassical growth model. Maliar and Maliar (2007) introduced the precomputation approach in the context of the current JEDC project. In the present paper, we give a more elaborate description of the precomputation approach.

A general version of the precomputation approach, applied to the model (1)–(3), constructs the intratemporal-choice manifolds $\Phi^j(\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1})$ and $\Theta^j(\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1})$ in (8) as follows:

- *Step* 1. Outside the main iterative cycle, choose a grid of P values for $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, i.e., $\{\boldsymbol{k}_p,\boldsymbol{a}_p,\boldsymbol{k}_p\}_{p=1,\ldots,P}$. For each grid point $p=1,\ldots,P$, solve Eqs. (2), (4) and (5) using a numerical solver with respect to consumption c_p^j and labor ℓ_p^j for $j=1,\ldots,N$.
- *Step* 2. Extend the constructed set functions to the relevant continuous domain using some interpolation method (a global polynomial approximation, piecewise linear polynomial approximation, splines, etc.), such that

$$c^j=\hat{\Phi}^j(\boldsymbol{k},\boldsymbol{a},\boldsymbol{k}') \quad \text{and} \quad \ell^j=\hat{\Theta}^j(\boldsymbol{k},\boldsymbol{a},\boldsymbol{k}'),\ j=1,\ldots,N, \tag{15}$$

  where $\hat{\Phi}^j$ and $\hat{\Theta}^j$ are the precomputed consumption and labor manifolds of a country j, and $(\boldsymbol{k},\boldsymbol{a},\boldsymbol{k}')\in\mathbb{R}^{3N}$.
- *Step* 3. In the main iterative cycle, use the precomputed manifolds $\hat{\Phi}^j(\boldsymbol{k},\boldsymbol{a},\boldsymbol{k}')$ and $\hat{\Theta}^j(\boldsymbol{k},\boldsymbol{a},\boldsymbol{k}')$ to find the intratemporal choice given the state, $(\boldsymbol{k}_t,\boldsymbol{a}_t)$, and the intertemporal choice, $\boldsymbol{k}_{t+1}$.

Many applications have enough structure to simplify the precomputation approach in two ways. First, it might be not necessary to precompute all the intratemporal-choice manifolds because some of these manifolds can be constructed analytically. Second, it may be possible to precompute the intratemporal choice in terms of a set of arguments that is smaller than $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$ or that is given by some function of $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$. We illustrate these two points by way of example for Model 5 described in Maliar and Maliar (2007). In this case, we precompute a single manifold, consumption of country 1, c_t^1, in terms of one argument, aggregate consumption, c_t (which is a function of $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$).

Example 3 (*Model 5*). Outside the main iterative cycle, take P values for aggregate consumption c_t, i.e., $\{c_p\}_{p=1,\ldots,P}$. For each c_p, use a numerical solver to find a solution c_p^1 to Eq. (14) written as

$$c_p^1+\sum_{j=2}^{N}\left[\frac{\tau^1}{\tau^j}(c_p^1)^{-1/\gamma^1}\right]^{-\gamma^j}=c_p. \tag{16}$$

Interpolate the constructed set function to a continuous domain to obtain the manifold $\hat{c}^1(c)$. Inside the main iterative cycle, given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, compute c_t from resource constraint (2), use the precomputed manifold to find consumption of country 1, $c_t^1=\hat{c}^1(c_t)$ and compute consumption of the other countries as $c_t^j=[(\tau^1/\tau^j)(c_t^1)^{-1/\gamma^1}]^{-\gamma^j}$, $j=2,\ldots,N$.

As with Model 5, precomputing a single intratemporal-choice manifold (either consumption or labor manifolds) is sufficient for Models 2 and 6; see the formulas in Appendix A. For Models 3, 4, 7 and 8, we can precompute N labor manifolds and find the corresponding consumption allocations using the formulas provided in Appendix A; note that for these models, we cannot precompute N consumption manifolds and find the corresponding labor allocations analytically: there is a closed-form expression for consumption given labor but there is no closed-form expression for labor given consumption.[8]

We can precompute the intratemporal choice in Model 2 in terms of two composite arguments (see Appendix B for details). For the remaining models—Models 3, 4, 6, 7 and 8—the intratemporal-choice manifolds must be precomputed in terms of $3N$ arguments $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$. To make the precomputation approach feasible for high-dimensional problems, we can precompute the intratemporal-choice manifolds on the ergodic set realized in equilibrium.[9] For SSA, the domain for precomputation is a set of simulated points; for CGA, it is a set of the clusters' centers obtained from simulated points. The domains of SSA and CGA are discussed in Section 5.

4.4. Combining iteration-on-allocation and precomputation

For Models 3, 4, 7 and 8, the pure iteration-on-allocation approach requires to iterate simultaneously on labor allocations of N countries $\{\ell_t^j\}^{j=1,\ldots,N}$ (see Appendix A). In turn, the pure precomputation approach requires to precompute the labor choice of each country as a function of $3N$ arguments $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$. We now show that under the assumption of additively separable production across countries, we can combine iteration-on-allocation and precomputation into a single

[8] The advantage of solving for labor over solving for consumption was exploited by Maliar and Maliar (2005) to simplify finding the intratemporal choice in the standard neoclassical growth model.

[9] To construct the domain for the intratemporal-choice manifolds, one can use a tensor-product grid of $3N$ arguments $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$. However, the number of grid points will grow exponentially with dimension, and the precomputation method will not be feasible even for a moderately large number of countries.

method that precomputes the labor manifolds in terms of three arguments and iterates on one allocation.[10] This is possible because conditions (4) and (5) implicitly define the intratemporal choice of each country j in terms of its own capital k_t^j, its own productivity level a_t^j and aggregate consumption c_t; i.e., $c_t^j=\Omega^j(k_t^j,a_t^j,c_t)$ and $\ell_t^j=\Lambda^j(k_t^j,a_t^j,c_t)$, $j=1,\ldots,N$. Model 2 is an example of the economy in which such manifolds can be constructed analytically (see Appendix B); generally, however, such manifolds must be precomputed numerically.[11]

We combine iteration-on-allocation and precomputation as follows:

- *Step* 1. Outside the main iterative cycle, for each country j, choose a grid of P values for k_t^j, a_t^j, c_t, i.e., $\{k_p^j,a_p^j,c_p\}_{p=1,\ldots,P}$. For each grid point $p=1,\ldots,P$, solve Eqs. (4) and (5) using a numerical solver with respect to c_p^j and ℓ_p^j for $j=1,\ldots,N$.
- *Step* 2. Interpolate the constructed set functions to the relevant continuous domain,

$$c^j=\hat{\Omega}^j(k^j,a^j,c)\quad\text{and}\quad \ell^j=\hat{\Lambda}^j(k^j,a^j,c),\ j=1,\ldots,N, \tag{17}$$

  where $\hat{\Omega}^j$ and $\hat{\Lambda}^j$ are the precomputed consumption and labor manifolds of a country j, and $(k^j,a^j,c)\in\mathbb{R}^3$.
- *Step* 3. Substitute the precomputed labor manifolds $\hat{\Lambda}^j(k^j,a^j,c)$ for $j=1,\ldots,N$ into resource constraint (2) to define the mapping of the form $\tilde{c}_t=g(c_t)$,

$$\tilde{c}_t=\sum_{j=1}^{N}\left[a_t^j f^j(k_t^j,\hat{\Lambda}^j(k_t^j,a_t^j,c_t))-\frac{\phi}{2}k_t^j\left(\frac{k_{t+1}^j}{k_t^j}-1\right)^2+k_t^j-k_{t+1}^j\right]. \tag{18}$$

  Inside the main iterative cycle, compute aggregate consumption, c_t, using the iteration-on-allocation approach. For each t, given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, assume some values for c_t and calculate $\tilde{c}_t$ from (18); if $c_t\neq\tilde{c}_t$, compute an input for the next iteration equal to $(1-\varsigma)c_t+\varsigma\tilde{c}_t$. Iterate until convergence.

Like the pure iteration-on-allocation and precomputation methods, their combination allows to solve for the intratemporal allocations (including aggregate consumption, c_t) with a high degree of accuracy.

5. Two ergodic-set algorithms

In this section, we describe two ergodic-set algorithms, SSA and CGA, that we use to solve the models (1)–(3). Both algorithms find a solution on the ergodic set. However, they differ in how they use information on the ergodic set: SSA uses simulated points both as a domain for finding the solution and as nodes for integration, while CGA uses such points exclusively for constructing the domain (it performs integration using deterministic methods unrelated to the estimated density function). For both methods, we re-write Euler equation (6) in a form suitable for parameterizing the capital policy function $k_{t+1}^j=K^j(\boldsymbol{k}_t,\boldsymbol{a}_t)$:

$$\begin{aligned}k_{t+1}^j&=E_t\left\{\beta\frac{u_c^j(c_{t+1}^j,\ell_{t+1}^j)}{u_c^j(c_t^j,\ell_t^j)\omega_t^j}[\pi_{t+1}^j+a_{t+1}^j f_k^j(k_{t+1}^j,\ell_{t+1}^j)]k_{t+1}^j\right\}\\&\approx\Psi^j(\boldsymbol{k}_t,\boldsymbol{a}_t;\boldsymbol{v}^j),\end{aligned} \tag{19}$$

where $\Psi^j(\boldsymbol{k}_t,\boldsymbol{a}_t;\boldsymbol{v}^j)$ is a flexible functional form used to parameterize the capital policy function, and $\boldsymbol{v}^j$ is a vector of coefficients.[12] See Appendix C for the specifications of (19) corresponding to Models 5–8. We denote by $\boldsymbol{v}$ a matrix composed of the vectors of polynomial coefficients of all countries, $\boldsymbol{v}\equiv(\boldsymbol{v}^1,\ldots,\boldsymbol{v}^j,\ldots,\boldsymbol{v}^N)$.

We assume that Ψ^j is given by a complete set of ordinary polynomials, i.e., $\Psi^j(\boldsymbol{k}_t,\boldsymbol{a}_t;\boldsymbol{v}^j)\equiv\boldsymbol{X}_t\boldsymbol{v}^j$, where $\boldsymbol{X}_t$ is a row vector composed of t-period monomial terms of the state variables, $\boldsymbol{k}_t$ and $\boldsymbol{a}_t$. For the first-degree polynomial function in the example given in (9), we have $\boldsymbol{X}_t=(1,k_t^1,\ldots,k_t^N,a_t^1,\ldots,a_t^N)\in\mathbb{R}^{1\times(2N+1)}$, and $\boldsymbol{v}^j=(v_0^j,v_1^j,\ldots,v_N^j,v_{N+1}^j,\ldots,v_{2N}^j)'\in\mathbb{R}^{(2N+1)\times1}$.

We should emphasize that the end-of-period capital stocks $\boldsymbol{k}_{t+1}$ are the only variables we approximate by functions of the state variables, $\boldsymbol{k}_t$ and $\boldsymbol{a}_t$. The remaining variables (consumption and labor) are either computed by the iteration-on-allocation solver or obtained from the precomputed intratemporal-choice manifolds in form (8).

Properly separating the intertemporal and intratemporal choices is crucial for the speed of SSA and CGA. Approximating the policy functions for capital, as in (19), has an important advantage over approximating the policy functions for other variables such as consumption and leisure. Namely, the equilibrium capital policy functions are the equilibrium capital laws of motion $k_{t+1}^j=K^j(\boldsymbol{k}_t,\boldsymbol{a}_t)$, $j=1,\ldots,N$. As a result, we can first construct a path for capital, $\{\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$, and

[10] Combining iteration-on-allocation and precomputation is also possible for Models 2 and 6, but does not provide any advantages over the pure iteration-on-allocation method described in Section 4.2.

[11] Maliar and Maliar (2001, 2003a) construct similar intratemporal-choice manifolds analytically for certain classes of heterogeneous-agent economies and use these manifolds to derive non-Gorman aggregation results.

[12] This kind of parameterization was used by Den Haan (1990) as a device to implement the simulation-based parameterized expectations algorithm in a model with more than one Euler equation.

subsequently fill in the corresponding intratemporal allocations $\{\boldsymbol{c}_t,\boldsymbol{\ell}_t\}_{t=1,\ldots,T}$. The iteration-on-allocation and precomputation methods can work with vectors and even matrices and can thus find the intratemporal choice in all periods, grid points, integration nodes at once rather than one by one.

5.1. Stochastic simulation algorithm

SSA simultaneously computes the ergodic distribution of state variables, its support and the associated policy functions. It proceeds as follows:

Fix the simulation length T and initial condition $(\boldsymbol{k}_0,\boldsymbol{a}_0)$. Draw and fix for all simulations a sequence of productivity levels $\{\boldsymbol{a}_t\}_{t=1,\ldots,T}$ using Eq. (3). If precomputation is used, construct the intratemporal-choice manifolds of type (8) as described in Section 4.3, and if precomputation is combined with iteration-on-allocation, do so as described in Section 4.4.

- *Step* 1. For an iteration s, fix some matrix of coefficients $\boldsymbol{v}(s)$. For each country $j=1,\ldots,N$, use the assumed capital policy function $k^j_{t+1}=\boldsymbol{X}_t\boldsymbol{v}^j$ to recursively calculate a sequence of capital stocks $\{\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$ corresponding to a given sequence of productivity levels $\{\boldsymbol{a}_t\}_{t=0,\ldots,T}$.
- *Step* 2. Given $\{\boldsymbol{k}_t,\boldsymbol{a}_t,\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$, calculate $\{\boldsymbol{c}_t,\boldsymbol{\ell}_t\}_{t=0,\ldots,T}$ using a vectorized version of either iteration-on-allocation or precomputation or their combination.
- *Step* 3. For each country $j=1,\ldots,N$, compute the integrand of (19),

$$y^j_t\equiv\beta\frac{u^j_c(c^j_{t+1},\ell^j_{t+1})}{u^j_c(c^j_t,\ell^j_t)\omega^j_t}[\pi^j_{t+1}+a^j_{t+1}f^j_k(k^j_{t+1},\ell^j_{t+1})]k^j_{t+1} \tag{20}$$

  for $t=0,\ldots,T-1$.
- *Step* 4. For each country $j=1,\ldots,N$, run a linear regression of the constructed variable y^j_t on a set of explanatory variables $\boldsymbol{X}_t$ using the numerically stable approximation methods described in JMM (2009, 2010b),

$$y^j_t=\boldsymbol{X}_t\boldsymbol{v}^j+\varepsilon^j_t, \tag{21}$$

  where ε^j_t is a t-period regression error corresponding to country j. Let the matrix of coefficients estimated on iteration s be called $\hat{\boldsymbol{v}}(s)$.
- *Step* 5. Compute the matrix of coefficients for the subsequent iteration $s+1$ using fixed-point iteration:

$$\boldsymbol{v}(s+1)=(1-\xi)\boldsymbol{v}(s)+\xi\hat{\boldsymbol{v}}(s), \tag{22}$$

  where $\xi\in(0,1)$ is a damping parameter.

Iterate on *Steps* 1–5 until a fixed point is found such that for $\vartheta>0$:

$$\frac{1}{T\cdot N}\sum_{t=1}^{T}\sum_{j=1}^{N}\left|\frac{k^j_{t+1}(s)-k^j_{t+1}(s+1)}{k^j_{t+1}(s)}\right|<10^{-\vartheta}\xi, \tag{23}$$

where $k^j_{t+1}(s)$ and $k^j_{t+1}(s+1)$ are the j-th country's capital stocks obtained on iterations, s and $s+1$, respectively, and $|\cdot|$ denotes the absolute value.

5.2. Cluster-grid algorithm

CGA is a projection method that computes a solution on a grid constructed from clusters of simulated points. It proceeds as follows: Make an initial guess about the capital policy functions $k^j_{t+1}=\boldsymbol{X}_t\boldsymbol{v}^j$, $j=1,\ldots,N$. Given initial condition $(\boldsymbol{k}_0,\boldsymbol{a}_0)$, draw a sequence of productivity levels $\{\boldsymbol{a}_t\}_{t=1,\ldots,T}$ using (3), and simulate the time series solution $\{\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$. Using simulated data, construct I clusters and compute the centers of the clusters, $\{\boldsymbol{k}_i,\boldsymbol{a}_i\}_{i=1,\ldots,I}$, to be used as a grid for projections; see Judd et al. (2010a) for a description of clustering methods and illustrative examples. In each grid point $i=1,\ldots,I$, construct a function $\Gamma^j(\boldsymbol{k}_i,\boldsymbol{a}_i,\boldsymbol{\varepsilon})$ that represents the integrand in (19),

$$\Gamma^j(\boldsymbol{k}_i,\boldsymbol{a}_i,\boldsymbol{\varepsilon})\equiv\beta\frac{u^j_c((c^j_i)',(\ell^j_i)')}{u^j_c(c^j_i,\ell^j_i)\omega^j}[(\pi^j_i)'+(a^j_i)'f^j_k((k^j_i)',(\ell^j_i)')](k^j_i)', \tag{24}$$

where $\boldsymbol{\varepsilon}\equiv(\varepsilon^1,\ldots,\varepsilon^N)$ is the next-period shock. Furthermore, if precomputation is used, construct the intratemporal-choice manifolds of type (8) as described in Section 4.3, and if precomputation is combined with iteration-on-allocation, do so as described in Section 4.4.

- *Step* 1. On an iteration s, fix a matrix of coefficients $\boldsymbol{v}(s)$. For each country j, use the assumed capital policy function to calculate the end-of-period capital stock in all grid points, $(k^j_i)'\equiv\boldsymbol{X}_i\boldsymbol{v}^j$ for $i=1,\ldots,I$.
- *Step* 2. Given $\{\boldsymbol{k}_i,\boldsymbol{a}_i,\boldsymbol{k}_i{}'\}_{i=1,\ldots,I}$, calculate $\{\boldsymbol{c}_i,\boldsymbol{\ell}_i\}_{i=1,\ldots,I}$ using a vectorized version of iteration-on-allocation, precomputation or their combination.

- *Step* 3. For each country $j=1,\ldots,N$, use a numerical integration method (such as non-product monomial rules or product Gauss–Hermite rule) to approximate the conditional expectations of Eq. (24). Call the result $(\hat{k}_i^j)'$, i.e.,

$$(\hat{k}_i^j)' \equiv E[I^j(\boldsymbol{k}_i,\boldsymbol{a}_i,\boldsymbol{\varepsilon})], \tag{25}$$

  where the expectation is computed with respect to $\boldsymbol{\varepsilon}\equiv(\varepsilon^1,\ldots,\varepsilon^N)$. To calculate the next-period intratemporal choice $\{\boldsymbol{c}_i',\boldsymbol{\ell}_i'\}_{i=1,\ldots,I}$, for each integration node, use a vectorized version of either iteration-on-allocation or precomputation or their combination as described in Section 4.
- *Step* 4. For each country $j=1,\ldots,N$, run a linear regression of the constructed variable $(\hat{k}_i^j)'$ on a set of explanatory variables $\boldsymbol{X}_i$ using the numerically stable approximation methods described in Judd et al. (2009, 2010b),

$$(\hat{k}_i^j)' = \boldsymbol{X}_i\boldsymbol{v}^j + \varepsilon_i^j, \tag{26}$$

  where ε_i^j is an i-grid-point regression error corresponding to country j. Let the matrix of coefficients estimated on iteration s be called $\hat{\boldsymbol{v}}(s)$.
- *Step* 5. Compute the matrix of coefficients for the subsequent iteration $s+1$ using fixed-point iteration (22).

Iterate on *Steps* 1–5 until a fixed point is found, such that for $\vartheta>0$:

$$\frac{1}{I\cdot N}\sum_{i=1}^{I}\sum_{j=1}^{N}\left|\frac{(k_i^j)'-(\hat{k}_i^j)'}{(k_i^j)'}\right| < 10^{-\vartheta}, \tag{27}$$

where $(k_i^j)'$ and $(\hat{k}_i^j)'$ are the end-of-period capital stocks before and after the iteration, respectively.

After achieving convergence, we should generally re-run CGA using the obtained policy functions for capital as an initial guess for simulation. Doing so controls for the possibility that the initial guess for the capital policy functions was far from the true solution, and the simulated series (and consequently, our cluster grid) did not adequately represent the true ergodic set.

6. Implementation details

In Section 6.1, we describe the baseline implementation of SSA and CGA that was used to generate the results presented in the comparison paper by Kollmann et al. (this issue-b). In Section 6.2, we discuss alternative implementations of these algorithms that are not included in KMMP (2010). Calibration of the models' parameters is provided in Juillard and Villemot (this issue).

6.1. Baseline implementation of SSA and CGA

Below, we describe the details of the baseline implementation of our methods, as well as the solution-output, hardware, software and measures of accuracy and cost.

Stochastic simulation algorithm: SSA computes solutions using the first-degree ordinary polynomial (9). To start the iterative process, we use an (arbitrary) initial guess: $k_{t+1}^j=0.9k_t^j+0.1a_t^j$ for all $j=1,\ldots,N$. Since the steady-state levels of capital and productivity are normalized to one, the above guess matches the steady-state level of capital. In terms of the vector of coefficients $\boldsymbol{v}^j$, this guess implies that $v_j^j=0.9$, $v_{N+j}^j=0.1$, $j=1,\ldots,N$, and that the remaining coefficients in $\boldsymbol{v}^j$ are equal to zero. Initial capital and productivity level are set at their steady-state values: $k_0^j=1$ and $a_0^j=1$ for all $j=1,\ldots,N$. The simulation length is $T=10{,}000$.

To estimate the coefficients in the linear regression (21), we use a least-squares truncated QR factorization method; see Judd et al. (2009) for a discussion. We set the damping parameter in (22) to be the largest values of ξ that lead to convergence: $\xi=0.05$ for Models 1 and 5, and at $\xi=0.03$ for the remaining models. We target seven digits of accuracy in the simulated data by fixing $\vartheta=7$ in convergence criterion (23). To rule out explosive and implosive behavior on initial iterations, we restrict the simulated series for capital using moving bounds as described in Maliar and Maliar (2003b); in most cases, however, the artificial bounds were not necessary as the initial guess led to a stationary simulated series.

Cluster-grid algorithm: CGA computes solutions using a second-degree ordinary polynomial. To start the iterative process, we use the first-degree polynomial solution computed by SSA as an initial guess. The SSA solution was used both to compute an initial guess for the matrix of coefficients $\boldsymbol{v}$ and to construct 500 clusters. The clusters were constructed by applying an hierarchical clustering algorithm with Ward's linkage to the principal components of the simulated data; see Judd et al. (2010a) for a description of the clustering methods and illustrative examples.

To estimate linear regression (26), we again use a least-squares truncated QR factorization method. We set the damping parameter in (22) at $\xi=0.1$ for Models 1 and 5, and $\xi=0.05$ for all other models. We use $\vartheta=7$ for convergence criterion (27). We solve the model twice: first, we compute the solution using a cheap non-product monomial rule $M1$ with $2N$ nodes, and then, we recompute it using an expensive product Gauss–Hermite rule $Q(2)$ with two nodes in each dimension (2^N nodes in total); see Judd et al. (2010a) for a description of these integration methods.

Iteration-on-allocation: In the baseline versions of both SSA and CGA, we solve for the intratemporal choice using the iteration-on-allocation approach. We use the damping parameter $\varsigma=0.01$ in all cases except for Models 1 and 5 under CGA

in which case we use $\varsigma = 0.05$. To start iterations under SSA, we assume that consumption and labor are equal to their steady-state values. Under CGA, we compute an initial guess for consumption and labor using the solution produced by SSA.

We would like to direct attention to an important aspect of the implementation of iteration-on-allocation. Finding consumption and labor allocations with a high degree of accuracy on each iteration requires a high computational cost and is in fact of no use, since on the next iteration, we must re-compute consumption and labor allocations for a different matrix of coefficients $\boldsymbol{v}$. We thus do not target any accuracy criteria in consumption and labor allocations in each iteration on $\boldsymbol{v}$, but instead perform 10 subiterations on mapping (10) as described in Section 4 (except for Models 1 and 5 under CGA in which we perform three subiterations). We store in memory consumption and labor allocations obtained after each round of subiterations and use these allocations as inputs for the next round of the iteration-on-allocation process. Thus, as the policy functions for capital (characterized by $\boldsymbol{v}$) are refined along the iterations, so do our consumption and labor allocations.

To enhance the numerical stability on initial iterations when the solution is inaccurate, we impose fixed lower and upper bounds (equal to 50% and 150% of the steady-state level, respectively) on consumption in Model 5 and on labor in Models 6–8. This technique is similar to the moving bounds used to restrict simulated series for capital under SSA. With the imposition of bounds, the iteration-on-allocation procedure was numerically stable and converged to a fixed point at a good pace in all of our experiments. In convergence criterion (11), we use $\theta = 7$.

Solution-output delivered to the testing bench of Juillard and Villemot (this issue): Under the iteration-on-allocation approach, SSA and CGA do not deliver explicit policy functions for consumption and labor. The only solution-output they produce is a matrix of the polynomial coefficients for the capital policy functions (laws of motion) of N heterogeneous countries, $\boldsymbol{v} = (\boldsymbol{v}^1, \ldots, \boldsymbol{v}^j, \ldots, \boldsymbol{v}^N)$. Thus, in addition to the polynomial coefficients $\boldsymbol{v}$, we supply to the testing bench of Juillard and Villemot (this issue) four iteration-on-allocation routines (one per each asymmetric model and its symmetric counterpart) that allow to find the intratemporal choice in simulation.[13]

The simulation of our solutions includes two steps: First, the capital laws of motion, $k^j_{t+1} = \Psi^j(\boldsymbol{k}_t, \boldsymbol{a}_t; \boldsymbol{v}^j)$, $j = 1, \ldots, N$, are used to generate the capital path $\{\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$. Then, given $\{\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$, the corresponding intratemporal choice $\{\boldsymbol{c}_t, \boldsymbol{\ell}_t\}_{t=0,\ldots,T}$ is filled in using the iteration-on-allocation method described in Section 4.2 and Appendix A. To begin the iteration-on-allocation process, we set consumption and labor equal to their steady-state values; we use the damping parameter $\varsigma = 0.01$, and we perform iterations until the results satisfy convergence criterion (11) with $\theta = 10$.

Software, *hardware*, *accuracy and cost*: Our programs are written in Matlab, version 7.6.0.324 (R2008a). We use a desktop computer with a Quad processor Intel(R) Core(TM) i7 CPU920 at 2.67 GHz, RAM 6,00 GB and Windows Vista 64 bits. For each model studied, we report the running time in seconds: for SSA, the running time is defined as the time needed to compute a linear solution starting from a given initial guess, and for CGA, the running time is defined as the time needed to compute a quadratic solution starting from a linear SSA solution. Accuracy tests are performed using the testing bench of Juillard and Villemot (this issue), and the results of these tests are described in Kollmann et al. (this issue-b).

6.2. Exploring alternative implementations

The current JEDC project was launched in 2003, and since then, we have implemented many versions of the studied methods. We now compare our baseline implementation of SSA and CGA to alternative implementations explored, some of which are illustrated with numerical results in Section 7.

Stochastic simulation algorithm: At an early stage of the project, Maliar and Maliar (2004, 2007) implemented a stochastic simulation approach using the simulation-based parameterized expectation algorithm (PEA) by Den Haan and Marcet (1990). Under the PEA, policy functions are parameterized by an exponentiated polynomial and are estimated using non-linear least-squares regression methods. The least-squares problem is typically ill-conditioned, which leads to numerical problems.[14] Moreover, non-linear regression methods require a good initial guess and involve costly computations of Jacobian and Heissian matrices. Also, such methods cannot be easily vectorized to estimate the policy functions of all countries simultaneously, which is critical for speed in multi-country settings; see Judd et al. (2010b) for an extensive discussion. In the present paper, we rely on numerically stable stochastic simulation approaches described in Judd et al. (2009): we use a linear regression model and employ a least-squares truncated QR factorization method suited for use with ill-conditioned problems. This approximation method (implemented in Matlab with the backslash operator) delivers the standard OLS estimator in the absence of ill-conditioning but removes highly collinear principal components in the presence of ill-conditioning.

We submitted for the comparison in Kollmann et al. (this issue-b) the first-degree polynomial approximation because under the chosen simulation length T=10,000, it was more accurate than the second-degree polynomial approximation. This result is explained in Judd et al. (2010b): the accuracy of Monte Carlo integration employed by SSA depends on how large the simulation length T is relative to the number of polynomial coefficients in $\boldsymbol{v}$.[15] The higher is the polynomial

[13] Using the iteration-on-allocation routines in simulation plays a key role in the overall accuracy of SSA and CGA because it allows us to solve for the intratemporal choice with essentially zero errors; see Table 4.3 in Kollmann et al. (this issue-b). This would not be possible if we constructed and supplied the standard explicit consumption and/or labor policy functions in terms of state variables.

[14] To be specific, polynomial terms in the approximating polynomial function are highly correlated (multicollinear), and the regression model cannot be estimated with the standard least-squares method; see Den Haan and Marcet (1990), Christiano and Fisher (2000) and Judd et al. (2009).

[15] In a recent paper, Judd et al. (2010c) develop a generalized stochastic simulation algorithm that uses deterministic integration methods and that delivers accuracy comparable to the highest accuracy attained in the comparison analysis of Kollmann et al. (this issue-b).

degree and/or the dimensionality of the problem, N, the larger is the number of the coefficients in $\boldsymbol{\nu}$, and the larger simulation length is needed to appropriately identify the coefficients. The simulation length assumed, T=10,000, is sufficient to accurately identify the coefficients of the first- but not the second-degree polynomial. In a model similar to Model 1 of the current JEDC project, Judd et al. (2010b) find that T should be increased from 10,000 to 50,000 and to 100,000 to make the second-degree polynomial approximation more accurate than the first-degree polynomial approximation for the model with up to N=4 and with up to N=6, respectively. Running such a long simulation would be costly both in terms of time and memory. As follows from the comparison in Kollmann et al. (this issue-b), even the linear solutions delivered by SSA are sufficiently accurate. This is because SSA fits a polynomial exclusively in the relevant area of the state space (the ergodic set) and also because it solves accurately for the intratemporal choice using the iteration-on-allocation method.

Cluster-grid algorithm: In the case of CGA, we submitted for the comparison in Kollmann et al. (this issue-b) the second-degree polynomial approximation. CGA relies on accurate numerical integration methods, and the second-degree polynomial approximation is considerably more accurate than the first-degree one. The third-degree polynomial approximation is even more accurate. In particular, Judd et al. (2010a) find that an increase in the polynomial degree used in CGA increases accuracy roughly by an order of magnitude in the examples considered. In Section 7, we compare the accuracy of the CGA method in the context of the current JEDC project using the first-, second- and third-degree ordinary polynomials, as well as using alternative Smolyak polynomials.

We also test how sensitive the CGA solutions are to the way in which the cluster grid is constructed. First, we tried to initialize CGA using a linear solution delivered by a log-linearization method instead of the one delivered by SSA.[16] Second, we tried to construct clusters using an alternative K-means clustering algorithm and/or different linkage methods instead of the baseline hierarchical algorithm with Ward's linkage. These modifications do not visibly affect the accuracy and speed of CGA. Finally, concerning the number of clusters, Judd et al. (2010a) find that oversampling (when there are more grid points than the polynomial coefficients) increases the accuracy and numerical stability of the CGA method compared to collocation (when the number of grid points is the same as the number of polynomial coefficients). In line with this finding, we choose to oversample and use 500 clusters to identify between 15 and 231 polynomial coefficients in models with N ranging from 2 to 10, respectively.

To perform numerical integration, we tried to choose the most accurate integration strategy feasible for problems of given dimensionality, $N \leq 10$. To this purpose, we design a two-step integration procedure that combines a low-cost monomial rule with $2N$ nodes (step one) and a costly monomial (quadrature) rule with 2^N nodes (step two). It turned out that, in the studied models, gains from so accurate integration are minimal relative to less costly integration alternatives. In Section 7, we investigate how accuracy and cost of CGA depend on the specific integration method.

Intratemporal-choice approaches: In addition to our baseline iteration-on-allocation procedure, we explored other approaches solving for the intratemporal choice. For all eight models studied, we computed the consumption and/or labor policy functions in terms of state variables within the main iterative cycle as described in Section 4.1, and we implemented a general version of the precomputation approach presented in Section 4.3. Also, for Model 5, we implemented the precomputation approach as described in Example 3, and for Models 6 and 7, we combined the precomputation approach and a numerical solver as described in Section 4.4.

To generate the results used for the comparison in Kollmann et al. (this issue-b), we opt for the most accurate method, which is iteration-on-allocation. However, our precomputation approach is a useful alternative to consider as it is faster than the iteration-on-allocation approach. In particular, it was adopted by Pichler (this issue) for his solution method. In Section 7, we compare the performance of alternative intratemporal-choice approaches in the context of Model 5.

7. Additional numerical results

Accuracy and speed of SSA and CGA under the baseline implementation is assessed in Kollmann et al. (this issue-b). In this section, we provide additional numerical results for CGA, which show how its accuracy and speed depend on the specific intratemporal-choice approach, approximating polynomial function and integration method. To assess the accuracy of solutions, we implement a test that computes the average and maximum solution errors along a stochastic simulation of 10,000 observations as described in Juillard and Villemot (this issue).

7.1. Comparison of intratemporal choice approaches

To illustrate the role of the specific intratemporal-choice approach in determining the accuracy of solutions, we use a two-country version of Model 5. We allow an intratemporal-choice approach used in the solution procedure differ from that used in the simulation procedure. We report the results obtained using the second-degree polynomial approximation; the regularities under the first-order polynomial approximation are similar.

[16] In Judd et al. (2010a), CGA is initialized without help of other methods: time series are simulated for an arbitrary initial guess and used to construct clusters on which a solution is computed. The obtained solution serves as a more accurate initial guess.

In the solution procedure, we consider four alternative intratemporal-choice approaches: (i) parameterize the consumption policy functions of both countries with a polynomial of the state variables and compute the polynomial coefficients inside the main iterative cycle; (ii) parameterize and compute only the consumption policy function of country 1 inside the main iterative cycle and find consumption of country 2 from closed-form expression (12); (iii) precompute the consumption manifold of country 1 outside the main iterative cycle in terms of aggregate consumption as described in Example 3 and find consumption of country 2 from (12); (iv) solve for consumption of both countries using the iteration-on-allocation approach, as described in Example 1.

In the simulation procedure, we solve for the intratemporal choice using four approaches that are parallel to those used in the solution procedure: (a) use the computed solution to construct the consumption policy functions for both countries in terms of the state variables (if not constructed by the solution method used); (b) use the solution to construct the consumption policy function of country 1 in terms of the state variables (if not constructed by the solution method used) and find consumption of country 2 from (12); (c) find consumption of country 1 using the consumption manifold precomputed by method (iii) and find consumption of country 2 from (12); (d) solve for consumption of both countries using the iteration-on-allocation approach.

To implement precomputation in (iii), we consider an interval for aggregate consumption equal to $\pm 20\%$ of the steady-state value, and we split this interval into 300 equally spaced points. Outside the main iterative cycle, for each value of aggregate consumption c_p, we compute c_p^1 numerically from (16). Inside the main iterative cycle, we compute aggregate consumption c_t from (2) and find the corresponding c_t^1 using a piecewise linear polynomial interpolation. We tried other interpolation schemes such as a global high-order polynomial approximations, piecewise cubic polynomial interpolation, splines, etc., and found that piecewise low-order polynomial interpolation schemes lead to more accurate solutions (though are more costly) than global high-order polynomial approximations.

In Table 1, we present the results of combining methods (i)–(iv) in the solution procedure with methods (a)–(d) in the simulation procedure (all errors that are less than 10^{-10} are replaced by $-\infty$). As the table indicates, if in both the solution and simulation procedures, we approximate a policy function for consumption using a second-degree polynomial of state variables, the resulting accuracy is low (namely, errors in the intratemporal-choice conditions including the resource constraint are large). If in the solution procedure, we solve for consumption very accurately (using precomputation and iteration-on-allocation) but in simulation, we solve for consumption not so accurately (using second-degree polynomials of state variables), the resulting accuracy is again low. Finally, if in the solution procedure, we solve for consumption not very accurately (using second-degree polynomials of state variables), but in simulation, we solve for consumption very accurately (using precomputation and iteration-on-allocation), the accuracy gets restored. These tendencies lead us to conclude that it is not so important for accuracy how we compute the intratemporal choice in the solution procedure but how we compute it in simulation (when running accuracy tests).

We would like to highlight two additional findings about accuracy in Table 1. First, accuracy does not depend significantly on whether we approximate one or more than one intratemporal-choice variable using second-degree polynomials of state variables; in both cases, we suffer approximately the same accuracy loss. Second, the methods solving for the intratemporal choice accurately lead to considerably larger Euler-equation errors than those solving for the intratemporal choice less accurately.

Table 1
Accuracy and time needed to run the test for the two-country version of Model 5 under alternative intratemporal-choice approaches.

Intratemporal choice in the solution procedure	Equation	Intratemporal choice in the simulation procedure											
		(a) Two policy functions			(b) One policy function			(c) Precomputation			(d) Iteration-on-allocation		
		Δ_{mean}	$\Delta_{\max}$	*TCPU*	Δ_{mean}	$\Delta_{\max}$	*TCPU*	Δ_{mean}	$\Delta_{\max}$	*TCPU*	Δ_{mean}	$\Delta_{\max}$	*TCPU*
(i) Two policy functions	Euler	−6.14	−4.55	15	−6.02	−4.55	15	−5.68	−4.29	17	−5.73	−4.29	226
	RC	−4.93	−3.48		−4.54	−3.09		−5.89	−5.61		$-\infty$	$-\infty$	
	Intrat.	−4.64	−3.22		$-\infty$	$-\infty$		$-\infty$	$-\infty$		$-\infty$	$-\infty$	
(ii) One policy function	Euler	−5.98	−4.57	15	−6.06	−4.57	15	−5.68	−4.29	17	−5.72	−4.28	226
	RC	−4.68	−3.60		−4.54	−3.09		−5.89	−5.61		$-\infty$	$-\infty$	
	Intrat.	−4.69	−3.17		$-\infty$	$-\infty$		$-\infty$	$-\infty$		$-\infty$	$-\infty$	
(iii) Precomputation	Euler	−5.98	−4.42	14	−5.91	−4.42	14	−5.69	−4.35	16	−5.74	−4.35	231
	RC	−4.66	−3.63		−4.43	−3.17		−5.89	−5.61		$-\infty$	$-\infty$	
	Intrat.	−4.65	−3.28		$-\infty$	$-\infty$		$-\infty$	$-\infty$		$-\infty$	$-\infty$	
(iv) Iteration-on-allocation	Euler	−5.99	−4.42	15	−5.92	−4.42	15	−5.69	−4.35	17	−5.74	−4.35	232
	RC	−4.66	−3.63		−4.42	−3.18		−5.89	−5.61		$-\infty$	$-\infty$	
	Intrat.	−4.65	−3.28		$-\infty$	$-\infty$		$-\infty$	$-\infty$		$-\infty$	$-\infty$	

Note: Δ_{mean} and $\Delta_{\max}$ are, respectively, the average and maximum errors in the corresponding optimality condition (in log 10 units) in the test on a stochastic simulation of 10,000 observations; *TCPU* is the time needed to run the test (in seconds). Abbreviations "Euler", "RC" and "Intrat." denote the Euler equations, resource constraint and intratemporal-choice conditions, respectively.

Finally, Table 1 also shows the time, *TCPU*, needed to run the test on a stochastic simulation of 10,000 observations. Under the precomputation approach, *TCPU* is only slightly larger than under the standard approach constructing the consumption and labor policy functions in terms of state variables, while under the iteration-on-allocation approach, *TCPU* is almost 20 times larger. The iteration-on-allocation approach performs slowly in the test because our testing procedure is not vectorized along the time dimension; i.e., we use the iteration-on-allocation solver 10,000 times to compute c_t^1 and c_t^2 period by period.

7.2. Cost of iteration-on-allocation

We now quantify the benefits of vectorizing the iteration-on-allocation approach along the time dimension. In Table 2, we compare the time necessary to simulate a time-series solution of length T under two alternative simulation procedures: one in which the intratemporal choice is computed using the standard policy functions represented by second-degree polynomials of state variables (*CPU* 1) and the other in which the intratemporal choice is computed using the iteration-on-allocation solver (*CPU* 2). As an initial guess for allocations in the latter procedure, we use the allocations obtained in the former procedure.

Since our simulation routines are written in a vectorized form, the cost of iteration-on-allocation depends dramatically on the simulation length. When we simulate only one period entry (i.e., $T=1$), the iteration-on-allocation approach is about 67 and 362 times more costly for Models 1 and 4, respectively, than the standard approach based on policy functions. However, as T increases, the relative cost of iteration-on-allocation decreases; in particular, for $T=10{,}000$, the iteration-on-allocation approach is about 4% and 250% more costly for Models 1 and 4, respectively, than the standard approach based on policy functions. The latter value is an upper bound. In other cases, the relative cost of iteration-on-allocation is even lower.

7.3. Approximating functions and integration methods

In Table 3, we assess the effect of the specific approximating function and integration method on the accuracy of CGA in the context of two-country versions of Models 5–8. For each model studied, we consider four alternative approximating functions: the first-, second- and third-degree ordinary polynomials, as well as the Smolyak polynomial used in Malin et al. (this issue). We also consider five alternative integration methods: the product Gauss–Hermite rule with 1, 2^N, 3^N nodes, denoted $Q(1)$, $Q(2)$ and $Q(3)$, respectively; and the monomial formulas with $2N$ and $2N^2+1$ nodes, denoted $M1$ and $M2$, respectively.

We observe the following regularities from the table. First, all of the integration rules considered, except for the one-node Gauss–Hermite rule $Q(1)$, deliver solutions of virtually the same accuracy, with errors that are identical to the fourth digit. The $Q(1)$ rule produces errors that are slightly larger; however, this rule has a substantially lower cost than the other

Table 2
Time for simulating Models 5–8 under two alternative simulation procedures: one using policy functions and the other using iteration-on-allocation.

N	$T=1$		$T=10$		$T=100$		$T=1{,}000$		$T=10{,}000$	
	CPU1	CPU2	CPU1	CPU2	CPU1	CPU2	CPU1	CPU2	CPU1	CPU2
Model 5										
2	0.0001	0.0071	0.0004	0.0090	0.0030	0.0182	0.0415	0.1038	5.4893	5.7720
4	0.0001	0.0041	0.0004	0.0055	0.0037	0.0144	0.0645	0.1218	22.1597	22.4737
6	0.0001	0.0033	0.0004	0.0046	0.0045	0.0171	0.2483	0.3546	48.7948	49.0005
8	0.0001	0.0028	0.0005	0.0040	0.0055	0.0173	0.5870	0.6553	82.8177	83.3027
10	0.0001	0.0022	0.0006	0.0036	0.0072	0.0184	0.9490	1.0325	124.5969	124.5281
Model 6										
2	0.0001	0.0035	0.0004	0.0056	0.0032	0.0219	0.0416	0.1754	5.4723	6.0610
4	0.0001	0.0019	0.0006	0.0038	0.0041	0.0204	0.0694	0.1941	22.4622	22.9424
6	0.0002	0.0011	0.0006	0.0027	0.0052	0.0167	0.2573	0.3730	49.1699	49.5512
8	0.0002	0.0015	0.0007	0.0033	0.0067	0.0205	0.5774	0.7385	83.1369	83.7333
Model 7										
2	0.0001	0.0302	0.0003	0.0430	0.0033	0.1544	0.0410	1.0800	5.5069	9.8106
4	0.0001	0.0200	0.0004	0.0422	0.0038	0.2006	0.0645	1.3623	22.4296	28.5446
6	0.0001	0.0227	0.0004	0.0506	0.0048	0.2958	0.2525	2.0232	49.0937	58.5057
Model 8										
2	0.0001	0.0381	0.0004	0.0622	0.0033	0.2730	0.0412	2.1903	5.5690	14.0949
4	0.0001	0.0555	0.0004	0.1076	0.0041	0.6396	0.0664	5.1589	22.4694	53.7229
6	0.0002	0.0694	0.0006	0.1595	0.0050	0.9718	0.2472	8.5013	49.2921	101.2122

Note: *CPU* 1 and *CPU* 2 are, respectively, the time necessary to simulate a time-series solution of length T under the simulation procedure using policy functions and the one using iteration-on-allocation (in seconds); N is the number of countries.

Table 3
Effect of the specific polynomial on accuracy of CGA under five integration rules in the two-country versions of Models 5–8.

Polynomial	$Q(3)$		$Q(2)$		$M2$		$M1$		$Q(1)$	
	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	Δ_{max}
Model 5										
1st	−4.90195	−3.13194	−4.90194	−3.13194	−4.90193	−3.13194	−4.90193	−3.13194	−4.88823	−3.13396
2nd	−6.38976	−4.34742	−6.38976	−4.34735	−6.38974	−4.34730	−6.38974	−4.34730	−5.86835	−4.30024
3rd	−7.15921	−5.15528	−7.15696	−5.15517	−7.15966	−5.15600	−7.15709	−5.15556	−5.89480	−4.96021
SMOL	−7.06458	−5.05292	−7.06445	−5.05265	−7.06427	−5.05233	−7.06425	−5.05236	−5.89095	−4.83609
Model 6										
1st	−4.82343	−3.02274	−4.82342	−3.02274	−4.82340	−3.02274	−4.82341	−3.02274	−4.75012	−3.03238
2nd	−6.27646	−4.30442	−6.27647	−4.30437	−6.27646	−4.30432	−6.27647	−4.30432	−5.70532	−4.23380
3rd	−7.15049	−5.15572	−7.15109	−5.15531	−7.15136	−5.15497	−7.15144	−5.15489	−5.72017	−4.78838
SMOL	−6.98459	−4.98077	−6.98441	−4.98053	−6.98414	−4.98026	−6.98409	−4.98026	−5.71619	−4.70282
Model 7										
1st	−4.77765	−3.03091	−4.77765	−3.03091	−4.77763	−3.03091	−4.77764	−3.03091	−4.73123	−3.03668
2nd	−6.07533	−4.24781	−6.07537	−4.24774	−6.07538	−4.24771	−6.07539	−4.24770	−5.65946	−4.20806
3rd	−7.06964	−4.99023	−7.07030	−4.99064	−7.07040	−4.99073	−7.07057	−4.99098	−5.67346	−4.75051
SMOL	−6.78548	−4.72708	−6.78540	−4.72691	−6.78532	−4.72679	−6.78525	−4.72677	−5.66775	−4.54921
Model 8										
1st	−4.58750	−2.80045	−4.58750	−2.80045	−4.58749	−2.80045	−4.58749	−2.80045	−4.53314	−2.80015
2nd	−5.87377	−3.83040	−5.87378	−3.83037	−5.87377	−3.83035	−5.87378	−3.83035	−5.52168	−3.79411
3rd	−6.81686	−4.38437	−6.81684	−4.38446	−6.81679	−4.38448	−6.81674	−4.38452	−5.57508	−4.27229
SMOL	−6.69808	−4.56910	−6.69794	−4.56898	−6.69782	−4.56889	−6.69777	−4.56889	−5.57173	−4.40082

Note: Δ_{mean} and $\Delta_{\max}$ are, respectively, the average and maximum absolute errors across all optimality conditions (in log 10 units) in the test on a stochastic simulation of 10,000 observations. Abbreviations "1st", "2nd", "3rd" and "SMOL" denote the first-, second-, third-degree ordinary polynomials and the Smolyak polynomial, respectively.

integration methods and thus allows to solve problems of much higher dimensionality. In particular, Judd et al. (2010a) use the $Q(1)$ rule to compute first- and second-degree polynomial solutions to a model (similar to Model 1 of the current project) with up to $N=200$ and 40 countries, respectively.

Second, when solutions are computed using ordinary polynomials, increasing the polynomial degree from one to two raises accuracy (reduces errors) by more than an order of magnitude; increasing the polynomial degree from two to three does so by slightly less than an order of magnitude. However, it is costly to increase the degree of a complete polynomial in high-dimensional problems. As is seen from Table 3, the Smolyak polynomial is a useful alternative for CGA: it leads to as nearly as accurate solutions as the third-degree complete polynomial, but its number of terms grows quadratically instead of cubically with dimension (the Smolyak polynomial has only four times more terms than the second-degree complete polynomial independently of dimension); see Malin et al. (this issue) for a discussion and definition of the Smolyak polynomial.

In Table 4, we investigate how the cost and accuracy of CGA depend on the specific integration method used. To this purpose, we recompute the solutions to Models 5–8 under four alternative integration methods: $Q(1)$, $Q(2)$, $M1$ and $M2$. The accuracy measures in our Table 4 are analogous to those reported in Table 5 of Kollmann et al. (this issue-b); however, our testing procedure uses random draws, which are different from those used by Juillard and Villemot (this issue). As Table 4 shows, the errors we found are very close to those shown in Table 5 of Kollmann et al. (this issue-b). Furthermore, all of the integration methods considered again lead to solutions of nearly the same accuracy, with the exception of the $Q(1)$ rule (which produces slightly less accurate solutions).

The key finding in our Table 4 is that CGA can compute solutions of the same accuracy as those submitted for the comparison in Kollmann et al. (this issue-b), but at a much lower cost. For example, we reduce the computational time for Model 5 with $N=10$ countries from about 35 h (reported in Table 3 of Kollmann et al., this issue-b) to 7 min (reported in our Table 4) without a visible loss in accuracy by replacing our costly, baseline two-step integration procedure with just its first step based on the $M1$ monomial rule with $2N$ nodes.[17] Alternatively, we can solve a 10-country version of Model 5 in about 2 min using the $Q(1)$ rule at the cost of a modest loss in accuracy. In Models 6–8, the cheaper integration rules reduce the computational time in roughly the same proportion as in Model 5. However, Models 6–8 are generally more costly to solve than Model 5 because of higher costs of computing the intratemporal choice. The computational time for these models can be reduced (at the cost of some accuracy loss) by combining the iteration-on-allocation and precomputation approaches,

[17] At the moment of submission of our solutions for the comparison in Kollmann et al. (this issue-b), we did not have a reliable accuracy test, and we submitted the solutions obtained under the most accurate integration procedure feasible for CGA, which is a two-step combination of the $M1$ and $Q(2)$ rules.

Table 4
Accuracy and speed of CGA for Models 5–8 under four integration rules.

N	$Q(2)$			$M2$			$M1$			$Q(1)$		
	Δ_{mean}	Δ_{max}	CPU	Δ_{mean}	$\Delta_{\max}$	CPU	Δ_{mean}	$\Delta_{\max}$	CPU	Δ_{mean}	$\Delta_{\max}$	CPU
Model 5												
2	−6.39	−4.35	72	−6.39	−4.35	91	−6.39	−4.35	73	−5.87	−4.30	63
4	−6.44	−4.45	145	−6.44	−4.45	227	−6.44	−4.45	105	−5.70	−4.36	76
6	−6.44	−4.66	575	−6.44	−4.66	661	−6.44	−4.66	161	−5.64	−4.50	94
8	−6.42	−4.76	4319	−6.42	−4.76	1822	−6.42	−4.76	290	−5.62	−4.57	115
10	−6.39	−4.74	144,327	−6.38	−4.75	4425	−6.38	−4.75	420	−5.62	−4.56	137
Model 6												
2	−6.28	−4.30	1231	−6.28	−4.30	1417	−6.28	−4.30	1234	−5.71	−4.23	963
4	−6.31	−4.45	2687	−6.31	−4.45	3804	−6.31	−4.45	1781	−5.52	−4.35	1104
6	−6.32	−4.62	8556	−6.32	−4.62	8128	−6.32	−4.62	2207	−5.46	−4.40	1052
8	−6.31	−4.66	38,392	−6.31	−4.66	19,635	−6.31	−4.66	3864	−5.44	−4.46	1444
Model 7												
2	−6.08	−4.25	759	−6.08	−4.25	912	−6.08	−4.25	768	−5.66	−4.21	614
4	−6.09	−4.21	1842	−6.09	−4.21	2745	−6.09	−4.21	1402	−5.52	−4.16	887
6	−6.09	−4.33	6254	−6.09	−4.33	7723	−6.09	−4.33	2449	−5.46	−4.23	1173
Model 8												
2	−5.87	−3.83	1185	−5.87	−3.83	1400	−5.87	−3.83	1177	−5.52	−3.79	894
4	−5.93	−4.13	3807	−5.93	−4.13	5631	−5.93	−4.13	2913	−5.38	−4.05	1790
6	−5.96	−4.22	13,414	−5.96	−4.22	14385	−5.96	−4.22	3869	−5.32	−4.09	1756

Note: Δ_{mean} and $\Delta_{\max}$ are, respectively, the average and maximum absolute errors across all optimality conditions (in log 10 units) in the test on a stochastic simulation of 10,000 observations; *CPU* is the time necessary to compute a solution (in seconds); N is the number of countries.

as described in Section 4.3. In addition, we can decrease the computational time for all models by reducing the number of clusters; see Judd et al. (2010a) for the corresponding experiments.

7.4. Hybrid of perturbation and accurate intratemporal-choice methods

In Section 7.1, we show that using accurate intratemporal-choice approaches in simulation can increase the accuracy of solution methods that compute the intratemporal choice with insufficient accuracy. A prominent example of such a method is perturbation, which in the studied models, produces small errors in the Euler equations but large errors in the intratemporal-choice conditions (especially, in the resource constraint); see Table 6 of Kollmann et al. (this issue-b) for the accuracy by equation for the first- and second-order perturbation methods by Kollmann, Kim and Kim (this issue-a) (referred to as PER1 and PER2, respectively). Consequently, there are potential benefits from constructing a hybrid of the standard perturbation method (used as a low-cost method for computing capital policy functions), and accurate intratemporal-choice methods (used to solve for consumption and labor after capital is computed).

To verify the above conjecture, we take the capital policy functions produced by the standard log-linearization method for two-country versions of Models 5–8 and accurately solve for consumption and labor in simulation using the iteration-on-allocation method. In Table 5, we compare the accuracy of the resulting hybrid method with that of the SSA, CGA, PER1 and PER2 methods, as reported in Table 5 of Kollmann et al. (this issue-b).

As Table 5 indicates, our hybrid method is far more accurate (by more than an order of magnitude) than PER1. It is even more accurate than PER2 and is only slightly less accurate than SSA. The hybrid method is still considerably less accurate than CGA. However, when comparing the hybrid method against CGA, we should take into account that the latter uses the

Table 5
Accuracy of the hybrid perturbation method and other solution methods for the two-country versions of Models 5–8.

Model	SSA		CGA		PER1		PER2		Hybrid	
	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$	Δ_{mean}	$\Delta_{\max}$
5	−4.79	−3.20	−6.39	−4.53	−3.69	−1.70	−5.13	−2.60	−4.50	−2.88
6	−4.79	−3.12	−6.38	−4.50	−3.53	−1.45	−4.84	−2.30	−4.56	−2.84
7	−4.08	−3.08	−6.15	−4.19	−3.05	−1.20	−4.21	−1.90	−4.57	−2.87
8	−4.62	−2.90	−5.98	−4.07	−3.11	−1.25	−4.35	−2.09	−4.36	−2.64

Note: Δ_{mean} and $\Delta_{\max}$ are, respectively, the average and maximum absolute errors across all optimality conditions (in log 10 units) in the test on a stochastic simulation of 10,000 observations: The results for SSA, CGA, PER1 and PER2 are reproduced from Kollmann et al. (this issue-b).

second-degree polynomial, while the former uses the first-degree polynomial. The second-order hybrid perturbation method is likely to be more accurate than the first-order one.

Finally, to construct the hybrid perturbation method, we can use any numerical procedure that can accurately solve the system of intratemporal-choice conditions with respect to consumption and labor; e.g., a standard Newton-type solver. However, as we argued before, the iteration-on-allocation solver has advantages over other solvers. It is a good candidate for a fusion with perturbation.

8. Conclusion

In this paper, we offer a mix of techniques that taken together allows us to address the challenges of high-dimensional problems. First, SSA and CGA operate on ergodic-set domains which in high-dimensional problems are normally just a tiny fraction of the standard hypercube domain used by other methods. Second, we use efficient and numerically stable linear approximation approaches described in Judd et al. (2009). Third, we rely on low-cost integration methods, namely, a Monte Carlo integration method combined with regression under SSA, and non-product monomial rules and the Gauss–Hermite rule with one node under CGA. Fourth, we solve for the intratemporal choice using the accurate iteration-on-allocation and precomputation methods. Fifth, we show that other polynomial families (such as Smolyak polynomials studied in Malin et al., this issue) can help increase accuracy and speed of our solution methods relative to our baseline family of ordinary polynomials. Finally, we argue that proper coordination of the approximation, integration and intratemporal-choice strategies is critical for accuracy, speed and numerical stability of our solution methods.

If one uses a standard desktop computer (as we do), it is crucial for speed to vectorize computations. We iterate on policy functions of all countries simultaneously rather than country by country, and we solve for the intratemporal choice in all points at once rather than point by point. In contrast, if one uses parallel computing tools, it is essential to separate computations by country, grid point, integration node, etc. We should emphasize that the methods described in the paper are naturally parallelizable.

In addition to our main SSA and CGA algorithms, we construct a hybrid solution algorithm that combines perturbation (used to compute policy functions for capital) and accurate intratemporal-choice methods (used to solve for consumption and labor allocations). We find that such a hybrid method delivers solutions that are more than an order of magnitude more accurate than those delivered by the pure perturbation method. This hybrid perturbation method can be useful for solving problems of much higher dimensionality than those studied in the present paper.

Acknowledgements

Lilia Maliar and Serguei Maliar acknowledge support from the Hoover Institution at Stanford University, the Stanford Institute for Theoretical Economics, the Center for Financial Studies in Frankfurt, the Paris School of Economics, 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. We thank Wouter Den Haan, Michel Juillard, Sébastien Villemot and an anonymous referee for useful comments. We thank Ben Malin and Paul Pichler for providing us with the Smolyak-polynomial terms and log-linear solutions, respectively.

Appendices

In this section, we present formulas used to implement the iteration-on-allocation and precomputation methods, as well as those used to parameterize the capital policy functions.

Appendix A

This section describes how we implement the iteration-on-allocation approach in Models 6–8 (and their corresponding symmetric counterparts, Models 2–4).

Model 6: Conditions (4), (5) and (2) can be represented as

$$\tilde{\ell}_t^j = \left[\frac{a_t^j (k_t^j)^{\alpha} \tau^1 b^1}{a_t^1 (k_t^1)^{\alpha} \tau^j b^j}\right]^{\eta^j/(1+\alpha\eta^j)} (\ell_t^1)^{\eta^j(1+\alpha\eta^1)/\eta^1(1+\alpha\eta^j)}, \quad j=2,\ldots,N, \tag{28}$$

$$\tilde{c}_t^j = \left[\frac{(1-\alpha)a_t^j A (k_t^j)^{\alpha} (l_t^j)^{-\alpha}}{b^j (l_t^j)^{1/\eta^j}}\right]^{\gamma^j} \quad \text{with} \begin{cases} l_t^j \equiv \ell_t^1, & j=1 \\ l_t^j \equiv \tilde{\ell}_t^j, & j=2,\ldots,N, \end{cases} \tag{29}$$

$$\tilde{\ell}_t^1 = \left[\frac{\sum_{j=1}^N \left[\tilde{c}_t^j + k_{t+1}^j - k_t^j + \frac{\phi}{2} k_t^j \left(\frac{k_{t+1}^j}{k_t^j} - 1 \right)^2 \right] - \sum_{j=2}^N a_t^j A (k_t^j)^\alpha (\tilde{\ell}_t^j)^{1-\alpha}}{a_t^1 A (k_t^1)^\alpha} \right]^{1/(1-\alpha)}, \tag{30}$$

where $\{\gamma^j, \eta^j, b^j\}^{j=1,\ldots,N}$ are the utility-function parameters, and α is the share of capital in production. Condition (28) is obtained by combining (4) and (5), and conditions (29) and (30) follow from (5) and (2), respectively. For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, Eqs. (28)–(30) define a mapping $\tilde{\ell}_t^1 = g(\ell_t^1)$. We iterate on labor of the first country, $\tilde{\ell}_t^1$, as follows: assume some initial ℓ_t^1; compute $\{\tilde{c}_t^j\}^{j=2,\ldots,N}$ from (28); find $\{\tilde{c}_t^j\}^{j=1,\ldots,N}$ from (29); obtain $\tilde{\ell}_t^1$ from (30); if $\ell_t^1 \neq \tilde{\ell}_t^1$, compute the next-iteration input as $(1-\xi)\ell_t^1 + \xi\tilde{\ell}_t^1$. Iterate until convergence.

Model 7: Conditions (5) and (4), respectively, are

$$\tilde{c}_t^j = \frac{\psi(L^e - \ell_t^j)}{1-\psi}(1-\alpha) a_t^j A (k_t^j)^\alpha (\ell_t^j)^{-\alpha}, \tag{31}$$

$$\tilde{\ell}_t^j = L^e - \left[(L^e - \ell_t^1)^{(1-\psi)(1-1/\gamma^1)} \frac{(\tilde{c}_t^1)^{\psi(1-1/\gamma^1)-1} \tau^1}{(\tilde{c}_t^j)^{\psi(1-1/\gamma^j)-1} \tau^j} \right]^{1/((1-\psi)(1-1/\gamma^j))}, \quad j = 2, \ldots, N, \tag{32}$$

where ψ is the utility-function parameter, and L^e is the labor endowment of the representative agent. The resource constraint is given by (30) and determines $\tilde{\ell}_t^1$. For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, Eqs. (30)–(32) define a mapping $\{\tilde{\ell}_t^j\}^{j=1,\ldots,N} = g(\{\ell_t^j\}^{j=1,\ldots,N})$. We iterate on labor of all countries, $\{\tilde{\ell}_t^j\}^{j=1,\ldots,N}$, as follows: assume some initial $\{\ell_t^j\}^{j=1,\ldots,N}$, find $\{\tilde{c}_t^j\}^{j=1,\ldots,N}$ from (31); compute $\{\tilde{\ell}_t^j\}^{j=2,\ldots,N}$ and $\tilde{\ell}_t^1$ from (32) and (30), respectively; if $\ell_t^j \neq \tilde{\ell}_t^j$ for $j=1,\ldots,N$, calculate the next-iteration input as $(1-\xi)\ell_t^j + \xi\tilde{\ell}_t^j$. Iterate until convergence.

Model 8: Conditions (5), (4) and (2), respectively, are

$$\tilde{c}_t^j = \left[\frac{(1-\alpha) a_t^j A (\ell_t^j)^{\mu^j - 1} (\alpha (k_t^j)^{\mu^j} + (1-\alpha)(\ell_t^j)^{\mu^j})^{1/\mu^j - 1}}{b^j} \right]^{\chi^j} (L^e - \ell_t^j), \tag{33}$$

$$\tilde{\ell}_t^j = L^e - \left[\frac{1}{b^j} \left(\frac{u_{c,t}^1 \tau^1}{(\tilde{c}_t^j)^{-1/\chi^j} \tau^j} \right)^{(1-1/\chi^j)/(1/\chi^j - 1/\gamma^j)} - \frac{(\tilde{c}_t^j)^{1-1/\chi^j}}{b^j} \right]^{1/(1-1/\chi^j)}, \quad j = 2, \ldots, N, \tag{34}$$

$$\tilde{\ell}_t^1 = \left[\left(\frac{f_t^1}{a_t^1 A (1-\alpha)^{1/\mu^1}} \right)^{\mu^j} - \frac{\alpha (k_t^1)^{\mu^1}}{1-\alpha} \right]^{1/\mu^1}, \tag{35}$$

where $\{\chi^j, \mu^j\}^{j=1,\ldots,N}$ are the utility-function parameters; $u_{c,t}^j$ for $j=1$ and f_t^1 are, respectively, the t-period marginal utility of consumption and output of country 1, defined as

$$u_{c,t}^j \equiv [(c_t^j)^{1-1/\chi^j} + b^j (L^e - \ell_t^j)^{1-1/\chi^j}]^{(1/\chi^j - 1/\gamma^j)/(1-1/\chi^j)} (c_t^j)^{-1/\chi^j}, \tag{36}$$

$$f_t^1 \equiv \sum_{j=1}^N \left[\tilde{c}_t^j + k_{t+1}^j - k_t^j + \frac{\phi}{2} k_t^j \left(\frac{k_{t+1}^j}{k_t^j} - 1 \right)^2 \right] - \sum_{j=2}^N a_{t+1}^j A (\alpha (k_t^j)^{\mu^j} + (1-\alpha)(\ell_t^j)^{\mu^j})^{1/\mu^j}. \tag{37}$$

For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, Eqs. (33)–(35) define a mapping $\{\tilde{\ell}_t^j\}^{j=1,\ldots,N} = g(\{\ell_t^j\}^{j=1,\ldots,N})$. We iterate on labor of all countries, $\{\tilde{\ell}_t^j\}^{j=1,\ldots,N}$, as follows: assume some initial $\{\ell_t^j\}^{j=1,\ldots,N}$; find $\{\tilde{c}_t^j\}^{j=1,\ldots,N}$ from (33); compute $\{\tilde{\ell}_t^j\}^{j=2,\ldots,N}$ and $\tilde{\ell}_t^1$ using (34) and (35), respectively; if $\ell_t^j \neq \tilde{\ell}_t^j$ for $j=1,\ldots,N$, calculate the next-iteration input as $(1-\xi)\ell_t^j + \xi\tilde{\ell}_t^j$. Iterate until convergence.

Appendix B

In this section, we show that in Model 2, the solution manifold for aggregate consumption can be precomputed in terms of two composite arguments independently of the number of agents. Since all agents are identical in preferences and have identical welfare weights, $\tau^j = 1$ for $j=1,\ldots,N$, the ratio of marginal utilities of any two agents in (4) is equal across agents.

As a result, $c_t^j = c_t/N$ for all j. From the intratemporal FOC (5), we obtain

$$\ell_t^j = \left[\frac{c_t^{1/\gamma} N^{-1/\gamma} b}{(1-\alpha) A a_t^j (k_t^j)^{\alpha}}\right]^{-\eta/(1+\alpha\eta)}. \tag{38}$$

Substituting (38) into resource constraint (2), we obtain

$$c_t = c_t^{-\eta(1-\alpha)/\gamma(1+\alpha\eta)} q_t + d_t, \tag{39}$$

with the composite variables q_t and d_t being defined as

$$q_t = \frac{\sum_{j=1}^{N} [A a_t^j (k_t^j)^{\alpha}]^{1+\eta(1-\alpha)/(1+\alpha\eta)}}{\left[\frac{N^{-1/\gamma} b}{(1-\alpha)}\right]^{\eta(1-\alpha)/(1+\alpha\eta)}}, \quad d_t = -\frac{\phi}{2} k_t^j \left(\frac{k_{t+1}^j}{k_t^j} - 1\right)^2 + k_t^j - k_{t+1}^j. \tag{40}$$

We can use Eq. (39) to precompute consumption c_t in terms of two variables q_t and d_t. (If welfare weights differ across agents and consequently, individual consumption is not equal to average consumption, we can still construct the intratemporal-choice manifolds in terms of the same composite variables; see Maliar and Maliar, 2001, 2003b, for related results).

Outside the main iterative cycle, take a grid of P values for q_t and d_t, i.e., $\{q_p, d_p\}_{p=1,\ldots,P}$. For each grid point $p=1,\ldots,P$, use a numerical solver to find a solution for c_p from Eq. (39) represented in a form suited for precomputation. Interpolate the constructed set function to the relevant continuous domain to obtain the manifold $\hat{c}(q,d)$. Inside the main iterative cycle, given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, compute q_t and d_t from (40) for each t, use the precomputed manifold to find aggregate consumption, $c_t = \hat{c}_t(q_t, d_t)$, and compute individual labor ℓ_t^j from (38) for $j=1,\ldots,N$.

Appendix C

In this section, we provide Euler equation (19) corresponding to Models 5–8.

Model 5:

$$k_{t+1}^j = E_t\left\{\beta \frac{(c_{t+1}^j)^{-1/\gamma^j}}{(c_t^j)^{-1/\gamma^j}\omega_t^j}\left[\theta_{t+1}^j + \alpha a_{t+1}^j A (k_{t+1}^j)^{\alpha-1}\right] k_{t+1}^j\right\}. \tag{41}$$

Model 6:

$$k_{t+1}^j = E_t\left\{\beta \frac{(c_{t+1}^j)^{-1/\gamma^j}}{(c_t^j)^{-1/\gamma^j}\omega_t^j}\left[\theta_{t+1}^j + \alpha a_{t+1}^j A (k_{t+1}^j)^{\alpha-1} (\ell_{t+1}^j)^{1-\alpha}\right] k_{t+1}^j\right\}. \tag{42}$$

Model 7:

$$k_{t+1}^j = E_t\left\{\beta \frac{\frac{[(c_{t+1}^j)^{\psi}(L^e - \ell_{t+1}^j)^{1-\psi}]^{1-1/\gamma^j}}{c_{t+1}^j}}{\frac{[(c_t^j)^{\psi}(L^e - \ell_t^j)^{1-\psi}]^{1-1/\gamma^j}}{c_t^j}\omega_t^j}\left[\theta_{t+1}^j + \alpha a_{t+1}^j A (k_{t+1}^j)^{\alpha-1} (\ell_{t+1}^j)^{1-\alpha}\right] k_{t+1}^j\right\}. \tag{43}$$

Model 8:

$$k_{t+1}^j = E_t\left\{\beta \frac{u_{c,t+1}^j}{u_{c,t}^j \omega_t^j}\left[\theta_{t+1}^j + \alpha a_{t+1}^j A (k_{t+1}^j)^{\mu^j - 1} (\alpha (k_{t+1}^j)^{\mu^j} + \alpha (\ell_{t+1}^j)^{\mu^j})^{1/\mu^j - 1}\right] k_{t+1}^j\right\}, \tag{44}$$

where $u_{c,t}^j$ is defined as in (36).

References

Christiano, L., Fisher, D., 2000. Algorithms for solving dynamic models with occasionally binding constraints. Journal of Economic Dynamics and Control 24, 1179–1232.

Den Haan, W., 1990. The optimal inflation path in a Sidrauski-type model with uncertainty. Journal of Monetary Economics 25, 389–409.

Den Haan, W., Judd, K., Juillard, M., this issue. Computational suite of models with heterogeneous agents: multi-country real business cycle models. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2010.09.010.

Den Haan, W., Marcet, A., 1990. Solving the stochastic growth model by parameterizing expectations. Journal of Business and Economic Statistics 8, 31–34.

Gaspar, J., Judd, K., 1997. Solving large-scale rational-expectations models. Macroeconomic Dynamics 1, 45–75.

Judd, K., 1992. Projection methods for solving aggregate growth models. Journal of Economic Theory 58, 410–452.

Judd, K., 1998. Numerical Methods in Economics. The MIT Press, Cambridge, MA, London, England.

Judd, K., Maliar, L., Maliar, S., 2009. Numerically stable stochastic simulation approaches for solving dynamic economic models. NBER Working Paper 15296.

Judd, K., Maliar, L., Maliar, S., 2010a. A cluster-grid projection method: solving problems with high dimensionality. NBER Working Paper 15965.

Judd, K., Maliar, L., Maliar, S., 2010b. Numerically stable stochastic simulation approaches for solving dynamic economic models. Manuscript.

Judd, K., Maliar, L., Maliar, S., 2010c. One-node quadrature beats Monte Carlo: a generalized stochastic simulation algorithm. Manuscript.
Juillard, M., Villemot, S., this issue. Multi-country real business cycle models: accuracy tests and testing bench. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2010.09.011.
Kollmann, R., Kim, S., Kim, J., this issue-a. Solving the multi-country real business cycle model using a perturbation method. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2010.09.012.
Kollmann, R., Maliar, S., Malin, B., Pichler, P., this issue-b. Comparison of solutions to the multi-country real business cycle model. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2010.09.013.
Maliar, L., Maliar, S., 2001. Heterogeneity in capital and skills in a neoclassical stochastic growth model. Journal of Economic Dynamics and Control 25, 1367–1397.
Maliar, L., Maliar, S., 2003a. The representative consumer in the neoclassical growth model with idiosyncratic shocks. Review of Economic Dynamics 6, 362–380.
Maliar, L., Maliar, S., 2003b. Parameterized expectations algorithm and the moving bounds. Journal of Business and Economic Statistics 21, 88–92.
Maliar, L., Maliar, S., 2004. Comparing numerical solutions of models with heterogeneous agents (Model A): a simulation-based parameterized expectations algorithm. Manuscript 〈http://www.stanford.edu/∼maliars〉.
Maliar, L., Maliar, S., 2005. Parameterized expectations algorithm: how to solve for labor easily. Computational Economics 25, 269–274.
Maliar, L., Maliar, S., 2007. Comparing numerical solutions of models with heterogeneous agents (Model A): a simulation-based parameterized expectations algorithm. Manuscript 〈http://www.stanford.edu/∼maliars〉.
Malin, B., Krueger, D., Kubler, F., this issue. Solving the multi-country real business cycle model using a Smolyak-collocation method. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2010.09.015.
Marcet, A., 1988. Solution of nonlinear models by parameterizing expectations. Manuscript, Carnegie Mellon University.
Pichler, P., this issue. Solving the multi-country real business cycle model using a monomial rule Galerkin method. Journal of Economic Dynamics and Control, doi:10.1016/j.jedc.2010.09.009.
Stroud, A., 1971. Approximate Integration of Multiple Integrals. Prentice Hall, Englewood Cliffs, NJ.