Solving the Multi-Country Real Business Cycle Model Using Ergodic Set Methods

Serguei Maliar, Lilia Maliar and Kenneth Judd*

July 30, 2010

Abstract

We apply the stochastic simulation algorithm, described in Judd, Maliar and Maliar (2009), and the projection cluster-grid algorithm, developed in Judd, Maliar and Maliar (2010a), to solving a collection of multi-country models. Four techniques 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 simulation combined with regression. We show that accuracy in intratemporal choice is crucial for the overall accuracy of solutions and offer two novel approaches, precomputation and iteration-on-allocation, that can solve for intratemporal allocations accurately and fast. We also propose a hybrid solution algorithm that combines the perturbation and accurate intratemporal-choice methods.

JEL classification : C6; C63; C68; C88

Key Words : Nonlinear dynamic models; Heterogeneous agents; Complete markets, Stochastic simulation; Parameterized expectations; Clusters; Perturbation

*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, Ben Malin, Sébastien Villemot and anonymous referee for useful comments.
Corresponding author: Serguei Maliar, T24, Hoover Institution, 434 Galvez Mall, Stanford University, Stanford, CA 94305-6010, USA, email: *maliars@stanford.edu*

1 Introduction

In the present paper, we show how to apply two ergodic-set algorithms to solving a collection of multi-country real business cycle models, proposed in Den Haan, Judd and Juillard (2010) (henceforth, DJJ) in the context of the current JEDC project. 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) (henceforth, JMM). The studied models include up to 10 countries (i.e., 20 state variables) and feature heterogeneity in fundamentals (preferences and technologies) and endogenous labor-leisure choice, as well as complete markets, adjustment costs, continuously valued state variables, and non-additively separable preferences and technologies.

Both the SSA and CGA build on strategies that allow us 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 domains that enclose the ergodic set. The SSA computes a solution on a grid composed from simulated points, whereas the CGA constructs a grid by clustering simulated data. Focusing on 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, we parameterize decision rules by a polynomial with additively separable terms, which allows us to estimate the polynomial coefficients using linear approximation methods. Such methods have low cost, are numerically stable and allow us to find the decision rules of all countries at once rather than on a country-by-country basis. Third, we compute fixed-point values of the polynomial coefficients rules using a fixed-point iteration method, the cost of which does not increase significantly with the dimensionality of the problem. Fourth, we evaluate conditional expectations using integration methods that are particularly suitable for high-dimensional applications: Namely, the SSA combines Monte Carlo simulation and regression in a manner that makes it possible to compute

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

[2]The SSA is similar to other stochastic simulation methods such as the simulation-based version of the parameterized expectations algorithm (PEA) by Marcet (1988), and Den Haan and Marcet (1990), the rules-of-thumb algorithm by Smith (1991); see JMM (2010b) for a discussion.

expectations (integrals) in all simulated points simultaneously, whereas the CGA performs numerical integration using inexpensive non-product monomial rules and a product Gauss-Hermite rule with small numbers of nodes (including the rule with one node).

The models studied in JMM (2010a, 2010b) are more challenging in the dimensional aspect than those considered in the current JEDC project since they include up to 200 countries; however, these models abstract from the labor-leisure choice. The presence of an endogenous labor-leisure choice complicates the solution procedure since one of its consequence is that intratemporal-choice variables (i.e., the control variables determined by a given law of motion for state variables) cannot be expressed analytically in terms of state variables and must be approximated using numerical methods. In our numerical experiments, the accuracy of the intratemporal choice plays an important role in the overall accuracy of the solutions. For example, in the context of a two-country model, we show that approximating at least one intratemporal-choice variable (e.g., consumption of the first country) with a polynomial function of the same (second) degree as that used to approximate the capital decision functions results in much larger errors in the intratemporal-choice conditions than in the intertemporal-choice conditions (Euler equations). Insufficient accuracy with regards to intratemporal choice drives down the overall accuracy of solutions. The importance of the intratemporal choice for accuracy is also seen from Table 6 of the comparison paper by Kollmann, Maliar, Malin and Pichler (2010) (henceforth, KMMP).

In the present paper, we describe two novel approaches to the problem of solving for intratemporal choice that enable us to attain a high degree of accuracy. Our first approach, called *iteration-on-allocation*, relies on a numerical solver that implements fixed-point iteration directly on the intratemporal-choice variables, i.e., the intratemporal-choice functions are never constructed explicitly! This approach allows us to achieve effectively zero errors in all intratemporal-choice conditions, including the budget 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, this approach can work with vectors and matrices, which makes it very fast when used in vectorized applications.

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

Our second approach, called *precomputation*, separates the intertemporal and intratemporal choices, precomputes the intratemporal-choice functions on an appropriately chosen grid of points outside the main iterative cycle and uses the constructed 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 yields very accurate intratemporal allocations in the examples considered.

One feature of the present analysis that merits a separate discussion is that our choice of which decision functions to parameterize affects the speed of the SSA and CGA. In the present paper, we parameterize a law of motion for capital stock in terms of the economy's state variables. An advantage of this parameterization is that we can construct the whole equilibrium path for capital without solving for the intratemporal-choice variables (consumption and labor). For a given capital path, the problem of finding intratemporal choices can be vectorized: we can compute all consumption and labor allocations at once rather than on a point-by-point basis. This makes it possible to exploit vectorized versions of the iteration-on-allocation and precomputation approaches.

The accuracy and speed of the SSA and CGA in the context of the given project are assessed in the comparison paper of KMMP (2010). In the present paper, we report a few additional experiments that demonstrate how the accuracy and speed of the CGA depend on the specific integration method, approximating polynomial function and approach used to solve for the intratemporal choice.

We find that the product Gauss-Hermite rule (with more than one node in each dimension) and non-product monomial formulas deliver essentially the same accuracy of solutions; in our experiments, solution errors are identical up to the fourth digit. Furthermore, a one-node Gauss-Hermite rule leads to solution errors that are no more than 20% larger than those produced by more accurate integration formulas. This is an important finding, since a low-cost one-node formula can be used to solve problems of a very high dimensionality. Finally, in the two-country examples, going from a second- to third-degree polynomial increases accuracy (i.e., decreases solution errors) by about an order of magnitude. The above three regularities are parallel to

[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 iterative cycle. Maliar and Maliar (2007) implement this approach in the context of the current JEDC project.

those reported by JMM (2010a) for a model with non-valued leisure.

We report the following new findings in the context of models with valued leisure. First, the Smolyak polynomial used in the Smolyak collocation algorithm by Malin, Krueger and Kubler (2010) (henceforth, MKK) is also a useful choice for the CGA. The Smolyak polynomial is composed of a complete second-degree polynomial and selected polynomial terms up to degree four. It has only four times more terms than the usual second-degree polynomial but can achieve nearly the same degree of accuracy as the third-degree complete polynomial. Second, the method's accuracy is more dependent on how accurately we compute the intratemporal choice in simulation procedure when we test accuracy than on how accurately we compute the intratemporal choice in the solution procedure. By recognizing this, we end up with a higher degree of accuracy. Finally, our results on the importance of accuracy in the intratemporal choice led us to propose a hybrid approach that combines a standard perturbation method and our accurate intratemporal-choice methods. Specifically, we use a decision rule for capital delivered by a log-linearization method, and we solve for the consumption and labor accurately using the iteration-on-allocation method. The hybrid solutions we obtain are more than an order of magnitude more accurate than those produced by the original log-linear decision rules. This hybrid approach can be useful for solving problems of very high dimensionality.

The rest of the paper is as follows: Section 2 presents the model and derives the first-order conditions. Section 3 describes how we address the challenges of high-dimensional problems. Section 4 develops two approaches for computing intratemporal-choice variables. Section 5 describes the steps of the SSA and CGA. Section 6 outlines the implementation of the SSA and CGA in the context of the current JEDC project. Section 7 presents the numerical results and introduces the 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.,

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

subject to

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

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 consumption, labor, capital, productivity level, utility function, production function and welfare weight of a country $j = 1, ..., N$, respectively; β is the discount factor; and ϕ is the adjustment-cost parameter. Initial condition $(\boldsymbol{k}_0, \boldsymbol{a}_0)$ is given, where $\boldsymbol{k}_0 \equiv \left(k_0^1, ..., k_0^N\right)$ and $\boldsymbol{a}_0 \equiv \left(a_0^1, ..., a_0^N\right)$. The process for productivity levels in country j is given by

$$\ln a_t^j = \rho \ln a_{t-1}^j + \sigma \left(e_t + e_t^j \right), \qquad e_t, e_t^j \sim \mathcal{N}(0,1), \tag{3}$$

where e_t and e_t^j are a common and country-specific productivity shocks, respectively; 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

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

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

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

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

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

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

In equations $(4)-(6)$ and hereafter, notation of type F_{x_m} stands for the first-order partial derivative of a function $F(..., x_m, ...)$ with respect to a variable x_m.

As part of the present JEDC project, eight models are considered. Models 1, 2, 3 and 4 have the same preferences and technologies as do Models 5, 6, 7 and 8, respectively, however, the former models assume identical preferences and technologies parameters for all countries, while the latter models have different parameters across countries. Models 1 and 5 do not have an endogenous labor-leisure choice, while the other models do have such a choice. A description of the models studied in the project, including the choice of the models' parameters, is provided in Juillard and Villemot (2010).

Intertemporal versus intratemporal choices For our numerical methods, it is convenient to make a distinction between intertemporal and intratemporal choices. In both of our algorithms, we parameterize a future capital stock, k_{t+1}^j, by a function of the current state variables,

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

where $\boldsymbol{k}_t \equiv (k_t^1, ..., k_t^N)$ and $\boldsymbol{a}_t \equiv (a_t^1, ..., a_t^N)$. We call k_{t+1}^j an *intertemporal-choice* variable because it captures dynamic aspects of the social planner's choice: it defines an equilibrium law of motion of the countries' capital stock. Once N future capital stocks $\boldsymbol{k}_{t+1}$ are fixed, we are left in each period t with the static problem of finding a solution to a system of $2N$ optimality conditions (i.e., one resource constraint (2), $N-1$ conditions (4) and N conditions (5)) with $2N$ unknowns - N consumption choices, $(c_t^1, ..., c_t^N) \equiv \boldsymbol{c}_t$, and N labor choices, $(\ell_t^1, ..., \ell_t^N) \equiv \boldsymbol{\ell}_t$. From this system, we need to find a solution in the form:

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

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 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 DJJ (2010) pose four challenges for numerical methods designed to find a global solution: (*i*) the large size of the domain on which the solution is computed, (*ii*) the large number of polynomial coefficients in an approximating polynomial function, (*iii*) the large number of integration nodes for approximating the conditional expectation function, and (*iv*) the high cost of solving for the intratemporal choice.

JMM (2010a, 2010b) show how to address the first three challenges in the context of the CGA and SSA methods, respectively. The problems that are solved in JMM (2010a, 2010b) are of larger dimensionality than those that are included in the current JEDC project as they include up to 200 countries; however, these problems are also simpler in the structure of their intratemporal choice, which can be characterized analytically. The strategies used by JMM (2010a, 2010b) to address challenges (*i*), (*ii*) and (*iii*) are discussed in Sections 3.1, 3.2 and 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 dimensions, is not studied in JMM (2010a, 2010b). In the present paper, solving accurately for intertemporal choice proved to be crucial for the overall accuracy of solutions. We address the intratemporal-choice challenge in a separate section, Section 4.

3.1 Multi-dimensional domain

To make a numerical method suitable for high-dimensional applications, we should restrict attention to a relatively small set of grid points in the multi-dimensional space.[5] Both the SSA and CGA achieve this goal by focusing on the ergodic set of points realized in equilibrium. In Figure 1a, we show the ergodic set constructed from a simulated series of $10,000$ observations, which are produced by a standard representative-agent neoclassical stochastic growth model. The SSA computes the solution on the given set of simulated points and keeps the number of simulated points fixed at $T = 10,000$ independently of the number of countries N; see JMM (2009) for details. The CGA chooses a more efficient representation of the ergodic set; namely, it

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

replaces a large number of closely situated simulated points with relatively few representative points. The CGA transforms correlated variables into uncorrelated principal components (denoted PC_t^1 and PC_t^2) using a principal components (PCs) decomposition (see Figure 1b); normalizes principal components to zero mean and unit variance (see Figure 1c); and constructs I clusters using a clustering algorithm. It subsequently uses the centers of the constructed clusters as a grid for projections; see JMM (2010a) for details. (Note that the 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 the SSA and CGA to compute a solution only in the relevant area of the state space (an ellipsoid area shown in Figure 1a) and thus allows us to eliminate an enormously large number of grid points that are never visited in equilibrium. For example, for a model with 100 state variables, our endogenous ergodic set domain is only about a $2 \cdot 10^{-70}$ fraction of a multi-dimensional hypercube which encloses the ergodic set; see JMM (2010a) for a further discussion.

3.2 Multi-dimensional polynomials

If we use complete polynomials for parameterizing decision rules, the number of polynomial terms in the first-, second- and third-degree polynomial grows linearly, quadratically and cubically, respectively, with the dimensionality of the problem; see Table 1 in JMM (2010a) for formulas and estimates of the number of polynomial terms in multi-dimensional complete polynomials.

To reduce the cost of finding polynomial coefficients in high dimensional problems, the SSA and CGA rely on three complementary techniques. First, both SSA and CGA parameterize decision rules by an additively separable polynomial function. For example, parameterizing the next-period capital stock of country j by a such a function of degree one yields:

$$k_{t+1}^j = v_0^j + v_1^j k_t^1 + ... + v_N^j k_t^N + v_{N+1}^j a_t^1 + ... + v_{2N}^j a_t^N, \tag{9}$$

where $\left(v_0^j, v_1^j, ..., v_N^j, v_{N+1}^j, ..., v_{2N}^j\right)' \equiv \boldsymbol{v}^j \in \mathbb{R}^{(2N+1)\times 1}$ is country's j vector of the polynomial coefficients. Additive separability of the polynomial function allows the SSA and CGA to estimate polynomial coefficients using the fast and numerically stable linear approximation methods described in JMM (2009, 2010b) including the least-squares method using SVD and QR factorization, Tikhonov regularization, least-absolute deviation methods and principal components method. Second, as noted in JMM (2010b), the additive separability of the polynomial function and linear approximation methods taken together allow an approximation of the decision rules of all N countries simultaneously rather than on a country-by-country basis. Finally, both the SSA and CGA find fixed point values of the polynomial coefficients using fixed-point iteration; this is a simple derivative-free iteration method for which cost does not increase considerably with the dimensionality of the problem.[6]

JMM (2010a, 2010b) evaluate how the cost of the CGA and SSA methods depends on the degree of the polynomial in the context of a multi-country model, which is similar to Model 1 of the current JEDC project. With regards to the CGA, JMM (2010a) report that complete first- and second-degree polynomials are feasible for models with up to $N = 200$ and up to $N = 40$ countries, respectively. With regards to the SSA, JMM (2010b) find that such polynomials are feasible for the models with up to $N = 200$ and up to $N = 30$ countries, respectively.

3.3 Multi-dimensional integration

Finding a solution requires the calculation of integrals that represent conditional expectation functions in the Euler equations. The SSA employs a Monte Carlo type of integration combined with regression, as is used in den Haan and Marcet (1990). The SSA integration procedure is efficient in the sense that a regression step makes it possible to infer expectations (to compute integrals) simultaneously in all T simulated points. If the length of simulations T is held fixed, the cost of this integration procedure does not grow substantially with dimensionality, though accuracy may decrease as more polynomial coefficients must be identified.

The CGA is a projection method, and it relies on deterministic methods of integration. The specific choice of integration method depends on the dimensionality of the problem. In a model with just one shock, 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 ex-

[6]Gaspar and Judd (1997) provide arguments in favor of fixed-point iteration over alternative iterative schemes, such as time iteration or Newton methods, for large-scale models.

pectation) by $E\left[g\left(\varepsilon\right)w\left(\varepsilon\right)\right]=\int_{\mathbb{R}}g\left(\varepsilon\right)w\left(\varepsilon\right)d\varepsilon\approx\sum_{h=1}^{H}w_{h}g\left(\varepsilon_{h}\right)$ for some nodes $\left\{\varepsilon_{h}\right\}_{h=1,...,H}$ and weights $\left\{w_{h}\right\}_{h=1,...,H}$. One can extend the Gaussian quadrature approach to a model with N exogenous shocks 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, p.271 and p.331) proposes to use non-product monomial integration formulas; a large collection of such formulas is available in Stroud (1971).

JMM (2010a) elaborate the monomial formulas for a heterogeneous-agent model similar to the one studied in the present paper, illustrate the use of such formulas by way of examples and provide an exhaustive comparison of the CGA's accuracy and cost under different integration strategies. Such strategies include the product Gauss-Hermite rule with 1, 2 and 3 nodes in each dimension (referred to as $Q\left(1\right)$, $Q\left(2\right)$ and $Q\left(3\right)$, respectively) and non-product monomial rules with $2N$ and $2N^{2}+1$ nodes (referred to as $M1$ and $M2$, respectively). Using second-degree polynomials, JMM (2010a) find that the integration formulas $Q\left(3\right)$, $Q\left(2\right)$, $M2$, $M1$ and $Q\left(1\right)$ are feasible for the models with up to $N=6$, $N=8$, $N=12$, $N=20$ and $N=40$ countries, respectively. Using first-degree polynomials, JMM (2010a) find that the formulas $M1$ and $Q\left(1\right)$ are feasible for the models with up to $N=100$ and $N=200$ countries, respectively. In the same spirit, the present paper builds the integration step of the CGA using non-product monomial formulas and compares the results to those obtained under the product Gausse-Hermite rules.

3.4 Coordinating computational strategies

As is argued in JMM (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 (measured in terms of unit-free Euler equation errors) of no more than 10^{-4}, it would be inefficient to compute integrals with accuracy of 10^{-10}, since doing so would increase costs without improving 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.

JMM (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. JMM (2010a) give an example of coordination between the approximation and integration strategies, which involves combining a second-degree polynomial with a low-cost, one-node Gauss-Hermite integration rule. This coordination makes it possible to increase N from 20 to 40 at a relatively low cost to accuracy.

In the presence of an 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 the intratemporal choice can drastically reduce the overall accuracy of solutions; the same result is also seen in Table 6 of the comparison paper by KMMP (2010).

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 present two novel approaches, iteration-on-allocation and precomputation, that allow us to solve for the intratemporal choice with a high degree of accuracy at a relatively low cost. Finally, in Section 4.4, we show for some of the models studied, combining the iteration-on-allocation and precomputation approaches can produce additional gains.

4.1 Standard intratemporal-choice approaches

Existing literature provides two approaches to computing the intratemporal choice. First, for a given k_t, a_t and k_{t+1}, one can solve a system of $2N$ equations, (2), (4) and (5), with respect to $2N$ unknowns $\left\{c_t^j\right\}^{j=1,\dots,N}$ and $\left\{\ell_t^j\right\}^{j=1,\dots,N}$ using a standard (quasi-)Newton method. This can be done both accurately and quickly for a one-country case. However, as dimensionality N increases, the cost of solving for the intratemporal choice can increase rapidly. This is because one should in general find a solution to the $2N$-dimensional system of equations on each grid point and/or integration node within an iterative cycle. The prohibitively high cost of this approach makes it unsuitable for high-dimensional applications.

Second, one can treat the intratemporal choice in the same way as the intertemporal choice, i.e., one can approximate the decision rules for the intratemporal-choice variables, $c_t^j = C^j(\boldsymbol{k}_t, \boldsymbol{a}_t)$ and $\ell_t^j = L^j(\boldsymbol{k}_t, \boldsymbol{a}_t)$, inside the main iterative cycle to satisfy the corresponding optimality conditions (2), (4) and (5). Note that we need not include $\boldsymbol{k}_{t+1}$ as an argument of the intratemporal-choice functions $C^j(\boldsymbol{k}_t, \boldsymbol{a}_t)$ and $L^j(\boldsymbol{k}_t, \boldsymbol{a}_t)$ because such functions are defined for the equilibrium intertemporal choice k_{t+1} determined by the state $(\boldsymbol{k}_t, \boldsymbol{a}_t)$ (in contrast, the intratemporal choice functions $\Phi^j(\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1})$ and $\Theta^j(\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1})$ in (8) are defined for any k_{t+1}). In our experiments, this approach did not produce sufficiently accurate results. Specifically, when we used a polynomial of the same (second) degree to approximate at least one intratemporal-choice function, as we do for approximating the intertemporal-choice (capital) functions, the intratemporal-choice errors were larger than the intertemporal-choice (Euler-equation) errors, which drove down the overall accuracy of solutions. Moreover, simultaneous iterations on the intertemporal- and intratemporal-choice functions both reduced numerical stability and decreased the speed of convergence.

4.2 Iteration-on-allocation

The first intratemporal-choice approach we use relies on a numerical solver called *(fixed-point) iteration-on-allocation*. This method's name emphasizes its application of fixed-point iteration 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 a polynomial approximating function. Iteration-on-allocation was first proposed by Maliar and Maliar (2004) in the context of the current JEDC project.

The iteration-on-allocation approach proceeds as follows:

- *Step* 1. Re-write conditions (2), (4) and (5) to define a function Γ that explicitly and uniquely maps a set of values $\boldsymbol{z}_t = (\boldsymbol{c}_t, \boldsymbol{\ell}_t)$ into a new set of values $\widetilde{\boldsymbol{z}}_t = \left(\widetilde{\boldsymbol{c}}_t, \widetilde{\boldsymbol{\ell}}_t\right)$. This is possible for all of the eight models studied by the current JEDC project):

$$\widetilde{\boldsymbol{z}}_t = \Gamma\left(\boldsymbol{z}_t\right). \tag{10}$$

- *Step* 2. Use some initial guess on $\boldsymbol{z}_t$ and calculate $\widetilde{\boldsymbol{z}}_t$ via mapping (10).
- *Step* 3. Use partial updating (damping) to compute an input for the next iteration as $(1-\varsigma)\,\boldsymbol{z}_t + \varsigma\widetilde{\boldsymbol{z}}_t$, where $\varsigma \in (0,1)$ is a damping parameter.

  Iterate until a fixed point is found such that $\boldsymbol{z}_t = \Gamma\left(\boldsymbol{z}_t\right)$ with a given degree of accuracy, i.e.,

$$\frac{1}{\varsigma \cdot T}\sum_{t=1}^{T}\left\|\frac{\widetilde{\boldsymbol{z}}_t - \boldsymbol{z}_t}{\boldsymbol{z}_t}\right\| < 10^{-\theta}. \tag{11}$$

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

On the initial iteration, we can assume that $\boldsymbol{z}_t$ is equal to a steady state value. Typically, we need not iterate on all $2N$ unknown elements of c_t and ℓ_t since there are explicit closed-form expressions relating these variables, and fixing one or a few of them allows us 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 the other models are given in Appendix A.

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

$$\widetilde{c}_t^j = \left[\left(c_t^1\right)^{-1/\gamma^1}\tau^1/\tau^j\right]^{-\gamma^j}, \qquad j = 2,...,N, \tag{12}$$

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

where $\left\{\gamma^j\right\}^{j=1,\ldots,N}$ *are the utility-function parameters, and* A *is a normalizing constant. For given* $\boldsymbol{k}_t$, $\boldsymbol{a}_t$ *and* $\boldsymbol{k}_{t+1}$, *equations* (12) *and* (13) *define a mapping*

$\widetilde{c}_t^1 = \Gamma\left(c_t^1\right)$. *We iterate on consumption of the first country,* c_t^1*, as follows: Assume some value for* c_t^1*; compute* $\left\{\widetilde{c}_t^j\right\}^{j=2,\dots,N}$ *from* (12)*; obtain* $\widetilde{c}_t^1$ *from* (13)*; if* $c_t^1 \neq \widetilde{c}_t^1$*, compute the input for the next iteration as* $(1-\varsigma)\, c_t^1 + \varsigma \widetilde{c}_t^1$*. Iterate until convergence.*

Iteration-on-allocation has two distinct advantages over the standard, Newton-type optimization methods. First, iteration-on-allocation does not require computing derivatives (such as Jacobian or Hessian) but instead performs direct calculations; 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 standard Newton-type methods compute a solution on a point-by-point basis and is thus more difficult to vectorize.

Convergence of fixed-point iteration is not in general guaranteed (for formal results about convergence of fixed-point iteration, see Judd 1998 p.165-166). However, damping can often be used to force convergence. In particular, by choosing an appropriate damping parameter ς, we are 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 that*

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

where c_t *is aggregate consumption that is given. Note that if* $\left\{\gamma^j\right\}^{j=1,\dots,N}$ *are of the same sign, then* $\Gamma'\left(c_t^1\right) < 0$*. There is a unique fixed-point value* c_t^1 *that satisfies* $c_t^1 = \Gamma\left(c_t^1\right)$ *(at this point,* $\Gamma\left(c_t^1\right)$ *crosses the* 45^o *line). However, convergence to this fixed point does not necessarily result from applying* Γ *iteratively to some initial guess* c_t^1*, i.e.,* $\Gamma\left(\dots\Gamma\left(\Gamma\left(c_t^1\right)\right)\right)$*. Depending on whether* Γ' *is larger than, smaller than or equal to minus one, the result will be convergence, divergence or cycling, respectively. (Note that slope of* Γ *depends on the model's parameters and welfare weights, as well as on the specific way in which* Γ *is constructed). Figures 2a and 2b demonstrate cases of fixed-point iteration convergence and divergence, respectively. Figures 2c and 2d illustrate fixed-point iteration with damping* $(1-\varsigma)\, c_t^1 + \varsigma\Gamma\left(c_t^1\right)$*. In particular, Figure 2d shows 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 functions outside of the main iterative cycle using either analytical derivations, numerical computations or a combination of both. This approach was originally proposed by Maliar and Maliar (2005) for constructing a labor-choice function 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, which we apply to the model (1) $-$ (3), consists of constructing the intratemporal-choice functions $\Phi^j(\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1})$ and $\Theta^j(\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1})$ in (8) as follows:

- *Step* 1. Outside of the main iterative cycle, choose a grid of P values $\left\{\boldsymbol{k}_p, \boldsymbol{a}_p, \boldsymbol{k}'_p\right\}_{p=1,...,P}$ for $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$. For each grid point $p = 1, ...P$, solve equations (2), (4) and (5) using a numerical solver with respect to consumption c^j_p and labor ℓ^j_p for $j = 1, ..., N$.

- *Step* 2. Extend the constructed set functions to the relevant domain using a preferred interpolation scheme (a global polynomial approximation, piecewise linear polynomial approximation, splines, etc.), such that

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

  where $\widehat{\Phi}^j$ and $\widehat{\Theta}^j$ are the precomputed consumption and labor functions 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 functions $\widehat{\Phi}^j(\boldsymbol{k}, \boldsymbol{a}, \boldsymbol{k}')$ and $\widehat{\Theta}^j(\boldsymbol{k}, \boldsymbol{a}, \boldsymbol{k}')$ to find the intratemporal choice given the state $(\boldsymbol{k}_t, \boldsymbol{a}_t)$ and the intertemporal choice $\boldsymbol{k}_{t+1}$.

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

Example 3 ***(Model 5).*** *Outside the main iterative cycle, take P values $\{c_p\}_{p=1,...,P}$ for aggregate consumption c_t. For each c_p, use a numerical solver to find a solution c_p^1 to* (4)*, presented in a form suitable for precomputation:*

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

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

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

With regards to the domain, we can precompute the intratemporal choice in Model 2 in terms of two composed arguments (see Appendix B for details). For the remaining models - Models 3, 4, 6, 7 and 8 - the intratemporal-choice functions 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 precompute the intratemporal choice on the same grid of values for the state variables $\boldsymbol{k}_t$ and $\boldsymbol{a}_t$ as the grid on which the capital decision functions are computed which is the ergodic set realized in equilibrium (augmented

[7]This fact was pointed out and exploited by Maliar and Maliar (2005) to compute the intratemporal choice in the standard neoclassical growth model.

appropriately to include the future capital stocks, $\boldsymbol{k}_{t+1}$).[8] For the SSA, the domain for precomputation is a set of simulated points; for the CGA, it is a set of the clusters' centers obtained from simulated points. Section 5 provides additional discussion of the SSA's and CGA's domains.

[maybe needed: Note that we cannot use $\boldsymbol{k}_{t+1}$ obtained from the equilibrium law of motion since in equilibrium, polynomial terms constructed using $\boldsymbol{k}_t$, $\boldsymbol{a}_t$ and $\boldsymbol{k}_{t+1}$ are linearly dependent. We therefore use a different set of values for $\boldsymbol{k}_{t+1}$, which is independent of $\boldsymbol{k}_t$ and $\boldsymbol{a}_t$].

4.4 Combining iteration-on-allocation and precomputation

For Models 3, 4, 7 and 8, the iteration-on-allocation approach requires iterating simultaneously on allocations of N countries $\left\{\ell_t^j\right\}^{j=1,\dots,N}$ (see Appendix A). In turn, the precomputation approach requires approximating the intratemporal 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 method that precomputes the labor functions in terms of three arguments and iterates on one allocation.[9] 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., $c_t^j = \Omega^j\left(k_t^j, a_t^j, c_t\right)$ and $\ell_t^j = \Lambda^j\left(k_t^j, a_t^j, c_t\right)$, $j = 1, \dots, N$. Model 2 is an example of an economy in which such functions can be constructed analytically (see Appendix B); generally, however, such functions must be precomputed numerically.[10]

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

[8]To construct the domain for the intratemporal-choice functions, 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 dimensionality, and the precomputation method will not be feasible for even a modestly large number of countries.

[9]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.

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

- *Step* 1. Outside the main iterative cycle, for each country j, choose a grid of P values $\{k_p^j, a_p^j, c_p\}_{p=1,...,P}$ for k_t^j,a_t^j, c_t. For each grid point $p = 1, ...P$, solve equations (4) and (5) using a numerical solver with respect to c_p^j and ℓ_p^j for $j = 1, ..., N$.

- *Step* 2. Interpolate the constructed labor function to the relevant domain,

$$c^j = \widehat{\Omega}^j \left(k^j, a^j, c\right) \quad \text{and} \quad \ell^j = \widehat{\Lambda}^j \left(k^j, a^j, c\right), \quad j = 1, ..., N, \tag{17}$$

  where $\widehat{\Omega}^j$ and $\widehat{\Lambda}^j$ are the precomputed consumption and labor functions of a country j, and $(k^j, a^j, c) \in \mathbb{R}^3$.

- *Step* 3. Substitute the precomputed labor functions $\widehat{\Lambda}^j (k^j, a^j, c)$ for $j = 1, ..., N$ in budget constraint (2) to define the mapping of the form $\widetilde{c}_t = \Gamma (c_t)$,

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

  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 value for c_t and calculate $\widetilde{c}_t$ from (18); if $c_t \neq \widetilde{c}_t$, compute an input for the next iteration equal to $(1 - \varsigma) c_t + \varsigma \widetilde{c}_t$. Iterate until convergence.

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

5 Two ergodic-set algorithms

To solve the model (1)$-$(3), we use two ergodic-set algorithms: the stochastic-simulation algorithm (SSA) and the cluster-grid algorithm (CGA). While both algorithms find a solution on the ergodic set, the way in which they use information on the ergodic set differs: The SSA uses information on both the density function and its support, while the CGA disregards the density function and uses information on the support only. For both methods, we rewrite the Euler equation (6) in a form suitable for parameterizing the capital decision rule $k_{t+1}^j = K^j(\boldsymbol{k}_t, \boldsymbol{a}_t)$:

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

where $\Psi^j(\boldsymbol{k}_t, \boldsymbol{a}_t; \boldsymbol{v}^j)$ is a flexible functional form used to parameterize the capital decision rule, and $\boldsymbol{v}^j$ is a vector of coefficients.[11] The matrix of the polynomial coefficients for all countries is denoted by $\boldsymbol{v} \equiv \left(\boldsymbol{v}^1, ..., \boldsymbol{v}^j, ..., \boldsymbol{v}^N\right)$.

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 $(\boldsymbol{k}_t, \boldsymbol{a}_t)$. For the polynomial function in example (9), we have $\boldsymbol{X}_t = \left(1, k_t^1, ..., k_t^N, a_t^1, ..., a_t^N\right) \in \mathbb{R}^{1\times(2N+1)}$, and $\left(v_0^j, v_1^j, ..., v_N^j, v_{N+1}^j, ..., v_{2N}^j\right)' \equiv \boldsymbol{v}^j \in \mathbb{R}^{(2N+1)\times 1}$.

We emphasize here that capital is the only variable we parameterize in terms of state variables k_t and a_t. The remaining variables (consumption and labor) are computed either by the iteration-on-allocation solver or by precomputation of the intratemporal-choice functions shown in form (8).

Properly separating the intertemporal and intratemporal choices is crucial to ensuring the speed of the SSA and CGA. Parameterizing the next-period capital decision rules in terms of the state variables, as in (19), has an important advantage over parameterizing other variables such as consumption and leisure. Namely, the equilibrium capital decision rule coincides with the equilibrium capital law of motion $k_{t+1}^j = K^j(\boldsymbol{k}_t, \boldsymbol{a}_t)$, $j = 1, ..., N$; this implies that we can first construct a path for state variables $\{\boldsymbol{k}_t, \boldsymbol{a}_t\}_{t=0,...,T}$, then subsequently select the corresponding intratemporal choice $\{\boldsymbol{c}_t, \boldsymbol{\ell}_t\}_{t=1,...,T}$. This feature allows us to increase the speed of our computations, because the iteration-on-allocation and precomputation methods can work with vectors and even matrices and can thus find the intratemporal choice for all periods or grid points at once instead of a slower, point-by-point basis.

[11]This kind of parameterization was originally 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.

5.1 Stochastic simulation algorithm

The stochastic simulation algorithm (SSA) simultaneously computes the ergodic distribution of state variables, its support and the associated decision rules. The SSA proceeds as follows:

Select and fix the simulations length T and initial condition $(\boldsymbol{k}_0, \boldsymbol{a}_0)$. Draw and fix for all simulations a sequence of productivity levels $\{\boldsymbol{a}_t\}_{t=1,...,T}$ using equation (3). If precomputation is used, construct the intratemporal-choice functions of type (8) as described in Sections 4.3 and 4.4.

- *Step* 1. For an iteration s, fix some vector of coefficients $\boldsymbol{v}(s)$. For each country $j = 1, ..., N$, use the assumed capital decision rule $k^j_{t+1} = \boldsymbol{X}_t \boldsymbol{v}^j$ to recursively calculate a sequence of capital stocks $\{\boldsymbol{k}_t\}_{t=1,...,T+1}$ corresponding to a given sequence of productivity levels $\{\boldsymbol{a}_t\}_{t=0,...,T}$.

- *Step* 2. Given $\{\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1}\}_{t=0,...,T}$, calculate $\{\boldsymbol{c}_t, \boldsymbol{\ell}_t\}_{t=0,...,T}$ using a vectorized version of either iteration-on-allocation or precomputation or the combination of the two methods, as described in Section 4.

- *Step* 3. For each country $j = 1, ..., N$, compute

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

  for $t = 0, ..., T-1$, which is the integrand of (19).

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

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

  where ϵ^j_t is a t-period approximation error corresponding to country j. Let the vector of coefficients estimated on iteration s be called $\widehat{\boldsymbol{v}}(s)$.

- *Step* 5. Compute the vector of coefficients for the subsequent $s+1$ iteration using fixed-point iteration:

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

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

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

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

where $k_{t+1}^{j}(s)$ and $k_{t+1}^{j}(s+1)$ are the j-th country capital stocks obtained on iterations, s and $s+1$, respectively, and $|\cdot|$ denotes the absolute value.

5.2 Cluster-grid algorithm

The cluster-grid algorithm (CGA) is a projection method that computes a solution on a grid constructed from clusters of simulated points. Let us represent the expression inside the conditional expectation in (19) as a new function $G^{j}(\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{\varepsilon}_{t+1})$,

$$\beta \frac{u_c^{j}\left(c_{t+1}^{j}, \ell_{t+1}^{j}\right)}{u_c^{j}\left(c_t^{j}, \ell_t^{j}\right) \omega_t^{j}} \left[\pi_{t+1}^{j} + a_{t+1}^{j} f_k^{j}\left(k_{t+1}^{j}, \ell_{t+1}^{j}\right)\right] k_{t+1}^{j} \equiv G^{j}(\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{\varepsilon}_{t+1}). \tag{24}$$

where $\boldsymbol{\varepsilon}_{t+1} \equiv \left(\varepsilon_{t+1}^{1}, ..., \varepsilon_{t+1}^{N}\right)$ and $\varepsilon_{t+1}^{j} \equiv e_{t+1} + e_{t+1}^{j}$. The CGA method then proceeds as follows:

Make an initial guess about the capital decision rule $k_{t+1}^{j} = \boldsymbol{X}_t \boldsymbol{v}^{j}$, $j = 1, ..., N$. Given initial condition $(\boldsymbol{k}_0, \boldsymbol{a}_0)$, draw a sequence of productivity levels $\{\boldsymbol{a}_t\}_{t=1,...,T}$ using (3), and simulate the time series solution $\{\boldsymbol{k}_t\}_{t=1,...,T+1}$. Construct I clusters on simulated data $\{\boldsymbol{k}_i, \boldsymbol{a}_i\}_{i=1,...,I}$, so that the centers of these clusters can be used as a grid for projections; see JMM (2010a) for a description of clustering methods and illustrative examples. If precomputation is used, construct the intratemporal-choice functions of type (8) as described in Sections 4.3 and 4.4.

- *Step* 1. On an iteration s, fix a vector of coefficients $\boldsymbol{v}(s)$. For each country j, use the assumed capital decision rule to calculate the next-period capital stock in all grid points, $\left(k_i^{j}\right)' \equiv \boldsymbol{X}_i \boldsymbol{v}^{j}$ for $i = 1, ..., I$.

- *Step* 2. Given $\{\boldsymbol{k}_i, \boldsymbol{a}_i, \boldsymbol{k}_i'\}_{i=1,...,I}$, calculate $\{\boldsymbol{c}_i, \boldsymbol{\ell}_i\}_{i=1,...,I}$ using a vectorized version of iteration-on-allocation, precomputation or the combination of the two methods as described in Section 4.

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

$$\left(\widehat{k}_i^j\right)' \equiv E\left[G^j\left(\boldsymbol{k}_i, \boldsymbol{a}_i, \boldsymbol{\varepsilon}\right)\right], \tag{25}$$

where E is computed with respect to $\boldsymbol{\varepsilon} \equiv \left(\varepsilon^1, ..., \varepsilon^N\right)$. To calculate the next-period intratemporal choice $\{\boldsymbol{c}_i', \boldsymbol{\ell}_i'\}_{i=1,...,I}$, 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. Run a linear regression of the constructed variable $\left(\widehat{k}_i^j\right)'$ on a set of the explanatory variables $\boldsymbol{X}_i$ for $j = 1, ..., N$, using the numerically stable approximation methods described in JMM (2009, 2010b),

$$\left(\widehat{k}_i^j\right)' = \boldsymbol{X}_i \boldsymbol{v}^j + \epsilon_i^j, \tag{26}$$

where ϵ_i^j is an i-grid approximation error corresponding to country j. Let the resulting vector of coefficients be called $\widehat{\boldsymbol{v}}\left(s\right)$.

- *Step* 5. Compute the coefficients for the subsequent $s+1$ iteration using fixed-point iteration:

$$\boldsymbol{v}\left(s+1\right) = \left(1-\xi\right)\boldsymbol{v}\left(s\right) + \xi\widehat{\boldsymbol{v}}\left(s\right), \tag{27}$$

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

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

$$\frac{1}{I \cdot N}\sum_{i=1}^{I}\sum_{j=1}^{N}\left|\frac{\left(k_i^j\right)' - \left(\widehat{k}_i^j\right)'}{\left(k_i^j\right)'}\right| < 10^{-\vartheta}, \tag{28}$$

where $\left(k_i^j\right)'$ and $\left(\widehat{k}_i^j\right)'$ are the next-period capital stocks on the grid before and after an iteration, respectively.

After achieving convergence, it is best to re-run the CGA using the obtained decision rule for capital as an initial guess for simulation. Doing so controls for the possibility that the initial guess for the capital decision rule was imprecise, and the simulated series (and consequently, our cluster grid) thus did not adequately represent the true ergodic set. Re-running the CGA a few additional times and updating the cluster gird as necessary can help correct for any errors created by a poor initial guess.

6 Methodology

In Section 6.1, we describe the baseline implementation of the SSA and CGA that is used to generate the results presented in the comparison paper by KMMP (2010). 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 (2010).

6.1 Baseline implementation of the SSA and CGA

Below, we describe the methodological 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 The 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 $j = 1, ...N$. Since the steady state levels of capital and productivity are normalized to one, this guess matches the steady state level of capital. In terms of the vector of coefficients $\boldsymbol{v}^{j}$, this guess implies that $v_{j}^{j} = 0.9$, $v_{N+j}^{j} = 0.1$, $j = 1, ...N$, 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 $j = 1, ..., N$. 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 JMM (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 $\xi = 0.03$ for the remaining models. We target seven digits of accuracy in the simulated data by fixing $\vartheta = 7$ in the 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 The CGA computes solutions using a second-degree ordinary polynomial. To start the iterative process, we use the first-degree polynomial solution computed by the SSA as an initial guess. The SSA solution was used both to compute an initial matrix of coefficients $\boldsymbol{v}$ and to construct 500 clusters. The clusters were constructed by applying the hierarchical clustering algorithm with Ward's linkage to the principal components of the simulated data (see JMM, 2010a, for a description of the clustering methods and illustrative examples). The outcome of the clustering process for a model with two state variables is illustrated in Figure 1d .

To estimate the regression equation (26), we again use a least-squares truncated QR factorization method. We set the damping parameter in (27) at $\xi = 0.1$ for Models 1 and 5, and $\xi = 0.05$ for all other models. We use $\vartheta = 7$ for the convergence criterion (28). We solve the model twice: first by computing the solution using a low-cost non-product monomial rule $M1$ with $2N$ nodes, and then again using a more costly product Gauss-Hermite rule $Q(2)$ with two nodes in each dimension and 2^N total nodes (see JMM (2010a) for a description of these integration methods).

Iteration-on-allocation In the baseline versions of both the 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 the CGA in which we use $\varsigma = 0.05$. To start iterations under the SSA, we assume that consumption and labor are equal to their steady-state values. Under the CGA, we compute an initial guess for consumption and labor using a time series solution produced by the SSA.

We would like to direct particular attention to an important aspect of the implementation of iteration-on-allocation. Finding consumption and labor allocations with a high degree of accuracy during each iteration requires a high computational cost and is of minimal use, since on the next iteration, we would re-compute consumption and labor allocations for a different vector of coefficients $\boldsymbol{v}$. We thus do not target any accuracy criteria in consumption and labor allocation for 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 the CGA in which we perform 3 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 decision function for capital (characterized by $\boldsymbol{v}$) is 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 upper and lower bounds (equal to 150% and 50% of the steady state level, respectively) on consumption in Model 5 and on labor in Models 6, 7 and 8. This trick is similar to the moving bounds used to restrict simulated series for capital under the 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. Finally, in the convergence criterion for consumption and labor allocations (11), we use $\theta = 7$.

Solution-output delivered to the testing bench of Juillard and Villemot (2010) When the iteration-on-iteration approach is used, the SSA and CGA do not deliver explicit decision rules for consumption and labor. The only solution-output they produce is a matrix of the polynomial coefficients for the capital decision rules (laws of motion) of N heterogeneous countries, $\boldsymbol{v} = \left(\boldsymbol{v}^1, ..., \boldsymbol{v}^j, ..., \boldsymbol{v}^N\right)$. Thus, in addition to the polynomial coefficients $\boldsymbol{v}$, we supply to the the testing bench of Juillard and Villemot (2010) four iteration-on-allocation routines (one per each asymmetric model and its symmetric counterpart) that allow to find the intratemporal choice in simulation.[12]

The simulation of our solutions includes two steps: First, the capital laws of motion, $k_{t+1}^j = \Psi^j\left(\boldsymbol{k}_t, \boldsymbol{a}_t; \boldsymbol{v}^j\right)$, $j = 1, ...N$ are used to generate the capital path $\{\boldsymbol{k}_{t+1}\}_{t=0,...,T}$. Then, given $\{\boldsymbol{k}_t, \boldsymbol{a}_t, \boldsymbol{k}_{t+1}\}_{t=0,...,T}$, the corresponding intratemporal choice $\{\boldsymbol{c}_t, \boldsymbol{\ell}_t\}_{t=0,...,T}$ is calculated 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 $\theta = 10$.

[12] Using the iteration-on-allocation routines in simulation plays a key role in the overall accuracy of the SSA and CGA because it allows us to solve for the intratemporal choice with essentially zero errors; see Table 4.3 in the comparison by KKMP (2010). This would not be possible if we constructed and supplied the standard explicit consumption and/or labor functions in terms of the state variables.

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 @2.67GHz, RAM 6,00GB and Windows Vista 64 bits. For each model studied, we report the running time in seconds: for the SSA, the running time is defined as the time needed to compute a first-order solution starting from a given initial guess, and for the CGA, the running time is defined as the time to compute a second-order solution starting from an initial guess consisting of a first-order SSA solution. Accuracy tests are performed using the testing bench of Juillard and Villemot (2010), and the results of these tests are described in KMMP (2010).

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 the SSA and CGA to several alternative implementations that have been 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 a simulation-based version of the parameterized expectation algorithm (PEA) by Den Haan and Marcet (1990). Under the PEA, decision rules 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. Moreover, non-linear regression methods require a good initial guess and involve costly computations of the Jacobian and Heissian matrices. Finally, such methods cannot be easily vectorized to estimate the decision rules of all countries simultaneously, which is critical to ensure adequate calculation speed in multi-country settings; see JMM (2010b) for an extensive discussion of this problem. In the present paper, we rely on the numerically stable stochastic simulation approaches described in JMM (2009): we use a linear regression model, normalize the data and employ a least-squares truncated QR factorization method that is suited for use with ill-conditioned problems. This approximation method (implemented in Matlab with backslash operator) use the original data and 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 comparison the first-degree polynomial approximation because it was more accurate than the second-degree polynomial approximation. This somewhat surprising results is explained in JMM (2010b): The accuracy of the Monte Carlo type of integration employed by the SSA depends on how long the simulation length T is relative to the number of polynomial coefficients in $\boldsymbol{v}$. The higher is the polynomial degree and/or dimensionality of the problem N, the larger is the number of the coefficients in $\boldsymbol{v}$, and the longer the simulation length is needed to appropriately identify of the coefficients. In a model similar to Model 1 of the current JEDC project, JMM (2010b) find that T should be increased from $10,000$ to $50,000$ to make the second-degree polynomial approximation more accurate than the first-degree polynomial approximation for the model with up to $N = 4$; if N is increased to 6, T should be increased to $100,000$. Since running a very lengthy simulation is costly both in terms of time and memory, in the present paper, we use $T = 10,000$ for the analysis in the present paper; this is sufficient to accurately identify the coefficients of the first-degree polynomial. As follows from the comparison in KMMP (2010), even the linear solutions delivered by the SSA are sufficiently accurate. This is because the SSA fits a polynomial exclusively in the relevant area of the state space (the ergodic set) and also because the SSA solves accurately for the intratemporal choice using the iteration-on-allocation method.

Cluster-grid algorithm In the case of the CGA, we submitted for comparison the second-degree polynomial approximation. The 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, JMM (2010a) find that an increase in the polynomial degree used in the CGA increases accuracy roughly by an order of magnitude in the examples considered. In Section 7, we compare the accuracy of the CGA in the context of the current JEDC project using the first-, second- and third-degree ordinary polynomials as well as under alternative Smolyak polynomials.

In addition to comparing the use of different polynomial degrees, we also test the sensitivity the CGA solutions are to the way in which the cluster grid is constructed. To do so, we first initialized the CGA using a linear solution delivered by a log-linearization method instead of one delivered by the SSA.[13] We then constructed clusters using an alternative K-means clustering algorithm with different linkages instead of the baseline hierarchical algorithm with Ward's linkage. These modifications do not visibly affect the accuracy and speed of the CGA. As far as the number of clusters is concerned, JMM (2010a) find that oversampling (when there are more grid points than the polynomial coefficients) increases the accuracy and numerical stability of the CGA compared to collocation (when the number of grid points is identical to the number of polynomial coefficients). In line with this finding, we chose to oversample and use 500 clusters to identify between 15 to 231 polynomial coefficients in models with N ranging from 2 to 10, respectively.

To implement numerical integration, we tried to choose the most accurate integration strategy feasible for problems of given dimensionality, $N \leq 10$. To this purpose, we designed a two-step integration procedure that combined a low-cost monomial rule with $2N$ nodes (step one) and a costly monomial (quadrature) rule with 2^N nodes (step two). Our results indicate that in the studied models, any accuracy gains from the above integration procedure are small relative to the gains obtained by using only less costly integration alternatives. We investigate the relationship between the specific integration method used and the accuracy and cost of the CGA in Section 7.

Intratemporal-choice approaches In addition to our baseline iteration-on-allocation procedure, we explored several alternative approaches to solving for the intratemporal allocations. For all eight models studied, we computed the consumption and/or labor functions in terms of the 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. Furthermore, 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 comparison results presented in KMMP (2010), we opt for the most accurate method, which is iteration-on-allocation. Our precomputation approach is however faster than iteration-on-allocation and was

[13]In JMM (2010a), the CGA is initialized using the CGA itself: A solution was first computed on an arbitrary grid of points, then used to simulate the model and to construct the clusters.

adopted by Pichler (2010) 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 the SSA and CGA under the baseline implementation is assessed in KMMP (2010). In this section, we provide additional numerical results for the CGA only, which demonstrate the dependence of its accuracy and speed on the specific intratemporal-choice approach, approximating polynomial function and integration method used. To assess the accuracy of solutions, we implement a test (described in Juillard and Villemot, 2010) that computes the average and maximum solution errors along a stochastic simulation of $10,000$ observations.

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 tendencies under the first-order polynomial approximation are similar.

In the solution procedure, we consider four alternative intratemporal-choice approaches: (i) parameterize consumption of both countries 1 and 2 with a polynomial of the state variables and compute the polynomials coefficients inside the main iterative cycle; (ii) parameterize and compute only consumption function of country 1 inside the main iterative cycle and find consumption of country 2 from closed-form expression (12); (iii) precompute the consumption function 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 solution to construct consumption functions for both countries 1 and 2 in terms of the state variables (if not constructed by the solution method used); (b) use the solution to construct consumption 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 function precomputed by the solution 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 the precomputation approach given by approach (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), $p = 1, ..., 300$. 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. In addition, we tried other interpolation schemes such as a piecewise cubic polynomial interpolation, splines, etc., and found that piecewise low-order polynomial interpolation schemes lead to more accurate solutions (though at a higher cost) than high-order global 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 Table 1 indicates, for both the solution and simulation procedures, approximating a consumption function using a second-degree polynomial of state variables results in accuracy is a low degree of accuracy (namely, the errors in the intratemporal-choice conditions including the budget constraint are large). If we solve for consumption very accurately in the solution procedure (using precomputation and iteration-on-allocation), then solve less accurately for consumption in simulation using second-degree polynomials of state variables, the result is also a low degree of accuracy . Finally, if we solve less accurately for consumption in the solution procedure using second-degree polynomials of state variables, then solve accurately for consumption in simulation using precomputation and iteration-on-allocation, high levels of accuracy are obtained. These results lead us to conclude that solution accuracy is less dependent on the computational method used to determine the intratemporal choice in the solution procedure than in it is on the computational method used to solve for the intratemporal choice in simulation.

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

Finally, Table 1 also shows $TCPU$, the amount of time needed to run each test on a stochastic simulation of $10,000$ observations. When the precomputation approach is used, $TCPU$ is only slightly larger than it is when the standard approach constructing the intratemporal-choice functions in terms of state variables is used; when the iteration-on-allocation approach is used, $TCPU$ is almost 20 times larger.The iteration-on-allocation approach performs slowly in the test because our testing procedure is not 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 in a period-by-period fashion.

7.2 Costs 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 explicitly-defined intratemporal decision functions represented by second-degree polynomials of state variables (CPU1) and the other in which the intratemporal choice is computed using the iteration-on-allocation solver (CPU2). As an initial guess for allocations in the latter procedure, we use the series 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 explicit decision rules. 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, then the standard explicit decision rules. The latter 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 the 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 Smolyak polynomials used in MKK (2010). 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.

The results shown in Table 3 demonstrate that 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 integration methods and thus allows the computation of models with much higher dimensions. In particular, JMM (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 $N = 40$ countries, respectively.

Furthermore, the results shown in Table 3 demonstrate that when solutions are computed using ordinary polynomials, increasing the polynomial degree from one to two raises accuracy by more than an order of magnitude; increasing the polynomial degree from two to three further increases accuracy by slightly less than an order of magnitude. However, increasing the degree of a complete polynomial in high-dimensional problems carries substantial costs. As Table 3 shows, the Smolyak polynomial is a useful alternative for the CGA: It leads to as nearly as accurate a solution as the third-degree complete polynomial, but its number of terms grows quadratically instead of cubically with dimension (independently of dimension, the Smolyak polynomial has only four times more terms than the complete second-degree polynomial; see MKK, 2010 for details).

Table 4 shows the results of our investigation of the relationship between the specific integration method used and the accuracy of the CGA. To obtain these results, we recompute the solutions to Models 5-8 under four alternative integration methods: $Q(1)$, $Q(2)$, $M1$ and $M2$. The accuracy results in our Table 4 are analogous to those reported in Table 5 of KMMP (2010); however, our testing procedure uses random draws, which are different from those used by Juillard and Villemot (2010). As Table 4 shows, the errors we found are very close to those shown in Table 5 of KMMP (2010). Furthermore, all of the integration methods considered continue to lead to solutions with similar levels of accuracy, with the exception of the $Q(1)$ rule (which produces slightly less accurate solutions).

The key finding of the results shown in Table 4 relates to the issue of computational cost. Specifically, we see that the CGA can compute solutions of the same accuracy as those submitted for comparison of KMMP (2010), but at a much lower cost. For example, we reduce the computational time for Model 5 with $N = 10$ countries from about 35 hours (reported in Table 3 of KMMP, 2010) to 7 minutes (reported in our Table 4) without a visible loss in accuracy by replacing our costly, two-step baseline integration procedure with just its first step, which is based on the M1 monomial rule with 2N nodes.[14] We can further reduce the time needed to solve a ten-country version of Model 5 to about 2 minutes using the $Q(1)$ rule, at the cost of a modest loss in accuracy. In Models 6-8, the low-cost integration rules reduce the computational time by roughly the same proportion as in Model 5. However, Models 6-8 are generally more costly to run than Model 5 due to the higher costs associated with solving for the intratemporal choice. The computational time for these models may be reduced by combining the iteration-on-allocation and precomputation approaches, as described in Section 4.3. In addition, we can decrease the computational time for all models by reducing the number of clusters; see JMM (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 even when these methods' own intratemporal choices are not sufficiently accurate. A prominent example of such a method is perturbation, which produces small errors in the Euler equations but large errors in the intratemporal-choice conditions

[14] At the moment of submission of our solutions for comparison of KMMP (2010), we did not have a reliable accuracy test, and we submitted the solutions obtained under the most accurate integration procedure feasible for the CGA, which is a two-step combination of the M1 and Q(2) rules.

- especially in the budget constraint; see Table 6 of KMMP (2010) for the accuracy, by equation, for the first- and second-order perturbation methods by Kollmann, Kim and Kim (2010) (referred to as PER1 and PER2, respectively). Consequently, there are potential benefits to be derived from constructing a hybrid of the standard perturbation method (used as a low-cost method for computing capital decision rules), and accurate intratemporal-choice methods (used to solve for consumption and labor after capital is computed).

To verify the above conjecture, we present a hybrid method that takes the capital decision rules produced by the standard log-linearization method for two-country versions of Models 5-8 and solves for consumption and labor in simulation using an accurate iteration-on-allocation method. In Table 5, we compare the accuracy of this hybrid method with that of the SSA, CGA, PER1 and PER2 methods, as reported in Table 5 of KMMP (2010).

As Table 5 indicates, our hybrid method is far more accurate (by more than an order of magnitude) than the PER1. It is even more accurate than the PER2 and is only slightly less accurate than the SSA. On the other hand, the hybrid method is considerably less accurate than the CGA. When comparing the hybrid method against the CGA, however, we should take into account that the latter uses the 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 previously noted, the iteration-on-allocation solver has advantages over other solvers. Thus, it is one of the better candidates for a fusion with perturbation.

8 Conclusion

In this paper, we offer a mix of techniques that, when taken together, allows us to address the challenges of solving high-dimensional problems. First, the SSA and CGA operate on the ergodic-set domain which, in high-dimensional problems, is normally a tiny fraction of the standard hypercube domain used by other methods.

Second, it is critical to vectorize computations for speed. Concerning the capital decision functions, we combine linearly-additive polynomials, linear approximation methods and fixed-point iteration to solve for the polynomial coefficients of all countries at once rather than on a country-by-country basis. Concerning consumption and labor allocations, we separate the intertemporal- and intratemporal-choice problems, solve for the state variables and find the control variables in all grid points / integration nodes / time periods at once rather than on a one-by-one basis using a vectorized (fixed-point) iteration-on-allocation solver.

Third, proper selection and coordination of the integration, approximation and intratemporal-choice strategies is critical for accuracy, speed and numerical stability of our solution methods. We use integration rules that have low cost in high dimensional problems, namely, a Monte Carlo type of integration combined with regression under the SSA, and non-product monomial rules and the Gauss-Hermite rule with one node under the CGA. We implement the approximation step using efficient and numerically-stable linear regression approaches described in JMM (2009). We solve for the intratemporal choice using accurate iteration-on-allocation method (this is important for the overall accuracy of our solutions). We also show that other polynomial families (such as Smolyak polynomials studied in MKK, 2010) can help increase accuracy and speed of our solution methods relative to our baseline ordinary-polynomial family.

Finally, in addition to our main SSA and CGA methods, we construct a hybrid method that combines perturbation (used to compute the capital decision rules) and accurate intratemporal choice methods (used to solve for consumption and labor). 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 very high dimensionality.

References

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

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

[3] Den Haan, W., and A. Marcet, (1994). Accuracy in simulations. Review of Economic Studies 6, 3-17.

[4] Den Haan W., K. Judd and M. Juillard, (2010). Computational suite of models with heterogeneous agents: Milti-country real business cycle models. Journal of Economic Dynamics and Control, this issue.

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

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

[7] Judd, K., (1998). Numerical methods in economics. London, England: The MIT Press, Cambridge Massachusetts.

[8] Judd, K., L. Maliar and S. Maliar, (2009). Numerically stable stochastic simulation approaches for solving dynamic economic models, NBER working paper 15296.

[9] Judd, K., L. Maliar and S. Maliar, (2010a). A cluster-grid projection method: solving problems with high dimensionality, NBER working paper 15965.

[10] Judd, K., L. Maliar and S. Maliar, (2010b). Numerically stable stochastic simulation approaches for solving dynamic economic models. Manuscript.

[11] Juillard, M. and S. Villemot, (2010). Multi-country real business cycle models: accuracy tests and testing bench. Journal of Economic Dynamics and Control, this issue.

[12] Kollmann, R., S. Kim, and J. Kim, (2010). Solving the multi-country real business cycle model using a perturbation method. Journal of Economic Dynamics and Control, this issue.

[13] Maliar, L. and S. Maliar, (2001). Heterogeneity in capital and skills in a neoclassical stochastic growth model. Journal of Economic Dynamics and Control 25, 1367-1397.

[14] Maliar, L. and S. Maliar, (2003a). The representative consumer in the neoclassical growth model with idiosyncratic shocks. Review of Economic Dynamics 6, 362-380.

[15] Maliar, L. and S. Maliar, (2003b). Parameterized expectations algorithm and the moving bounds. Journal of Business and Economic Statistics 21, 88-92.

[16] Maliar, L. and S. Maliar, (2004). Comparing numerical solutions of models with heterogeneous agents (Model A): a simulation-based parameterized expectations algorithm. Manuscript.

[17] Maliar, L. and S. Maliar, (2005). Parameterized expectations algorithm: how to solve for labor easily. Computational Economics 25, 269-274.

[18] Maliar, L. and S. Maliar, (2007). Comparing numerical solutions of models with heterogeneous agents (Model A): a simulation-based parameterized expectations algorithm. Manuscript.

[19] Malin, B., D. Krueger, and F. Kubler, (2010). Solving the multi-country real business cycle model using a Smolyak-collocation method. Journal of Economic Dynamics and Control, this issue.

[20] Marcet, A., (1988). Solution of nonlinear models by parameterizing expectations. Carnegie Mellon University. Manuscript.

[21] Pichler, P., (2010). Solving the multi-country real business cycle model using a monomial rule Galerkin method. Journal of Economic Dynamics and Control, this issue.

[22] Smith, A., (1991). Solving stochastic dynamic programming problems using rules of thumb. Queen's University. Economics Department. Discussion Paper 816.

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

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

9 Appendix

In this section, we present some supplementary results.

9.1 Appendix A

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

Model 6 Combining FOCs (4) and (5) gives

$$\widetilde{\ell}_t^j = \left[\frac{a_t^j \left(k_t^j\right)^\alpha \tau^1 b^1}{a_t^1 \left(k_t^1\right)^\alpha \tau^j b^j}\right]^{\frac{\eta^j}{1+\alpha\eta^j}} \left(\ell_t^1\right)^{\frac{\eta^j\left(1+\alpha\eta^1\right)}{\eta^1\left(1+\alpha\eta^j\right)}} \qquad j=2,...,N. \tag{29}$$

FOC (5) and budget constraint (2), respectively, can be written as

$$\widetilde{c}_t^j = \left[\frac{(1-\alpha)\, a_t^j A \left(k_t^j\right)^\alpha \left(l_t^j\right)^{-\alpha}}{b^j \left(l_t^j\right)^{1/\eta^j}}\right]^{\gamma^j}, \quad \text{with} \left\{\begin{array}{l} l_t^j \equiv \ell_t^1,\ j=1 \\ l_t^j \equiv \widetilde{\ell}_t^j,\ j=2,...,N \end{array}\right., \tag{30}$$

$$\widetilde{\ell}_t^1 = \left[\frac{\sum_{j=1}^N \left[\widetilde{c}_t^j + k_{t+1}^j - k_t^j + \frac{\phi}{2} k_t^j \left(\frac{k_{t+1}^j}{k_t^j} - 1\right)^2\right] - \sum_{j=2}^N a_t^j A \left(k_t^j\right)^\alpha \left(\widetilde{\ell}_t^j\right)^{1-\alpha}}{a_t^1 A \left(k_t^1\right)^\alpha}\right]^{\frac{1}{1-\alpha}}, \tag{31}$$

where $\{\gamma^j, \eta^j, b^j\}^{j=1,...,N}$ are the utility-function parameters, and α is the share of capital in production. For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, equations (29) – (31) define a mapping $\widetilde{\ell}_t^1 = \Gamma\left(\ell_t^1\right)$. We iterate on labor of the first country, $\widetilde{\ell}_t^1$, as follows: Assume some initial ℓ_t^1; compute $\left\{\widetilde{\ell}_t^j\right\}^{j=2,...,N}$ from (29); find $\left\{\widetilde{c}_t^j\right\}^{j=1,...,N}$ from (30); obtain $\widetilde{\ell}_t^1$ from (31); if $\ell_t^1 \neq \widetilde{\ell}_t^1$, compute the next-iteration input as $(1-\xi)\,\ell_t^1 + \xi\widetilde{\ell}_t^1$. Iterate until convergence.

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

$$\widetilde{c}_t^j = \frac{\psi\left(L^e - \ell_t^j\right)}{1-\psi} (1-\alpha)\, a_t^j A \left(k_t^j\right)^\alpha \left(\ell_t^j\right)^{-\alpha}, \tag{32}$$

$$\widetilde{\ell}_t^j = L^e - \left[\left(L^e - \ell_t^1\right)^{(1-\psi)\left(1-1/\gamma^1\right)} \frac{\left(\widetilde{c}_t^1\right)^{\psi\left(1-1/\gamma^1\right)-1}\tau^1}{\left(\widetilde{c}_t^j\right)^{\psi\left(1-1/\gamma^j\right)-1}\tau^j} \right]^{\frac{1}{(1-\psi)\left(1-1/\gamma^j\right)}}, \quad j = 2, ..., N, \tag{33}$$

where $\{\gamma^j\}^{j=1,\ldots,N}$ and ψ are the utility-function parameters, α is the share of capital in production, and L^e are the labor endowment of the representative agent. The resource constraint is given by (31) and determines $\widetilde{\ell}_t^1$. For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, equations (31), (32) and (33) define a mapping $\left\{\widetilde{\ell}_t^j\right\}^{j=1,\ldots,N} = \Gamma\left(\left\{\ell_t^j\right\}^{j=1,\ldots,N}\right)$. We iterate on labor of all countries, $\left\{\widetilde{\ell}_t^j\right\}^{j=1,\ldots,N}$, as follows: Assume some initial $\left\{\ell_t^j\right\}^{j=1,\ldots,N}$, find $\left\{\widetilde{c}_t^j\right\}^{j=1,\ldots,N}$ from (32); compute $\left\{\widetilde{\ell}_t^j\right\}^{j=2,\ldots,N}$ and $\widetilde{\ell}_t^1$ from (33) and (31), respectively; if $\ell_t^j \neq \widetilde{\ell}_t^j$ for $j = 1, ..., N$, calculate the next-iteration input as $(1-\xi)\,\ell_t^j + \xi\widetilde{\ell}_t^j$. Iterate until convergence.

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

$$\widetilde{c}_t^j = \left[\frac{(1-\alpha)\,a_t^j A\left(\ell_t^j\right)^{\mu^j-1}\left(\alpha\left(k_t^j\right)^{\mu^j} + (1-\alpha)\left(\ell_t^j\right)^{\mu^j}\right)^{1/\mu^j-1}}{b^j}\right]^{\chi^j}\left(L^e - \ell_t^j\right), \tag{34}$$

$$\widetilde{\ell}_t^j = L^e - \left[\frac{1}{b^j}\left(\frac{u_{c,t}^1\tau^1}{\left(\widetilde{c}_t^j\right)^{-1/\chi^j}\tau^j}\right)^{\frac{1-1/\chi^j}{1/\chi^j-1/\gamma^j}} - \frac{\left(\widetilde{c}_t^j\right)^{1-1/\chi^j}}{b^j}\right]^{\frac{1}{1-1/\chi^j}}, \quad j = 2, ..., N, \tag{35}$$

$$\widetilde{\ell}_t^1 = \left[\left(\frac{f_t^1}{a_t^1 A\,(1-\alpha)^{1/\mu^1}}\right)^{\mu^j} - \frac{\alpha\left(k_t^1\right)^{\mu^1}}{1-\alpha}\right]^{\frac{1}{\mu^1}}, \tag{36}$$

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

$$u_{c,t}^1 \equiv \left[\left(\widetilde{c}_t^1\right)^{1-1/\chi^1} + b^1\left(L^e - \ell_t^1\right)^{1-1/\chi^1}\right]^{\frac{1/\chi^1-1/\gamma^1}{1-1/\chi^1}}\left(\widetilde{c}_t^1\right)^{-1/\chi^1},$$

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

For given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, equations (34) – (36) define a mapping $\left\{\widetilde{\ell}_t^j\right\}^{j=1,...,N} = \Gamma\left(\left\{\ell_t^j\right\}^{j=1,...,N}\right)$. We iterate on labor of all countries, $\left\{\widetilde{\ell}_t^j\right\}^{j=1,...,N}$, as follows: Assume some initial $\left\{\ell_t^j\right\}^{j=1,...,N}$; find $\left\{\widetilde{c}_t^j\right\}^{j=1,...,N}$ from (34); compute $\left\{\widetilde{\ell}_t^j\right\}^{j=2,...,N}$ and $\widetilde{\ell}_t^1$ using (35) and (36), respectively; if $\ell_t^j \neq \widetilde{\ell}_t^j$ for $j = 1, ..., N$, calculate the next-iteration input as $(1-\xi)\,\ell_t^j + \xi\widetilde{\ell}_t^j$. Iterate until convergence.

9.2 Appendix B

This section describes how to use the precomputation approach for Models 2. Since all agents are identical in preferences and have identical welfare weights, $\tau^j = 1$ for $j = 1, ..., N$, 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 get

$$
\ell_t^j = \left[\frac{c_t^{1/\gamma} N^{-1/\gamma} b}{(1-\alpha)\, A a_t^j \left(k_t^j\right)^{\alpha}}\right]^{-\frac{\eta}{1+\alpha\eta}}. \tag{37}
$$

Substituting (37) into budget constraint (2), we obtain

$$
c_t = c_t^{-\frac{\eta(1-\alpha)}{\gamma(1+\alpha\eta)}} q_t + d_t, \tag{38}
$$

with variables q_t and d_t being defined as

$$
q_t = \frac{\sum_{j=1}^{N}\left[A a_t^j \left(k_t^j\right)^{\alpha}\right]^{1+\frac{\eta(1-\alpha)}{1+\alpha\eta}}}{\left[\frac{N^{-1/\gamma} b}{(1-\alpha)}\right]^{\frac{\eta(1-\alpha)}{1+\alpha\eta}}}, \qquad d_t = -\frac{\phi}{2}k_t^j\left(\frac{k_{t+1}^j}{k_t^j}-1\right)^2 + k_t^j - k_{t+1}^j. \tag{39}
$$

We can use equation (38) to precompute consumption c_t in terms of two variables q_t and d_t.

Outside the main iterative cycle, take a grid of P values $\{q_P, d_P\}_{p=1,...,P}$ for q_t and d_t. For each grid point $p = 1, ..., P$, use a numerical solver to find a solution c_p to equation (38) represented in a form suited for precomputation. Interpolate the constructed set function to continuous domain to get $\widehat{c}\,(q, d)$. Inside the main iterative cycle, for each t, given $\boldsymbol{k}_t$, $\boldsymbol{a}_t$, $\boldsymbol{k}_{t+1}$, compute q_t and d_t from (39), use the precomputed function to find aggregate consumption, $c_t = \widehat{c}_t\,(q_t, d_t)$, and compute individual labor ℓ_t^j from (37) for $j = 1, ..., N$.

Note that if welfare weights were not assumed identical but were determined endogenously from a lifetime budget constraint, we could still express the individual intratemporal choice in terms of aggregate variables (even though individual and average consumption are not equal any more). This result is shown by Maliar and Maliar (2001, 2003b) who use non-Gorman aggregation to solve large-scale macroeconomic models.

9.3 Appendix C

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

Model 5

$$k_{t+1}^j = E_t\left\{\beta\frac{\left(c_{t+1}^j\right)^{-1/\gamma^j}}{\left(c_t^j\right)^{-1/\gamma^j}\omega_t^j}\left[\theta_{t+1}^j + \alpha a_{t+1}^j A\left(k_{t+1}^j\right)^{\alpha-1}\right]k_{t+1}^j\right\}. \tag{40}$$

Model 6

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

Model 7

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

Model 8

$$k_{t+1}^{j}=E_{t}\left\{\beta\frac{u_{c,t+1}^{j}}{u_{c,t}^{j}\omega_{t}^{j}}\left[\theta_{t+1}^{j}+\alpha a_{t+1}^{j}A\left(k_{t+1}^{j}\right)^{\mu^{j}-1}\left(\alpha\left(k_{t+1}^{j}\right)^{\mu^{j}}+\alpha\left(\ell_{t+1}^{j}\right)^{\mu^{j}}\right)^{1/\mu^{j}-1}\right]k_{t+1}^{j}\right\}, \tag{43}$$

where $u_{c,t}^{j}$ is defined as

$$u_{c,t}^{j}\equiv\left[\left(c_{t}^{j}\right)^{1-1/\chi^{j}}+b^{j}\left(L^{e}-\ell_{t}^{j}\right)^{1-1/\chi^{j}}\right]^{\frac{1/\chi^{j}-1/\gamma^{j}}{1-1/\chi^{j}}}\left(c_{t}^{j}\right)^{-1/\chi^{j}}.$$

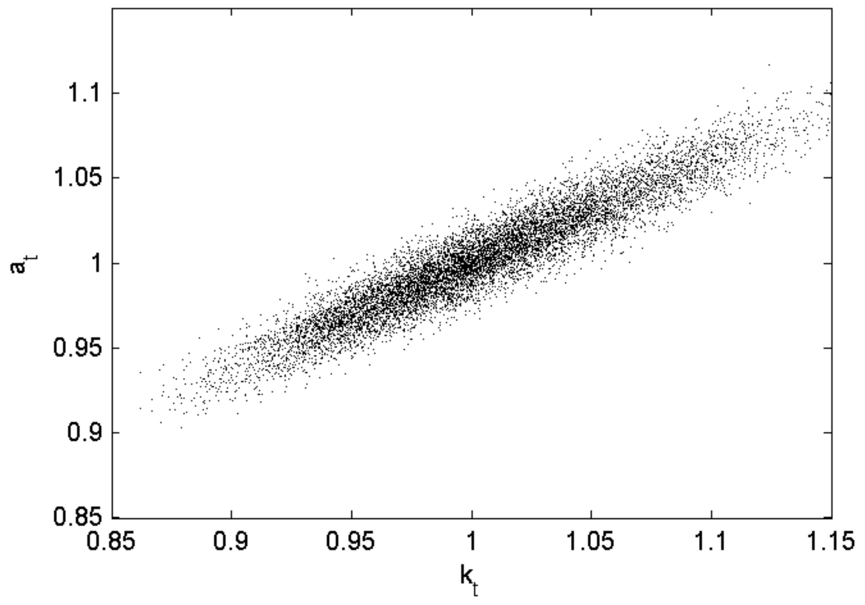

Figure 1a. Ergodic distribution.

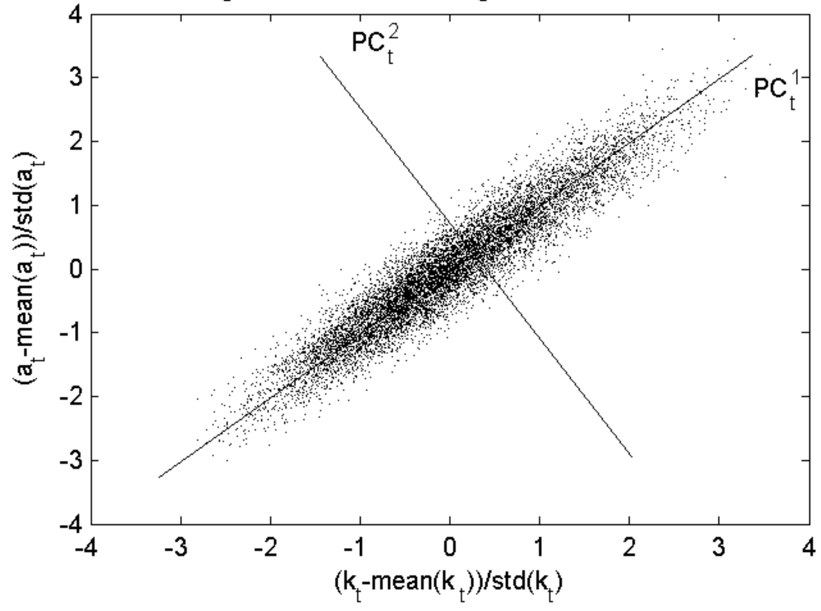

Figure 1b. Normalized ergodic distribution.

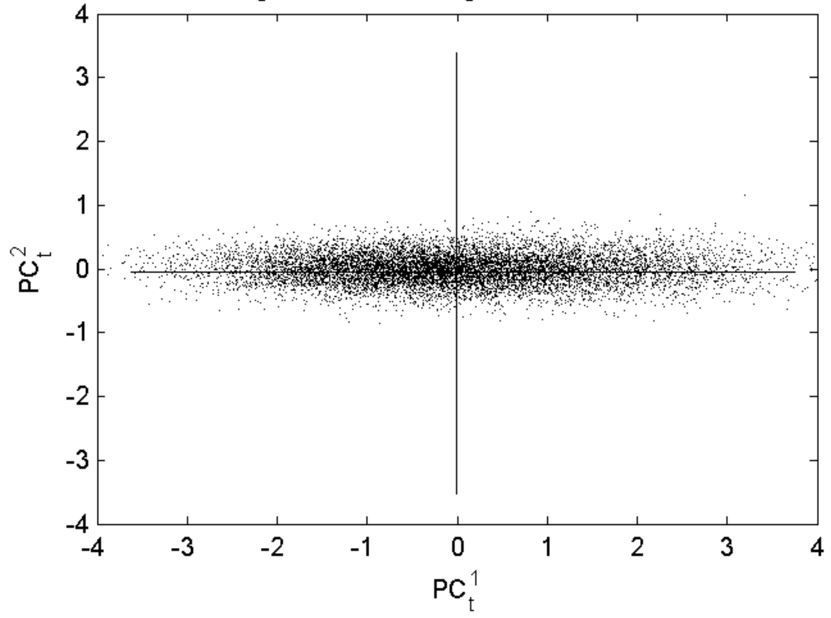

Figure 1c. PCs of ergodic distribution.

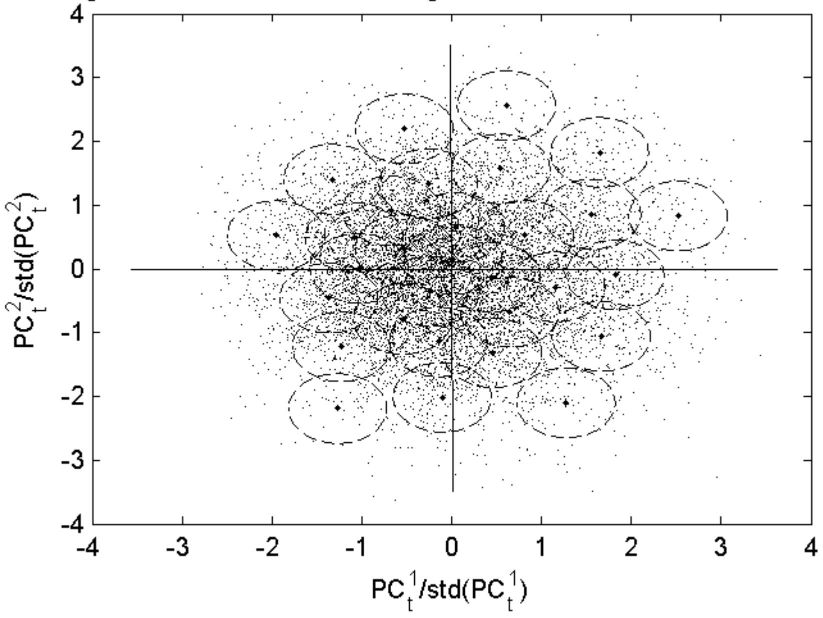

Figure 1d. Normalized PCs of ergodic distribution and 30 clusters.

Figure 2a. Iteration-on-allocation: convergence without damping.

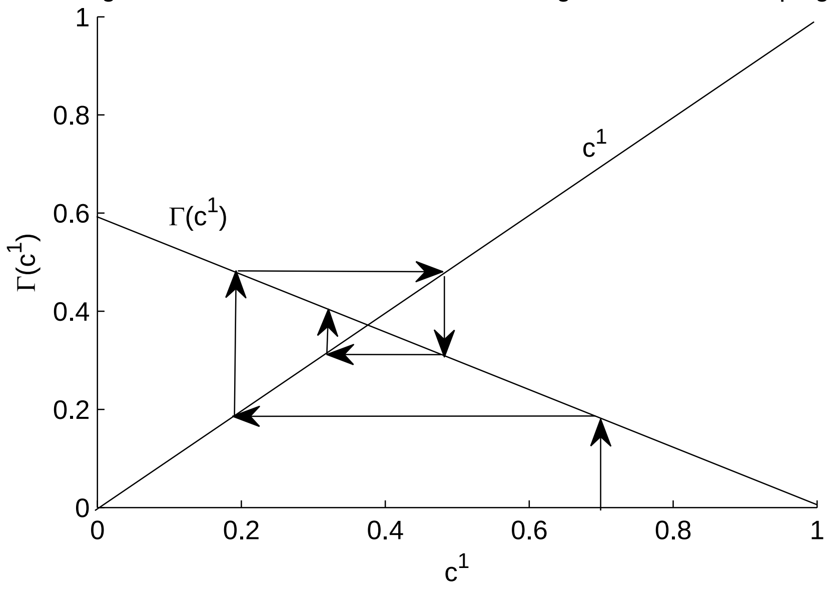

Figure 2b. Iteration-on-allocation: divergence without damping.

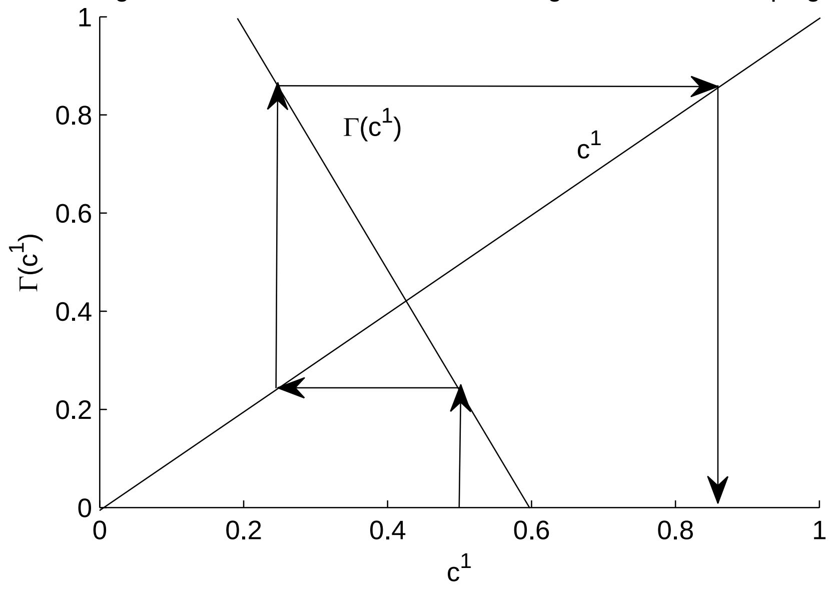

Figure 2c. Iteration-on-allocation: convergence with damping.

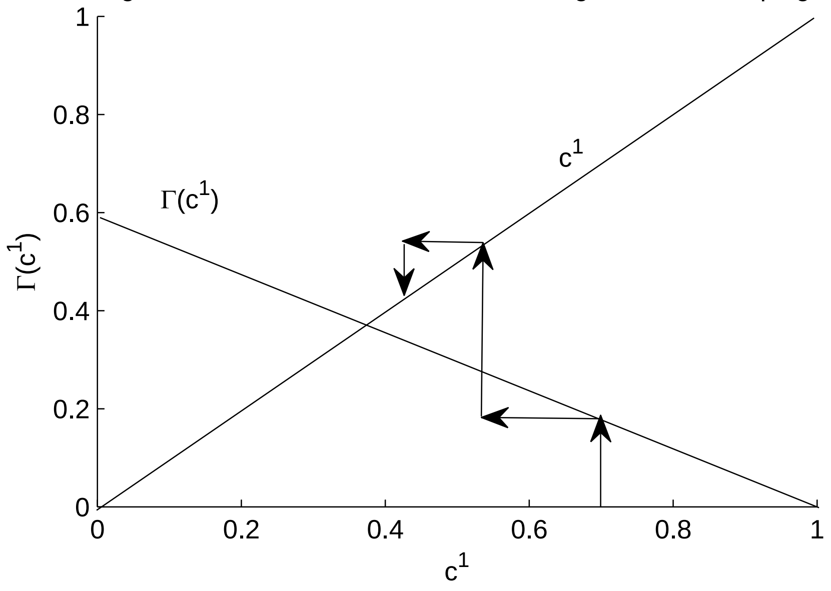

Figure 2d. Iteration-on-allocation: convergence with damping.

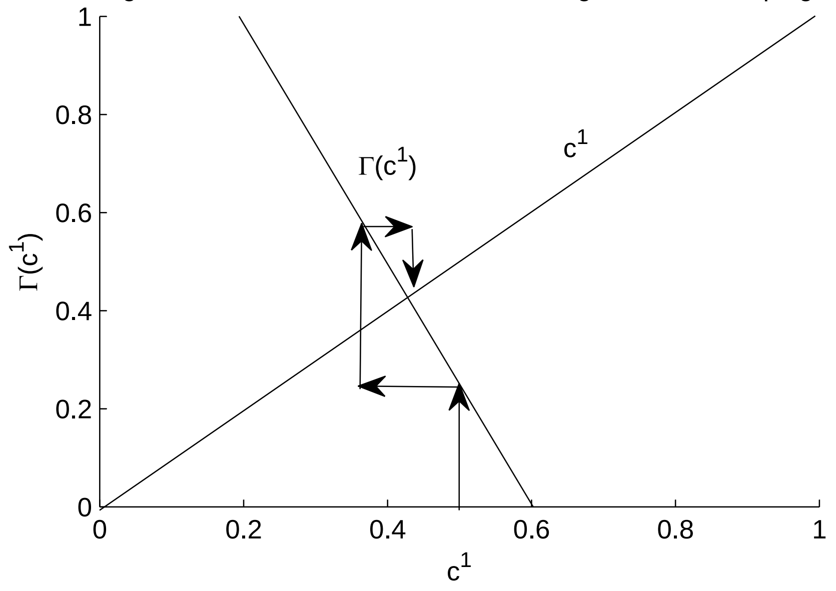

Table 1. Accuracy and test time under alternative intratemporal-choice approaches for Model 5.

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

Remark: $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum errors in the corresponding optimality conditions (in log10 units) in test on stochastic simulation of 10,000 observations; and TCPU is running time of test (in seconds). Abbreviations "Euler", "BC" and "Intrat" denote the average errors in the Euler equations, budget constraint and the intratemporal optimality conditions, respectively. See Juillard and Villemot (2010) for a definition of errors.

Table 2. Time for simulating Models 5-8 under two alternative intratemporal-choice approaches: approximating consumption with a function of the state variables and 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

Remark: *CPU1* and *CPU2* are, respectively, time for simulating model *T* periods forward (in seconds) approximating consumption with a function of the state variables and using iteration-on-allocation.

Table 3. The effect of specific polynomial on accuracy of the CGA under five integration rules for Models 5-8 with *N=2* countries.

Polyn.	*Q(3)*		*Q(2)*		*M2*		*M1*		*Q(1)*	
	ε_{mean}	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}	ε_{max}	ε_{mean}	ε_{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

Remark: ε_{mean} and ε_{max} are, respectively, the average and maximum absolute errors across all optimality conditions (in log10 units) in test on stochastic simulation of 10,000 observations; and abbreviations 1st, 2nd, 3rd and "SMOL" denote the first-, second-, third-degree ordinary polynomials and Smolyak polynomials, respectively. See Juillard and Villemot (2010) for a definition of errors.

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

N	*Q(2)*			*M2*			*M1*			*Q(1)*		
	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	*CPU*
Model 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	144327	-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	38392	-6.31	-4.66	19635	-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	13414	-5.96	-4.22	14385	-5.96	-4.22	3869	-5.32	-4.09	1756

Remark: $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum absolute errors across all optimality conditions (in log10 units) in test on stochastic simulation of 10,000 observations; and CPU is time for computing solution (in seconds). See Juillard and Villemot (2010) for a definition of errors.

Table 5. Accuracy of the hybrid perturbation and other solution methods for Models 5-8 with *N=2* countries.

Model	SSA		CGA		PER1		PER2		Hybrid	
	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$	$\mathcal{E}_{mean}$	$\mathcal{E}_{max}$
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

Remark: $\mathcal{E}_{mean}$ and $\mathcal{E}_{max}$ are, respectively, the average and maximum absolute errors across all optimality conditions (in log10 units) in test on stochastic simulation of 10,000 observations; the results for the SSA, CGA, PER1 and PER2 are reproduced from KMMP (2010).