NBER WORKING PAPER SERIES

ONE-NODE QUADRATURE BEATS MONTE CARLO: A GENERALIZED STOCHASTIC SIMULATION ALGORITHM

Kenneth Judd
Lilia Maliar
Serguei Maliar

Working Paper 16708
http://www.nber.org/papers/w16708

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
January 2011

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

One-node Quadrature Beats Monte Carlo: A Generalized Stochastic Simulation Algorithm
Kenneth Judd, Lilia Maliar, and Serguei Maliar
NBER Working Paper No. 16708
January 2011
JEL No. C63

ABSTRACT

In conventional stochastic simulation algorithms, Monte Carlo integration and curve fitting are merged together and implemented by means of regression. We perform a decomposition of the solution error and show that regression does a good job in curve fitting but a poor job in integration, which leads to low accuracy of solutions. We propose a generalized notion of stochastic simulation approach in which integration and curve fitting are separated. We specifically allow for the use of deterministic (quadrature and monomial) integration methods which are more accurate than the conventional Monte Carlo method. We achieve accuracy of solutions that is orders of magnitude higher than that of the conventional stochastic simulation algorithms.

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

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

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

1 Introduction

Numerical methods for solving dynamic economic models differ substantially in their accuracy and speed. For example, in the comparison studies by Den Haan (2010), and Kollmann, Maliar, Malin and Pichler (2011) (henceforth, KMMP), these differences amount to several orders of magnitude. To develop efficient solution methods, we must understand what accounts for the differences across methods and identify computational techniques that lead to best results. This is not easy to do because existing solution methods differ in many dimensions (solution domain, interpolation method, integration method, iterative procedure, etc.) and contain steps that are not directly comparable.

In the present paper, we assess computational techniques that are part of the basis of two broad classes of solution methods, the stochastic simulation and projection.[1] The first class approximates solutions on a set of simulated points using Monte Carlo integration, while the second class approximates solutions on a fixed grid of points using deterministic numerical integration. Furthermore, in stochastic simulation methods, the integration and curve-fitting steps are merged into one by means of regression, whereas in projection methods, these steps are performed separately. Finally, stochastic simulation methods are simpler to implement and are relatively less expensive in high-dimensional problems than projection methods, but they are also less accurate; see Maliar, Maliar and Judd (2011) and KMMP (2011).

We introduce a notion of generalized stochastic simulation algorithm ($GSSA$) that breaks down the fusion of integration and curve-fitting that is present in the conventional stochastic simulation algorithm. We specifically allow for the use of deterministic (quadrature and monomial) integration methods which are characteristic for the projection class of algorithms. $GSSA$ effectively lies between a pure stochastic simulation and pure projection algorithms and includes both of these algorithms as limiting cases. By separating the integration and curve-fitting steps, we are able to decompose the total solution error into errors associated with integration, curve-fitting and the choice of a solution domain.

Within our generalized framework, we show that errors associated with

[1] Examples of stochastic simulation methods are Den Haan and Marcet (1990), Smith (1991), Maliar and Maliar (2005), Judd, Maliar and Maliar (2010b) (henceforth, JMM), and examples of projection methods are Judd (1992), Christiano and Fisher (2000), Krueger and Kubler (2004), and JMM (2010a).

curve-fitting decrease rapidly with the degree of the approximating polynomial function. Furthermore, errors associated with a stochastically generated solution domain decrease rapidly with the simulation length. Finally, the integration errors also decrease with the simulation length but slowly.[2] Under the conventional one-node Monte Carlo method, the integration errors are considerably larger than other kinds of errors and restrict the overall accuracy of solutions. After we substitute the conventional one-node Monte-Carlo integration method with a one-node Gauss-Hermite quadrature integration method, the integration errors decrease dramatically. As a result, $GSSA$ combines high accuracy of projection methods and low cost in high-dimensional problems of stochastic simulation methods.[3]

We first test the performance of $GSSA$ on the example of the standard representative-agent growth model. We find that for a wide range of parameters, replacing conventional one-node Monte Carlo integration with one-node Gauss-Hermite quadrature integration reduces the solution errors by about $2-3$ orders of magnitude, while replacing it with several-node quadrature integration reduces such errors by up to 5 orders of magnitude. For example, for Monte Carlo integration with with 1, 30 and 3000 nodes, the solution errors (measured by the size of Euler equation errors on a stochastic simulation) are at most $5.1 \cdot 10^{-4}$, $3.1 \cdot 10^{-4}$ and $1.2 \cdot 10^{-5}$, respectively, while for quadrature integration with 1, 2 and 10 nodes, the solution errors are at most $6.3 \cdot 10^{-7}$, $1.7 \cdot 10^{-9}$ and $1.6 \cdot 10^{-9}$, respectively. Surprisingly, the quadrature integration method with just one node leads to more accurate solutions than a Monte Carlo method with thousands of nodes.

We next study the performance of $GSSA$ in the context of a heterogeneous-agent growth model, namely, we consider a multi-country model with up to thirty heterogeneous countries. We find that for a second-degree polynomial, the one-node Gauss-Hermite quadrature integration method produces errors that are up to two orders of magnitude smaller than those produced by the conventional one-node Monte Carlo integration method. For example, after we substitute one-node Monte Carlo integration by one-node quadrature in-

[2] Galant and Nychka (1987) analyze a relation between the choice of an approximating polynomial function and the sample size in the context of semi-parametric maximum likelihood estimation.

[3] Stochastic simulation algorithms operate on the ergodic set realized in equilibrium. This allows to avoid costs associated with finding a solution in areas of state space that are never realized in equilibrium. The higher is the dimensionality of the problem, the larger is the gain from focusing on the ergodic set; see JMM (2010b) for a discussion.

tegration, the solution error in a thirty-country model goes down from $5 \cdot 10^{-3}$ to $2.5 \cdot 10^{-5}$.

We finally point out that the use of numerically stable approximation methods is crucial for the successful performance of $GSSA$. The problem of recovering policy functions from simulated data is often ill-conditioned, and standard least-squares methods (such as ordinary least-squares and Gauss-Newton methods) work only under low-degree polynomial approximations. JMM (2009, 2010b) describe a variety of approximation methods that can handle ill-conditioned problems in the context of stochastic simulation algorithms. The numerically stable approximation methods include the least-squares method based on SVD, Tikhonov regularization, least absolute deviation methods and principal components method. These methods help restore numerical stability under high-degree polynomial approximations and allow us to achieve high accuracy of solutions.

The rest of the paper is as follows: In Section 2, we describe the representative agent model (the heterogeneous agents model is outlined in Appendix A). In Section 3, we present $GSSA$. In Section 4, we discuss the determinants of accuracy of $GSSA$. In Section 5, we describe the numerical experiments performed for both the representative- and heterogeneous-agent models. Finally, in Section 6, we conclude.

2 The model

We study the standard representative-agent neoclassical stochastic growth model:

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

$$\text{s.t. } c_t + k_{t+1} = \left(1-d\right) k_t + \theta_t f\left(k_t\right), \tag{2}$$

$$\ln \theta_{t+1} = \rho \ln \theta_t + \epsilon_{t+1}, \qquad \epsilon_{t+1} \sim \mathcal{N}\left(0, \sigma^2\right), \tag{3}$$

where initial condition (k_0, θ_0) is given. Here, E_t is the operator of conditional expectation; c_t, k_t and θ_t are, respectively, consumption, capital and productivity level; $\delta \in (0,1)$ is the discount factor; $d \in (0,1]$ is the depreciation rate of capital; $\rho \in (-1,1)$ is the autocorrelation coefficient; and $\sigma \geq 0$ is the standard deviation. The utility and production functions, u and f, respectively, are assumed to be strictly increasing, continuously differentiable and concave.

An interior solution to problem $(1)-(3)$ satisfies the following Euler equation:

$$u'(c_t) = \delta E_t \left\{ u'(c_{t+1}) \left[1 - d + \theta_{t+1} f'(k_{t+1}) \right] \right\}, \tag{4}$$

where u' and f' are the first derivatives of the utility and production functions, respectively. In this paper, we look for a solution to problem $(1)-(3)$ in the form of capital policy function, $k_{t+1} = K(k_t, \theta_t)$, satisfying conditions $(2)-(4)$. To approximate the capital policy function, we use the following representation of Euler equation (4),

$$k_{t+1} = E_t \left[G(k_t, \theta_t, \epsilon_{t+1}) \right], \tag{5}$$

where $G(k_t, \theta_t, \epsilon_{t+1})$ is given by

$$G(k_t, \theta_t, \epsilon_{t+1}) \equiv \delta \frac{u'(c_{t+1})}{u'(c_t)} \left[1 - d + \theta_{t+1} f'(k_{t+1}) \right] k_{t+1}, \tag{6}$$

with c_t and c_{t+1} being determined by conditions (2) and (3) and by the capital policy function, $K(k_t, \theta_t)$. Condition (5) holds because $u'(c_t) \neq 0$ and because k_{t+1} is t-measurable.[4]

3 Generalized stochastic simulation algorithm

We parameterize end-of-period capital, k_{t+1}, in the left side of the Euler equation (5) with a flexible functional form,

$$k_{t+1} = \Psi(k_t, \theta_t; \beta), \tag{7}$$

where β is a vector of coefficients. To find β, the generalized stochastic simulation algorithm ($GSSA$) proceeds as follows:

Choose a simulation length, T. Fix initial condition (k_0, θ_0). Draw a sequence of productivity shocks, $\{\epsilon_t\}_{t=1,\ldots,T}$. Compute a sequence of productivity levels, $\{\theta_t\}_{t=0,\ldots,T}$, using (3), and fix it for all simulations. For initial iteration $s = 1$, fix a vector of coefficients $\beta^{(1)}$.

[4] In a similar way, one can use Euler equation (4) to express other t-measurable variables, e.g., $\ln(k_{t+1})$, c_t and $u'(c_t)$. JMM (2009) show that the choice of a policy function to approximate can affect the numerical stability of stochastic simulation methods.

- Step 1. On iteration s, use the assumed policy function $\Psi\left(k_t, \theta_t; \beta^{(s)}\right)$ in (7) to compute forward the capital path, $\{k_{t+1}\}_{t=0,...,T}$, for the given sequence $\{\theta_t\}_{t=0,...,T}$. Calculate the consumption path, $\{c_t\}_{t=0,...,T}$, from budget constraint (2).

- Step 2. Approximate conditional expectation (integral) $E_t\left[G\left(k_t, \theta_t, \epsilon_{t+1}\right)\right]$ in (5) for $t = 0, ..., T-1$ as a weighted sum of the integrand, G, evaluated in H nodes:

$$Y_t \equiv \sum_{h=1}^{H} G\left(k_t, \theta_t, \epsilon_{t+1,h}\right) w_{t,h}, \tag{8}$$

  where G is defined in (6) and $\{\epsilon_{t+1,h}\}_{h=1,...,H}$ and $\{w_{t,h}\}_{h=1,...,H}$ are integration nodes and weights, respectively; k_{t+1} is given by (7); c_t follows directly from (2); and $c_{t+1,h}$ is determined by (2) conditional on future productivity shock $\epsilon_{t+1,h}$ with $\theta_{t+1,h} = \theta_t^{\rho} \exp\left(\epsilon_{t+1,h}\right)$ and $k_{t+2,h} \equiv \Psi\left(\Psi\left(k_t, \theta_t; \beta\right), \theta_t^{\rho} \exp\left(\epsilon_{t+1,h}\right); \beta\right)$.

- Step 3. Find a vector of coefficients $\widehat{\beta}$ that minimizes the distance between Y_t and $\Psi\left(k_t, \theta_t; \beta\right)$ by running a regression.

- Step 4. Compute the vector of coefficients to be used on the next-iteration $\beta^{(s+1)}$ using fixed-point iteration, namely,

$$\beta^{(s+1)} = (1-\mu)\,\beta^{(s)} + \mu\widehat{\beta}, \tag{9}$$

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

  Iterate on Steps 1-4 until convergence is achieved,

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

  where $\left\{k_{t+1}^{(s)}\right\}_{t=1,...,T}$ and $\left\{k_{t+1}^{(s+1)}\right\}_{t=1,...,T}$ are the capital paths obtained on iterations s and $s+1$, respectively, and parameter $\vartheta > 0$ determines the convergence criterion. (Note that convergence of the capital path implies convergence of the polynomial coefficients, β).

$GSSA$ is similar to the stochastic simulation algorithm (SSA) in JMM (2009, 2010b) except for the integration procedure in Step 2. SSA relies on a specific type of Monte Carlo integration, whereas $GSSA$ can use any integration methods including those unrelated to the estimated density function. Formula (8) represents two alternative integration methods, Monte Carlo and Gauss-Hermite quadrature ones, however other methods can be used as well.

Monte Carlo integration method For each period t and each node h, an H-node Monte Carlo integration method, $MC(H)$, draws H shocks for the next period, $\{\epsilon_{t+1,h}\}_{h=1,...,H}$, and computes (8) by assigning equal weights to all nodes, i.e. $w_{t,h} = 1/H$ for all h and t. We refer to $GSSA$ using the H-node Monte Carlo integration method as $GSSA - MC(H)$. One-node Monte Carlo integration, $MC(1)$, is the conventional integration procedure used by stochastic simulation methods; see Den Haan and Marcet (1990) and JMM (2009, 2010b). It approximates the conditional expectation in (5) by the value of integrand (6) realized in period $t+1$, i.e., $H = 1$, $\epsilon_{t+1,1} = \epsilon_{t+1}$ and $w_{t,1} = 1$ (this implies that $\theta_{t+1,1} = \theta_t^{\rho} \exp(\epsilon_{t+1})$).

Gauss-Hermite quadrature integration method An H-node Gauss-Hermite quadrature integration method, $Q(H)$, evaluates (8) using nodes and weights that are constructed by Gauss-Hermite quadrature integration. For example, two-node Gauss-Hermite quadrature, $Q(2)$, uses $\epsilon_{t+1,1} = -\sigma$, $\epsilon_{t+1,2} = \sigma$ and $w_{t,1} = w_{t,2} = \frac{1}{2}$, and three-node Gauss-Hermite quadrature, $Q(3)$, uses $\epsilon_{t+1,1} = 0$, $\epsilon_{t+1,2} = \sqrt{\frac{3}{2}}$, $\epsilon_{t+1,3} = -\sqrt{\frac{3}{2}}$ and $w_{t,1} = \frac{2\sqrt{\pi}}{3}$, $w_{t,2} = w_{t,3} = \frac{\sqrt{\pi}}{6}$ for all t; see Judd (1998 p.261). We refer to $GSSA$ using H-node Gauss-Hermite quadrature integration as $GSSA - Q(H)$. A one-node Gauss-Hermite quadrature integration method, $Q(1)$, approximates the conditional expectation in (5) by the value of integrand (6) under the assumption of zero shock in period $t+1$, i.e., $H = 1$, $\epsilon_{t+1,1} = 0$ and $w_{t,1} = 1$ (this implies that $\theta_{t+1,1} = \theta_t^{\rho}$).

4 Determinants of accuracy

We now explore factors that determine the accuracy of $GSSA$. First, since the integration method used in Step 2 is not exact, we have an integration

error ε_t, namely,

$$\varepsilon_t = E_t[\cdot] - Y_t, \tag{11}$$

where $E_t[\cdot]$ is a compact notation for the exact conditional expectation in (5), and Y_t is the approximation given by (8).

Second, since we use a finite-degree polynomial for approximation, there is an error from omitting high-degree polynomial terms (curve-fitting error). Let $\Psi(k_t, \theta_t; \beta) = X_t^m \beta^m$ be a complete ordinary polynomial of degree m composed of $n+1$ terms, where $X_t^m = \left(1, k_t, \theta_t, k_t^2, k_t\theta_t, \theta_t^2, ..., \theta_t^m\right) \in \mathbb{R}^{1\times(n+1)}$ and $\beta^m = (\beta_0^m, \beta_1^m, ..., \beta_n^m)' \in \mathbb{R}^{(n+1)\times 1}$. Furthermore, let $X_t^\infty \beta^\infty$ be an infinite-degree polynomial, which is equivalent to the true policy function $K(k_t, \theta_t)$. Then, the error from omitting high-degree polynomial terms is

$$X^{\infty-m}\beta^{\infty-m} \equiv X^\infty \beta^\infty - X^m \beta^m, \tag{12}$$

where $X^\infty \equiv \left(X_0^\infty, ..., X_{T-1}^\infty\right)' \in \mathbb{R}^{T\times\infty}$, $X^m \equiv \left(X_0^m, ..., X_{T-1}^m\right)' \in \mathbb{R}^{T\times(n+1)}$, and β^m is the initial segment of $\beta^\infty = (\beta_0^\infty, \beta_1^\infty, ...)' \in \mathbb{R}^{\infty\times 1}$.

If we use an approximating polynomial of an infinite degree, $X^\infty\beta^\infty$, but our integration method produces errors of type (11), the regression model in Step 3 is

$$E = X^\infty \beta^\infty + \varepsilon, \tag{13}$$

where $E \equiv (E_0[\cdot], ..., E_{T-1}[\cdot])' \in \mathbb{R}^{T\times 1}$ and $\varepsilon \equiv (\varepsilon_0, ..., \varepsilon_{T-1})' \in \mathbb{R}^{T\times 1}$. Let us assume that the coefficients in regression equation (13) are estimated using the ordinary least-squares (OLS) method,

$$\widehat{\beta}^\infty = \left[(X^\infty)' X^\infty\right]^{-1} (X^\infty)' E = \beta^\infty + \left[(X^\infty)' X^\infty\right]^{-1} (X^\infty)' \varepsilon, \tag{14}$$

where $\widehat{\beta}^\infty$ denotes the OLS estimator of β^∞.[5] The more accurate is the integration rule, the smaller is the integration error, ε, and the closer is $\widehat{\beta}^\infty$ to β^∞.

Since, in practice, we use a finite- rather than infinite-degree polynomial, the regression model in Step 3 is misspecified,

$$E = X^m \beta^m + v, \tag{15}$$

[5] OLS is used in the discussion of this section for the sake of expository convenience. In the context of the stochastic simulation class of methods, OLS is numerically unstable. See JMM (2009, 2010b) for a description of numerically stable approaches that can be used in the approximation step of a stochastic simulation algorithm.

where $v \equiv (v_0, ..., v_{T-1})' \in \mathbb{R}^{T\times 1}$ is an error term. Substituting the true regression model, (13), into the OLS estimator corresponding to (15) and using (12), we get

$$\widehat{\beta}^m = \left[(X^m)' X^m\right]^{-1} (X^m)' (X^\infty \beta^\infty + \varepsilon) = \\ \beta^m + \left[(X^m)' X^m\right]^{-1} (X^m)' \varepsilon + \left[(X^m)' X^m\right]^{-1} (X^m)' X^{\infty-m} \beta^{\infty-m}. \quad (16)$$

As a result, the OLS estimator, $\widehat{\beta}^m$, contains errors of both types, (11) and (12). To measure accuracy of a solution, we compute the difference in capital allocations produced by the true policy function, $X^\infty \beta^\infty$, and our approximate policy function, $X^m \widehat{\beta}^m$,

$$R \equiv X^\infty \beta^\infty - X^m \widehat{\beta}^m = -P^m \varepsilon - \left[P^m - I_T\right] X^{\infty-m} \beta^{\infty-m}, \quad (17)$$

where I_T is a $T \times T$ identity matrix, and $P^m \equiv X^m \left[(X^m)' X^m\right]^{-1} (X^m)'$ is a matrix known in econometrics as a projection matrix.

Below, we assess the magnitude of errors of types (11) and (12) in our model. We do not intend to give formal proofs concerning bounds on the errors but lay out a series of arguments that helps us expose what we find later to be quantitatively important.

Integration errors Let us consider an infinitely dimensional polynomial approximation $X^\infty \beta^\infty$ leading to estimator (14), and let us assume that the integration error, ε_t, is *i.i.d.* with zero mean and constant variance, σ_ε^2 (i.e., we neglect a bias resulting from omitting the high-degree polynomial terms). Under these assumptions, we have the standard version of the Central Limit Theorem, namely, the asymptotic distributions of OLS estimator (14) and the capital-allocation errors in (17) are given, respectively, by

$$\sqrt{T}\left(\widehat{\beta}^m - \beta^m\right) \sim \mathcal{N}\left(0, \left[(X^\infty)' X^\infty\right]^{-1} \sigma_\varepsilon^2\right), \quad (18)$$

$$\sqrt{T} R_t \sim \mathcal{N}\left(0, P^\infty \sigma_\varepsilon^2\right). \quad (19)$$

Thus, a rate of convergence of our approximate solution to the true solution is $\sqrt{T}$. This means that to increase accuracy of the solution by an order of magnitude, we must increase the simulation length, T, by two orders of magnitude.

Consider the conventional one-node Monte Carlo integration method, $MC(1)$, which approximates the expectation in (5) with the next-period realization of the integrand, so that the integration error is

$$\varepsilon_t = E_t\left[G\left(k_t, \theta_t, \epsilon_{t+1}\right)\right] - G\left(k_t, \theta_t, \epsilon_{t+1}\right), \tag{20}$$

where $G\left(k_t, \theta_t, \epsilon_{t+1}\right)$ is given by (6). In a typical real business cycle model, fluctuations in variables like $G\left(k_t, \theta_t, \epsilon_{t+1}\right)$ amount to several percents. If integration error (20) is on average 1%, (i.e., $\sigma_\varepsilon = 0.01$), then regression model (13) with $T = 10,000$ observations allows us to approximate the conditional expectation with accuracy of about $\sigma_\varepsilon/\sqrt{T} = 10^{-4}$. To achieve accuracy of 10^{-5}, we need to increase the simulation length to $T = 1,000,000$. Thus, a very long simulation is needed to approximate the conditional expectation with a high degree of accuracy if the integration method used is not sufficiently accurate.[6]

We now present an integration theorem regarding our second integration method, the Gauss-Hermite quadrature one; see, e.g., Judd (1998, p. 261).

Theorem 1 *Under an H-node Gauss-Hermite quadrature integration method, the integration error is equal to*

$$\varepsilon_t = E_t\left[G\left(k_t, \theta_t, \epsilon_{t+1}\right)\right] - Y_t = \frac{H!\sqrt{\pi}}{2^H\,(2H)!} G_3^{(2H)}\left(k_t, \theta_t, \epsilon\right), \tag{21}$$

where $G\left(k_t, \theta_t, \epsilon_{t+1}\right)$ and Y_t are given by (6) *and* (8)*, respectively; $G_3^{(2H)}$ denotes a $2H$-th order partial derivative of G with respect to the third argument; and ϵ is some real number, $\epsilon \in (-\infty, \infty)$.*

For a function G that is smooth and has little curvature, integration error ε_t decreases rapidly with the number of quadrature nodes, H. This is because the term $\frac{H!\sqrt{\pi}}{2^H(2H)!}$ goes rapidly to zero with H. For example, for H equal to 1, 2, 3 and 10, this term is 0.44, 0.037, 0.0018 and 3×10^{-15}, respectively. In particular, Gauss-Hermite quadrature integration is exact for functions that are linear in ϵ since for such functions, we have $G_3^{(2H)}\left(k_t, \theta_t, \epsilon\right) = 0$ for all $H \geq 1$.

[6]The Monte Carlo integration method with H nodes, $MC(H)$, has the convergence rate of $\sqrt{TH}$ and has the cost proportional to TH. Roughly speaking, an increase in the number of integration nodes from $H = 1$ to $H > 1$ under $MC(H)$ has the same effect on accuracy and cost as an increase in the simulation length from T to TH under $MC(1)$.

The properties of the integration errors produced by Gauss-Hermite quadrature are in general unknown. If ε_t has a non-zero expected value, the Central Limit Theorem does not apply, and $\widehat{\beta}^{\infty}$ does not converge to β^{∞} asymptotically. However, provided that the integration errors are very small in size, our approximation to the conditional expectation will be still very accurate.

Error from omitting high-degree polynomial terms (curve-fitting error) Suppose now that the only source of errors in (16) and (17) is the omission of high-degree polynomial terms, $X^{\infty-m}\beta^{\infty-m}$ (i.e., we assume that the integration procedure is exact so that integration errors are absent). Below, we evaluate errors resulting from omitting high-degree polynomial terms in a one-dimensional case under the Chebyshev polynomial representation; see Judd (1998, p. 209). Chebyshev polynomial is defined with the following recursive formula: $T_0(x) = 1$, $T_1(x) = x$ and $T_m(x) = 2xT_{m-1}(x) - T_{m-2}(x)$ for $m \geq 2$.

Theorem 2 *Let a function $\psi : [-1, 1] \to \mathbb{R}$ be ℓ times differentiable and have a Chebyshev polynomial representation*

$$\psi(x) = \frac{1}{2}\eta_0 + \sum_{m=1}^{\infty} \eta_m T_m(x), \tag{22}$$

where $\{\eta_m\}_{m=0,\ldots,\infty}$ are the Chebyshev polynomial coefficients. Then, there exists a constant η such that $|\eta_m| \leq \frac{\eta}{m^{\ell}}$, $m \geq 1$.

Thus, the Chebyshev polynomial coefficients decline rapidly with the polynomial degree m if function $\psi(x)$ is sufficiently smooth. Provided that the coefficients on high-degree polynomial terms are small in size, the error from omitting such terms, $X^{\infty-m}\beta^{\infty-m}$, is also small in size. Other polynomial representations (e.g., ordinary polynomials, Hermite polynomials, Legendre polynomials) are linear combinations of Chebyshev polynomials so that result (22) applies to them as well.

Errors associated with stochastically generated solution domain In the above discussion, we treated the dependent variables, X^m, in the regression as being exogenous. However, such variables are constructed by $GSSA$ endogenously as a part of the solution procedure. It might happen that the simulated points produced by $GSSA$ do not adequately represent

the relevant solution domain. First, the simulation length, T, might be not sufficiently large. Second, the solution might depend on a specific random sequence of productivity levels, $\{\theta_t\}_{t=0,\ldots,T}$. Third, the capital series produced by our approximate policy function, $X^m\widehat{\beta}^m$, might differ from those implied by the true policy function, $X^\infty\beta^\infty$. Finally, stochastic simulation overrepresents the center of the ergodic distribution and underrepresents the tails of the ergodic distribution. The above factors can be also important for accuracy of solutions.

5 Numerical experiments

In this section, we discuss the implementation details of $GSSA$ and describe the results of our numerical experiments. We first study the representative-agent model of Section 2, and we later extend this model to include heterogeneous agents.

5.1 Representative-agent model

We investigate how the simulation length, polynomial degree and integration method affect accuracy of the $GSSA$ solutions in the context of the representative-agent model.

5.1.1 Implementation

We assume a constant relative risk aversion (CRRA) utility function, $u(c_t) = \frac{c_t^{1-\gamma}-1}{1-\gamma}$, with risk-aversion coefficient $\gamma \in (0,\infty)$ and a Cobb-Douglas production function, $f(k_t) = k_t^\alpha$, with capital share $\alpha = 0.36$. We set the discount factor and depreciation rate at $\delta = 0.99$ and $d = 0.025$, respectively. We set the damping parameter for fixed-point iteration in (9) at $\mu = 0.1$, and we set the convergence parameter in (10) at $\vartheta = 9$ (i.e., we target nine-digit precision).

We implement one- and H-node Monte Carlo and quadrature integration methods as discussed in Section 3. For a detailed description of quadrature and monomial formulas and examples of their use, see JMM (2010a). Our regression method in Step 3 is the numerically stable least-squares method based on singular value decomposition, SVD; see JMM (2009, 2010b).

For each experiment, we report the time necessary for computing a solution (in seconds) and unit-free Euler equation errors (defined as the difference between the left and right sides of (5) divided by k_{t+1}) on a stochastic simulation of 1,000 observations. To compute conditional expectation in the test, we use an accurate ten-node Gauss-Hermite quadrature rule. We run the computational experiments on a desktop computer ASUS with Intel(R) Core(TM)2 Quad CPU Q9400 (2.66 GHz). Our programs are written in Matlab, version 7.6.0.324 (R2008a).

5.1.2 Monte Carlo integration

We first solve model $(1)-(3)$ using $GSSA-MC(H)$. We parameterize the CRRA utility function by $\gamma = 1$, and we parameterize the process for shocks (3) by $\rho = 0.95$ and $\sigma = 0.01$. We implement Monte Carlo integration using H randomly drawn nodes in each simulated point with $H \in \{1, 30, 3000\}$; we vary the simulation length by $T \in \{30, 300, 3000, 10000\}$, and we compute polynomial approximations up to degree five, $m \in \{1, ..., 5\}$. $GSSA-MC(1)$ corresponds to SSA as is used in JMM (2009). The results are reported in Table 1.

When the simulation length, T, and the number of integration nodes, H, are small, $GSSA - MC(H)$ fails to deliver high-degree polynomial approximations and the accuracy of solutions is low. Increasing T and H helps restore numerical stability and increase accuracy. The highest accuracy is achieved in the experiment with the largest values of T and H considered (i.e., $T = 10,000$ and $H = 3,000$) under the third-degree polynomial, $m = 3$, namely, the Euler equation errors are of size 10^{-6}.

5.1.3 Gauss-Hermite quadrature integration

We next solve model $(1)-(3)$ using $GSSA-Q(H)$. We specifically repeat the experiments reported in Section 5.1.2 using Gauss-Hermite quadrature integration with $H \in \{1, 2, 10\}$ nodes instead of the Monte Carlo integration with $H \in \{1, 30, 3000\}$ nodes. We report the results in Table 2.

The accuracy of the solutions increases with the polynomial degree, the sample size and the number of integration nodes. Any of these three factors can restrict the overall accuracy of solutions. Specifically, if we use a rigid first-degree polynomial, the maximum error is of order 10^{-4}; if we use a small sample size of $T = 30$, it is of order 10^{-5}; and if we use the least

accurate one-node integration rule, it is of order 10^{-7}. In turn, under our most accurate case of fifth-degree polynomial, $T = 10,000$ and the most accurate ten-node integration, the maximum error is of order 10^{-9}. Under quadrature integration, the simulation length plays a less important role than under Monte Carlo integration. This is because in the former case, the simulation length affects only the solution domain, while in the latter case, it affects both the solution domain and the number of integration nodes. The key result of Table 2 is that the change in the integration procedure leads to a large increase in accuracy. The least accurate solution, produced by $GSSA - Q(H)$ under $T = 30$ and $H = 1$, is still more accurate than the most accurate solution, delivered by $GSSA - MC(H)$ under $T = 10,000$ and $H = 3000$. In order $GSSA - MC(H)$ had a comparable degree of accuracy as $GSSA - Q(H)$, its running time must be of orders of magnitude larger.

5.1.4 Sensitivity experiments

In Table 3, we present the results of the sensitivity experiments, namely, we consider $\gamma \in \{0.2, 1, 5\}$, $\rho \in \{0.95, 0.99\}$ and $\sigma \in \{0.01, 0.03\}$. We use $T = 10,000$, polynomial approximations up to degree five and two alternative integration methods such as $MC(1)$ and $Q(H)$ with $H = \{1, 2, 10\}$.

As is seen from the table, the tendencies, observed in the sensitivity experiments, are similar to those observed in Tables 1 and 2. Under some parameterizations, the difference between the mean and maximum errors is large (up to two orders of magnitude), which indicates that the approximation obtained is not uniformly accurate. Since stochastic simulation overrepresents the center and underrepresents the tails of the ergodic distribution, the accuracy of solutions on tails might be low. To achieve more uniform accuracy, we should use an approximation method that minimizes the maximum error instead of the least-squares error. Finally, we find that the results are not visibly affected by a specific random draw of productivity shocks.

5.2 Heterogeneous-agent economy

In this section, we apply $GSSA$ for solving a multi-country (heterogeneous-agent) variant of representative-agent model $(1) - (3)$. A formal description of the heterogeneous-agent model, the parameters choice and the implementation details are provided in Appendix A. We compute the second-degree polynomial approximations, and we employ two alternative integration meth-

ods, namely, the conventional (one-node) Monte Carlo integration method, $MC\,(1)$, and the one-node Gauss-Hermite quadrature integration method, $Q\,(1)$. The results are shown in Table 4.

Under $MC\,(1)$, the accuracy of solutions visibly depends on the sample size, T, relative to the dimensionality of the problem, J (the number of the regression coefficients, $n+1$). For a given T, an increase in J leads to a lower degree of accuracy because there are more regression coefficients to identify; for example, under $T = 10,000$, going from $J = 2$ to $J = 30$ increases the mean solution error by about a factor of 10, namely, from $8.3 \cdot 10^{-5}$ to $9.3 \cdot 10^{-4}$. For a given J, an increase in T leads to a higher degree of accuracy; for example, under $J = 8$, going from $T = 10,000$ to $T = 100,000$ decreases the mean solution error by about a factor of 3, namely, from $2.4 \cdot 10^{-4}$ to $7.4 \cdot 10^{-5}$. These results are in line with a $\sqrt{T}$-rate of convergence predicted by the Central Limit Theorem.

Replacing $MC\,(1)$ with $Q\,(1)$ increases the accuracy of solutions between one and two orders of magnitude. A reduction in the simulation length from $T = 10,000$ to $T = 3,000$ has virtually no effect on the accuracy of solutions. With a small number of countries, we can reduce T even further without a visible decrease in accuracy (these experiments are not reported), however, we must have at least as many observations as the polynomial coefficients to identify; for example, for the model with $J = 30$, we cannot reduce T below $n + 1 = 1,891$.

The computational time for $MC\,(1)$ and $Q\,(1)$ is similar: it ranges from 3 minutes to 3 hours depending on the number of countries, J, and simulation length, T. This time is quite modest given that we use a standard desktop computer and solve computationally costly high-dimensional problems.

Finally, in the studied models, $GSSA - Q\,(1)$ delivers accuracy that is comparable to the highest accuracy achieved in the literature; see KMMP (2011) for accuracy comparison of different solution methods. If $Q\,(1)$ happens to be not sufficiently accurate in other applications, we can extend $GSSA$ to include more accurate integration methods such as low-cost non-product monomial integration rules; see JMM (2010a).

6 Conclusion

In this paper, we generalize the stochastic simulation approach to make it compatible with both Monte Carlo and deterministic (quadrature and mono-

mial) integration methods. At the conceptual level, our analysis provides a link between the stochastic simulation and projection classes of algorithms and makes it possible to assess different kinds of approximation errors. At the practical level, our generalized framework allows us to select the integration method most suitable for a given application. Any existing stochastic simulation algorithm can be easily extended to include deterministic integration. A version of $GSSA$ based on one-node Gauss-Hermite quadrature integration is particularly easy to implement and is as simple and intuitive as the conventional stochastic simulation method based on one-node Monte Carlo integration. $GSSA$ can solve high-dimensional problems that are computationally demanding and even intractable for earlier solution procedures, and it delivers accuracy comparable to the highest accuracy attained in the literature.

References

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

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

[3] Gallant, R. and D. Nychka (1987). Semi-Nonparametric maximum likelihood estimation. Econometrica 55, 363-390.

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

[5] Judd, K., (1998). Numerical Methods in Economics. London, England: The MIT Press, Cambridge Massachusetts.

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

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

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

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

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

[11] Maliar, L. and S. Maliar, (2005). Solving nonlinear stochastic growth models: iterating on value function by simulations. Economics Letters 87, 135-140.

[12] Maliar, S., L. Maliar and K. Judd, (2011). Solving the multi-country real business cycle model using ergodic set methods. Journal of Economic Dynamic and Control 35, 207-228.

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

7 Appendix A

In this section, we describe the heterogeneous-agent model studied in Section 5.2. For the sake of comparison, we consider the same setup as the one used in JMM (2010a) and JMM (2010b) for testing the performance of the cluster-grid and stochastic simulation approaches, respectively (the notation in this section corresponds to that in JMM, 2010b).

A world economy consists of J countries. Each country is populated by a representative consumer. The social planner's problem is

$$\max_{\left\{c_t^j,k_{t+1}^j\right\}_{t=0,...,\infty}^{j=1,...,J}} E_0 \sum_{j=1}^{J} \upsilon^j \left[\sum_{t=0}^{\infty} \delta^t u^j\left(c_t^j\right)\right] \tag{23}$$

subject to

$$\sum_{j=1}^{J} c_t^j + \sum_{j=1}^{J} k_{t+1}^j = \sum_{j=1}^{J} k_t^j (1-d) + \sum_{j=1}^{J} \theta_t^j f^j \left(k_t^j\right), \tag{24}$$

$$\ln \theta_{t+1}^j = \rho \ln \theta_t^j + \left(\epsilon_{t+1} + \xi_{t+1}^j\right), \quad j = 1, ..., J, \tag{25}$$

where initial condition $\left\{k_0^j, \theta_0^j\right\}^{j=1,...,J}$ is given, and $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, $\xi_t^j \sim \mathcal{N}(0, \sigma^2)$. We denote by c_t^j, k_{t+1}^j, θ_{t+1}^j and υ^j a country's j consumption, end-of-period capital, productivity level and welfare weight, respectively. A common productivity shock, ϵ_{t+1}, and a country-specific technology shock, ξ_{t+1}^j, determines country's j productivity level. The utility and production functions, u^j and f^j, are both increasing, continuously differentiable and concave.

We assume that $u^j = u$ and $f^j = f$ for $j = 1, ..., J$, which implies that $\upsilon^j = 1$ and that $c_t^j = c_t \equiv 1/J \sum_{j=1}^{J} c_t^j$ for $j = 1, ..., J$. We parameterize the capital policy functions of each country, $k_{t+1}^j = K^j \left(\left\{k_t^j, \theta_t^j\right\}^{j=1,...,J}\right)$, by a polynomial $\Psi^j \left(\left\{k_t^j, \theta_t^j\right\}_{j=1}^{J}; \beta^j\right)$ using a set of Euler equations:

$$k_{t+1}^j = E_t \left\{ \delta \frac{u'\left(c_{t+1}\right)}{u'\left(c_t\right)} \left[1 - d + \theta_{t+1}^j f'\left(k_{t+1}^j\right)\right] k_{t+1}^j \right\} \approx \Psi \left(\left\{k_t^j, \theta_t^j\right\}_{j=1}^{J}; \beta^j\right), \tag{26}$$

for $j = 1, ..., J$. Note that since the countries are identical in their fundamentals, we could have computed just one capital policy function for all countries. However, we treat the countries as fully heterogeneous and compute a separate policy function for each country considered. This approach allows us to make a judgement regarding accuracy and costs in models with heterogeneous fundamentals. $GSSA$ for solving the heterogeneous-agent model is similar to the one presented in Section 3 for the case of the representative-agent model, however, instead of doing each step just once, we do it for each country $j = 1, ..., J$.

We parameterize the model by $\gamma = 1$; $\rho = 0.95$ and $\sigma = 0.01$. We set the damping parameter for the first- and second-degree polynomial approximations at $\mu = 0.1$ and $\mu = 0.05$, respectively; we use $\vartheta = 6$ (i.e., six-digit precision) for all experiments. To estimate the regression coefficients, we normalize the data and use the OLS method. In the studied model, data normalization is sufficient for stabilizing the stochastic simulation approach

under low-degree polynomial approximations; see JMM (2009, 2010b) for a discussion. An important advantage of linear approximation methods is that they can be vectorized in multi-dimensional applications; this enables us to iterate on policy functions of all heterogeneous countries simultaneously which reduces the costs considerably. To compute the Euler equation errors, we evaluate the conditional expectation in (26) using a $2J$-node monomial rule; JMM (2010a) shows that this rule is sufficiently accurate for the purpose of accuracy checks.

Table 1. Solving the benchmark model using Monte Carlo integration.

	T=30			*T=300*			*T=3000*			*T=10000*		
	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
Monte Carlo integration with P=1 draw												
1st degree	8,98(-4)	2,16(-3)	1	2,97(-4)	1,36(-3)	4	5,38(-5)	3,05(-4)	1(1)	5,5(-5)	3,0(-4)	1(1)
2nd degree	-	-	-	3,80(-4)	2,03(-3)	3	9,26(-5)	3,94(-4)	1(1)	4,0(-5)	2,0(-4)	3(1)
3rd degree	-	-	-	-	-	-	1,03(-4)	5,91(-4)	1(1)	5,1(-5)	3,3(-4)	2(1)
4th degree	-	-	-	-	-	-	1,38(-4)	1,01(-3)	1(1)	5,4(-5)	9,0(-4)	3(1)
5th degree	-	-	-	-	-	-	1,53(-4)	1,64(-3)	1(1)	7,1(-5)	5,1(-4)	3(1)
Monte Carlo integration with P=30 draws												
1st degree	1,28(-4)	7,10(-4)	4	6,00(-5)	3,15(-4)	1(1)	4,40(-5)	2,58(-4)	6(1)	4,7(-5)	3,0(-4)	2(2)
2nd degree	-	-	-	1,84(-4)	1,02(-3)	1(1)	1,60(-5)	6,68(-5)	4(1)	8,6(-6)	4,1(-5)	2(2)
3rd degree	-	-	-	2,27(-4)	2,61(-3)	1(1)	1,89(-5)	8,87(-5)	5(1)	1,5(-5)	1,4(-4)	1(2)
4th degree	-	-	-	5,79(-4)	1,02(-2)	5(1)	2,04(-5)	2,59(-4)	5(1)	1,6(-5)	1,6(-4)	1(2)
5th degree	-	-	-	-	-	-	2,96(-5)	3,35(-4)	6(1)	1,7(-5)	3,1(-4)	1(2)
Monte Carlo with integration with P=3000 draws												
1st degree	1,28(-4)	5,26(-4)	4(2)	5,41(-5)	3,82(-4)	6(2)	4,36(-5)	2,86(-4)	3(3)	4,9(-5)	3,1(-4)	7(3)
2nd degree	7,28(-4)	5,39(-3)	3(2)	1,39(-5)	1,14(-4)	6(2)	1,74(-6)	1,92(-5)	1(3)	1,2(-6)	1,0(-5)	5(3)
3rd degree	2,08(-3)	9,51(-2)	6(2)	3,07(-5)	3,55(-4)	5(2)	2,29(-6)	4,00(-5)	1(3)	1,1(-6)	9,5(-6)	2(3)
4th degree	-	-	-	1,32(-4)	3,19(-3)	4(2)	2,45(-6)	5,58(-5)	1(3)	1,1(-6)	1,0(-5)	3(3)
5th degree	-	-	-	7,66(-4)	2,68(-2)	5(2)	2,51(-6)	2,73(-5)	1(3)	1,2(-6)	1,2(-5)	2(3)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively; *CPU* is computational time in seconds; and the notation *x(m)* means $x \cdot 10^m$.

Table 2. Solving the benchmark model using the Gauss-Hermite quadrature integration method.

	T=30			*T=300*			*T=3000*			*T=10000*		
	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
Gauss-Hermite quadrature with P=1 node												
1st degree	4,49(-5)	2,66(-4)	2	5,19(-5)	3,62(-4)	5	4,50(-5)	2,90(-4)	1(1)	5,0(-5)	3,0(-4)	5(1)
2nd degree	1,05(-6)	1,12(-5)	2	1,18(-6)	1,32(-5)	4	9,11(-7)	9,29(-6)	1(1)	1,1(-6)	1,0(-5)	4(1)
3rd degree	1,92(-6)	3,25(-5)	8(-2)	4,84(-7)	1,45(-6)	1	4,83(-7)	7,07(-7)	8	4,8(-7)	9,5(-7)	3(1)
4th degree	8,05(-7)	1,78(-5)	1(-1)	4,82(-7)	6,41(-7)	6(-1)	4,86(-7)	6,30(-7)	3	4,9(-7)	6,3(-7)	2(1)
5th degree	5,57(-7)	7,61(-6)	2(-1)	4,90(-7)	6,63(-7)	9(-2)	4,86(-7)	6,26(-7)	1	4,9(-7)	6,3(-7)	9
Gauss-Hermite quadrature with P=2 nodes												
1st degree	4,68(-5)	2,71(-4)	2	5,19(-5)	3,68(-4)	4	4,36(-5)	2,85(-4)	1(1)	4,8(-5)	3,1(-4)	5(1)
2nd degree	9,08(-7)	1,15(-5)	2	1,03(-6)	1,33(-5)	2	7,62(-7)	9,60(-6)	1(1)	8,6(-7)	1,0(-5)	3(1)
3rd degree	1,34(-6)	2,00(-5)	1(-1)	4,77(-8)	1,15(-6)	2	2,46(-8)	3,56(-7)	8	3,3(-8)	6,5(-7)	2(1)
4th degree	9,25(-7)	1,71(-5)	1(-1)	3,27(-9)	8,65(-8)	1	1,24(-9)	2,23(-8)	4	2,1(-9)	4,1(-8)	1(1)
5th degree	3,74(-7)	1,08(-5)	2(-1)	1,57(-8)	4,21(-7)	6(-2)	5,21(-10)	7,22(-9)	2	1,1(-10)	1,7(-9)	9
Gauss-Hermite quadrature with P=10 nodes												
1st degree	4,68(-5)	2,71(-4)	1	5,19(-5)	3,68(-4)	8	4,36(-5)	2,85(-4)	4(1)	4,8(-5)	3,1(-4)	2(2)
2nd degree	9,08(-7)	1,15(-5)	2	1,03(-6)	1,33(-5)	8	7,62(-7)	9,60(-6)	3(1)	8,6(-7)	1,0(-5)	1(2)
3rd degree	1,34(-6)	2,00(-5)	1(-1)	4,78(-8)	1,15(-6)	2	2,46(-8)	3,56(-7)	2(1)	3,3(-8)	6,5(-7)	5(1)
4th degree	9,25(-7)	1,71(-5)	1(-1)	3,28(-9)	8,65(-8)	7(-1)	1,25(-9)	2,25(-8)	1(1)	2,1(-9)	4,1(-8)	4(1)
5th degree	3,03(-7)	6,79(-6)	2(-1)	1,57(-8)	4,22(-7)	9(-2)	5,24(-10)	7,00(-9)	4	1,1(-10)	1,6(-9)	2(1)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively; *CPU* is computational time in seconds; and the notation *x(m)* means $x \cdot 10^m$.

Table 3. Comparison of the Monte Carlo and Gauss-Hermite quadrature integration methods: sensitivity results.

	$\gamma = 1/5$			$\gamma = 5$			$\rho = 0.99$			$\sigma = 0.03$		
	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
Monte Carlo integration with P=1 random draw												
1st degree	1,9(-5)	1,9(-4)	8	4,5(-4)	3,1(-3)	8(1)	1,1(-4)	7,1(-4)	8	4,50(-4)	3,12(-3)	1(1)
2nd degree	1,2(-5)	1,2(-4)	1(1)	2,2(-4)	3,8(-4)	2(2)	6,3(-5)	1,4(-4)	4(1)	1,20(-4)	5,39(-4)	5(1)
3rd degree	1,5(-5)	1,3(-4)	1(1)	2,7(-4)	2,3(-3)	2(2)	1,1(-4)	7,9(-4)	3(1)	1,57(-4)	1,12(-3)	4(1)
4th degree	1,9(-5)	4,6(-4)	1(1)	3,4(-4)	3,4(-3)	3(2)	1,1(-4)	9,6(-4)	4(1)	1,65(-4)	2,51(-3)	5(1)
5th degree	2,2(-5)	4,2(-4)	2(1)	-	-	-	1,4(-4)	1,3(-3)	4(1)	2,11(-4)	2,72(-3)	6(1)
Gauss-Hermite quadrature with P=1 node												
1st degree	1,5(-5)	1,4(-4)	2(1)	4,3(-4)	3,2(-3)	3(2)	1,2(-4)	7,2(-4)	5(1)	5,03(-4)	3,21(-3)	5(1)
2nd degree	2,2(-6)	4,1(-6)	2(1)	1,1(-4)	2,3(-4)	3(2)	6,0(-6)	3,0(-5)	4(1)	2,94(-5)	3,14(-4)	5(1)
3rd degree	2,1(-6)	2,3(-6)	2(1)	1,1(-4)	1,4(-4)	2(2)	5,1(-6)	6,8(-6)	3(1)	5,44(-6)	6,43(-5)	4(1)
4th degree	2,1(-6)	2,2(-6)	1(1)	1,1(-4)	1,3(-4)	2(2)	5,1(-6)	5,7(-6)	2(1)	4,35(-6)	1,24(-5)	3(1)
5th degree	2,1(-6)	2,2(-6)	4	1,1(-4)	1,3(-4)	1(2)	5,1(-6)	5,7(-6)	2(1)	4,36(-6)	8,17(-6)	2(1)
Gauss-Hermite quadrature with P=2 nodes												
1st degree	1,4(-5)	1,5(-4)	2(1)	4,5(-4)	3,2(-3)	3(2)	1,2(-4)	7,3(-4)	4(1)	4,91(-4)	3,22(-3)	5(1)
2nd degree	2,6(-7)	3,2(-6)	2(1)	1,3(-5)	1,6(-4)	3(2)	3,5(-6)	3,3(-5)	4(1)	2,67(-5)	3,23(-4)	5(1)
3rd degree	1,0(-8)	1,9(-7)	1(1)	7,0(-7)	1,5(-5)	2(2)	2,9(-7)	4,8(-6)	3(1)	3,18(-6)	6,61(-5)	4(1)
4th degree	4,1(-10)	7,6(-9)	6	6,0(-8)	1,5(-6)	1(2)	3,8(-8)	5,6(-7)	2(1)	5,54(-7)	1,33(-5)	3(1)
5th degree	8,8(-11)	5,4(-10)	2	5,7(-9)	1,5(-7)	1(2)	5,3(-9)	6,6(-8)	2(1)	9,18(-8)	2,30(-6)	2(1)
Gauss-Hermite quadrature with P=10 nodes												
1st degree	1,4(-5)	1,5(-4)	8(1)	4,5(-4)	3,2(-3)	9(2)	1,2(-4)	7,3(-4)	2(2)	4,91(-4)	3,22(-3)	2(2)
2nd degree	2,6(-7)	3,2(-6)	5(1)	1,3(-5)	1,6(-4)	7(2)	3,5(-6)	3,3(-5)	2(2)	2,67(-5)	3,23(-4)	2(2)
3rd degree	1,0(-8)	1,9(-7)	3(1)	7,0(-7)	1,5(-5)	5(2)	2,9(-7)	4,8(-6)	1(2)	3,18(-6)	6,61(-5)	1(2)
4th degree	4,2(-10)	7,6(-9)	1(1)	6,0(-8)	1,5(-6)	3(2)	3,8(-8)	5,6(-7)	7(1)	5,55(-7)	1,33(-5)	8(1)
5th degree	6,5(-11)	5,0(-10)	5	5,4(-9)	1,5(-7)	3(2)	5,3(-9)	6,6(-8)	6(1)	9,18(-8)	2,31(-6)	6(1)

Remark: e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively; *CPU* is computational time in seconds; and the notation *x(m)* means $x \cdot 10^m$.

Table 4. Solving the multi-country model using Monte-Carlo and Gauss-Hermite quadrature integration: comparison results.

J	$n+1$	*MC(1), T = 10,000*			*MC(1), T = 100,000*			*Q(1), T=10,000*			*Q(1), T=3000*		
		e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*	e_{mean}	e_{max}	*CPU*
J=2	15	8.3(-5)	1.1(-3)	8(2)	2.6(-5)	2.4(-4)	9(3)	7,19(-6)	4,06(-5)	2(3)	7,31(-6)	3,59(-5)	6(2)
J=4	45	1.4(-4)	1.3(-3)	1(3)	4.5(-5)	2.7(-4)	1(4)	9,56(-6)	4,54(-5)	4(3)	9,60(-6)	4,59(-5)	1(3)
J=6	91	2.0(-4)	1.6(-3)	2(3)	5.7(-5)	3.5(-4)	2(4)	1,04(-5)	3,55(-5)	5(3)	1,02(-5)	3,53(-5)	1(3)
J=8	153	2.4(-4)	1.4(-3)	3(3)	7.4(-5)	4.5(-4)	2(4)	1,09(-5)	3,20(-5)	9(3)	1,06(-5)	3,22(-5)	2(3)
J=10	231	2.9(-4)	1.7(-3)	5(3)	-	-	-	1,10(-5)	3,01(-5)	1(4)	1,08(-5)	3,09(-5)	5(3)
J=12	325	3.6(-4)	2.1(-3)	6(3)	-	-	-	1,12(-5)	2,97(-5)	2(4)	1,09(-5)	3,01(-5)	2(3)
J=16	561	4.4(-4)	2.4(-3)	2(4)	-	-	-	1,13(-5)	3,17(-5)	3(4)	1,11(-5)	3,56(-5)	5(3)
J=20	861	5.6(-4)	3.1(-3)	2(4)	-	-	-	1,14(-5)	2,93(-5)	5(4)	1,12(-5)	3,75(-5)	6(3)
J=30	1891	9.3(-4)	5.0(-3)	1(5)	-	-	-	1,15(-5)	2,54(-5)	1(5)	1,10(-5)	3,81(-5)	2(4)

Remark: *J* is the number of countries; $n+1$ is the number of polynomial coefficients in the parameterized decision rule of one country; e_{mean} and e_{max} are the average and maximum Euler equation errors, respectively; *T* is the simulation length; *CPU* is computational time in seconds; and the notation *x(m)* means $x \cdot 10^m$.