NBER WORKING PAPER SERIES

HOW TO SOLVE DYNAMIC STOCHASTIC MODELS COMPUTING EXPECTATIONS JUST ONCE

Kenneth L. Judd
Lilia Maliar
Serguei Maliar

Working Paper 17418
http://www.nber.org/papers/w17418

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
September 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, and the Generalitat Valenciana under the grants BEST/2011/283 and BEST/2011/282, respectively. The views expressed herein are those of the authors and do not necessarily reflect the views of the National Bureau of Economic Research.

How to Solve Dynamic Stochastic Models Computing Expectations Just Once
Kenneth L. Judd, Lilia Maliar, and Serguei Maliar
NBER Working Paper No. 17418
September 2011
JEL No. C63

ABSTRACT

We introduce a technique called "precomputation of integrals" that makes it possible to compute conditional expectations in dynamic stochastic models in the initial stage of the solution procedure. This technique can be applied to any set of equations that contains conditional expectations, in particular, to the Bellman and Euler equations. After the integrals are precomputed, we can solve stochastic models as if they were deterministic. We illustrate the benefits of precomputation of integrals using one- and multi-agent numerical examples.

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

Serguei Maliar
Office T-24 Hoover Institution
Stanford University
CA 94305-6010, USA
maliars@stanford.edu

Lilia Maliar
Office T-24 Hoover Institution
Stanford University
CA 94305-6010, USA
maliarl@stanford.edu

1 Introduction

In existing global methods for solving dynamic stochastic models, a conditional expectation function is computed in each iteration.[1] Re-computing expectations in each iteration is costly, and the cost grows rapidly when the number of random variables increases (because the dimensionality of integrals increases), when more accurate integration methods are used (because the number of integration nodes increases), and when models become more complex (because numerical solvers are used more intensively, which involves additional evaluations of integrals).

In this paper, we introduce a simple technique that makes it possible to compute conditional expectations in the initial stage of the solution procedure. The key idea is to approximate the integrand – expression inside the conditional expectation – with a parametric function whose basis functions are separable in endogenous and exogenous state variables. (This property is satisfied for many polynomial families including ordinary, Chebyshev and Hermite polynomials as well as for many nonpolynomial families of functions.) With such basis functions, we can construct integrals of exogenous state variables for any given values of endogenous state variables before the beginning of iterations. Subsequently, in the iterative procedure, values of conditional expectation functions can be immediately derived using the precomputed integrals. Effectively, this allows us to solve stochastic problems as if they were deterministic, with the corresponding reduction in cost. We call the proposed technique *precomputation of integrals*.

Precomputation of integrals works under very general assumptions and can be applied to any set of equations that contains conditional expectations, including the Bellman and Euler equations. Furthermore, precomputation of integrals is compatible with essentially all computational techniques used by existing global solution methods, including a variety of approximating functions, solution domains, integration rules, fitting methods and iterative schemes for finding unknown parameters of approximating functions. That is, apart from precomputation of integrals, the rest of our solution procedure is standard. Finally, given that we must approximate integrals just once, we can use very accurate integration methods that would be intractable inside an iterative cycle.

We illustrate the technique of precomputation of integrals in two applications. One is a simple, one-agent neoclassical growth model, and the other is a more challenging, multi-agent model with up to 20 heterogeneous agents. We study both the Bellman and Euler equation algorithms. In our numerical examples, solution algorithms based on precomputation of integrals proved to be accurate, fast and reliable.

The rest of the paper is as follows: In Section 2, we set up the test model. In Section 3, we show how to precompute integrals for ordinary polynomial functions of state variables.

[1] For reviews of methods for solving dynamic economic models, see Taylor and Uhlig (1990), Rust (1996, 2008), Gaspar and Judd (1997), Judd (1998), Marimon and Scott (1999), Santos (1999), Christiano and Fisher (2000), Miranda and Fackler (2002), Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006), Stachursky (2009), Den Haan (2010), and Kollmann, Maliar, Malin and Pichler (2011).

In Section 4, we apply the precomputation technique to the Bellman and Euler equations. In Section 5, we discuss generalizations of the precomputation technique. In Section 6, we discuss compatibility of precomputation of integrals with other computational techniques in the literature. In Section 7, we present our numerical results. Finally, in Section 8, we conclude.

2 Neoclassical stochastic growth model

To illustrate the technique of precomputation of integrals, we use the standard neoclassical growth model.

2.1 The set up

The representative agent solves

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

$$\text{s.t. } c_t + k_{t+1} = (1-\delta) k_t + a_t f(k_t), \tag{2}$$

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

where the initial condition (k_0, a_0) is given; E_t is the expectation operator conditional on information at time t; c_t, k_t and a_t are consumption, capital and productivity level, respectively; $\beta \in (0,1)$ is the discount factor; $\delta \in (0,1]$ is the depreciation rate of capital; $\rho \in (-1,1)$ is the autocorrelation coefficient of the productivity level; $\sigma \geq 0$ is the standard deviation of the productivity shock; u and f are the utility and production functions, respectively, both of which are strictly increasing, continuously differentiable and concave.

2.2 Bellman equation

We can characterize a solution to the model (1)–(3) by using dynamic programming approach. We solve for the value function $V(k,a)$ that satisfies the Bellman equation,

$$V(k,a) = \max_{k',c} \{u(c) + \beta E[V(k',a')]\} \tag{4}$$

$$\text{s.t. } k' = (1-\delta) k + a f(k) - c, \tag{5}$$

$$\ln a' = \rho \ln a + \epsilon', \qquad \epsilon' \sim \mathcal{N}\left(0, \sigma^2\right), \tag{6}$$

where the primes on variables denote next-period values, and $E[V(k',a')] \equiv E[V(k',a') | k, a]$ is an expectation of $V(k',a')$ conditional on state (k,a).

2.3 Euler equation

We can also characterize a solution of the model (1)–(3) by formulating the Lagrangian function and deriving the first-order conditions (without formulating the Bellman equation). This yields the Euler equation

$$u_1(c) = \beta E\left[u_1(c')\left(1-\delta+a'f_1(k')\right)\right], \tag{7}$$

where F_i denotes a first-order derivative of a function F with respect the i-th argument. Under the Euler equation approach, we solve for policy functions $c = C(k,a)$ and $k' = K(k,a)$ that satisfy (5), (6) and (7).

3 Precomputation of integrals under ordinary polynomials

The Bellman and Euler equations contain conditional expectation functions, $E[V(k',a')]$ and $E[u_1(c')(1-\delta+a'f_1(k'))]$, respectively. The key idea of our precomputation technique is to approximate an integrand (which is $V(k',a')$ and $u_1(c')(1-\delta+a'f_1(k'))$ in the Bellman and Euler equations, respectively) with a parametric function for which conditional expectation can be characterized for any parameter vector and any state of the world. An example of such a function is a complete ordinary polynomial,

$$\mathcal{P}(k,a;b) = b_0 + b_1k + b_2a + b_3k^2 + b_4ka + b_5a^2 + ... + b_na^L, \tag{8}$$

where $b \equiv (b_0, b_1, ..., b_n) \in \mathbb{R}^{n+1}$ is a vector of polynomial coefficients, and L is a degree of the polynomial. Taking into account that k' is known at present and that $a' = a^\rho \exp(\epsilon')$, we can represent conditional expectation of $\mathcal{P}(k',a';b)$ as follows

$$\begin{aligned}
&E[\mathcal{P}(k',a';b)] \\
&\quad = E\left[b_0 + b_1k' + b_2a^\rho\exp(\epsilon') + b_3(k')^2 + b_4k'a^\rho\exp(\epsilon') + ... + b_na^{L\rho}\exp(L\epsilon')\right] \\
&\quad = b_0 + b_1k' + b_2a^\rho E[\exp(\epsilon')] + b_3(k')^2 + b_4k'a^\rho E[\exp(\epsilon')] + ... + b_na^{L\rho}\exp(L\epsilon') \\
&\quad = b_0\mathcal{I}_0 + b_1\mathcal{I}_1k' + b_2\mathcal{I}_2a^\rho + b_3\mathcal{I}_3(k')^2 + b_4\mathcal{I}_4k'a^\rho + ... + b_n\mathcal{I}_na^{L\rho} \\
&\quad = \theta_0 + \theta_1k' + \theta_2a^\rho + \theta_3(k')^2 + \theta_4k'a^\rho + ... + \theta_na^{L\rho} \\
&\quad \equiv \mathcal{P}(k',a^\rho;\theta),
\end{aligned} \tag{9}$$

where $\theta \equiv (\theta_0, \theta_1, ..., \theta_n) \in \mathbb{R}^{n+1}$, and the coefficients θ_i and b_i are related by

$$\theta_i = b_i\mathcal{I}_i, \qquad i = 0, 1, ..., n, \tag{10}$$

where $\mathcal{I}_0 = \mathcal{I}_1 = \mathcal{I}_3 = \mathcal{I}_6 ... = 1$, $\mathcal{I}_2 = \mathcal{I}_4 = \mathcal{I}_7 = ... = E[\exp(\epsilon')]$, $\mathcal{I}_5 = \mathcal{I}_8 = ... = E[\exp(2\varepsilon')]$, ..., $\mathcal{I}_n = E[\exp(L\epsilon')]$.

The integrals $\mathcal{I}_0, ..., \mathcal{I}_n$ can be computed up-front without solving the model (i.e., precomputed). Let e_l be the l-th moment of the function $\exp(\epsilon')$, $l = 0, 1, ..., L$. Since, by assumption, $\epsilon' \sim \mathcal{N}(0, \sigma^2)$, we have

$$e_l \equiv E[\exp(l\epsilon')] = \frac{1}{\sqrt{2\pi\sigma}} \int_{-\infty}^{+\infty} \exp(l\epsilon') \exp\left(-\frac{(\epsilon')^2}{2\sigma^2}\right) d\epsilon'. \tag{11}$$

Note that there is a pattern in the sequence of $\mathcal{I}$'s in terms of e's,

$$\{\mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3, \mathcal{I}_4, \mathcal{I}_5, \mathcal{I}_6, \mathcal{I}_7, \mathcal{I}_8, \mathcal{I}_9, ..., \mathcal{I}_n\} = \{1, 1, e_1, 1, e_1, e_2, 1, e_1, e_2, e_3, ..., e_L\}, \tag{12}$$

where $e_0 = 1$. Once $\mathcal{I}$'s are computed, an evaluation of conditional expectation becomes trivial. Namely, conditional expectation of a polynomial function is given by the same polynomial function but evaluated at a different coefficients vector, i.e., $E[\mathcal{P}(k', a'; b)] = \mathcal{P}(k', a^\rho; \theta)$.

4 Characterizing the solution under precomputation of integrals

In this section, we reformulate the Bellman and Euler equations using the technique of precomputation of integrals.

4.1 Bellman equation algorithm

We parameterize the true value function $V(k, a)$ with a flexible functional form $\widehat{V}(k, a; b)$ given by the polynomial (8). Using the precomputation result (9), we rewrite the Bellman equation (4)–(6) as

$$\widehat{V}(k, a; b) \doteq \max_{k', c} \left\{ u(c) + \beta \widehat{V}(k', a^\rho; \theta) \right\}, \tag{13}$$

$$\text{s.t. } k' = (1 - \delta) k + a f(k) - c, \tag{14}$$

$$\ln a' = \rho \ln a + \epsilon', \qquad \epsilon' \sim \mathcal{N}(0, \sigma^2), \tag{15}$$

$$\theta_i = b_i \mathcal{I}_i, \qquad i = 0, 1, ..., n, \tag{16}$$

where $\doteq$ indicates that the corresponding equality is satisfied approximately. In the transformed Bellman equation (13)–(16), the effect of uncertainty on the solution is summarized by the moments (11) which determine the relation between b and θ. To solve the Bellman equation, we proceed in two steps.

Algorithm 1. Bellman equation algorithm
Precomputation of integrals. Construct $\{\mathcal{I}_0, ..., \mathcal{I}_n\}$ using (11), (12).
Transformed problem. Find b that solves (13)–(16).

Apart from the fact that b and θ differ, the transformed Bellman equation (13)–(16) is standard and can be solved using a variety of solution algorithms available in the literature; we describe an example of such an algorithm in Appendix B.

4.2 Euler equation algorithm

The precomputation technique introduced in Section 3 assumes that the function we parameterize is the same as the function for which we need to compute the expectation. This is true for the Bellman equation approach that parameterizes $V(k, a)$ and requires us to compute $E[V(k', a')]$. However, this is not true for a Euler equation algorithm that typically parameterizes policy functions like $C(k, a)$ or $K(k, a)$ but needs to compute $E[u_1(c')(1 - \delta + a' f_1(k'))]$. To adapt the precomputation technique to the Euler equation, we make a change of variables, namely, we introduce a new variable q' that stands for the integrand in (7), which implies

$$q \equiv u_1(c)[1 - \delta + a f_1(k)]. \tag{17}$$

In terms of q and q', the Euler equation (7) is

$$\frac{q}{1 - \delta + a f_1(k)} = \beta E[q']. \tag{18}$$

If we approximate the function $q = Q(k, a)$, we have the same function under the expectation, $E[Q(k', a')]$, and thus, the results of Section 3 apply.

We parameterize the function $Q(k, a)$ with a flexible functional form $\widehat{Q}(k, a; b)$ given by the polynomial (8). Using the precomputation result (9), we rewrite the Euler equation (18) as

$$\frac{\widehat{Q}(k, a; b)}{1 - \delta + a f'(k)} \doteq \beta \widehat{Q}(k', a^{\rho}; \theta). \tag{19}$$

Again, after integrals are precomputed, all the effect of uncertainty on the solution is compressed into a mapping between the vectors b and θ. To solve the model, we proceed in two steps.

Algorithm 2. Euler equation algorithm
Precomputation of integrals. Construct $\{\mathcal{I}_0, ..., \mathcal{I}_n\}$ using (11), (12).
Transformed problem. Find b that solves (14)–(17) and (19).

As in the case of the Bellman equation, we can use a variety of solution algorithms to solve the transformed problem; see Appendix B for an example of such an algorithm.

In some applications, it might be preferable to approximate policy functions other than the function $Q(k,a)$ because such policy functions are better behaved or more convenient for certain purposes than others; see Maliar, Maliar and Judd (2011) for examples. We can extend the precomputation technique to those cases by approximating other policy functions jointly with $Q(k,a)$. As an example, in Appendix C, we show how to use the precomputation technique for approximating the capital policy function $k' = K(k,a)$.

4.3 Relation between the Bellman and Euler equations under precomputation of integrals

If a solution to the Bellman equation is interior (4)–(6), it satisfies the first-order and envelope conditions, which, respectively, are

$$u_1(c) = \beta E\left[V_1(k',a')\right], \tag{20}$$
$$V_1(k,a) = u_1(c)\left[1-\delta+af_1(k)\right]. \tag{21}$$

The envelope condition (21) implies that the supplementary variable q, introduced in (17), is effectively the derivative of the value function, $q = Q(k,a) = V_1(k,a)$.

Unlike the original Euler equation (7), the conditions (20) and (21) are directly suitable for precomputation of integrals since they contain $V_1(k,a)$ and $E\left[V_1(k',a')\right]$. The precomputation result (9) can be applied to the derivative of the value function $V_1(k,a)$ in the same way as it was applied to the value function $V(k,a)$. The resulting Euler equation is identical to (19) up to notations, namely, instead of $\widehat{Q}(k,a;b)$ and $\widehat{Q}(k',a^{\rho};\theta)$, we have $\widehat{V}_1(k,a;b)$ and $\widehat{V}_1(k',a^{\rho};\theta)$, respectively. That is, in the studied example, approximating the derivative of the value function $V_1(k,a)$ leads to the same solution algorithm (Algorithm 2) as approximating the supplementary function $Q(k,a)$.

5 Generalization of precomputation of integrals

In this section, we provide a general description of the precomputation technique. Let $x \in \mathbb{R}^{n_x}$ and $z \in \mathbb{R}^{n_z}$ be vectors of endogenous and exogenous state variables, respectively. We make the following three assumptions:

A1. Each basis function of an approximating function is multiplicatively separable in x and z, i.e.,

$$\mathcal{P}(x,z;b) = \sum_{i=0}^{n} b_i \psi_i(x)\,\varphi_i(z), \tag{22}$$

where $b \equiv (b_0, ..., b_n) \in \mathbb{R}^{n+1}$; $\psi_i(x)\,\varphi_i(z)$ is the i-th basis function; by convention, $\psi_0(x)\,\varphi_0(z)$ is equal to 1; $\psi_i : \mathbb{R}^{n_x} \to \mathbb{R}$ and $\varphi_i : \mathbb{R}^{n_z} \to \mathbb{R}$ for $i = 1, ..., n$.

A2. Next-period endogenous state variables x' are t-measurable.

A3. Exogenous state variables follow a stochastic process $z' = Z(z, \epsilon')$, where $\epsilon' \in \Omega \subseteq \mathbb{R}^{n_\epsilon}$ is the vector of disturbances with a density function $w : \mathbb{R}^{n_\epsilon} \to \mathbb{R}^+$ that is known at t and that satisfies $\int_{\epsilon' \in \Omega} w(\epsilon')\, d\epsilon' = 1$. Moreover, integrals of type $\int_{\epsilon' \in \Omega} \varphi_i(Z(z, \epsilon'))\, w(\epsilon')\, d\epsilon'$ exist and are finite.

Proposition 1. Under the assumptions *A1*–*A3*, we have

$$E[\mathcal{P}(x', z'; b)\,|x, z] = \mathcal{P}(x', Z(z, \mu); \theta), \tag{23}$$

where $\mu \equiv E[\epsilon']$, and $\theta \equiv (\theta_0, ..., \theta_n)$ and b are related by

$$\theta_i = b_i \int_{\epsilon' \in \Omega} \frac{\varphi_i(Z(z, \epsilon'))}{\varphi_i(Z(z, \mu))} w(\epsilon')\, d\epsilon'. \tag{24}$$

Proof: This result is obtained as follows

$$\begin{aligned}
E[\mathcal{P}(x', z'; b)\,|x, z] &= \int_{\epsilon \in \Omega} \mathcal{P}(x', z'; b)\, w(\epsilon')\, d\epsilon' \\
&\overset{A1}{=} \int_{\epsilon' \in \Omega} \sum_{i=0}^{n} b_i \psi_i(x')\, \varphi_i(z')\, w(\epsilon')\, d\epsilon' \overset{A2}{=} \sum_{i=0}^{n} b_i \psi_i(x') \int_{\epsilon' \in \Omega} \varphi_i(Z(z, \epsilon'))\, w(\epsilon')\, d\epsilon' \\
&\overset{A3}{=} \sum_{i=0}^{n} b_i \psi_i(x')\, \varphi_i(Z(z, \mu)) \int_{\epsilon' \in \Omega} \frac{\varphi_i(Z(z, \epsilon'))}{\varphi_i(Z(z, \mu))} w(\epsilon')\, d\epsilon' \\
&= \sum_{i=0}^{n} \theta_i \psi_i(x')\, \varphi_i(Z(z, \mu)) = \mathcal{P}(x', Z(z, \mu); \theta). \qquad \blacksquare
\end{aligned}$$

Few comments about our assumptions are in order. *A1* holds not only for the ordinary polynomials (8) but also for orthogonal polynomial families (like Chebyshev, Hermite and Legendre families), as well as for many nonpolynomial families (for example, $\psi_i(x)$ and $\varphi_i(z)$ can be trigonometric functions). *A2* means that next-period endogenous state variables are known with certainty at present. *A3* means that endogenous state variables have finite moments. To construct the conditional expectation function, we set the next-period exogenous state variables at the values $Z(z, \mu)$. In the example of Section 3, such reference values allow us to separate out the exogenous state variables and the integrals of disturbances, making the integrals (11) independent of the economy's state. Using the result of Proposition 1, we can precompute conditional expectation functions in the Bellman and Euler equations under very general assumptions; see Appendix A.

6 Combining precomputation of integrals with other techniques in the literature

Precomputation of integrals is compatible with a variety of computational techniques used by existing global solution methods. First, we can use different solution domains such as a tensor-product grid, a set of simulated points (these two domains are used in the present paper), a Smolyak grid (introduced in Krueger and Kubler, 2004), and a cluster grid (proposed in Judd, Maliar and Maliar, 2010). Second, we can use a variety of approximation methods for fitting the unknown functions to the data such as least-squares methods based on singular value decomposition or QR factorization, Tikhonov regularization, least-absolute deviations method, and principal component regression method; see Judd, Maliar and Maliar (2011). Finally, we can use different integration formulas and different iterative schemes for finding unknown parameters of approximating functions; these techniques are discussed in the remainder of the section.

6.1 Integration formulas

Numerical integration is an approximation of integrals by a weighted sum of values that an integrand g takes in a finite set of nodes,

$$E\left[g\left(\epsilon'\right)\right]=\int_{\epsilon'\in\Omega}g\left(\epsilon'\right)w\left(\epsilon'\right)d\epsilon'\doteq\sum_{j=1}^{J}\omega_{j}g\left(\epsilon'_{j}\right), \tag{25}$$

where ϵ'_j and ω_j are the j-th integration node and weight, respectively. Integration formulas differ in their choices of integration nodes and weights. Typically, there is a trade off between the accuracy and cost: integration formulas with more nodes (and thus, with a higher cost) lead to more accurate approximations.

Unidimensional integrals can be accurately computed by discretizing an autoregressive process into a finite-state Markov chain (see Tauchen, 1986) or by using Gaussian quadrature formulas (see, e.g., Judd, 1992). These techniques can be extended to multidimensional case using a product rule, however, the cost grows exponentially with the number of random variables ("curse of dimensionality"). A feasible and accurate alternative for high-dimensional integrals is non-product monomial formulas; see, e.g., Judd (1988, pp. 271-275) for a description of some monomial formulas. The cost of monomial formulas grows with the dimensionality only polynomially (e.g., linearly and quadratically). Another tractable alternative in high-dimensional applications is Monte Carlo integration (see Rust, 1996, and Geweke, 1996, for a discussion), however, it delivers significantly lower accuracy than that attained by deterministic integration methods; see Judd et al. (2011) for comparison results.

Precomputation of integrals can be performed using any of the numerical integration methods discussed above (for some stochastic processes for shocks, integrals of type (25)

can be precomputed analytically; see Maliar and Maliar, 2005, for an example). Since precomputation of integrals is a one-time fixed-cost operation, we can use very accurate integration methods that would be too expensive to use inside an iterative cycle.

6.2 Iterative methods for finding fixed points

The number of evaluations of integrals (and hence, the gains from precomputation of integrals) highly depends on the technique used for finding fixed-point parameters of approximating functions. Three techniques that are used in the literature are time iteration, fixed-point iteration and (quasi)-Newton methods; see Judd (1998, pp. 553-558 and 103-119) for a review. All three techniques can be applied to both the Bellman and Euler equations.

Time iteration requires us to find a numerical solution to a non-linear optimization problem in each grid point (namely, to find a maximum of the Bellman operator or to find a solution to the Euler equation). An integral of type (25) must be evaluated many times in each iteration, which makes time iteration expensive; see Aruoba et al. (2006). An endogenous-grid technique of Carroll (2005) can reduce the cost of time iteration, however, this technique is not trivial to implement for complex models.[2]

In turn, fixed-point iteration does not require us to find a solution to the Bellman and Euler equations in each iteration but ensures that successive approximations converge to a fixed-point in the limit.[3] In this case, integrals are evaluated just once per iteration. However, the rate of convergence of fixed-point iteration is generally slower than that of time iteration, and the number of integral evaluations might be still very large.

Finally, (quasi)-Newton methods perform multiple evaluations of integrals when computing derivatives of the objective function with respect to parameters of approximating functions. The number of integral evaluations varies substantially with a quality of initial guess, a parameterization of the model and a design of the optimization method; see Judd (1992) for a discussion and numerical experiments.

Precomputation of integrals is compatible with all three iterative techniques, which can be implemented in the usual way. The difference is that, in our case, iterations will not involve evaluations of integrals.

[2]For our model, Carroll's (2005) technique requires us to introduce a new variable $Y' \equiv a'f(k') + (1-d)k'$ and to rewrite the Bellman equation as $V(Y,a) = \max\{u(Y-k') + \delta E[V(Y',a')]\}$. Using an endogenous grid on k' (instead of k) makes it possible to compute $E[V(Y',a')]$ with just one evaluation per iteration. However, in a similar model with labor, $Y' = a'f(k',\ell') + (1-d)k'$ depends on labor ℓ', and $E[V(Y',a')]$ cannot be computed without solving for control variables; see Barillas and Fernández-Villaverde (2007) for a method that constructs an endogenous grid by iteration on the labor policy function.

[3]Fixed-point iteration is common for solution algorithms iterating on the Euler equation; see, e.g., Marcet (1988), Miranda and Helmberger (1988), Marcet and Lorenzoni (1999). It is also used in the context of an algorithm iterating on the Bellman equation; see Maliar and Maliar (2005).

7 Numerical examples

In this section, we assess the performance of the global Bellman and Euler equation methods developed in Sections 4.1 and 4.2.

7.1 Implementation details

To parameterize the model (1)–(3), we assume $u(c_t) = \frac{c_t^{1-\gamma}-1}{1-\gamma}$ with $\gamma = \left\{\frac{1}{3}, 1, 3\right\}$, $f(k_t) = Ak_t^{\alpha}$ with $\alpha = 0.36$ and $A = \frac{1/\beta-(1-\delta)}{\alpha}$ (the latter value normalizes the deterministic steady state of capital to one), and $\beta = 0.99$, $\delta = 0.025$, $\rho = 0.95$, $\sigma = 0.01$. To precompute the moments in (11), we use a 10-node Gauss-Hermite quadrature rule $e_l \doteq \sum_{j=1}^{10} \left\{\omega_j \exp\left(l\epsilon_j'\right)\right\}$, where ϵ_j' and ω_j are the j-th integration node and weight, respectively.

We use the simplest possible solution domain, namely, a rectangular, uniformly spaced grid of 10×10 points for capital and productivity within the ergodic range. We compute polynomial approximations of degrees up to 5. To solve for the unknown polynomial coefficients, we use fixed-point iteration. A detailed description of the global methods is provided in Appendix B.

As a measure of accuracy, we report the mean and maximum of unit-free Euler equation errors on a stochastic simulation of 10,000 observations. We use MATLAB software, version 7.6.0.324 (R2008a) and a desktop computer ASUS with Intel(R) Core(TM)2 Quad CPU Q9400 (2.66 GHz), RAM 4MB.

7.2 Results

In Table 1, we provide the results for the global Bellman equation algorithm.

Table 1: Accuracy and speed of the Bellman equation algorithm.[a]

Polynomial	$\gamma = 1/3$			$\gamma = 1$			$\gamma = 3$		
degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
1st	-1.64	-1.63	2.72	-1.64	-1.63	8.60	-1.64	-1.63	13.45
2nd	-3.83	-3.50	0.10	-3.65	-3.42	0.14	-3.20	-2.67	0.20
3rd		failed		-4.83	-4.39	0.16	-4.12	-3.29	0.22
4th	-6.27	-5.81	0.12	-5.96	-5.36	0.17	-5.06	-4.12	0.26
5th		failed		-7.12	-6.43	0.18	-6.04	-4.92	0.25

[a] Notes: L_1 and L_∞ are, repectively, the average and maximum of absolute approximation errors across optimality condition and test points (in log10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); γ is the coefficient of risk aversion.

This algorithm is successful in finding solutions under $\gamma = 1$ and $\gamma = 3$ however fails under $\gamma = \frac{1}{3}$ for polynomials of degrees 3 and 5. The reason is that parametric

approximations of value function do not necessarily satisfy the property of monotonicity; see Judd (1998, p. 438) for a discussion. We do not resolve this issue because it is related to interpolation and not to precomputation of integrals.[4] Whenever our approximation of value function is monotone, the approximation errors decrease with each polynomial degree by one or more order of magnitude. High-degree polynomial solutions are fairly accurate, in particular, for the fifth-degree polynomial, the errors are of order $10^{-5} - 10^{-6}$.

The results for the global Euler equation algorithm are provided in Table 2.

Table 2: Accuracy and speed of the Euler equation algorithm.[a]

Polynomial	$\gamma = 1/3$			$\gamma = 1$			$\gamma = 3$		
degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
1st	-3.59	-3.37	0.74	-3.39	-3.24	1.20	-2.98	-2.68	1.90
2nd	-5.00	-4.49	0.74	-4.64	-4.21	1.16	-3.91	-3.53	1.84
3rd	-5.98	-5.54	0.54	-5.68	-5.19	0.72	-4.81	-4.31	1.48
4th	-7.24	-6.69	0.48	-6.83	-6.18	0.73	-5.79	-5.07	1.23
5th	-8.44	-7.89	0.27	-8.01	-7.32	0.40	-6.63	-5.85	0.91

[a] Notes: L_1 and L_∞ are, repectively, the average and maximum of absolute approximation errors across optimality condition and test points (in log10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); γ is the coefficient of risk aversion.

The performance of this algorithm is superior to that of the Bellman equation algorithm: it is successful in finding the solutions in all cases and produces much smaller approximation errors (of order $10^{-6} - 10^{-8}$). The accuracy is higher because the Euler equation algorithm parameterizes the derivative of the value function while the Bellman equation algorithm parameterizes the value function itself and effectively, loses one polynomial degree.

Finally, in Appendix D, we illustrate precomputation of integrals in the context of a model with up to 20 heterogeneous agents that was studied in Judd et al. (2011) using a generalized stochastic simulation algorithm (GSSA). In the context of GSSA, Judd et al. (2011) found that there is a significant trade-off between accuracy and speed depending on the integration method used (Monte Carlo, Gaussian quadrature and monomial integration methods). Precomputation of integrals effectively eliminates the above trade-off. Given that we need to compute integrals just once, we can do it very accurately at a relatively low (fixed) cost. As a result, GSSA based on precomputation of integrals possesses two valuable features – high accuracy of solutions (characteristic for the versions of GSSA based on accurate but expensive multipoint Gaussian quadrature and monomial rules)

[4] Two techniques can ensure monotonicity of parametric approximations. One is to impose restrictions on the derivatives of the estimated functions; this is easy to do under linear programming methods described in Judd et al. (2011). The other is to use parametric functions that preserve shape; see Judd (1998, p. 440).

and a low cost (characteristic for versions of GSSA based on cheap but far less accurate Monte Carlo methods).

8 Conclusion

Many existing solution methods in the literature rely on parametric functions that satisfy the assumption of separability used in the present paper. For such methods, we can precompute integrals in the stage of initialization. The resulting transformed stochastic problem has the same computational complexity as a similar deterministic problem. Our technique of precomputation of integrals is very general and can be applied to essentially any set of equations that contains conditional expectations. Precomputation of integrals can save programming efforts, reduce a computational burden and increase accuracy of solutions. It is of special value in computationally intense applications.

References

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

[2] Barillas, F. and J. Fernandez-Villaverde, (2007). A generalization of the endogenous grid method. Journal of Economic Dynamics and Control, Elsevier 31, 2698-2712.

[3] Carroll, K. (2005). The method of endogenous grid points for solving dynamic stochastic optimal problems, Economic letters 91, 312-320.

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

[5] Den Haan, W., (2010), Comparison of solutions to the incomplete markets model with aggregate uncertainty. Journal of Economic Dynamics and Control 34, 4–27.

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

[7] Geweke, J., (1996). Monte Carlo simulation and numerical integration. In Handbook of Computational Economics (H. Amman, D. Kendrick, and J. Rust, eds.), Amsterdam: Elsevier Science, pp. 733-800.

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

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

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

[11] Judd, K., L. Maliar and S. Maliar, (2011). Numerically stable and accurate stochastic simulation approaches for solving dynamic models. Quantitative Economics 2, 173-210.

[12] 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.

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

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

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

[16] Marcet, A. (1988). Solving non-linear models by parameterizing expectations. Unpublished manuscript, Carnegie Mellon University, Graduate School of Industrial Administration.

[17] Marcet, A., and G. Lorenzoni (1999). The parameterized expectation approach: some practical issues. In Computational Methods for Study of Dynamic Economies (R. Marimon and A. Scott, eds.). Oxford University Press, New York, pp. 143-171.

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

[19] Miranda, M. and P. Fackler, (2002). Applied Computational Economics and Finance. Cambridge: MIT Press.

[20] Miranda, M. and P. Helmberger (1988). The effects of commodity price stabilization programs. American Economic Rewiev 78, 46-58.

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

[22] Rust, J., (2008). Dynamic programming. In The New Palgrave Dictionary of Economics (S. Durlauf and L. Blume eds.), Palgrave Macmillan.

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

[24] Stachursky, J., (2009). Economic Dynamics: Theory and Computation. Cambridge: MIT Press.

[25] Tauchen, G., (1986). Finite state Markov chain approximations to univariate and vector autoregressions. Economic Letters 20, 177-181.

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

Supplement to "How to Solve Dynamic Stochastic Models Computing Expectations Just Once": Appendices

Kenneth L. Judd
Lilia Maliar
Serguei Maliar

Appendix A: Precomputation of integrals in the Bellman and Euler equations in a general setup

In this section, we show how to precompute integrals in the Bellman and Euler equations in a general setup. We focus on a class of stochastic infinite-horizon optimization problems in which both state and control variables take a continuum of possible values.

Precomputing integrals in the Bellman equation We assume that the problem has a recursive formulation, so that its solution satisfies the Bellman equation in the relevant area of the state space

$$V(x,z) = \max_{x',y} \{R(y) + \beta E[V(x',z')|x,z]\}, \tag{26}$$

$$\text{s.t} \quad G(y,x,z,x') = 0, \tag{27}$$

$$z' = Z(z,\epsilon'), \tag{28}$$

where $x \in \mathbb{R}^{n_x}$, $z \in \mathbb{R}^{n_z}$ and $y \in \mathbb{R}^{n_y}$ are vectors of endogenous state variables, exogenous state variables and control variables, respectively; the return function $R : \mathbb{R}^{n_y} \to \mathbb{R}$ is continuously differentiable, strictly increasing and strictly concave; $G : \mathbb{R}^{n_y+2n_x+n_z} \to \mathbb{R}^G$ is a vector-valued function which is continuously differentiable in y, y', x, x'; the set $\{(x,x') : G(y,x,z,x') = 0\}$ is convex and compact; the vector of disturbances $\epsilon' \in \Omega \subseteq \mathbb{R}^{n_\epsilon}$ has a probability distribution which is known at present; z' in (28) has finite moments; and x' is known at present.

A solution is a value function $V(x,z)$ (and the corresponding set of policy functions $x' = X(x,z)$ and $y = Y(x,z)$) that satisfies (26)–(28) in the relevant area of the state space. We assume that the functions R, G and Z satisfy jointly a set of regularity conditions that ensures that the solution exists and is unique. To simplify the presentation, we did not include x and z in R, and we did not include y' and z' in G, however, the technique of precomputation of integrals applies to these cases as well.

Under the result of Proposition 1, precomputation of integrals leads us to the following Bellman equation,

$$\widehat{V}(x,z;b) \doteq \max_{x',y} \left\{ R(y) + \beta \widehat{V}(x', Z(z,\mu);\theta) \right\} \tag{29}$$
$$\text{s.t. (23), (24), (27) and (28),}$$

where the approximating function $\widehat{V}(x,z;b)$ is of type (22).

Precomputing integrals in the Euler equation To derive first-order conditions, we use the Bellman equation. An interior solution to (26)–(28) satisfies

$$\frac{\partial R(y)}{\partial y_j} - \lambda \frac{\partial G(y,x,z,x')}{\partial y_j} = 0, \tag{30}$$

$$\beta E\left\{ \frac{\partial V(x',z')}{\partial x_i'} \right\} - \lambda \frac{\partial G(y,x,z,x')}{\partial x_i'} = 0, \tag{31}$$

where $y_j \in y$, $x_i \in x$, and $\lambda \in \mathbb{R}^G$ is a vector of Lagrange multipliers associated with the constraints (27). Furthermore, by the envelope theorem, we have

$$\frac{\partial V(x,z)}{\partial x_i} = -\lambda \frac{\partial G(y,x,z,x')}{\partial x_i}. \tag{32}$$

Taking (32) one period ahead and substituting the resulting condition into (31), we obtain a set of the Euler equations

$$\lambda \frac{\partial G(y,x,z,x')}{\partial x_i'} = -\beta E\left\{ \lambda' \frac{\partial G(y',x',z',x'')}{\partial x_i'} \right\}. \tag{33}$$

As in Section 4.2, we replace the integrands of (33) with new variables $\left(q_1', ..., q_{n_x}'\right)$,

$$-\lambda' \frac{\partial G(y'x',z',x'')}{\partial x_i'} \equiv q_i' = Q^i(x',z'). \tag{34}$$

Under the result of Proposition 1, we can precompute the integrals in the Euler equation (33) to obtain

$$\widehat{Q}^i(x,z;b) \doteq \beta \widehat{Q}^i(x', Z(z,\mu);\theta) \frac{\partial G(y,x,z,x')}{\partial x_i} \left[\frac{\partial G(y,x,z,x')}{\partial x_i'} \right]^{-1}, \tag{35}$$

where the approximating functions $\widehat{Q}^i(x,z;b)$, $i = 1, ..., n_x$, are of type (22).

Finally, for the general model (26)–(28), we have two results that were previously discussed in the context of the one-agent model of Sections 2–4. First, the supplementary

functions $Q^i(x,z)$, $i=1,...,n_x$, coincide with the corresponding derivatives of the value function $\frac{\partial[V(x,z)]}{\partial x_i}$. Second, the technique of precomputation of integrals can be extended to the case when we parameterize and approximate policy functions of variables other than q_i's. In particular, we can premultiply both sides of the i-th Euler equation (35) by the corresponding state variable x_i' and re-arrange the terms to obtain

$$x_i' \doteq \beta \frac{\widehat{Q}^i(x', Z(z,\mu);\theta)}{\widehat{Q}^i(x,z;b)} \frac{\partial G(y,x,z,x')}{\partial x_i} \left[\frac{\partial G(y,x,z,x')}{\partial x_i'}\right]^{-1} x_i'. \tag{36}$$

We can now solve for $\widehat{X}^i(x,z;v)$ jointly with $\widehat{Q}^i(x,z;b)$. Namely, we can iterate on the coefficients of the endogenous state variables v using the function $\widehat{Q}^i(x,z;b)$ as a tool for computing conditional expectations; in Appendices C and D, we illustrate this procedure by way of examples.

Appendix B: Bellman and Euler equation algorithms for solving the neoclassical stochastic growth model

In this section, we provide a description of the Bellman and Euler equation algorithms, which we use to solve the neoclassical stochastic growth model described in Section 2. Both algorithms use fixed-point iteration, FPI, for finding the coefficients of the approximating functions.

Algorithm 1–FPI (Bellman equation algorithm based on FPI)

- Initialization:

  (a) Parameterize the value function $V(k,a)$ using a flexible functional form $\widehat{V}(k,a;b)$ (we use the ordinary polynomials (8)).

  (b) Choose an initial guess on the coefficients vector $b^{(1)}$ (we use a low-accuracy linear solution).

  (c) Construct a grid $\{k_m, a_m\}_{m=1,...,M}$ to be used as a solution domain (we simulate the linear solution 10,000 periods forward, compute the minimum and maximum values of capital and productivity in the simulation, and split the obtained capital and productivity intervals into 10 equally spaced intervals to construct a product grid of 100 points).

  (d) Precompute L moments of $\exp(\epsilon')$ using (11) and construct the sequence $\{\mathcal{I}_0,...,\mathcal{I}_n\}$ using (12).

- Step 1. At iteration p, use $b^{(p)}$ to compute the values $\widehat{V}_1\left(k_m, a_m; b^{(p)}\right)$ in each grid point $m = 1, ..., M$. Compute the corresponding values of c_m using (21) and find k'_m using (14). Use the precomputation result (16), to infer the value function on the grid,

$$\mathcal{V}_m \equiv u\left(c_m\right) + \beta V\left(k'_m, a^{\rho}_m; \theta^{(p)}\right).$$

- Step 2. Find $\widehat{b}$ that minimizes the errors ε_m in the regression equation,

$$\mathcal{V}_m = \widehat{V}\left(k_m, a_m; b\right) + \varepsilon_m.$$

(We use a linear least-squares regression method based on QR factorization).

- Step 3. Check for convergence and end the solution procedure if

$$\frac{1}{M}\sum_{m=1}^{M}\left|\frac{\left(k'_m\right)^{(p)} - \left(k'_m\right)^{(p-1)}}{\left(k'_m\right)^{(p)}}\right| < 10^{-11},$$

where $\left\{\left(k'_m\right)^{(p)}\right\}_{m=1,...,M}$ and $\left\{\left(k'_m\right)^{(p-1)}\right\}_{m=1,...,M}$ are the capital choices obtained on iterations p and $p-1$, respectively, and $|\cdot|$ denotes absolute value.

- Step 4. Compute $b^{(p+1)}$ for iteration $p+1$ using fixed-point iteration with damping

$$b^{(p+1)} = \left(1 - \xi\right) b^{(p)} + \xi\widehat{b},$$

where the damping parameter is $\xi = 0.1$. Go to Step 1.

Algorithm 2–FPI (Euler equation algorithm based on FPI)

- Initialization:

(a) Parameterize the function $Q\left(k, a\right)$ using a flexible functional form $\widehat{Q}\left(k, a; b\right)$ (we use the ordinary polynomials (8)).

(b)-(d) The same as in Algorithm 1–FPI.

- Step 1. At iteration p, use $b^{(p)}$ to compute the values $\widehat{Q}\left(k_m, a_m; b^{(p)}\right)$ in each grid point $m = 1, ..., M$. Compute the corresponding values of k'_m using (14). Find the values of q_m on the grid by combining the precomputation result (16) with the condition (18)

$$q_m = \beta\widehat{Q}\left(k'_m, a^{\rho}_m; \theta^{(p)}\right)\left[1 - \delta + af_1\left(k_m\right)\right].$$

- Step 2. Find $\widehat{b}$ that minimizes the errors ε_m in the regression equation,

$$q_m = \widehat{Q}\left(k_m, a_m; b\right) + \varepsilon_m.$$

(We use a linear least-squares regression method based on QR factorization).

- Steps 3 and 4. The same as in Algorithm 1–FPI.

Accuracy check We generate a new random draw of $10,200$ points and discard the first 200 points. At each point (k_τ, a_τ), we compute unit-free Euler equation error using a 10-node Gauss-Hermite quadrature rule,

$$\mathcal{E}(k_\tau, a_\tau) \equiv \sum_{j=1}^{J^{\text{test}}} \omega_j^{\text{test}} \left[\beta \frac{u'\left(c'_{\tau,j}\right)}{u'(c_\tau)} \left[1 - \delta + a_\tau^\rho \exp\left(\epsilon_j'^{\,\text{test}}\right) f'(k'_\tau)\right] \right] - 1,$$

where $\epsilon_j'^{\,\text{test}}$ and ω_j^{test} are the j-th integration node and weight; c_τ and $c'_{\tau,j}$ follow from the budget constraint (14).

Appendix C: Euler equation approach: alternative parameterizations

Depending on the application, some policy functions can be more convenient for the purpose of approximation than others (for example, because some functions are better behaved or preferable for certain computations).[5] We can use a simple transformation of the Euler equation that allows us to approximate any policy function. As an example, let us consider the capital policy function $k' = K(k, a)$. Premultiplying both sides of the Euler equation (7) by k' and rearranging the terms, we obtain an equivalent representation of the Euler equation (as long as $k' > 0$),

$$k' = \beta E \left[\frac{u_1(c')}{u_1(c)} \left(1 - \delta + a' f_1(k')\right) k' \right]. \tag{37}$$

We have expressed the variable k' in two ways: as an outcome of the policy function and as a conditional expectation of a random variable. We can compute the capital policy function as a fixed point of (37) by evaluating conditional expectation functions inside the iterative cycle as is done in Judd et al. (2010, 2011).

In this section, we show how to implement the iteration on (37) by precomputing integrals. Using the results of Section 4.2, we rewrite the Euler equation (37) in the form

$$k' \doteq \beta \frac{\widehat{Q}(k', a^\rho; \theta)}{\widehat{Q}(k, a; b)} \left[1 - \delta + a f'(k)\right] k'. \tag{38}$$

We parameterize the capital policy function $k' = K(k, a)$ with a flexible functional form $\widehat{K}(k, a; v)$ where v is the coefficients vector, and we solve for both functions $\widehat{K}(k, a; v)$ and $\widehat{Q}(k', a^\rho; \theta)$ simultaneously. We again proceed in two steps.

[5] For example, in the context of multi-country growth models, Maliar et al. (2011) argue that a capital policy function has less curvature than consumption and labor policy functions and hence, can be approximated more accurately. Moreover, knowing the capital policy function makes it possible to compute endogenous state variables in all grid points or integration nodes or time periods without solving for control variables which allows us to vectorize computations and to increases the speed in some applications.

Algorithm 3. Euler equation algorithm with capital-policy-function parameterization
Precomputation of integrals. Construct $\{\mathcal{I}_0, ..., \mathcal{I}_n\}$ using (11) and (12).
Transformed problem. Find v and b that solve (14), (15), (16), (17) and (38).

Below, we describe a version of Algorithm 3 that relies on FPI. We iterate only on the coefficients of the capital policy function v, and we use $\widehat{Q}(k, a; b)$ just as a supplementary tool for computing the expectations.

Algorithm 3–FPI (Euler equation algorithm with capital-policy-function parameterization based on FPI)

- Initialization:

  (a) Parameterize the policy functions $k' = K(k, a)$ and $q = Q(k, a)$ with flexible functional forms $\widehat{K}(k, a; v)$ and $\widehat{Q}(k, a; b)$, respectively (in both cases, we use the ordinary polynomials (8)).

  (b)-(d) The same as in Algorithm 1–FPI.

- Step 1. At iteration p, use $v^{(p)}$ to compute the values $k'_m = \widehat{K}\left(k_m, a_m; v^{(p)}\right)$ in each grid point $m = 1, ..., M$. Compute the corresponding values of c_m using (14) and of q_m using (17), namely,

$$q_m = u_1(c_m)\left[1 - \delta + a_m A f_1(k_m)\right].$$

- Step 2. Find $\widehat{b}$ that minimizes the errors ε_m in the regression equation,

$$q_m = \widehat{Q}(k_m, a_m; b) + \varepsilon_m.$$

  Find the values of $\widehat{Q}\left(k'_m, a^{\rho}_m; \widehat{\theta}\right)$ on the grid that correspond to $\widehat{Q}\left(k_m, a_m; \widehat{b}\right)$ using the precomputation result (16). Use (38) to compute the value of next-period capital on the grid, denoted by

$$y'_m \equiv \beta \frac{\widehat{Q}\left(k'_m, a^{\rho}_m; \widehat{\theta}\right)}{\widehat{Q}\left(k_m, a_m; \widehat{b}\right)} \left(1 - \delta + a f_1(k_m)\right) k'_m.$$

  Find $\widehat{v}$ that minimizes the errors ι_m in the regression equation,

$$y'_m = \widehat{K}(k_m, a_m; v) + \iota_m.$$

  (To implement both regressions, we use a linear least-squares regression method based on QR factorization).

- Steps 3 and 4. The same as in Algorithm 1–FPI.

Results In Table 3, we provide the results for the Euler equation method parameterizing the capital policy function (Algorithm 3–FPI).

Table 3: Accuracy and speed of the Euler equation algorithm (capital-policy-function parameterization) in the one-agent model.[a]

Polynomial	$\gamma = 1/3$			$\gamma = 1$			$\gamma = 3$		
degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
1st	-3.84	-3.58	0.16	-3.65	-3.32	0.09	-3.24	-2.76	0.05
2nd	-5.26	-4.66	0.14	-4.98	-4.35	0.07	-4.30	-3.61	0.04
3rd	-6.10	-5.80	0.11	-5.92	-5.39	0.05	-4.96	-4.16	0.03
4th	-7.38	-6.85	0.11	-7.09	-6.29	0.05	-5.59	-4.82	0.03
5th	-8.56	-7.99	0.06	-8.13	-7.28	0.03	-6.33	-5.67	0.03

[a] Notes: L_1 and L_∞ are, repectively, the average and maximum of absolute approximation errors across optimality condition and test points (in log10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds); γ is the coefficient of risk aversion.

The performance of this version of the Euler equation algorithm is somewhat better than that of the Euler equation algorithm parameterizing the function $Q(k, a)$. Namely, the accuracy of Algorithm 3–FPI is higher by about a half order of magnitude and the running time is significantly smaller than those of Algorithm 2–FPI. Presumably, this is because the capital policy function is close to linear and is easier to approximate numerically.

Appendix D: A multicountry analysis

In this section, we describe the multicountry model analyzed in Section 7. This model is identical to the one used in Judd et al. (2011) to assess the performance of the GSSA method.

The model A social planner maximizes a weighted sum of the expected lifetime utilities of N agents (interpreted as countries) subject to the aggregate resource constraint, i.e.,

$$\max_{\left\{c_t^h, k_{t+1}^h\right\}_{t=0,\ldots,\infty}^{h=1,\ldots,N}} E_0 \sum_{h=1}^{N} \tau^h \left(\sum_{t=0}^{\infty} \beta^t u^h \left(c_t^h\right) \right) \tag{39}$$

subject to

$$\sum_{h=1}^{N} c_t^h = \sum_{h=1}^{N} \left[a_t^h f^h \left(k_t^h\right) + k_t^h (1-\delta) - k_{t+1}^h \right], \tag{40}$$

where c_t^h, k_t^h, a_t^h, u^h, f^h and τ^h are consumption, capital, productivity level, utility function, production function and welfare weight of a country $h \in \{1, ..., N\}$, respectively; β is the discount factor; and δ is the depreciation rate. The initial condition $(\boldsymbol{k}_0, \boldsymbol{a}_0)$ is given, where $\boldsymbol{k}_t \equiv \left(k_t^1, ..., k_t^N\right)$ and $\boldsymbol{a}_t \equiv \left(a_t^1, ..., a_t^N\right)$. The process for productivity shocks in the country h is given by

$$\ln a_t^h = \rho \ln a_{t-1}^h + \epsilon_t^h, \tag{41}$$

where $\epsilon_t^h \equiv \epsilon_t + \varpi_t^h$ with $\epsilon_t \sim N(0, \sigma)$ and $\varpi_t^h \sim N(0, \sigma)$ being a common-for-all-countries and country-specific productivity shocks, respectively; ρ is the autocorrelation coefficient of the productivity shock; and σ is the standard deviation of the productivity shocks. Thus, the shocks follow a multivariate Normal distribution $\left(\epsilon_t^1, ..., \epsilon_t^N\right)^\top \sim \mathcal{N}(0_N, \Sigma)$ with $0_N \in \mathbb{R}^N$ being a vector of zero means and $\Sigma = \begin{pmatrix} 2\sigma^2 & ... & \sigma^2 \\ ... & ... & ... \\ \sigma^2 & ... & 2\sigma^2 \end{pmatrix} \in \mathbb{R}^{N \times N}$ being a variance-covariance matrix.

Optimality conditions An interior solution to the planner's problem (39)–(41) satisfies the following first-order conditions

$$\tau^h u_1^h\left(c_t^h\right) = \lambda_t, \tag{42}$$

$$\lambda_t = \beta E_t\left\{\lambda_{t+1}\left[1 - \delta + a_{t+1}^h f_1^h\left(k_{t+1}^h\right)\right]\right\}, \tag{43}$$

where λ_t is the Lagrange multiplier associated with the aggregate constraint (40). Also, the solution satisfies a set of transversality conditions, $\lim_{t \to \infty} \lambda_t k_{t+1}^h = 0$ for $h = 1, ..., N$.

Optimality conditions after precomputation of integrals Let us introduce a new variable q_{t+1}^h that represents the expression inside conditional expectation in (43), which implies

$$q_t^h \equiv \tau^h u_1^h\left(c_t^h\right)\left[1 - \delta + a_t^h f_1^h\left(k_t^h\right)\right], \tag{44}$$

where the condition (42) was used to eliminate λ_t. In terms of q_t^h and q_{t+1}^h, the Euler equation (43) is

$$\frac{q_t^h}{1 - \delta + a_t^h f_1^h\left(k_t^h\right)} = \beta E\left[q_{t+1}^h\right]. \tag{45}$$

We parameterize the function $q_t^h = Q^h(\boldsymbol{k}_t, \boldsymbol{a}_t)$ with a flexible functional form $\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)$, $h = 1, ..., N$. Under the results of Proposition 1, the integral $E\left[q_{t+1}^h\right]$ can be precomputed. If we use a complete ordinary polynomial, we need to compute conditional expectations of polynomial terms of type $E_t\left[\left(a_{t+1}^1\right)^{l^1}\left(a_{t+1}^2\right)^{l^2} ... \left(a_{t+1}^N\right)^{l^N}\right]$, where a power l^h on a_{t+1}^h can

take a value from 0 to L with L being the polynomial degree. Under (23) and (24), this requires us to approximate multidimensional integrals of type

$$E_t\left[\exp\left(l^1\epsilon_{t+1}^1 + l^2\epsilon_{t+1}^2 + ... + l^N\epsilon_{t+1}^N\right)\right] = \int_{-\infty}^{+\infty} ... \int_{-\infty}^{+\infty} \exp\left(l^1\epsilon_{t+1}^1 + l^2\epsilon_{t+1}^2 + ... + l^N\epsilon_{t+1}^N\right) d\epsilon_{t+1}^1 d\epsilon_{t+1}^2 .... d\epsilon_{t+1}^N. \quad (46)$$

There is a variety of numerical integration techniques in the literature that can be used for approximating the integral (46). See, for example, Judd et al. (2011) for a detailed description and examples of multidimensional integration methods including Gauss-Hermite quadrature product rules and non-product monomial formulas. Prior to integration, correlated random variables must be normalized using a Cholesky decomposition.

We next formulate a version of the Euler equation (45) in terms of $\widehat{Q}^h$,

$$\frac{\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)}{1 - \delta + a_t^h f_1^h\left(k_t^h\right)} \doteq \beta \widehat{Q}^h\left(\boldsymbol{k}_{t+1}, \boldsymbol{a}_t^\rho; \theta^h\right). \quad (47)$$

If the approximating functions $\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)$, $h = 1, ..., N$, are given by the ordinary polynomial (22), the coefficients vectors b^h and θ^h are related by conditions of type (10).

Policy functions to parameterize We search for a solution in the form of N capital policy functions $k_{t+1}^h = K^h\left(\boldsymbol{k_t}, \boldsymbol{a}_t\right)$, $h = 1, ..., N$. We parameterize each capital policy function by a flexible functional form $\widehat{K}^h\left(\boldsymbol{k_t}, \boldsymbol{a}_t; v^h\right)$, where v^h is the coefficients vector, and we rewrite (44) and (47) as

$$\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right) \doteq \tau^h u_1^h\left(c_t^h\right)\left[1 - \delta + a_t^h f_1^h\left(k_t^h\right)\right], \quad (48)$$

$$k_{t+1}^h \doteq \beta \frac{\widehat{Q}^h\left(\boldsymbol{k}_{t+1}, \boldsymbol{a}_t^\rho; \theta^h\right)}{\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)}\left[1 - \delta + a_t^h f_1^h\left(k_t^h\right)\right] k_{t+1}^h. \quad (49)$$

In the transformed Euler equation (48), the effect of uncertainty on the solution is again captured by a set of precomputed integrals (46) that determines the relation between b^h and θ^h.

Algorithm 4. Euler equation algorithm for the multicountry model
Precomputation of integrals. Construct $\{\mathcal{I}_0, ..., \mathcal{I}_n\}$ using (46).
Transformed problem. Find v^h and b^h, $h = 1, ..., N$ that solve (10), (39), (40), (41), (48) and (49).

We next describe a version of Algorithm 4 based on stochastic simulation and FPI. As in the case of Algorithm 3–FPI used to solve the one-agent model, we will iterate on $\widehat{K}^h\left(\boldsymbol{k_t}, \boldsymbol{a}_t; v^h\right)$ and use $\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)$, $h = 1, ..., N$, just as a supplementary tool for computing conditional expectations.

Algorithm 4 –FPI (Euler equation algorithm based on stochastic simulation and FPI)

- Initialization:

  (a) Parameterize $k_{t+1}^h = K^h(\boldsymbol{k}_t, \boldsymbol{a}_t)$ and $q_t^h = Q^h(\boldsymbol{k}_t, \boldsymbol{a}_t)$ with flexible functional forms $\widehat{K}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; v^h\right)$ and $\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)$, respectively (in both cases, we use the ordinary polynomials (8)).

  (b) Choose an initial guess on the coefficients vectors $(v^1)^{(1)}, ..., \left(v^h\right)^{(1)}$ (we use a deterministic steady state).

  (c) Fix the simulations length T and the 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 the equation (41) (we use $T = 2,000$ and $(\boldsymbol{k}_0, \boldsymbol{a}_0) = (\mathbf{1}, \mathbf{1})$ where $\mathbf{1} \equiv (1, ..., 1) \in \mathbb{R}^N$).

  (d) Precompute the moments $\{e_1, ..., e_L\}$ in (46) and construct the sequence $\{\mathcal{I}_0, ..., \mathcal{I}_n\}$ (we use a monomial integration formula with $2N^2 + 1$ integration nodes, as described in Judd et al., 2011).

- Step 1. At iteration p, use the assumed capital policy functions $k_{t+1}^h = \widehat{K}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; v^h\right)$, $h = 1, ..., N$, to recursively calculate a sequence of capital stocks $\{\boldsymbol{k}_{t+1}\}_{t=0,...,T}$ 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 $\left\{c_t^h\right\}_{t=0,...,T}$, $h = 1, ..., N$, satisfying (40) and (42).

- Step 3. For each country $h = 1, ..., N$, compute q_t^h using (44). Find $\widehat{b}^h$ that minimizes the errors ε_t^h in the regression equation,

$$q_t^h = \widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right) + \varepsilon_t^h.$$

  Find the values of $\widehat{Q}^h\left(\boldsymbol{k}_{t+1}, \boldsymbol{a}_t^\rho; \widehat{\theta}^h\right)$ that correspond to $\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; \widehat{b}^h\right)$ using the precomputation results (10). Compute the right side of (49) at $t = 0, ..., T-1$,

$$y_t^h \equiv \beta \frac{\widehat{Q}^h\left(\boldsymbol{k}_{t+1}, \boldsymbol{a}_t^\rho; \theta^h\right)}{\widehat{Q}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; b^h\right)} \left[1 - \delta + a_t^h f_1^h\left(k_t^h\right)\right] k_{t+1}^h.$$

- Step 4. For each country $h = 1, ..., N$, find $\widehat{v}^h$ that minimizes the errors ι_t^h in the regression equation,

$$y_t^h = \widehat{K}^h\left(\boldsymbol{k}_t, \boldsymbol{a}_t; v^h\right) + \iota_t^h.$$

- Step 5. For each country $h = 1, ..., N$, compute the coefficients vectors for the subsequent iteration $p + 1$ using fixed-point iteration,

$$\left(v^h\right)^{(p+1)} = (1 - \xi)\left(v^h\right)^{(p)} + \xi\left(\widehat{v}^h\right)^{(p+1)}, \tag{50}$$

where $\xi = 0.1$ is a damping parameter.

Iterate on Steps 1–5 until convergence,

$$\frac{1}{T \cdot N}\sum_{t=1}^{T}\sum_{h=1}^{N}\left|\frac{\left(k_{t+1}^h\right)^{(p)} - \left(k_{t+1}^h\right)^{(p-1)}}{\left(k_{t+1}^h\right)^{(p)}}\right| < 10^{-7}\xi, \tag{51}$$

where $\left(k_{t+1}^h\right)^{(p)}$ and $\left(k_{t+1}^h\right)^{(p-1)}$ are the h-th country's capital stocks obtained on iterations, p and $p - 1$, respectively.

Parameterization of the model and accuracy evaluation To solve the model, we assume $u^h\left(c_t^h\right) = \ln c_t^h$, $f\left(k_t\right) = Ak_t^\alpha$ with $\alpha = 0.36$, $\beta = 0.99$, $\delta = 0.025$, $\rho = 0.95$ and $\sigma = 0.01$. We assume that the planner assigns equal weights to all countries $\tau^h = 1$, which implies that all countries have an identical consumption, $c_t^h = c_t$ for all $h = 1, .., N$. To evaluate the accuracy of solutions, we report unit-free Euler equation errors on a simulation of 10,000 observations. To approximate integrals in our accuracy checks, we use a monomial integration rule with $2N^2 + 1$ nodes as in Judd et al. (2011).

Results In Table 4, we provide the results produced by the Euler equation algorithm based on stochastic simulation and FPI (Algorithm 4–FPI).

Table 4: Accuracy and speed of the Euler equation algorithm (capital-policy-function parameterization) in the multicountry model.[a]

Polynomial	$N = 2$			$N = 4$			$N = 6$			$N = 20$		
degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
1st	-4.45	-3.01	10	-4.41	-3.04	27	-4.36	-2.98	35	-4.53	-3.06	76
2nd	-5.71	-3.87	80	-5.44	-3.73	123	-5.34	-3.74	214	-5.34	-3.80	5352
3rd	-6.87	-4.94	32	-6.20	-4.41	455	-6.24	-4.35	2555	-	-	-
4th	-7.78	-5.64	105	-7.23	-5.14	3317	-	-	-	-	-	-
5th	-7.54	-5.36	277	-	-	-	-	-	-	-	-	-

[a] Notes: L_1 and L_∞ are, respectively, the average and maximum of absolute approximation errors across optimality conditions and test points (in log10 units). CPU is the time necessary for computing a solution (in seconds); N is the number of heterogeneous countries.

As is seen from the table, we obtain highly accurate solutions: the maximum approximation error ranges from $10^{-5.36}$ for the model with $N = 2$ to $10^{-3.8}$ for the model with

$N = 20$. These accuracy levels are comparable to those attained by the versions of GSSA that use accurate integration methods; see Judd et al. (2011, Table 5). The difference between the version of GSSA introduced in Judd et al. (2011) and the one studied in the present paper is that in the former case, integrals are re-computed in each iteration while in the latter case, integrals are precomputed. As a result, our cost of finding solutions is generally much lower than that in Judd et al. (2011). For example, it takes us about 43 minutes (2555 seconds) to compute a polynomial approximation of degree 3 for the model with $N = 6$. For comparison, the version of GSSA in Judd et al. (2011) needs about 18 hours (65,663 seconds) to find a solution of comparable accuracy; see Judd et al. (2011, Table 5).