Quantitative Economics 8 (2017), 851–893 1759-7331/20170851

How to solve dynamic stochastic models computing expectations just once

Kenneth L. Judd
Hoover Institution, Stanford University

Lilia Maliar
Department of Economics, Stanford University

Serguei Maliar
Department of Economics, Santa Clara University

Inna Tsener
Department of Applied Economics, University of the Balearic Islands

We introduce a computational technique—*precomputation of integrals*—that makes it possible to construct conditional expectation functions in dynamic stochastic models in the initial stage of a solution procedure. This technique is very general: it works for a broad class of approximating functions, including piecewise polynomials; it can be applied to both Bellman and Euler equations; and it is compatible with both continuous-state and discrete-state shocks. In the case of normally distributed shocks, the integrals can be constructed in a closed form. After the integrals are precomputed, we can solve stochastic models as if they were deterministic. We illustrate this technique using one- and multi-agent growth models with continuous-state shocks (and up to 60 state variables), as well as Aiyagari's (1994) model with discrete-state shocks. Precomputation of integrals saves programming efforts, reduces computational burden, and increases the accuracy of solutions. It is of special value in computationally intense applications. MATLAB codes are provided.

Keywords. Dynamic model, precomputation, numerical integration, dynamic programming, value function iteration, Bellman equation, Euler equation, envelope condition method, endogenous grid method, Aiyagari model.

JEL classification. C61, C63, C68.

Kenneth L. Judd: kennethjudd@mac.com
Lilia Maliar: maliarl@stanford.edu
Serguei Maliar: smaliar@scu.edu
Inna Tsener: inna.tcener@uib.es

We are indebted to the editor and three anonymous referees for many thoughtful comments and suggestions. Errors are ours. Support from the Hoover Institution and Department of Economics at Stanford University, University of Alicante, University of the Balearic Islands, Santa Clara University, and the MINECO/FEDER Grant ECO2015-70540-P is gratefully acknowledged.

DOI: 10.3982/QE329

1. Introduction

Existing global methods for solving dynamic stochastic models compute conditional expectation functions in their iterative cycles.[1] Recomputing expectation functions in each iteration is costly and the cost grows rapidly (i) when the number of random variables increases (because the dimensionality of integrals increases), (ii) when more accurate integration methods are used (because the number of integration nodes increases), and (iii) when models become more complex (because numerical solvers are used more intensively, and this involves additional evaluations of integrals).

In this paper, we introduce a computational technique that makes it possible to construct conditional expectation functions in the initial stage of the solution procedure; we refer to this technique as *precomputation of expectation functions* or *precomputation of integrals*. The idea is simple and can be seen from the following example: Assume that the value function of a stylized growth model is approximated using a linear polynomial function $V(k,z) \approx b_0 + b_1 k + b_2 z$, where b_0, b_1, and b_2 are polynomial coefficients, k is capital, and z is productivity that follows a first-order autoregressive process, $z' = z^\rho \exp(\varepsilon')$, with $\rho \in (-1,1)$ and ε' being a random shock. The key step of our precomputation analysis is to notice that expectation of an ordinary polynomial function can be derived in a closed form as $E[b_0 + b_1 k' + b_2 z'] = b_0 + b_1 k' + b_2 z^\rho \mathcal{I}$, where $\mathcal{I} \equiv E[\exp(\varepsilon')]$. Given a distribution function of ε', the integral $\mathcal{I}$ can be constructed either analytically or numerically. In particular, it can be constructed analytically in the case of normally distributed shocks, used by a vast majority of economic models, namely, for $\varepsilon' \sim \mathcal{N}(0,\sigma^2)$, we have $\mathcal{I} = \exp(\frac{\sigma^2}{2})$. Importantly, we need to construct $\mathcal{I}$ just once, in the stage of initialization. Within the main iterative cycle on bs, expectation functions can be evaluated by using a closed-form expression that contains no random variables, that is, $E[V(k',z')] \approx b_0 + b_1 k' + b_2 z^\rho \mathcal{I}$. In effect, precomputation of integrals allows us to solve a stochastic problem as if it was a deterministic problem.

In our example, the advantage of using precomputation of integrals is twofold: First, we attain higher accuracy of numerical solutions because we construct integrals exactly, whereas the related literature constructs integrals approximately, by using some numerical integration method (e.g., Monte Carlo, quasi-Monte Carlo, quadrature, monomials).[2] Second, we are able to reduce the cost of constructing numerical solutions because in the main iterative cycle we evaluate future value function in just one composite future state, whereas the existing global solution methods approximate future value function as a weighted average across a possibly large number of future states.

Of course, our example is very special. However, it turns out that integrals can be precomputed in a variety of other contexts: First, precomputation of integrals can be implemented for any set of equations that contain conditional expectation functions, including the Bellman and Euler equations. Second, integrals can be precomputed not only

[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), Kollmann, Maliar, Malin, and Pichler (2011), and Maliar and Maliar (2014).

[2] See Judd, Maliar, and Maliar (2017) for a discussion of alternative accuracy measures of numerical solutions.

for ordinary polynomial functions but also for any other approximating families whose bases are separable in endogenous and exogenous state variables, including orthogonal polynomial families such as Chebyshev, Smolyak, Hermite, and piecewise polynomial functions, as well as many nonpolynomial families.[3] Third, precomputation of integrals can be combined with other computational techniques used by existing global solution methods, including a variety of solution domains, integration rules, fitting methods, and iterative schemes for finding unknown parameters of approximating functions. Fourth, precomputation of expectation functions is also possible for models with a discrete set of shocks and a discrete set of controls. Fifth, integrals can be computed analytically not only for univariate, but also for multivariate normally distributed shocks, including the case when shocks are correlated. Finally, in those cases when integrals cannot be constructed analytically, we can construct them numerically using very accurate methods since they should be constructed just once (i.e., this is a one time fixed cost).

We emphasize that precomputation of integrals is not a new solution method but an analytical and numerical manipulation of the model's equations that simplifies the construction of conditional expectation functions. Moreover, precomputation of integrals is not related to any specific solution method: any numerical solution method that can be applied to solve the original model's equations can also be applied to solve the model's equations, obtained after precomputing the integrals. In particular, we show that the precomputation technique can enhance the performance of five existing solution methods: conventional value function iteration, the endogenous grid method of Carroll (2006), the envelope condition method of Maliar and Maliar (2013), and two versions of the Euler equation methods. In the context of the stylized one-agent neoclassical growth model, we show that precomputation of integrals reduces the running time of the value-iterative and Euler equation methods up to three and five times, respectively, depending on the degree of the polynomial approximation and the specific solution method considered. Furthermore, we show that precomputation of integrals can be implemented in more complex models such as a growth model with elastic labor supply. MATLAB codes are provided in a supplementary file on the journal website, http://qeconomics.org/supp/329/code_and_data.zip. It is noteworthy that precomputation of integrals leads to much larger gains in terms of accuracy and speed in models with multiple shocks than in models with one shock. We solve a multicountry growth model with up to 30 heterogeneous countries (60 state variables, including 30 correlated shocks) using a generalized stochastic simulation algorithm (GSSA) in line with Judd, Maliar, and Maliar (2011). In the latter paper, integrals are approximated numerically using deterministic integration methods such as a Gauss Hermite product rule and two monomial integration methods; for these methods, the number of integration nodes grows with the number of shocks exponentially, quadratically, and linearly, respectively. In contrast, precomputation of integrals always means just one integration node. We find that precomputation of integrals can reduce the running time by many orders of magnitude with multivariate shocks compared to numerical approximations

[3]See Judd (1998) for a survey of polynomial approximating functions. Also, see Krueger and Kubler (2004), and Judd, Maliar, Maliar, and Valero (2014) for a discussion of Smolyak approximating functions.

of integrals. In particular, in the model with 30 countries, only the solution method using precomputation of integrals was able to deliver accurate second-order numerical solutions, while similar solution methods, which use numerical approximations of integrals, were too expensive.

Finally, we show that expectation functions can be also precomputed in models with a discrete set of shocks. As an illustration, we use Aiyagari's (1994) model in which decision functions are parameterized by piecewise linear polynomial functions. Here, the reduction in running time depends critically on the number of states in the associated Markov chain for exogenous shocks. In our baseline case of a seven-state Markov chain, the running time is reduced by around 50%, but far larger reductions in cost are possible when the number of states increases. Our analysis suggests that the gains from precomputation of expectation functions will be especially large in models with multiple shocks in which Markov chains are characterized by a very large number of states.

In the case of value-iterative methods, precomputation of integrals is always beneficial: it does not affect the way in which value function iteration is implemented; it just makes it faster. In the case of the Euler equation methods, precomputation has benefits but may also have costs. To precompute expectation functions, we must reparameterize the Euler equation in a particular way and we must construct a policy function for a specific variable that is the integrand of the expectation function in the Euler equation.[4] In some applications, it could happen that this new policy function is harder to approximate accurately than conventional policy functions for capital or consumption. We observe this effect in the context of Aiyagari's (1994) model: an algorithm iterating on the reparameterized Euler equation delivers slightly less accurate solutions than a similar algorithm iterating on the conventional Euler equation because the integrand has a spike in the area of the kink. However, such an accuracy reduction is very minor, even in this special case.

The rest of the paper is as follows. In Section 2, we introduce the technique of precomputation of integrals in the context of an optimal growth model with one continuous-state shock. In Section 3, we show the precomputation results for models with multivariate continuous-state shocks. In Section 4, we show how to precompute expectation functions in models with discrete-state shocks. In Section 5, we describe possible generalizations of the precomputation technique, and we discuss some of its limitations. Finally, in Section 6, we conclude. Appendices are available in a supplementary file on the journal website, http://qeconomics.org/supp/329/supplement.pdf.

2. Univariate continuous-state expectations

We show the technique for precomputing univariate (one-dimensional) continuous-state integrals in the context of the standard one-agent neoclassical stochastic growth model.

[4] It turns out that the reparameterization of the Euler equation that is necessary for precomputing integrals leads to the same system of equations that does the version of the envelope condition method that iterates on the derivative of the value function (ECM-DVF); see Maliar and Maliar (2013) and Arellano, Maliar, Maliar, and Tsyrennikov (2016). Thus, our analysis suggests that the ECM method produces systems of equations that are suitable for precomputation of integrals by construction.

2.1 *Neoclassical stochastic growth model*

The representative agent solves

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

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

$$\ln z_{t+1} = \rho \ln z_t + \varepsilon_{t+1}, \tag{3}$$

where (k_0, z_0) is given, E_t is an operator of conditional expectation, c_t, k_t, and z_t are consumption, capital, and productivity level, respectively, $\beta \in (0,1)$, $\delta \in (0,1]$, and $\rho \in (-1,1)$ are parameters, $\varepsilon_{t+1} \sim \mathcal{N}(0,\sigma^2)$ is a productivity shock, and u and f are the utility and production functions, respectively, both of which are strictly increasing, continuously differentiable, and concave.

Bellman equation We can characterize a solution to the model (1)–(3) by using a dynamic programming approach. The Bellman equation is

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

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

$$\ln z' = \rho \ln z + \varepsilon', \tag{6}$$

where $\varepsilon' \sim \mathcal{N}(0,\sigma^2)$ and $E[V(k',z')] \equiv E[V(k',z')|k,z]$ is expectation conditional on state (k,z); here and later in the text, the primes on variables denote their next-period values. We solve for value function $V(k,z)$ that satisfies (4)–(6).

Euler equation We can also characterize a solution to (1)–(3) by a set of the first-order conditions. The Euler equation is

$$u'(c) = \beta E[u'(c')(1-\delta + z' f'(k'))]. \tag{7}$$

We solve for the policy functions such as $c = C(k,z)$ and $k' = K(k,z)$ that satisfy (5), (6), and (7).

Numerical approximation of expectation functions (integrals) To implement iteration on Bellman and Euler equations, we must construct and evaluate expectation functions, $E[V(k',z')]$ and $E[u'(c')(1-\delta+z'f'(k'))]$, respectively. The related literature approximates expectation functions using numerical integration methods, such as Monte Carlo, quasi-Monte Carlo, quadrature, and monomial. All such methods approximate the value of an integral of a given function G by a weighted sum of the integrand function evaluated in a finite number of nodes,

$$E(G(\varepsilon')) = \int_{-\infty}^{+\infty} G(\varepsilon')\omega(\varepsilon')\,d\varepsilon' \approx \sum_{j=1}^{J} w_j G(\varepsilon'_j), \tag{8}$$

where $\omega(\cdot)$ is a probability density function of ε', and $\{\varepsilon'_j\}$ and $\{w_j\}$ are the integration nodes and weights, respectively, $j = 1, \ldots, J$. However, integration methods differ in the number and placement of the integration nodes and in the choice of integration weights. Typically, there is a trade-off between the accuracy and cost: integration formulas with more nodes (and thus, with a higher evaluation cost) lead to more accurate approximations. Importantly, all the existing solution methods recompute expectation functions using (8) each time when they are evaluated in the solution procedure, which involves a substantial cost. We will show that this cost can be significantly reduced, while attaining the highest possible accuracy in approximation of integrals.

2.2 *Precomputation of univariate integrals under ordinary polynomial approximations*

In this section, we show that for a class of ordinary polynomial functions, conditional expectation functions can be constructed prior to solving the model, that is, precomputed. Our precomputation analysis builds on the following fact: If a state-contingent function is parameterized by an ordinary polynomial function, then its conditional expectation function can be analytically characterized for any vector of the polynomial coefficients and any state of the world. (We had shown this result for a linear polynomial function in the Introduction, and we now generalize this result for polynomials of higher degrees.)

Let a function $\mathcal{P}(k, z)$ be approximated with a complete ordinary polynomial function, that is,

$$\mathcal{P}(k, z; b) = b_0 + b_1 k + b_2 z + b_3 k^2 + b_4 kz + b_5 z^2 + \cdots + b_n z^L, \tag{9}$$

where $b \equiv (b_0, b_1, \ldots, b_n) \in \mathbb{R}^{n+1}$ is a vector of polynomial coefficients and L is a degree of polynomial. Taking into account that $k' = K(k, z)$ is known at present and that future productivity depends on a random draw $z' = z^\rho \exp(\varepsilon')$, we can represent conditional expectation of $\mathcal{P}(k', z'; b)$ as

$$\begin{aligned}
&E\big[\mathcal{P}\big(k', z'; b\big)\big] \\
&\quad = E\big[b_0 + b_1 k' + b_2 z^\rho \exp\big(\varepsilon'\big) + b_3\big(k'\big)^2 + b_4 k' z^\rho \exp\big(\varepsilon'\big) + \cdots + b_n z^{L\rho} \exp\big(L\varepsilon'\big)\big] \\
&\quad = b_0 + b_1 k' + b_2 z^\rho E\big[\exp\big(\varepsilon'\big)\big] + b_3\big(k'\big)^2 + b_4 k' z^\rho E\big[\exp\big(\varepsilon'\big)\big] + \cdots \\
&\qquad + b_n z^{L\rho} E\big[\exp\big(L\varepsilon'\big)\big] \\
&\quad = b_0 \mathcal{I}_0 + b_1 \mathcal{I}_1 k' + b_2 \mathcal{I}_2 z^\rho + b_3 \mathcal{I}_3\big(k'\big)^2 + b_4 \mathcal{I}_4 k' z^\rho + \cdots + b_n \mathcal{I}_n z^{L\rho} \\
&\quad = b'_0 + b'_1 k' + b'_2 z^\rho + b'_3\big(k'\big)^2 + b'_4 k' z^\rho + \cdots + b'_n z^{L\rho} \\
&\quad \equiv \mathcal{P}\big(k', z^\rho; b'\big),
\end{aligned} \tag{10}$$

where $b' \equiv (b'_0, b'_1, \ldots, b'_n) \in \mathbb{R}^{n+1}$. The coefficients b'_i and b_i, for $i = 0, 1, \ldots, n$, are related by

$$b'_i = b_i \mathcal{I}_i, \tag{11}$$

where

$$\mathcal{I}_i = E\big[\exp\big(l_i\varepsilon'\big)\big], \tag{12}$$

with l_i being a power on z' in the ith monomial term. For example, z does not enter in the monomial terms $i = 0, 1, 3, 6$, so $l_0 = l_1 = l_3 = l_6 = 0$ and the corresponding $\mathcal{I}$s are $\mathcal{I}_0 = \mathcal{I}_1 = \mathcal{I}_3 = \mathcal{I}_6 = 1$; z does enter linearly in monomial terms $i = 2, 4, 7$, so that we have $l_2 = l_4 = l_7 = 1$ and $\mathcal{I}_2 = \mathcal{I}_4 = \mathcal{I}_7 = E[\exp(\varepsilon')]$; similarly, we obtain $l_5 = l_8 = 2$ and $\mathcal{I}_5 = \mathcal{I}_8 = E[\exp(2\varepsilon')], \ldots$, and $\mathcal{I}_n = E[\exp(L\varepsilon')]$.

The integrals $\mathcal{I}_0, \ldots, \mathcal{I}_n$ depend only on the properties of exogenous shock ε' and can be computed up-front without specifying the values of the coefficients b (which are unknown before the model is solved), that is, the integrals can be precomputed. Once $\mathcal{I}$s are constructed, evaluation of conditional expectation becomes trivial in the iterative procedure. Namely, conditional expectation of a polynomial function is given by the same polynomial function but evaluated at a different coefficients' vector, that is, $E[\mathcal{P}(k', z'; b)] = \mathcal{P}(k', z^\rho; b')$, where a relation between b' and b is determined by (11) and (12). In particular, for the case of normally distributed innovations $\varepsilon' \sim \mathcal{N}(0, \sigma^2)$, the integrals (12) can be computed exactly (in a closed form); we do this in Section 2.4. For those probability distributions for which analytical characterizations are not possible, integrals can be precomputed numerically very accurately, since we need to construct them just once at the beginning of the iterative cycle (i.e., this is a one time fixed cost).

Remark 1. The precomputation result (11) and (12) also holds for piecewise polynomial approximations. For example, let us parameterize $\mathcal{P}(k, z)$ on a rectangular grid $[k_1, \ldots, k_M] \times [z_1, \ldots, z_N]$ by using a collection of piecewise linear polynomial functions. Then, in each local area $[k_m, k_{m+1}] \times [z_n, z_{n+1}]$, we have a local polynomial function

$$\mathcal{P}_{[k_m, k_{m+1}]\times[z_n, z_{n+1}]}(k, z) = \beta_0^{(m,n)} + \beta_1^{(m,n)} k + \beta_2^{(m,n)} z,$$

where $n \in \{1, \ldots, N\}$ and $m \in \{1, \ldots, M\}$. The expectation function in each local area is given by

$$E\big[\mathcal{P}_{[k_m, k_{m+1}]\times[z_n, z_{n+1}]}\big(k', z'\big)\big] = \beta_0^{(m,n)} + \beta_1^{(m,n)} k' + \beta_2^{(m,n)} z^\rho E\big[\exp\big(\varepsilon'\big)\big],$$

i.e., (11) and (12) hold in each local area. The precomputation analysis can be generalized to higher-order piecewise polynomial approximations in a similar way.

2.3 *Characterizing the solution under precomputation of integrals*

In this section, we show how to precompute expectations in the Bellman and Euler equations.

2.3.1 *Bellman equation with precomputation of integrals* Precomputation of integrals is straightforward in the Bellman equation. We parameterize the true value function $V(k, z)$ with a flexible functional form $\widehat{V}(k, z; b)$ given by polynomial (9). Using (10),

we rewrite the Bellman equation (4)–(6) as

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

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

$$b_i' = b_i\mathcal{I}_i, \quad i=0,1,\ldots,n, \tag{15}$$

where $\doteq$ indicates that the equality is satisfied approximately, $b \equiv (b_0, b_1, \ldots, b_n) \in \mathbb{R}^{n+1}$ and $b' \equiv (b_0', b_1', \ldots, b_n') \in \mathbb{R}^{n+1}$, and the $\mathcal{I}_i$s in (15) are determined by (12). In the transformed Bellman equation (13)–(15), the effect of uncertainty on the solution is summarized by expected values in (12) that determine a relation between b and b'. To solve the transformed Bellman equation, we proceed in two steps. First, construct $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12); second, find b that solves (13)–(15). Apart from the fact that b and b' differ, the transformed Bellman equation (13)–(15) is standard and can be solved using a variety of solution algorithms available in the literature; we show some algorithms in Section 2.5. The only difference is that we are able to construct expectation functions faster and/or more accurately.

2.3.2 *Euler equation with precomputation of integrals* Precomputation of integrals is trickier in the Euler equation. The precomputation technique introduced in Section 2.2 assumes that a function we parameterize is the same as a function for which we need to compute expectation. This is true for the Bellman equation approach when we parameterize $V(k,z)$ and compute $E[V(k',z')]$. However, this is not true for the existing Euler equation approaches that parameterize such policy functions as the expectation function (Den Haan and Marcet (1990)), the capital function (Judd (1992)), and the labor function (Maliar and Maliar (2005b)), because these functions do not coincide with the integrand of the expectation function in the Euler equation $E[u'(c')(1-\delta+z'f'(k'))]$. None of these parameterizations allows us to precompute the integrals in the Euler equation in the way we did in the Bellman equation.[5]

To adapt the precomputation technique to the Euler equation methods, we reparameterize the Euler equation, namely, we introduce a new variable q that represents the integrand of the expectation function in the Euler equation (7):

$$q \equiv u'(c)\big[1-\delta+zf'(k)\big]. \tag{16}$$

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

$$\frac{q}{1-\delta+zf'(k)} = \beta E\big[q'\big]. \tag{17}$$

We now have the same function on the left side $q = Q(k,z)$ as the one inside the expectation on the right side $E[Q(k',z')]$ and we can use the precomputation result. By

[5]In particular, the parameterized expectation algorithm of Den Haan and Marcet (1990) parameterizes the expectation function but to precompute integrals, we need to parameterize the integrand of the expectation function.

parameterizing $Q(\cdot)$ with a flexible functional form $\widehat{Q}(\cdot; b)$ given by polynomial (9), we rewrite the Euler equation (17) as

$$\frac{\widehat{Q}(k, z; b)}{1 - \delta + zf'(k)} \doteq \beta \widehat{Q}\big(k', z^{\rho}; b'\big), \tag{18}$$

where b and b' are related by (11). Again, after the integrals are precomputed, all the effect of uncertainty on the solution is compressed into a mapping between vectors b and b'. To solve the transformed Euler equation, we proceed in two steps. First, we construct $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12); second, we find b that solves (5), (6), and (18). As in the case of the Bellman equation, we can use a variety of solution algorithms to solve the transformed problem; we show an example of such algorithms in Section 2.5.

In fact, the proposed change of variables makes it possible to precompute integrals in any set of equations that contains expectation functions: we just need to introduce new variables that represent the integrands of expectations functions and rewrite the model's equations in terms of these new variables.

2.3.3 *Precomputation of integrals and envelope condition method* There is a relation between the technique of precomputation of integrals and the envelope condition method (ECM) introduced in Maliar and Maliar (2013) and developed in Arellano et al. (2016). If a solution to the Bellman equation (4)–(6) is interior, it satisfies a first-order condition and envelope condition, which, respectively, are

$$u'(c) = \beta E\big[V_1\big(k', z'\big)\big], \tag{19}$$

$$V_1(k, z) = u'(c)\big[1 - \delta + zf'(k)\big], \tag{20}$$

where F_i denotes a first-order derivative of a function F with respect to the ith argument and g' denotes a first derivative of a function g. Combining (19) and (20) yields the Euler equation (7) in terms of the derivatives of the value function:

$$\frac{V_1(k, z)}{1 - \delta + zf'(k)} = \beta E\big[V_1\big(k', z'\big)\big]. \tag{21}$$

This is an equation that is used to implement a version of the ECM method iterating on the derivative of the value function (ECM-DVF). Notice that Euler equation (21) used by ECM-DVF is identical (up to notation) to the Euler equation (18) reparameterized for precomputation of integrals, namely, $\widehat{Q}(k, z; b)$ and $\widehat{Q}(k', z^{\rho}; b')$ coincide with $\widehat{V}_1(k, z; b)$ and $\widehat{V}_1(k', z^{\rho}; b')$, respectively. That is, the ECM method delivers systems of equations that are directly suitable for precomputation of integrals.

2.4 *Analytical versus numerical construction of integrals*

In this section, we compare the accuracy and computational expense of analytical and numerical approximation of integrals.

Analytical construction of integrals under precomputation In our precomputation analysis, integrals (12) can be constructed analytically for the case of normally distributed shock $\varepsilon' \sim \mathcal{N}(0, \sigma^2)$, that is,

$$\begin{aligned}\mathcal{I}_i &= E\big[\exp\big(l_i\varepsilon'\big)\big] = \frac{1}{\sqrt{2\pi\sigma^2}}\int_{-\infty}^{+\infty}\exp\big(l_i\varepsilon'\big)\exp\left(-\frac{\big(\varepsilon'\big)^2}{2\sigma^2}\right)d\varepsilon' \\ &= \frac{1}{\sqrt{2\pi\sigma^2}}\int_{-\infty}^{+\infty}\exp\left(-\frac{\big(\varepsilon' - l_i\sigma^2\big)^2}{2\sigma^2}\right)\exp\left(\frac{\sigma^2 l_i^2}{2}\right)d\varepsilon' \\ &= \exp\left(\frac{\sigma^2 l_i^2}{2}\right),\end{aligned} \tag{22}$$

where, to obtain the last result, we use the fact $\int_{-\infty}^{+\infty} f(x)\,dx = 1$ for a density function f of a normally distributed variable x with mean $l_i\sigma^2$ and variance σ^2. While this is a special case, it is a very important one as the assumption of normally distributed shocks is employed by nearly all the literature on dynamic economic models. Maliar and Maliar (2004, 2005a) provide related examples of analytical precomputation of integrals in economic models.

Numerical construction of integrals We consider two numerical integration methods that are commonly used in the literature, namely, a Monte Carlo method and a Gauss Hermite quadrature method. Both methods approximate integrals by a weighted average (8), but differ in the choice of the integration nodes and their weights. The Monte Carlo method generates nodes by using a random draw and assigns equal weights to all nodes, while the Gauss Hermite quadrature method uses a set of nodes and weights that mimic the shape of the normal distribution. For the Monte Carlo method, we use 100 and 10,000 nodes, and for the Gauss Hermite quadrature method, we use 2, 5, and 10 nodes; we refer to these methods as MC(100), MC(10,000), GH(2), GH(5), and GH(10), respectively; see Judd, Maliar, and Maliar (2011) for a description of these methods.

Comparison of the accuracy Since we have exact closed-form expressions for the integrals, we can assess the accuracy of approximations produced by numerical integration methods. Specifically, we compute a percentage difference between the exact value of an integral (evaluated using analytical expression (22)) and its approximate value (computed using a given numerical integration method).

As expected, the numerical integration methods become less accurate as the degrees of uncertainty increase. In Table 1, we report the exact value (22) of integral $\mathcal{I}_i$ for $l_i = 1, \ldots, 5$ under the standard deviation of shocks $\sigma = 0.2$; this size of shocks roughly corresponds to models with idiosyncratic risk such as Aiyagari's (1994) model.

GH(10) delivers the most accurate results with 11 digits of accuracy; GH(5) guarantees 4 digits of accuracy and MC(10,000) ensures 2 digits of accuracy, while the rest of the methods are less accurate and produce errors of up to 6.4%. (For the Monte Carlo methods, the value of the integral is a random variable that depends on a specific random draw, and the numbers in the table show a typical experiment.)

TABLE 1. Accuracy of univariate numerical integration methods.

Integral	Exact Value Under Precomputation	Numerical Integration Errors (%)				
		GH(2)	GH(5)	GH(10)	MC(100)	MC(10,000)
$E[\exp(\varepsilon')]$	1.0202	−0.0132	−0.0000	0	1.7833	0.0327
$E[\exp(2\varepsilon')]$	1.0833	−0.2044	−0.0000	0	3.3320	0.0628
$E[\exp(3\varepsilon')]$	1.1972	−0.9816	−0.0000	0	4.4131	0.1425
$E[\exp(4\varepsilon')]$	1.3771	−2.8823	−0.0003	−0.0000	4.7574	0.3434
$E[\exp(5\varepsilon')]$	1.6487	−6.4074	−0.0025	−0.0000	4.0906	0.7724

Note: GH(2), GH(5), and GH(10) are Gauss Hermite quadrature methods with 2, 5, and 10 nodes, respectively; MC(100) and MC(10,000) are Monte Carlo integration methods with 100 and 10,000 nodes, respectively.

We repeat the calculations under a lower standard deviation of shocks $\sigma = 0.02$; this size of shocks is typical for business cycle models. Here, all deterministic methods are very accurate: in particular, GH(10) and GH(5) guarantee more than 14 digits of accuracy, while GH(2) is slightly less accurate and guarantees 5 digits of accuracy. MC integration methods are considerably less accurate; for example, with 100 nodes, the absolute unit-free errors range between 0.2 and 0.9%, respectively, and with 10,000 nodes, they range between 0.005 and 0.02%, respectively, in a typical simulation. (To save on space, the results for $\sigma = 0.02$ are not reported.)

Comparison of costs In Table 2, we illustrate the difference in costs between the exact and approximate integration methods. Specifically, we report the factors that show how much the time for evaluating expectation function (8) increases when we use numerical integration methods relative to the case when we use the closed-form expression (22). In particular, the first line of the table shows the results for the linear polynomial function $E[b_0 + b_1 k' + b_2 z']$.

Our results show that evaluating integrals using precomputation can be considerably faster than doing it numerically. The difference in costs is especially large for Monte Carlo integration (orders of magnitude). Notice that the reported difference in costs is just for a one time evaluation of integral. However, when constructing numerical solutions to dynamic economic models, we must evaluate integrals many times in an it-

TABLE 2. Computational cost of univariate numerical integration methods.

Polynomial Degree	Evaluation Time Under Precomputation (in sec)	Speedup From Precomputation				
		GH(2)	GH(5)	GH(10)	MC(100)	MC(10,000)
1st	0.1498(−3)	×1.39	×2.10	×2.64	×17.24	×1104.3
2nd	0.1652(−3)	×1.52	×1.78	×2.95	×27.92	×2254.8
3rd	0.1142(−3)	×1.69	×1.59	×2.94	×31.63	×2802.1
4th	0.0821(−3)	×1.72	×2.69	×5.92	×52.24	×5093.7
5th	0.0747(−3)	×1.78	×4.04	×7.95	×74.69	×7478.5

Note: GH(2), GH(5), and GH(10) are Gauss Hermite quadrature methods with 2, 5, and 10 nodes, respectively; MC(100) and MC(10,000) are Monte Carlo integration methods with 100 and 10,000 nodes, respectively.

erative cycle, and the corresponding cumulative difference in cost can be even more substantial, as we will show in the next section.

2.5 *Numerical assessment of gains from precomputation of integrals in Bellman and Euler equation methods*

We must emphasize that the technique of precomputation of integrals is not a new numerical solution method but a computational technique that can improve the performance of many existing solution methods in terms of their accuracy and computational expense. As an illustration, we show how to precompute integrals in two publicly available MATLAB codes: one code solves a neoclassical growth model with inelastic labor supply by using seven alternative solution methods described in Arellano et al. (2016); the other code solves a similar model with elastic labor supply by using four alternative solution methods, described in Maliar and Maliar (2013). We show that precomputation of integrals can substantially reduce the cost of value-iterative methods, including conventional value function iteration, the endogenous grid method of Carroll (2006), and the envelope condition method of Maliar and Maliar (2013), as well as of two variants of the Euler equation method. All the methods analyzed in this section are elaborated in Appendices A and B.

2.5.1 *Methodology and implementation details* Tables 1 and 2 suggest that precomputation of integrals can both increase the accuracy and reduce the cost of analyzing dynamic stochastic models. How large the gains from precomputation of integrals are will critically depend on a numerical solution method used for comparison. In our comparison analysis, we solve the model (1)–(3) by using two otherwise identical numerical solution methods: one method precomputes the integrals by using exact formula (22) and the other recomputes the integrals on each iteration using an accurate Gauss Hermite quadrature method with five integration nodes, GH(5). In both cases, the numerical solutions will be essentially the same, and we will only observe the difference in running times. Thus, we focus on savings in cost maintaining the same (high) accuracy of solutions. If a somewhat less accurate integration method is used for comparison (e.g., Monte Carlo), there will be also considerable accuracy gains from precomputation of integrals.

To parameterize the model (1)–(3), we assume $u(c_t)=\frac{c_t^{1-\gamma}-1}{1-\gamma}$ with $\gamma=\{\frac{1}{3},3\}$, $f(k_t)=Ak_t^{\alpha}$ with $\alpha=0.36$, and $A=\frac{1/\beta-(1-\delta)}{\alpha}$ (this value normalizes the deterministic steady state of capital to unity), and we use $\beta=0.99$, $\delta=0.025$, $\rho=0.95$, and $\sigma=0.01$.

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 from 2 to 5. As a measure of accuracy, we report the mean and maximum of absolute unit-free Euler equation residuals on a stochastic simulation of 10,000 observations. We use MATLAB R2015b software and a serial desktop computer with Intel Core i5 (3.2 GHz) and 16 MB RAM.

Remark 2. We draw attention to a novel methodological aspect of our accuracy evaluation procedure that is derived from our precomputaion analysis. Namely, to construct

the residuals, we use the transformed Euler equation

$$u'(c) \doteq \beta \widehat{Q}\big(k', z^{\rho}; b'\big) \tag{23}$$

instead of the conventional Euler equation (7). This fact allows us to evaluate the expectation functions exactly when evaluating accuracy, while in the conventional Euler equation, the analysis of residuals relies on approximate numerical integration, which may contaminate the inferences about the accuracy.

2.5.2 *Numerical results for value-iterative methods* Conventional value function iteration (VFI) is expensive; see Aruoba, Fernández-Villaverde, and Rubio-Ramírez (2006). Two recent value-iterative methods help us to reduce the cost of iterating on value function, namely, the endogenous grid method (EGM) of Carroll (2006) and the envelope condition method () of Maliar and Maliar (2013). We show that the technique of precomputation of integrals can reduce the cost of conventional value function iteration, as well as the cost of these two recent methods. Notice that if it is used in the context of the value-iterative methods, precomputation of integrals produces just benefits without adding any costs, namely, it does not affect the way in which such methods are implemented but it just makes them faster.

Conventional VFI Conventional VFI requires finding k' that maximizes the right side of Bellman equation (4) for each state (k, z). By combining (4) and (5), we get

$$V(k, z) = \max_{k'} \big\{ u\big((1-\delta)k + zf(k) - k'\big) + \beta E\big[V\big(k', z'\big)\big]\big\}. \tag{24}$$

We must solve (24) with respect to k' in each point of the grid (k, z). To find a maximizer k' in (24), we must explore many different candidate points k' and for each of such points, we must interpolate V to future values (k', z') and approximate the corresponding conditional expectation function $E[V(k', z')]$. Precomputation of integrals reduces the cost of VFI by constructing the conditional expectation function up-front, before solving the model.

In Table 3, we compare the performance of two versions of conventional VFI: one in which conditional expectation is computed at each iteration and the other in which it is precomputed up-front before starting the solution procedure.

Both versions of conventional VFI deliver the same levels of accuracy for all polynomial approximations considered. However, VFI with precomputation is nearly two times faster than the version without precomputation. The overall cost of finding a solution is still high for both methods. Precomputation helps us to reduce the numerical cost of the integral evaluation but there is still a large cost associated with solving a nonlinear equation in each grid point.

Endogenous grid method Carroll (2006) introduces an EGM that reduces the cost of conventional VFI. The key idea of EGM is to construct a grid on future endogenous state variables k' instead of current endogenous state variables k. By combining (4) and (5), we get

$$V(k, z) = \max_{k'} \big\{ u\big((1-\delta)k + zf(k) - k'\big) + \beta W\big(k', z\big)\big\}, \tag{25}$$

TABLE 3. Accuracy and cost of conventional VFI with and without precomputation.

	$\gamma = 1/3$						$\gamma = 3$					
Polynomial	No Precomputation			Precomputation			No Precomputation			Precomputation		
Degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−4.02	−3.52	4.09	−4.02	−3.52	1.56	−3.43	−2.43	6.86	−3.43	−2.43	4.42
3rd	−5.38	−4.64	6.45	−5.38	−4.64	3.00	−4.38	−3.11	12.19	−4.38	−3.11	7.87
4th	−6.65	−5.77	7.92	−6.65	−5.77	3.87	−5.27	−3.82	16.33	−5.27	−3.82	10.37
5th	−7.97	−6.85	8.73	−7.97	−6.85	4.36	−6.05	−4.45	19.76	−6.05	−4.45	12.41

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute approximation errors across optimality condition and test points (in log 10 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.

where $W(k', z) \equiv E[V(k', z')]$ is an expectation function that EGM computes at the beginning of each iteration (this is possible to do because the values of k' are fixed). We must solve (25) with respect to k in each point (k', z) of the grid.[6] Note that our technique of precomputation of integrals allows us to go one step further and to construct integrals just once, at the beginning of the solution procedure.

In Table 4, we compare the performance of two versions of EGM: one in which conditional expectation is recomputed in each iteration and the other in which it is precomputed up-front before starting the solution procedure.

The gains from precomputation are smaller under EGM than under conventional VFI. This is because in the former case, conditional expectations are evaluated just once per iteration on value function, while in the latter case, they are evaluated multiple times per iteration when value function is evaluated for different candidate points k'. Still, the savings from precomputation of integrals are sizable even for the EGM method.

TABLE 4. Accuracy and cost of the EGM algorithm with and without precomputation.

	$\gamma = 1/3$						$\gamma = 3$					
Polynomial	No Precomputation			Precomputation			No Precomputation			Precomputation		
Degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−3.89	−3.55	1.12	−3.89	−3.55	0.77	−3.43	−2.44	2.37	−3.43	−2.44	1.87
3rd	−5.21	−4.56	2.09	−5.21	−4.56	1.52	−4.39	−3.12	4.25	−4.39	−3.12	3.40
4th	−6.36	−5.61	2.76	−6.36	−5.61	2.00	−5.30	−3.84	5.80	−5.30	−3.84	4.60
5th	−7.60	−6.65	3.13	−7.60	−6.65	2.32	−6.10	−4.48	7.13	−6.10	−4.48	5.62

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in log 10 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.

[6]This problem still requires a numerical solver. Carroll (2006) proposes a change of variables that can avoid the use of a numerical solver for this specific model. Precomputation of integrals is also possible with Carroll's (2006) change of variables and can speed up computations there as well; we do not study this method in the paper.

Envelope condition method The ECM of Maliar and Maliar (2013) relies on a conventional exogenous grid (k, z) and simplifies the root-finding using a different mechanism. Namely, to construct policy functions, ECM uses the envelope condition instead of the first-order conditions (FOCs) used by conventional VFI and EGM. For problem (4)–(6), the envelope condition provides a convenient closed-form expression for consumption policy function:

$$c = u'^{-1}\left(\frac{V_1(k,z)}{1-\delta+zf'(k)}\right). \tag{26}$$

Given c, we compute k' from budget constraint (5) and, finally, evaluate $E[V(k', z')]$ on the right side of the Bellman equation (4) to construct $V(k, z)$ on the left side. In this model, ECM is simpler than Carroll's (2006) EGM since all the policy functions can be constructed analytically and a numerical solver need not be used at all (not even once). We compare the accuracy and speed of the ECM with and without precomputation of integrals in Table 5.

The gains from precomputation of integrals are higher for ECM than for previously considered VFI and EGM. Since ECM avoids finding the roots of nonlinear equations, the cost of approximating integrals constitutes a larger fraction of total costs. As a result, savings in the cost of integration are also more substantial in percentage terms for ECM than for the other methods.

Remark 3. Our results should not be interpreted as a comparison of ECM and EGM. First, as we said earlier, the cost of EGM in this specific model can be reduced by using the change of variables proposed in Carroll (2006); second, in more complicated models (e.g., the one with valued leisure), neither ECM nor EGM can avoid root-finding completely, although they both can simplify it.[7] Maliar and Maliar (2013, 2014) show that in

Table 5. Accuracy and cost of the ECM method with and without precomputation.

	$\gamma = 1/3$						$\gamma = 3$					
Polynomial	No Precomputation			Precomputation			No Precomputation			Precomputation		
Degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−4.02	−3.52	0.46	−4.02	−3.52	0.14	−3.43	−2.43	0.61	−3.43	−2.43	0.18
3rd	−5.38	−4.64	0.62	−5.38	−4.64	0.19	−4.38	−3.11	1.03	−4.38	−3.11	0.32
4th	−6.65	−5.77	0.83	−6.65	−5.77	0.26	−5.27	−3.82	1.48	−5.27	−3.82	0.46
5th	−7.97	−6.85	0.87	−7.97	−6.85	0.27	−6.05	−4.45	1.83	−6.05	−4.45	0.59

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute approximation errors across optimality condition and test points (in log 10 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.

[7] For the studied model, Carroll's (2006) technique requires us to introduce a new variable $Y' \equiv z'f(k') + (1-\delta)k'$ and to rewrite the Bellman equation as $V(Y, z) = \max\{u(Y-k') + \beta E[V(Y', z')]\}$. Using an endogenous grid on k' (instead of k) makes it possible to compute $E[V(Y', z')]$ with just one evaluation per iteration. However, in a similar model with labor, $Y' = z'f(k', \ell') + (1-d)k'$ depends on labor ℓ', and $E[V(Y', z')]$ cannot be computed without solving for labor policy function; see Barillas and Fernandez-Villaverde (2007) for a method that recomputes the endogenous grid by iterating on the labor policy function, and see Maliar and Maliar (2013) for a method that computes labor using a numerical solver.

a model with valued leisure, the performance of ECM and EGM is very similar in terms of both accuracy and computational expense. Our point is that for all such methods, precomputation of integrals can help save on costs and increase the accuracy of solutions.

2.5.3 *Numerical results for Euler equation methods* We next demonstrate that precomputation of integrals can also reduce the cost of Euler equation methods. Our benchmark solution method is similar to projection methods in Judd (1992) in that it uses a rectangular grid and deterministic (quadrature) integration methods, but it also uses fixed-point iteration with damping as in Den Haan and Marcet (1990) and Judd, Maliar, and Maliar (2011).

Parameterizing Q function Our first Euler equation method parameterizes the new variable q and uses a change of variables that eliminates consumption. In terms of q, the Euler equation is given by (17), and the budget constraint is

$$k' = (1-\delta)k + zf(k) - u'^{-1}\left(\frac{q}{1-\delta+zf'(k)}\right). \tag{27}$$

The results of our numerical experiments for the Euler equation algorithm considered are provided in Table 6.

The main finding in the table is that savings in costs from precomputation of integrals are much larger for the Euler equation method than for the previously considered VFI and EGM methods, and are similar to those under the ECM method. Here, the running time is typically reduced by a factor of 3. Such a large reduction in costs is due to the fact that the considered Euler equation method avoids finding the roots of nonlinear equations and spends the largest fraction of total costs on integration.

Parameterizing capital function jointly with Q function In some applications, it might be preferable to approximate policy functions other than q because such policy functions are better behaved or more convenient for some computational purposes.[8] Also, it

TABLE 6. Accuracy and cost of the Euler equation algorithm, parameterizing Q function, with and without precomputation.

	$\gamma = 1/3$						$\gamma = 3$					
Polynomial	No Precomputation			Precomputation			No Precomputation			Precomputation		
Degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−4.02	−3.52	0.53	−4.02	−3.52	0.16	−3.44	−2.46	0.76	−3.44	−2.46	0.23
3rd	−5.38	−4.64	0.73	−5.38	−4.64	0.22	−4.38	−3.11	1.37	−4.38	−3.11	0.41
4th	−6.65	−5.77	0.98	−6.65	−5.77	0.30	−5.26	−3.82	1.92	−5.26	−3.82	0.59
5th	−7.97	−6.85	1.06	−7.97	−6.85	0.32	−6.05	−4.45	2.34	−6.05	−4.45	0.76

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in log 10 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.

[8] For example, a capital policy function may have less curvature than consumption and labor policy functions and, hence, it can be approximated more accurately. Moreover, knowing the capital policy function

could happen that recovering the decision variables such as c and k' from q requires us to solve some nonlinear equations numerically. In this case, the advantages of precomputing the integrals might be offset by disadvantages of numerical root-finding. While precomputation of integrals is not directly suitable for approximating functions other than q, it is still possible to approximate other policy functions jointly with q (and thus, sidestep the root-finding step). As an example, we show how to use the precomputation technique for approximating the capital policy function $k' = K(k, z)$.

Premultiplying both sides of Euler equation (17) by k' and rearranging the terms, we obtain an equivalent representation of the Euler equation (as long as $k' > 0$),

$$k' = \beta \frac{E[q']}{q}\big(1 - \delta + zf'(k)\big)k' \doteq \widehat{K}(k, z; v), \tag{28}$$

where $\widehat{K}(\cdot; v)$ is a flexible functional form and v is the coefficients vector. Thus, we have expressed k' in two ways: one is as today's choice variable k' parameterized by $\widehat{K}(\cdot; v)$; the other is as a combination of the model's variables that involves conditional expectation of a random variable $E[q']$. We can compute the capital policy function as a fixed point of (28) by iterating on v until convergence. However, to perform such iterations, we also need to construct q, defined by (16), and $E[q']$. We again compare two otherwise identical numerical solution methods: one that recomputes $E[q']$ in each step of the iterative procedure using a five-node Gauss Hermite quadrature method; the other that precomputes $E[q']$ in the beginning.

In Table 7, we provide the results for the Euler equation method approximating the capital policy function K jointly with Q.

The accuracy of this version of the Euler equation algorithm is similar to that of the Euler equation algorithm that parameterizes only Q; however, the running time is somewhat larger. This is because such a method is less numerically stable than the other methods and we had to use damping to enhance numerical stability, namely, we update the policy function only by 15% from one iteration to another.

TABLE 7. Accuracy and speed of the Euler equation algorithm, jointly parameterizing functions Q and K, with and without precomputation.

	$\gamma = 1/3$						$\gamma = 3$					
Polynomial	No Precomputation			Precomputation			No Precomputation			Precomputation		
Degree	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−4.02	−3.52	3.18	−4.02	−3.52	0.90	−3.44	−2.46	4.86	−3.44	−2.46	0.87
3rd	−5.38	−4.64	3.69	−5.38	−4.64	1.05	−4.38	−3.11	7.26	−4.38	−3.11	1.56
4th	−6.65	−5.77	4.36	−6.65	−5.77	1.26	−5.26	−3.82	9.44	−5.26	−3.82	2.24
5th	−7.42	−6.53	4.46	−7.42	−6.53	1.33	−6.05	−4.45	10.77	−6.05	−4.45	2.70

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in log 10 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.

makes it possible to compute endogenous state variables in all grid points without solving for control variables, which is convenient for parallel computation.

The gains from precomputation of integrals are even larger for this version of the Euler equation method than for the one parameterizing the Q function; in particular, under high risk aversion coefficient, the running time is reduced by up to a factor of 5. Combining precomputation of integrals with approximations of several functions simultaneously will be useful in more complex models in which there are several Euler equations and several conditional expectation functions that need to be precomputed; this case is studied in Section 3.

Conventional Euler equation method parameterizing the capital function In the case of the Euler equation methods, precomputation of integrals has benefits, but may also have costs. Specifically, to be able to precompute expectation functions, we must reparameterize the Euler equation in a particular way and we must compute a decision function for a specific variable, which is the integrand of the expectation function. It could happen that in some applications, this alternative implementation of the Euler equation iteration produces less accurate solutions than the conventional Euler equation method.

To see whether or not this is the case in our model, we solve the model by using the conventional Euler equation method that solves for the capital policy function $k' = K(k, z)$ satisfying

$$k' = (1-\delta)k + zf(k) - u'^{-1}\big(\beta E\big\{u'\big(c'\big(k', z'\big)\big)\big[1-\delta+z'f'\big(k'\big)\big]\big\}\big), \tag{29}$$

where the above Euler equation follows by combining (5) and (7). The results are shown in Table 8. As we observe, the accuracy of the conventional Euler equation algorithm is similar to that of other methods considered in this section. The running time for this method is slightly smaller than that of the Euler equation method that parameterizes capital policy function jointly with Q, however, is significantly larger than that of the Euler equation method that parameterizes only function Q. Thus, in this particular case, a reparameterization of the Euler equation in terms of Q has no negative side effects on either accuracy or cost.

TABLE 8. Accuracy and cost of a conventional Euler equation algorithm parameterizing K without precomputation.

Polynomial Degree	$\gamma = 1/3$			$\gamma = 3$		
	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−4.02	−3.53	0.54	−3.44	−2.46	0.79
3rd	−5.38	−4.64	0.73	−4.38	−3.11	1.37
4th	−6.65	−5.77	0.97	−5.26	−3.82	1.87
5th	−7.94	−6.83	1.02	−6.05	−4.45	2.23

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals 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.

Model with elastic labor supply We finally show that precomputation of the expectation functions can simplify the construction of numerical solutions to more complex models such as the model with elastic labor supply. In such a model, the representative agent solves

$$V(k,z) = \max_{k',c,l}\{u(c,l) + \beta E[V(k',z')]\} \tag{30}$$

$$\text{s.t.} \quad k' = (1-\delta)k + zf(k,l) - c, \tag{31}$$

$$\ln z' = \rho \ln z + \varepsilon', \tag{32}$$

where l denotes labor, u is a utility function that is strictly increasing in consumption and strictly decreasing in labor, f is a production functions that is strictly increasing in both arguments, and both utility and production functions, are continuously differentiable and concave. Our goal is to solve for value function $V(k,z)$ and policy functions $c = C(k,z)$, $l = L(k,z)$, and $k' = K(k,z)$.

As far as value-iterative methods are concerned, value function can be precomputed in the model with elastic labor supply in the same way as in the model with inelastic labor supply studied in Section 2.5.2. Let us show how the precomputation analysis can be generalized to the Euler equation methods. An interior solution to (30)–(32) satisfies first-order conditions

$$u_1(c,l) = \beta E[u_1(c',l')(1-\delta + z'f_1(k',l'))], \tag{33}$$

$$u_2(c,l) = -u_1(c,l)zf_2(k,l). \tag{34}$$

Again, we rewrite the Euler equation in the form that is suitable for precomputation,

$$\frac{q}{1-\delta + zf_1(k,l)} = \beta E[q'], \tag{35}$$

where q denotes the integrand of the Euler equation (33),

$$q \equiv u_1(c,l)[1-\delta + zf_1(k,l)]. \tag{36}$$

Since we must parameterize q in terms of the state variables $q = Q(k,z)$, the construction of the intratemporal choice requires us to solve numerically two equations (34) and (36) with respect to two unknowns c and l given (k,z). How does this problem compare to the one we face under the conventional Euler equation methods? The answer depends on what function is parameterized in terms of state variables. If we parameterize the capital function $k' = K(k,z)$, we must also solve numerically two equations (31) and (34) with respect to two unknowns c and l given (k,z), so the cost will be comparable; the same is true if we parameterize the consumption function $c = C(k,z)$. However, if we parameterize the labor function $l = L(k,z)$, the intratemporal choice c and k' satisfying (31) and (34) can be derived in a closed form under conventional utility functions such as Cobb–Douglas or addilog, and the cost will be considerably lower; see Maliar and Maliar (2005b). Finally, it is possible to reduce the cost of iteration on either the conventional or transformed Euler equations by constructing the intratemporal choice

TABLE 9. Accuracy and cost of the ECM and Euler equation algorithms in the model with elastic labor supply with and without precomputation.

Polynomial Degree	$\gamma=1/3$						$\gamma=3$					
	No Precomputation			Precomputation			No Precomputation			Precomputation		
	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
2nd	−3.42	−2.43	5.73	−3.42	−2.43	3.73	−3.43	−2.47	4.13	−3.43	−2.47	3.39
3rd	−4.44	−3.19	8.03	−4.44	−3.19	5.73	−4.44	−3.20	7.67	−4.44	−3.20	6.41
4th	−5.41	−3.95	10.01	−5.41	−3.95	7.35	−5.41	−3.94	14.40	−5.41	−3.94	9.45
5th	−6.28	−4.62	11.50	−6.28	−4.62	8.61	−6.28	−4.62	18.02	−6.28	−4.62	12.57

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute approximation errors across optimality condition and test points (in log 10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds).

functions outside the main iterative cycle on some prespecified grid of points; see Maliar and Maliar (2005b, 2014) for this different kind of precomputation results.

As in the case of inelastic labor supply, the transformed Euler equation (35) is identical to that under ECM-DVF (up to notation), and, therefore, we can use the code accompanying the paper of Maliar and Maliar (2013) as a basis for our precomputation analysis. Following that paper, we parameterize the model (30)–(32) by an additively separable utility function $u(c,l)=\frac{c^{1-\gamma}-1}{1-\gamma}+B\frac{(1-l)^{1-\mu}-1}{1-\mu}$ for which the two equations (34) and (36) can be combined into one equation relating q and l:

$$q=\frac{B(1-l)^{-\mu}}{zf_2(k,l)}\big[1-\delta+zf_1(k,l)\big]. \tag{37}$$

We assume that the production function is Cobb–Douglas $f(k,l)=Ak^{\alpha}l^{1-\alpha}$ with $\alpha=0.33$ and $A=\frac{1/\beta-(1-\delta)}{\alpha l^{1-\alpha}}$ (this value normalizes the deterministic steady state of capital to 1). We fix $\gamma=5$ and $\mu=5$, and we calibrate the parameters β, δ, and B to reproduce the steady state labor, capital–output, and consumption–output ratios of $l=1/3$, $\pi_k=10$, and $\pi_c=3/4$, respectively, and we assume $\rho=0.95$ and $\sigma=0.01$. The implementation of the solution methods with elastic labor supply is similar to that with inelastic labor supply; see Appendix B for details. To solve for labor choices, we solve (37) numerically in each grid point by using a Newton solver.

In Table 9, we compare the results for the value-iterative and Euler equation methods with and without precomputation of integrals. Again, the introduction of precomputation of integrals has no effect on accuracy, but reduces the running time by about 30% for both value-iterative and Euler equation methods. Here, the reduction in cost is not as large as in the model with inelastic labor supply because a large fraction of the total cost comes from applying a numerical solver to multiple grid points.

3. MULTIVARIATE CONTINUOUS-STATE EXPECTATIONS

Although precomputation of integrals can visibly speed up computations even in models with just one shock, the real case of interest is models with multivariate shocks, in

which the cost of integrals evaluation can increase rapidly with the dimensionality of the problem. As an example, we describe a multicountry model with N stochastic shocks analyzed in Judd, Maliar, and Maliar (2011). This model provides a convenient framework for testing how the accuracy and cost of numerical solution methods change with the number of state variables, in particular, with the number of exogenous shocks.

3.1 *Heterogeneous neoclassical stochastic growth 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, that is,

$$\max_{\{c_t^h,k_{t+1}^h\}_{t=0,\ldots,\infty}^{h=1,\ldots,N}} E_0 \sum_{h=1}^{N} \tau^h \left(\sum_{t=0}^{\infty} \beta^t u\big(c_t^h\big) \right) \tag{38}$$

subject to

$$\sum_{h=1}^{N} c_t^h = \sum_{h=1}^{N} \big[z_t^h f\big(k_t^h\big) + (1-\delta) k_t^h - k_{t+1}^h \big], \tag{39}$$

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

$$\ln z_t^h = \rho \ln z_{t-1}^h + \varepsilon_t^h, \tag{40}$$

where $\boldsymbol{\varepsilon}_t = (\varepsilon_t^1,\ldots,\varepsilon_t^N) \sim \mathcal{N}(0_N, \Sigma)$ is generated by a multivariate normal distribution with zero mean 0_N and variance–covariance matrix Σ. Here, we assume that all the countries have identical utility and production functions; however, it is straightforward to extend our analysis for the case when the utility and production functions differ across countries; see Maliar, Maliar, and Judd (2011).

Bellman equation We can represent problem (38)–(40) in a dynamic programming form,

$$V(\boldsymbol{k},\boldsymbol{z}) = \max_{\{(k^h)',c^h\}^{h=1,\ldots,N}} \left\{ \sum_{h=1}^{N} \tau^h u\big(c^h\big) + \beta E\big[V\big(\boldsymbol{k}',\boldsymbol{z}'\big)\big] \right\} \tag{41}$$

$$\text{s.t.} \quad \sum_{h=1}^{N} c^h = \sum_{h=1}^{N} \big[z^h f\big(k^h\big) + (1-\delta) k^h - \big(k^h\big)' \big], \tag{42}$$

$$\ln\big(z^h\big)' = \rho \ln z^h + \big(\varepsilon^h\big)', \tag{43}$$

where V is an optimal value function, $\boldsymbol{\varepsilon}' = ((\varepsilon^1)',\ldots,(\varepsilon^N)') \sim \mathcal{N}(0_N,\Sigma)$; $\boldsymbol{k} \equiv (k^1,\ldots,k^N)$, and $\boldsymbol{z} \equiv (z^1,\ldots,z^N)$.

Euler equation An interior solution to the planner's problem (38)–(40) satisfies a set of N first-order conditions

$$u'\big(c^h\big)=\beta E\big\{u'\big(\big(c^h\big)'\big)\big[1-\delta+\big(z^h\big)'f'\big(\big(k^h\big)'\big)\big]\big\}, \tag{44}$$

where $h=1,\ldots,N$. Under the Euler equation approach, we solve for policy functions $c^h=C^h(\boldsymbol{k},z)$ and $(k^h)'=K^h(\boldsymbol{k},z)$, $h=1,\ldots,N$, that satisfy (39), (40), and (44).

3.2 *Precomputation of multivariate integrals under polynomial approximations*

It turns out that multivariate integrals can be precomputed in essentially the same way as univariate integrals in Section 2.2. Let $\boldsymbol{k}=(k^1,\ldots,k^N)$ and $\boldsymbol{z}=(z^1,\ldots,z^N)$ be the economy's state variables and let a policy function $\mathcal{P}(\boldsymbol{k},\boldsymbol{z})$ be approximated with ordinary polynomial function

$$\begin{aligned}\mathcal{P}(\boldsymbol{k},\boldsymbol{z};b)=\;&b_0+b_1k^1+\cdots+b_Nk^N+b_{N+1}z^1+\cdots\\&+b_{2N}z^N+b_{1,1}\big(k^1\big)^2+\cdots+b_{N,N}\big(k^N\big)^2\\&+\sum_{i,j\in\{1,\ldots,N\},i\neq j}b_{i,j}k^ik^j+\sum_{i\in\{1,\ldots,N\},j\in\{N+1,\ldots,2N\}}b_{i,j}k^iz^j\\&+\sum_{i,j\in\{N+1,\ldots,2N\}}b_{i,j}z^iz^j+\cdots\\&+b_{N+1,\ldots,N+1}\big(z^1\big)^L+\cdots+b_{2N,\ldots,2N}\big(z^N\big)^L,\end{aligned} \tag{45}$$

where $b=(b_0,b_1,\ldots,b_{2N},b_{1,1},\ldots,b_{2N,2N},\ldots,b_{1,\ldots,1},\ldots,b_{2N,\ldots,2N})\in\mathbb{R}^{n+1}$ is a vector of polynomial coefficients and L is a polynomial degree. Again, taking into account that $\boldsymbol{k}'\equiv((k^1)',\ldots,(k^N)')$ is known and that $(z^h)'=(z^h)^\rho\exp((\varepsilon^h)')$ has a known distribution function, we can represent a conditional expectation of $\mathcal{P}(\boldsymbol{k}',\boldsymbol{z}';b)$ as

$$\begin{aligned}E\big[\mathcal{P}\big(\boldsymbol{k}',\boldsymbol{z}';b\big)\big]=E\Big[&b_0+b_1\big(k^1\big)'+\cdots+b_N\big(k^N\big)'+b_{N+1}\big(z^1\big)'+\cdots\\&+b_{2N}\big(z^N\big)'+b_{1,1}\big(\big(k^1\big)'\big)^2+\cdots+b_{N,N}\big(\big(k^N\big)'\big)^2\\&+\sum_{i,j\in\{1,\ldots,N\}}b_{i,j}\big(k^i\big)'\big(k^j\big)'+\sum_{i\in\{1,\ldots,N\},j\in\{N+1,\ldots,2N\}}b_{i,j}\big(k^i\big)'\big(z^j\big)'\\&+\sum_{i,j\in\{N+1,\ldots,2N\}}b_{i,j}\big(z^i\big)'\big(z^j\big)'+\cdots\\&+b_{N+1,\ldots,N+1}\big(\big(z^1\big)'\big)^L+\cdots+b_{2N,\ldots,2N}\big(\big(z^N\big)'\big)^L\Big]\\=\;&\mathcal{P}\big(\boldsymbol{k}',\boldsymbol{z}^\rho;b'\big),\end{aligned} \tag{46}$$

where $b'\equiv(b'_0,b'_1,\ldots,b'_{2N},b'_{1,1},\ldots,b'_{2N,2N},\ldots,b'_{1,\ldots,1},\ldots,b'_{2N,\ldots,2N})\in\mathbb{R}^{n+1}$, and the coefficients b'_i and b_i are related by

$$b'_i=b_i\mathcal{I}_i,\quad i=0,1,\ldots,n, \tag{47}$$

where

$$\mathcal{I}_i = E\big[\exp\big(\boldsymbol{l}_i^\top \boldsymbol{\varepsilon}'\big)\big] = E\big[\exp\big(l_i^1\big(\varepsilon^1\big)' + l_i^2\big(\varepsilon^2\big)' + \cdots + l_i^N\big(\varepsilon^N\big)'\big)\big], \tag{48}$$

with $(l_i^1, \ldots, l_i^N) \equiv \boldsymbol{l}_i$ being the powers on $(z^1)', \ldots, (z^N)'$ in the ith monomial term, respectively.

As in the one-dimensional case, $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ depend only on the properties of exogenous shock $\boldsymbol{\varepsilon}'$ and can be computed up-front (i.e., precomputed). Once $\mathcal{I}$s are constructed, we again obtain that conditional expectation of a polynomial function is given by the same polynomial function but evaluated at a different coefficients vector, that is, $E[\mathcal{P}(\boldsymbol{k}', z'; b)] = \mathcal{P}(\boldsymbol{k}', \boldsymbol{z}^\rho; b')$; this result is parallel to what we had in the univariate case in Section 2.2.

3.3 *Characterizing the solution under precomputation of integrals*

We now show how to precompute expectations in the Bellman and Euler equations in the case of multivariate shocks.

3.3.1 *Bellman equation with precomputation of integrals* Using the precomputation result (46), we rewrite the Bellman equation (41) as

$$\widehat{V}(\boldsymbol{k}, z; b) \doteq \max_{\{(k^h)', c^h\}^{h=1,\ldots,N}} \left\{ \sum_{h=1}^{N} \tau^h u\big(c^h\big) + \beta\big[\widehat{V}\big(\boldsymbol{k}', z^\rho; b'\big)\big] \right\}, \tag{49}$$

$$\text{s.t.} \quad (42), (43), \text{ and } (46),$$

where a relation between b and b' is summarized by (47) and (48). To solve the transformed Bellman equation, we proceed in two steps. First, we construct $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (48); second, we find b that solves the Bellman equation (49).

3.3.2 *Euler equation with precomputation of integrals* Let us introduce a new variable q^h that represents the integrand of the expectation function in (44),

$$q^h \equiv u'\big(c^h\big)\big[1 - \delta + z^h f'\big(k^h\big)\big], \tag{50}$$

where $h = 1, \ldots, N$. In terms of q^h and $(q^h)'$, the Euler equation (44) is

$$\frac{q^h}{1 - \delta + z^h f'\big(k^h\big)} = \beta E\big[\big(q^h\big)'\big]. \tag{51}$$

We parameterize the function $q^h = Q^h(\boldsymbol{k}, z)$ with a flexible functional form $\widehat{Q}^h(\boldsymbol{k}, z; b^h)$, $h = 1, \ldots, N$. Using the precomputation result (46), we formulate a version of the Euler equation (51) in terms of $\widehat{Q}^h$,

$$\frac{\widehat{Q}^h\big(\boldsymbol{k}, z; b^h\big)}{1 - \delta + z^h f'\big(k^h\big)} \doteq \beta \widehat{Q}^h\big(\boldsymbol{k}', \boldsymbol{z}^\rho; \big(b^h\big)'\big), \tag{52}$$

where b^h and $(b^h)'$ are related by (47) and (48) for all $h = 1, \ldots, N$. To solve the transformed Euler equation, we proceed in two steps. First, we construct $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (48); second, we find b that solves (42), (43), (47), (48), and (52).

3.4 *Analytical versus numerical construction of integrals*

In this section, we compare the accuracy and computational expense of analytical and numerical approximation of integrals.

Analytical construction of integrals under precomputation For an important special case of a multivariate normal distribution $\boldsymbol{\varepsilon}' = ((\varepsilon^1)', \ldots, (\varepsilon^N)') \sim \mathcal{N}(0_N, \Sigma)$, we can compute $\mathcal{I}_i$ analytically. As in the unidimensional case, we complete the square of the expression inside of the exponential function,

$$\begin{aligned}
\mathcal{I}_i &= E\big[\exp\big(l_i^1(\varepsilon^1)' + \cdots + l_i^N(\varepsilon^N)'\big)\big] = E\big[\exp\big(\boldsymbol{l}_i^\top \boldsymbol{\varepsilon}'\big)\big] \\
&= \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \exp\big(\boldsymbol{l}_i^\top \boldsymbol{\varepsilon}'\big) \exp\left(-\frac{1}{2}\big(\boldsymbol{\varepsilon}'\big)^\top \Sigma^{-1} \boldsymbol{\varepsilon}'\right) d\boldsymbol{\varepsilon}' \\
&= \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \exp\left(-\frac{1}{2}\big\{\big(\boldsymbol{\varepsilon}'\big)^\top \Sigma^{-1} \boldsymbol{\varepsilon}' - 2\boldsymbol{l}_i^\top \boldsymbol{\varepsilon}'\big\}\right) d\boldsymbol{\varepsilon}' \\
&= \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} \exp\left(-\frac{1}{2}\big\{\big(\boldsymbol{\varepsilon}' - \Sigma \boldsymbol{l}_i\big)^\top \Sigma^{-1}\big(\boldsymbol{\varepsilon}' - \Sigma \boldsymbol{l}_i\big) - \boldsymbol{l}_i^\top \Sigma \boldsymbol{l}_i\big\}\right) d\boldsymbol{\varepsilon}' \\
&= \exp\left(\frac{\boldsymbol{l}_i^\top \Sigma \boldsymbol{l}_i}{2}\right),
\end{aligned} \tag{53}$$

where in the last equality, we use the fact that $\int_{-\infty}^{+\infty} \cdots \int_{-\infty}^{+\infty} f(\boldsymbol{x})\, d\boldsymbol{x} = 1$ for a density function $f(\boldsymbol{x})$ of a normally distributed variable $\boldsymbol{x}$ with mean $\Sigma \boldsymbol{l}_i$ and variance–covariance matrix Σ.

Numerical construction of integrals Similarly to the univariate case, multivariate numerical integration methods approximate integrals using a weighted sum (8), where the integration nodes and weights are constructed in the multidimensional space. Gauss Hermite quadrature can be extended to the multivariate case using a product rule; however, product rules are prohibitively costly even for a moderately large number of shocks N; in particular, the cost of GH(2), GH(5), and GH(10) grows exponentially as 2^N, 5^N, and 10^N, respectively. A tractable and sufficiently accurate alternative to product rules is nonproduct monomial integration methods; see Stroud (1971) for a collection of monomial formulas, and see Judd, Maliar, and Maliar (2011) for a description of two monomial integration methods with $2N$ and $2N^2 + 1$ nodes, referred to as M1 and M2, respectively. A quasi-Monte Carlo method is another class of numerical integration methods that is tractable in problems with high dimensionality; see Rust (1996) and Geweke (1996) for a discussion (we do not study such methods).

Comparison of the accuracy Constructing the expectation functions of polynomial approximations (46) includes evaluation of expectations of exponents of different linear combination of shocks in (48). Our numerical experiments show that for the same sum of the powers on $(z^1)', \ldots, (z^N)'$ in the ith monomial term, $l_i^1 + \cdots + l_i^N$, numerical integration methods produce the largest approximation errors whenever all l_is, except one, are zeros. For example, $E[\exp(2\varepsilon^1)]$ is approximated less accurately than $E[\exp(\varepsilon^1 + \varepsilon^2)]$;

TABLE 10. Accuracy of multivariate numerical integration methods.

Integral	Exact Value Under Precomputation	Numerical Integration Errors (%)				
		$N=2$	$N=5$	$N=10$	$N=20$	$N=30$
M1						
$E[\exp(\varepsilon')]$	1.0202	−0.0066	0.0132	0.0465	0.1145	0.1843
$E[\exp(2\varepsilon')]$	1.0833	−0.1044	0.2022	0.7353	1.8884	3.1659
$E[\exp(3\varepsilon')]$	1.1972	−0.5141	0.9569	3.6498	10.0497	18.0266
$E[\exp(4\varepsilon')]$	1.3771	−1.5615	2.7482	11.2132	34.0097	67.0056
$E[\exp(5\varepsilon')]$	1.6487	−3.6167	5.9206	26.3606	90.3823	200.4211
M2						
$E[\exp(\varepsilon')]$	1.0202	−0.0000	−0.0001	−0.0003	−0.0014	−0.0035
$E[\exp(2\varepsilon')]$	1.0833	−0.0027	−0.0044	−0.0183	−0.0917	−0.2321
$E[\exp(3\varepsilon')]$	1.1972	−0.0296	−0.0471	−0.2024	−1.0700	−2.8572
$E[\exp(4\varepsilon')]$	1.3771	−0.1549	−0.2458	−1.0888	−6.2073	−17.8343
$E[\exp(5\varepsilon')]$	1.6487	−0.5399	−0.8522	−3.9235	−24.6011	−77.4623

Note: M1 and M2 are the monomial integration methods with $2N$ and $2N^2+1$ nodes, respectively.

similarly, $E[\exp(5\varepsilon^1)]$ is approximated less accurately than $E[\exp(\varepsilon^1+\varepsilon^2+\varepsilon^3+\varepsilon^4+\varepsilon^5)]$ and $E[\exp(2\varepsilon^1+3\varepsilon^2)]$. Therefore, the upper bound on the integration errors in the multivariate case can be seen from the univariate analysis reported in Table 1 for both the Monte Carlo and Gauss Hermite quadrature methods, and this upper bound will be the same for any N.

Hence, in the multivariate case, we restrict attention to assessing the accuracy of nonproduct monomial integration methods that we did not study in the univariate case. The results for two monomial rules are reported in Table 10. For $\sigma=0.2$, we observe three tendencies: First, the M1 monomial rule with $2N$ nodes produces less accurate approximation than the M2 rule with $2N^2+1$ nodes; this result is not surprising since more nodes normally provide a more accurate representation of functions for constructing averages. Second, the accuracy decreases with the order of exponentiation; this is because monomial formulas are constructed to integrate accurately some low-order monomials but not high-order monomials. Finally, the accuracy of approximation of integrals decreases with the number of shocks N; presumably, this is because the number of nodes grows less rapidly with N than the volume of multidimensional space.

For low volatility of shocks $\sigma=0.02$, the monomial methods are very accurate; in particular, M1 and M2 deliver respectively three and six digits of accuracy even for $N=30$. The Monte Carlo integration methods are considerably less accurate and unit-free residuals can be 100% or larger in a typical simulation (again, the results for $\sigma=0.02$ are not reported).

Comparison of costs In Table 11, we report the cost of evaluation of expectation of complete ordinary polynomial (46). In the second column, we provide the time for the evaluation of the expectation function using analytical result (53), and in the remaining columns, we report the factors by which the analytical precomputation of integrals differs from considered numerical integration methods. The cost of product rules GH(2)

TABLE 11. Cost of multivariate numerical integration methods.

Polynomial Degree	Evaluation Time Under Precomputation (sec)	Speedup From Precomputation					
		GH(2)	GH(5)	M1	M2	MC(100)	MC(10,000)
$N = 2$							
1st	0.1802(−3)	×2.04	×5.27	×1.93	×2.81	×17.23	×1469.1
2nd	0.1841(−3)	×2.30	×7.06	×1.88	×2.88	×27.86	×2383.3
3rd	0.1838(−3)	×2.70	×12.85	×2.38	×4.49	×49.07	×4443.7
4th	0.2941(−3)	×2.92	×17.01	×2.80	×6.02	×66.28	×6040.4
5th	0.5122(−3)	×3.44	×21.61	×3.39	×7.69	×84.47	×7979.9
$N = 5$							
1st	0.1835(−3)	×4.59	×471.8	×2.04	×5.73	×10.63	×1034.9
2nd	0.1786(−3)	×10.12	×922.6	×2.52	×9.90	×19.14	×3026.8
3rd	0.1158(−2)	×28.71	×3309.4	×6.84	×34.88	×68.86	×10,595.3
4th	0.1923(−2)	×28.54	×3369.5	×6.86	×34.87	×68.52	×10,477.9
$N = 10$							
1st	0.2088(−3)	×112.2	–	×3.64	×23.45	×12.32	×1320.7
2nd	0.2210(−3)	×227.1	–	×5.19	×47.49	×22.05	×2553.7
3rd	0.2523(−2)	×1170.9	–	×19.42	×233.99	×95.96	×11,509.3
$N = 20$							
1st	0.1556(−3)	×182,255.0	–	×8.29	×124.33	×16.10	–
2nd	0.1780(−3)	×527,079.2	–	×19.36	×364.32	×42.07	–
$N = 30$							
1st	0.1661(−3)	–	–	×10.04	×246.32	×15.43	–
2nd	0.2361(−3)	–	–	×32.89	×923.70	×53.68	–

Note: The second column shows the time for evaluating a complete polynomial of a given degree by using the precomputation method (in seconds). Numbers in the third through seventh columns are factors by which the evaluation time of the given integration method differ from that of the precomputation method. GH(2) and GH(5) are the Gauss Hermite quadrature methods with 2 and 5 nodes, respectively; M1 and M2 are the monomial integration methods with $2N$ and $2N^2 + 1$ nodes, respectively; MC(100) and MC(10,000) are the Monte Carlo integration methods with 100 and 10,000 nodes, respectively.

and GH(5) increases exponentially, and these methods become too expensive for $N = 10$ and $N = 20$, even for low-order polynomials. In contrast, the cost of monomial formulas M1 and M2 grows polynomially, linearly, and quadratically, respectively. The monomial formulas are tractable for very large N although they can become orders of magnitude more expensive than precomputation of integrals. Finally, the cost of the Monte Carlo integration methods does not grow rapidly with the number of shocks because the number of nodes is fixed but the computer runs out of memory for large N. We must emphasize that to construct numerical solutions to dynamic economic models, we must evaluate integrals many times in an iterative cycle and the corresponding difference in costs will be much larger.

3.5 *Numerical assessment of gains from precomputation in the multi-agent model*

In the context of the generalized stochastic simulation method (GSSA), Judd, Maliar, and Maliar (2011) found that there is a significant trade-off between the accuracy and cost of numerical solutions depending on the integration method used (Monte Carlo,

Gaussian quadrature, and monomial integration methods). Precomputation of integrals eliminates this trade-off: it allows us to produce the most accurate solutions (where integrals are evaluated exactly using closed-form expressions) at the lowest possible cost. We introduce precomputation of integrals into a MATLAB code accompanying the paper by Judd, Maliar, and Maliar (2011), and we assess the improvements in the accuracy and cost in the model with up to $N = 30$ heterogeneous countries (60 state variables). The numerical methods used in this section are elaborated in Appendix C.

3.5.1 *Generalized stochastic simulation algorithm* In the representative agent model, we construct numerical solutions on two-dimensional tensor product grids. However, tensor product grids are not tractable for problems with a large number of state variables. Therefore, for the multicountry model, we use a grid of points produced by stochastic simulation. The advantage of such a grid is that it covers only the high probability area of state space, and we avoid constructing the solution in an enormously large set of points that have extremely low probability of occurrence; see Judd, Maliar, and Maliar (2011) for a discussion.

However, the critical issue is how to evaluate expectation functions in the context of simulation-based methods. For example, Den Haan and Marcet (1990) propose a parameterized expectation algorithm (PEA) that approximates the expectation function in (8) by using a Monte Carlo method with a single random draw, MC(1), that is, $E_t[G(\varepsilon'_{t+1})] \approx G(\varepsilon'_{t+1})$, where $\varepsilon'_{t+1} \sim \mathcal{N}(0, \sigma^2)$. To some extent, the sampling errors of this crude integration method are offset by one another on the regression step; however, the low quality of approximation of integrals still dramatically restricts the overall accuracy of the PEA solutions; see Judd, Maliar, and Maliar (2011) and Maliar and Maliar (2014) for numerical illustrations. Judd, Maliar, and Maliar (2011) propose the GSSA algorithm that also uses simulation points as a grid for constructing a solution but that approximates integrals accurately by using deterministic integration methods (such as quadrature and monomial methods). GSSA produces highly accurate solutions, and we use it as a benchmark for comparison to the precomputation-based methods.

3.5.2 *Methodology and implementation details* We parameterize the model by $u(c_t^h) = \ln c_t^h$, $f(k_t) = Ak_t^\alpha$ with $\alpha = 0.36$, and $A = \frac{1/\beta-(1-\delta)}{\alpha}$; this value normalizes the deterministic steady state of capital to unity for each country. We assume that $\beta = 0.99$, $\delta = 0.025$, $\rho = 0.95$, and $\sigma = 0.01$, and we use equal Pareto weights across all countries $\tau^h = 1$, which implies that all countries have identical consumption, $c_t^h = c_t$ for all $h = 1, \ldots, N$. We assume that each country's shock in (40) has both common-for-all-countries and country-specific components, $\varepsilon_t^h \equiv \varepsilon_t + \varpi_t^h$ with $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ and $\varpi_t^h \sim \mathcal{N}(0, \sigma^2)$. That is, countries' shocks follow a multivariate normal distribution $(\varepsilon_t^1, \ldots, \varepsilon_t^N) \sim \mathcal{N}(0_N, \Sigma)$, where $0_N \in \mathbb{R}^N$ is a vector of zero means and

$$\Sigma = \begin{pmatrix} 2\sigma^2 & \ldots & \sigma^2 \\ \ldots & \ldots & \ldots \\ \sigma^2 & \ldots & 2\sigma^2 \end{pmatrix} \in \mathbb{R}^{N\times N}$$

is a variance–covariance matrix.

3.5.3 *Euler equation algorithm with precomputation of integrals* For the multicountry case, we implement an Euler equation method that approximates N capital policy functions $k_{t+1}^h = K^h(\boldsymbol{k}_t, z_t)$ jointly with N expectation functions $Q^h(\boldsymbol{k}_t, z_t)$ for $h = 1, \ldots, N$ (this method is similar to the Euler equation method that parameterizes both capital function K and the integrand function Q in the univariate case). We parameterize each capital and integrand function by a polynomial function of type (45), $\widehat{K}^h(\boldsymbol{k}_t, z_t; v^h)$ and $\widehat{Q}^h(\boldsymbol{k}_t, z_t; b^h)$, respectively, where v^h and b^h are the coefficients vectors, and we rewrite (51) as

$$k_{t+1}^h \doteq \beta \frac{E\big[\widehat{Q}^h\big(\boldsymbol{k}_{t+1}, z_{t+1}; b^h\big)\big]}{\widehat{Q}^h\big(\boldsymbol{k}_t, z_t; b^h\big)}\big[1 - \delta + z_t^h f'\big(k_t^h\big)\big]k_{t+1}^h. \tag{54}$$

Our first solution method is a variant of GSSA that recomputes expectation functions in (54) at each iteration using one of the following three numerical integration methods: (i) a Gauss Hermite quadrature rule with 2^N nodes, (ii) monomial rules with $2N$ nodes, and (iii) monomial rule $2N^2 + 1$ nodes, referred to as GH(2), M1 and M2, respectively.

Our second solution method relies on precomputation of integrals in (54), namely, we use (52) that is rewritten in a way parallel to (54):

$$k_{t+1}^h \doteq \beta \frac{\widehat{Q}^h\big(\boldsymbol{k}_{t+1}, z_t^\rho; \big(b^h\big)'\big)}{\widehat{Q}^h\big(\boldsymbol{k}_t, z_t; b^h\big)}\big[1 - \delta + z_t^h f'\big(k_t^h\big)\big]k_{t+1}^h. \tag{55}$$

Integrals $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ in $\widehat{Q}^h(\boldsymbol{k}_{t+1}, z_t^\rho; (b^h)')$ of (55) are precomputed using closed-form expression (53).

In Table 12, we report the solution to the model with $N = 2$, 5, 10, 20, and 30 countries. As a measure of accuracy, we report the mean and maximum of absolute unit-free Euler equation residuals on a stochastic simulation of 10,000 observations. Again, to avoid approximation errors from numerical integration when evaluating accuracy, we construct residuals in the transformed Euler equation (55) by using the precomputation of integrals. We observe the following tendencies. First, all integration methods considered produce the same mean and maximum Euler equation residuals at least up to the third digit. The solutions are very accurate: for $N = 2$, the maximum residuals range from about -3 to -6 for polynomial approximations of degrees from 2 to 5, respectively. Therefore, in this case, gains from precomputation of integrals come primarily in terms of a cost reduction.

As we see in the table, the savings in costs from precomputation of integrals depend on the dimensionality of the problem, the degree of polynomial approximation, and a specific numerical integration method used for comparison. The cost of the GH(2) method grows exponentially with N; as expected, this method is intractable even for moderately large models. The cost of the M1 and M2 methods grows polynomially with N, but these methods still become increasingly expensive relatively to the precomputation method. Only precomputation-based method was able to produce a quadratic solution to the model with $N = 30$ countries in a reasonable time.

TABLE 12. Accuracy and cost of the Euler equation algorithm for the multicountry model: GSSA parameterizing Q and K.

Polynomial Degree	Precomputation			M1			M2			GH(2)		
	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
$N=2$												
1st	−2.77	−1.81	61	−2.77	−1.81	83	−2.77	−1.81	98	−2.77	−1.81	87
2nd	−3.88	−2.61	223	−3.88	−2.61	379	−3.88	−2.61	605	−3.88	−2.61	397
3rd	−4.94	−3.55	382	−4.94	−3.55	558	−4.94	−3.55	811	−4.94	−3.55	583
4th	−6.05	−4.68	574	−6.05	−4.68	1361	−6.05	−4.68	1215	−6.05	−4.68	1164
5th	−7.15	−5.79	738	−7.15	−5.79	2068	−7.15	−5.79	1829	−7.15	−5.79	1728
$N=5$												
1st	−2.86	−1.94	75	−2.86	−1.94	159	−2.86	−1.94	467	−2.86	−1.94	291
2nd	−3.99	−2.81	319	−3.99	−2.81	927	−3.99	−2.81	3443	−3.99	−2.81	2294
3rd	−5.10	−3.84	2550	−5.10	−3.84	10,349	−5.10	−3.84	13,179	−5.10	−3.84	6246
4th	−6.18	−4.89	7033	−6.18	−4.89	22,876	–	–	–	–	–	–
$N=10$												
1st	−2.87	−1.89	125	−2.87	−1.89	385	−2.87	−1.89	3308	−2.87	−1.89	12,405
2nd	−4.00	−2.80	666	−4.00	−2.80	2465	−4.00	−2.80	20,989	–	–	–
3rd	−4.99	−3.88	14,837	−4.99	−3.88	20,006	–	–	–	–	–	–
$N=20$												
1st	−3.12	−2.09	152	−3.12	−2.09	1324	−3.12	−2.09	22,084	–	–	–
2nd	−4.36	−3.26	3303	−4.36	−3.26	13,840	–	–	–	–	–	–
$N=30$												
1st	−3.15	−2.08	221	−3.15	−2.08	2784	–	–	–	–	–	–
2nd	−4.22	−3.22	13,543	–	–	–	–	–	–	–	–	–

Note: The main columns correspond to variants of GSSA that evaluate integrals by using the precomputation method, the monomial integration methods with $2N$ and $2N^2+1$ nodes and the Gauss Hermite quadrature method with two nodes, respectively; the statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in log 10 units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds).

3.5.4 *Conventional Euler equation algorithm without precomputation* For the sake of comparison, we also implement the conventional GSSA method as in Judd, Maliar, and Maliar (2011). Such a method parameterizes N capital policy functions $k^h_{t+1} = K^h(\boldsymbol{k}_t, \boldsymbol{z}_t; v^h)$ using

$$k^h_{t+1} = \beta E\left[\frac{u'(c_{t+1})}{u'(c_t)}\left[1-\delta+z_{t+1}f'\left(k^h_{t+1}\right)\right]\right]k^h_{t+1}. \tag{56}$$

GSSA recomputes expectation functions in each iteration by using some numerical integration method. To save on space, we report the results for this method in Table C.1 of Appendix C. For this method, we also compute the solutions to the model with $N=2$, 5, 10, 20, and 30 countries.

The accuracy of solutions produced by conventional GSSA is very similar to that produced by the Euler equation method with precomputation of integrals. The running time depends on the specific experiment considered and it is generally larger for the conventional GSSA method than for the GSSA method with precomputation of in-

tegrals. Importantly, the running time of conventional GSSA increases far more rapidly with the dimensionality of the problem than the time of the GSSA method with precomputation of integrals. Again, only a precomputation-based method was able to produce a quadratic solution to the model with $N = 30$ countries in a reasonable time. We conclude that precomputation of integrals can significantly reduce the computational expense of high-dimensional problems and it can help us solve models that are intractable with conventional solution methods.

4. Discrete-state expectations

In the previous sections, we focused on precomputing expectation functions in models in which shocks belong to continuous intervals. In this section, we show precomputation results for the discrete-shock case. As an example, we analyze a general equilibrium growth model that has idiosyncratic but no aggregate risk, in line with Bewley (1977), Deaton (1991), Carroll (1992), Huggett (1993), and Aiyagari (1994); our analysis is closest to the last paper.

4.1 *A growth model with discrete shocks*

The economy is populated by a continuum of ex ante identical agents uniformly distributed on interval $[0, 1]$. The agents supply inelastically their total time endowment (equal to 1) to the market. The labor productivities of agents follow a first-order Markov chain. The agents maximize expected lifetime utility by choosing consumption and asset holdings (to simplify notation, we omit the agent's subscript)

$$\max_{\{c_t,k_{t+1}\}_{t=0}^{\infty}} \quad E_0\left[\sum_{t=0}^{\infty}\beta^t u_t(c_t)\right] \tag{57}$$

$$\text{s.t.} \quad c_t + k_{t+1} = Wz_t + (1+R)k_t, \tag{58}$$

$$\pi_{j\ell} = \Pr(z_{t+1} = z_j | z_t = z_\ell) \text{ is given}, \tag{59}$$

$$k_{t+1} \geq -\phi, \tag{60}$$

where c_t, k_t, and z_t denote consumption, capital, and labor productivity, respectively, R and W are the interest rate and real wage, respectively, $\phi > 0$ is a borrowing limit, $\beta \in (0, 1)$ is the discount factor, $z_t \in \{z_1, \ldots, z_J\}$, and J is a finite number of possible productivity states, $0 < z_1 < \cdots < z_J < \infty$. Transition probability $\pi_{j\ell}$ shows the probability that tomorrow's state is j given that today's state is ℓ, where $j, \ell \in \{1, \ldots, J\}$. To ensure the existence of a solution, we assume that $\beta(1 + R) < 1$ and that k_t is bounded from above by $\overline{k}$. The production side consists of a representative firm that owns a production technology $f(K_t, N_t) + (1 - \delta)K_t$ and maximizes period-by-period profits, where K_t is capital, N_t is labor, $\delta \in (0, 1]$, and u and f are strictly increasing, continuously differentiable, and strictly concave.

Bellman equation In the model (57)–(60) with J exogenous states, the optimal value function can be represented by J state-contingent value functions $V^j(k)$ that satisfy the Bellman equation

$$V^\ell(k) = \max_{k',c} \left\{ u(c) + \beta \sum_{j=1}^{J} \pi_{j\ell} V^j\big(k'\big) \right\}, \tag{61}$$

$$\text{s.t.} \quad k' = W z_\ell + (1+R)k - c, \tag{62}$$

$$k' \geq -\phi. \tag{63}$$

where $j, \ell \in \{1, \ldots, J\}$ and $\pi_{j\ell}$ satisfies (59).

Euler equation The policy functions for the problem (57)–(60) can also be represented as J state-contingent functions. A consumption policy function $C^j(k)$ satisfies the Euler equation

$$u'\big(C^\ell(k)\big) - \eta = \beta \sum_{j=1}^{J} \pi_{j\ell} \big\{ u'\big(C^j\big(k'\big)\big)(1+R) \big\}, \tag{64}$$

where $j, \ell \in \{1, \ldots, J\}$ and $\eta \geq 0$ is a Lagrange multiplier, which is associated with the borrowing constraint (63), that satisfies a Kuhn–Tucker condition $\eta(k' + \phi) = 0$.

4.2 *Precomputation of expectation functions*

Let us show how to precompute conditional expectation functions under the assumption of ordinary polynomial approximation. Each state-contingent value or policy function is approximated by a polynomial function

$$\mathcal{P}\big(k; b^\ell\big) = b_0^\ell + b_1^\ell k + b_2^\ell k^2 + \cdots + b_n^\ell k^L, \tag{65}$$

where $\ell = 1, \ldots, J$, $b^\ell \equiv (b_0^\ell, b_1^\ell, \ldots, b_n^\ell) \in \mathbb{R}^{n+1}$ is a vector of polynomial coefficients, and L is the degree of polynomial. Given today's state ℓ, conditional expectation of (65) is

$$\begin{aligned} E\big[\mathcal{P}\big(k'; b^j\big) | k, z_\ell\big] &= \sum_{j=1}^{J} \pi_{j\ell} \mathcal{P}\big(k'; b^j\big) = \pi_{1\ell} \mathcal{P}\big(k'; b^1\big) + \cdots + \pi_{J\ell} \mathcal{P}\big(k'; b^J\big) \\ &= \pi_{1\ell}\big[b_0^1 + b_1^1 k' + \cdots + b_n^1 \big(k'\big)^L\big] + \cdots \\ &\quad + \pi_{J\ell}\big[b_0^J + b_1^J k' + \cdots + b_n^J \big(k'\big)^L\big] \\ &= \big[\pi_{1\ell} b_0^1 + \cdots + \pi_{J\ell} b_0^J\big] + \big[\pi_{1\ell} b_1^1 + \cdots + \pi_{J\ell} b_1^J\big] k' + \cdots \\ &\quad + \big[\pi_{1\ell} b_n^1 + \cdots + \pi_{J\ell} b_n^J\big] \big(k'\big)^L \\ &= \mathcal{P}\big(k'; \big(b^\ell\big)'\big), \end{aligned} \tag{66}$$

where $(b^\ell)' \equiv ((b_0^\ell)', (b_1^\ell)', \ldots, (b_n^\ell)') \in \mathbb{R}^{n+1}$ is given by

$$\left(b_i^\ell\right)' = \sum_{j=1}^{J} \pi_{j\ell} b_i^j, \quad i = 0, 1, \ldots, n. \tag{67}$$

Condition (67) provides a basis for precomputation of conditional expectation functions in the discrete-state case. As in the continuous-state case, conditional expectation of a polynomial function is given by the same polynomial function, but evaluated at a different coefficients vector, that is, $E[\mathcal{P}(k'; b^j)|k, z_\ell] = \mathcal{P}(k'; (b^\ell)')$, where $(b^\ell)'$ is determined by (67).

Remark 4. Similar to the continuous-state case, the precomputation result (66) and (67) also holds for piecewise polynomial approximations. As an example, consider again a collection of piecewise linear polynomial functions on a grid $[k_1, \ldots, k_M]$ such that in each state $\ell = 1, \ldots, J$, we have a local polynomial function

$$\mathcal{P}_{[k_m, k_{m+1}]}\left(k; b^{(\ell,m)}\right) = b_0^{(\ell,m)} + b_1^{(\ell,m)} k,$$

where $m \in \{1, \ldots, M\}$. Then the expectation function in each interval $[k_m, k_{m+1}]$ is given by

$$E\left[\mathcal{P}_{[k_m, k_{m+1}]}\left(k'; b^{(\ell,m)}\right)\right] = \sum_{j=1}^{J} \pi_{j\ell}\left(b_0^{(j,m)} + b_1^{(j,m)} k'\right) = \left(b_0^{(\ell,m)}\right)' + \left(b_1^{(\ell,m)}\right)' k',$$

where $(b_0^{(\ell,m)})' = [\pi_{1\ell} b_0^{(1,m)} + \cdots + \pi_{J\ell} b_0^{(J,m)}]$ and $(b_1^{(\ell,m)})' = [\pi_{1\ell} b_1^{(1,m)} + \cdots + \pi_{J\ell} b_1^{(J,m)}]$, i.e., (66) and (67) hold in each local area. The precomputation results for higher-order piecewise polynomial approximations can be shown in a similar manner.

4.3 *Characterizing the solution under precomputation of expectation functions*

In this section, we show how to precompute expectations in the Bellman and Euler equations in the discrete-shock case.

4.3.1 *Bellman equation with precomputation of expectation functions* Using precomputation result (66), we rewrite the Bellman equation (61)–(63) as

$$\widehat{V}\left(k; b^\ell\right) \doteq \max_{k', c} \left\{u(c) + \beta \widehat{V}\left(k'; \left(b^\ell\right)'\right)\right\} \tag{68}$$

$$\text{s.t.} \quad (62), (63), (66), \text{ and } (67),$$

where $\ell \in \{1, \ldots, J\}$. In the transformed Bellman equation (68), the effect of uncertainty on the solution is captured by a relation between b^j and $(b^\ell)'$, which is described by (66) and (67).

4.3.2 *Euler equation with precomputation of expectation functions* To make the Euler equation (64) suitable for precomputation of expectation functions, we use the same change of variables as in the continuous-state case:

$$q \equiv u'(c)[1+R]. \tag{69}$$

We next rewrite the Euler equation (64) and budget constraint (62) by eliminating c and by expressing them in terms of q approximated using a polynomial function $Q(k; b^j)$,

$$\frac{Q(k; b^\ell)}{1+R} - \eta = \beta \sum_{j=1}^{J} \pi_{j\ell} Q\big(k'; b^j\big), \tag{70}$$

where $j, \ell \in \{1, \ldots, J\}$ and Lagrange multiplier η satisfies the same conditions as in (64). Finally, under polynomial approximation (65), we can use the precomputation result (66) to rewrite the Euler equation (70) as

$$\frac{\widehat{Q}(k; b^\ell)}{1+R} - \eta \doteq \beta \widehat{Q}\big(k'; \big(b^\ell\big)'\big), \tag{71}$$

where $\ell \in \{1, \ldots, J\}$ and $(b^\ell)'$ is determined by (67). Again, after expectation functions are precomputed, all the effect of uncertainty on the solution is compressed into a mapping between the vectors b^ℓ and $(b^\ell)'$, which is described by (66) and (67).

4.4 *Numerical assessment of gains from precomputation of expectation functions in Aiyagari's (1994) model*

We assess gains from precomputation of expectation functions in Aiyagari's (1994) model. With discrete-state shocks, expectation functions are computed exactly both with and without precomputation analysis. Thus, all the gains will come in terms of a cost reduction, unlike in the case of continuous-state shocks, in which we may have both a lower cost and higher accuracy. We consider a version of the Euler equation solution method described in Maliar and Maliar (2006), and we show how to precompute expectation functions in the MATLAB code accompanying that paper. A detailed description of the solution algorithm is provided in Appendix D.

4.4.1 *Methodology and implementation details* Following the literature, we study a stationary equilibrium in which aggregate variables are constant over time. A stationary equilibrium is defined as a stationary probability measure x, optimal consumption and capital policy functions $C^j(k)$ and $K^j(k)$, respectively, $j = 1, \ldots, J$, and positive real number (R, W) such that

(i) x satisfies $x = \int_{\mathcal{K}\times\mathcal{Z}} P(k, z, B)\, dx$ for all $B \in \mathcal{B}$,

(ii) $C^j(k)$ and $K^j(k)$ solve (57)–(60) for given (R, W),

(iii) $R = f_1(K, N) - \delta$ and $W = f_2(K, N)$,

(iv) $N = \int_{\mathcal{Z}} z_t\, dx$ and $K = \int_{\mathcal{K}\times\mathcal{Z}} K(k, z)\, dx$,

where $\mathcal{Z} \equiv \{z_1, \ldots, z_J\}$ is a set of all possible productivity levels, $\mathcal{K} \equiv [-\phi, \overline{k}]$ is an interval of possible asset holdings, $\mathcal{B}$ is a collection of Borel subsets of all possible individual states $\mathcal{K} \times \mathcal{Z}$, and $P(k, z, B)$ is conditional probability that an agent with today's state (k, z) will be in a set $B \in \mathcal{B}$ in the next period.

To approximate the capital decision functions $K^j(k)$, $j = 1, \ldots, J$, we use piecewise linear polynomial approximation. We use an unevenly spaced grid of 1000 points: we placed a dense grid of 900 grid point in the area of the borrowing limit in which the solution is highly nonlinear, and we placed the remaining 100 grid points to cover the rest of the solution domain. Precomputation of expectation functions can be easily implemented in MATLAB in the one-dimensional case by using a routine *interp1*, which produces linear, cubic, and spline approximations, and which returns both the knots and coefficients. We take these coefficients as input, and we construct the precomputation coefficients as implied by (66) and (67).

In our calibration procedure, we follow Aiyagari (1994). The model's period is 1 year. We assume $u(c_t) = \frac{c_t^{1-\gamma}-1}{1-\gamma}$ with $\gamma \in \{\frac{1}{3}, 3\}$ and $f(K, N) = K^\alpha N^{1-\alpha}$ with $\alpha = 0.36$ (we use a normalization $N = 1$ for convenience). We use $\beta = 0.96$ and $\delta = 0.08$. We set the debt limit at $\phi = 0$ and the upper bound on capital at $\overline{k} = \delta^{1/(\alpha-1)}$. As in Aiyagari (1994), we assume that idiosyncratic shocks follow an AR(1) process: $\log z_{t+1} = \rho \log z_t + \sigma(1-\rho^2)^{1/2}\varepsilon_{t+1}$, where $\rho \in \{0.6, 0.9\}$, $\sigma \in \{0.2, 0.4\}$, and $\varepsilon_{t+1} \sim \mathcal{N}(0, 1)$, and we discretize this process into a seven-state Markov chain using Tauchen's (1986) procedure; see also Tauchen and Hussey (1991) for a discussion of this method.[9] We solve for the equilibrium interest rate by using stochastic simulation and a bisection method as in Aiyagari (1994), and we construct stationary probability distribution as in Rios-Rull (1997). Finally, as a measure of accuracy, we report the mean and maximum of absolute unit-free residuals in the transformed Euler equation (71) on a stochastic simulation of 10,000 observations.

4.4.2 *Numerical results* We illustrate a numerical solution to Aiyagari's (1994) model in Figure 1.

The properties of the solution to Aiyagari's (1994) model are well known. The capital function is close to linear except in the kink area, and the consumption function has strong nonlinearities in the area of the kink. The constructed probability distribution indicates that our upper bound is chosen large enough so that probability of reaching this bound is low (in our simulations, it was actually never reached). But the limit on borrowing is important and is occasionally reached.

In Table 13, we compare the accuracy and cost for two versions of the algorithm with and without precomputation of expectation functions.

As we can see, the residuals are again practically identical for the methods with and without precomputation. The solutions are very accurate everywhere in the solution domain (average error is of order 10^{-6} percent), except for a very close neighborhood of the

[9]We discretize the process for shocks to provide a numerical illustration of the precomputation technique for discrete-state shocks. Alternatively, we could have solved Aiyagari's (1994) model by applying the precomputation technique to the original AR(1) process with continuous-state shock.

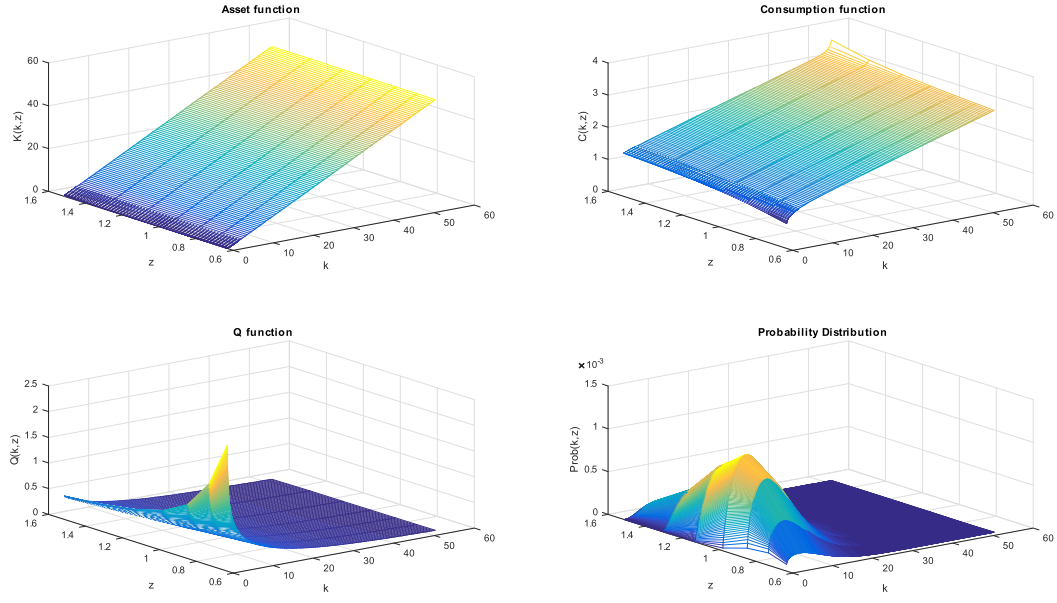

FIGURE 1. Numerical solution to Aiyagari's (1994) model.

borrowing limit, where the maximum residuals are much larger and sensitive to small changes in the construction of the numerical solutions.

To gain insight into the accuracy deterioration near the kink, we note that q has a spike in the lowest asset and productivity states in Figure 1. As follows from the ECM analysis in Section 2.3.3, q is the marginal value function, and it is largest when consumption is lowest. It is typically better to approximate numerically functions that do not have much curvature, for example, it is usually better to approximate consumption rather than the marginal utility of consumption (especially with high risk aversion). This reasoning suggests that a highly nonlinear variable q may not be the best candidate for approximating in the context of Aiyagari's (1994) model.

To check on this conjecture, namely, to see how the choice of a function to parameterize affects the accuracy of solutions, we also solved the model by using the stan-

TABLE 13. Accuracy and cost of the Euler equation algorithm with and without precomputation.

	$\gamma = 1/3$						$\gamma = 3$					
	No Precomputation			Precomputation			No Precomputation			Precomputation		
Parameterization	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
$\rho = 0.6, \sigma = 0.2$	−6.34	−2.99	150.3	−6.34	−2.99	72.6	−4.93	−2.42	154.7	−4.93	−2.42	72.8
$\rho = 0.6, \sigma = 0.4$	−6.18	−3.58	140.6	−6.18	−3.58	70.0	−4.67	−1.87	143.3	−4.67	−1.87	68.1
$\rho = 0.9, \sigma = 0.2$	−6.78	−3.14	151.0	−6.78	−3.14	68.8	−5.36	−2.71	155.6	−5.36	−2.71	71.2
$\rho = 0.9, \sigma = 0.4$	−6.62	−3.17	135.7	−6.62	−3.17	65.9	−5.04	−1.96	134.4	−5.04	−1.96	65.1

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in log 10 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.

TABLE 14. Accuracy and cost of the Euler equation algorithm parameterizing K.

	$\gamma = 1/3$			$\gamma = 3$		
Parameterization	L_1	L_∞	CPU	L_1	L_∞	CPU
$\rho = 0.6$, $\sigma = 0.2$	−6.23	−3.04	111.5	−4.75	−3.02	115.4
$\rho = 0.6$, $\sigma = 0.4$	−6.05	−3.67	104.2	−4.46	−1.88	106.6
$\rho = 0.9$, $\sigma = 0.2$	−6.67	−3.16	109.3	−5.15	−2.93	115.3
$\rho = 0.9$, $\sigma = 0.4$	−6.47	−3.07	100.9	−4.82	−2.10	100.0

Note: The statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in $\log 10$ 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.

dard Euler equation algorithm that parameterizes only the capital policy function and that sidesteps precomputation. In Table 14, we report the average and maximum Euler equation errors produced by this standard method.

The maximum residuals produced by the method parameterizing k' are somewhat smaller than those produced by the algorithm parameterizing q, although the average residuals are very similar. These results suggest that both methods are very accurate away from the kink, but their accuracy deterioration near the kink is somewhat larger for the method parameterizing q than for the method parameterizing k'. More generally, these result suggest that in some problems, it might be preferable to approximate decision functions others than q and to compute expectations in the conventional way even if this implies a higher cost.

Our main result is that precomputation of expectation functions reduces the running time by about 50%. This reduction in running times corresponds to a Markov chain with seven states that we used following Aiyagari (1994); see also Maliar, Maliar, and Valli (2010). If we use a Markov chain with more states, the savings in costs from precomputation of expectation functions will be much larger. We conjecture that the gains from precomputation of expectations functions will be especially sizable in models with multivariate shocks in which Markov chains may have a very large number of states.

5. APPROXIMATING FUNCTIONS CONSISTENT WITH PRECOMPUTATION OF EXPECTATION FUNCTIONS

All our precomputation results are obtained for ordinary polynomial functions. In this section, we establish other families of approximating functions for which expectation functions can be precomputed. The cases of continuous-state and discrete-state exogenous shocks are analyzed in Sections 5.1 and 5.2, respectively.

5.1 *Continuous-state problems*

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.

ASSUMPTION 1. *Each basis function of an approximating polynomial is multiplicatively separable in x and z, that is,*

$$\mathcal{P}(x, z; b) = \sum_{i=0}^{n} b_i \psi_i(x) \varphi_i(z), \tag{72}$$

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

ASSUMPTION 2. *Next-period endogenous state variables x' are t-measurable.*

ASSUMPTION 3. *Exogenous state variables follow a stochastic process $z' = Z(z, \varepsilon')$, where $\varepsilon' \in \Omega \subseteq \mathbb{R}^{n_\varepsilon}$ is a vector of disturbances with a density function $w : \mathbb{R}^{n_\varepsilon} \to \mathbb{R}^+$ that is known at t and that satisfies $\int_{\varepsilon' \in \Omega} w(\varepsilon')\, d\varepsilon' = 1$. Moreover, all integrals of type $\int_{\varepsilon' \in \Omega} \varphi_i(Z(z, \varepsilon'))w(\varepsilon')\, d\varepsilon'$ exist and are finite, up to the order of approximation in* (72).

PROPOSITION 1. *Under Assumptions* 1–3, *we have*

$$E\big[\mathcal{P}\big(x', z'; b\big)|x, z\big] = \mathcal{P}\big(x', Z(z, \mu); b'(z)\big), \tag{73}$$

where $\mu \equiv E[\varepsilon']$ and $b'(z) \equiv (b'_0(z), \ldots, b'_n(z))$, and b are related by

$$b'_i(z) = b_i \int_{\varepsilon' \in \Omega} \frac{\varphi_i\big(Z\big(z, \varepsilon'\big)\big)}{\varphi_i\big(Z(z, \mu)\big)} w\big(\varepsilon'\big)\, d\varepsilon'. \tag{74}$$

PROOF. This result is obtained as

$$\begin{aligned}
E\big[\mathcal{P}\big(x', z'; b\big)|x, z\big] &= \int_{\varepsilon' \in \Omega} \mathcal{P}\big(x', z'; b\big) w\big(\varepsilon'\big)\, d\varepsilon' \\
&\overset{\text{Assumption 1}}{=} \int_{\varepsilon' \in \Omega} \sum_{i=0}^{n} b_i \psi_i\big(x'\big)\varphi_i\big(z'\big) w\big(\varepsilon'\big)\, d\varepsilon' \\
&\overset{\text{Assumption 2}}{=} \sum_{i=0}^{n} b_i \psi_i\big(x'\big) \int_{\varepsilon' \in \Omega} \varphi_i\big(Z\big(z, \varepsilon'\big)\big) w\big(\varepsilon'\big)\, d\varepsilon' \\
&\overset{\text{Assumption 3}}{=} \sum_{i=0}^{n} b_i \psi_i\big(x'\big)\varphi_i\big(Z(z, \mu)\big) \int_{\varepsilon' \in \Omega} \frac{\varphi_i\big(Z\big(z, \varepsilon'\big)\big)}{\varphi_i\big(Z(z, \mu)\big)} w\big(\varepsilon'\big)\, d\varepsilon' \\
&\overset{(74)}{=} \sum_{i=0}^{n} b'_i(z) \psi_i\big(x'\big)\varphi_i\big(Z(z, \mu)\big) \overset{(72)}{=} \mathcal{P}\big(x', Z(z, \mu); b'(z)\big).
\end{aligned}$$

□

A few comments about our assumptions are in order. Assumption 1 holds not only for ordinary polynomials (9), 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). Assumption 1 also holds for piecewise approximations such as piecewise linear functions and cubic splines, as well

as mixed approximations that combine higher-order polynomials for some variables with piecewise functions for others. Furthermore, Assumption 1 is also consistent with anisotropic approximating functions that allow use for different degrees of polynomial approximation in different model's variables; see, for example, an anisotropic Smolyak method introduced in Judd et al. (2014). Anisotropic families may be especially useful in high-dimensional problems if there is more curvature in some directions than in others or if some of the state variables are exogenous so that higher-order smoothness in those variables is not really required.

Assumption 2 means that next-period endogenous state variables such as k' are known with certainty at present. There are some interesting economic models in which this assumption is not satisfied, including asset-pricing models, dynamic games, input–output models with private shocks, and research and development models. It would be of interest to extend precomputation technique to the case when all or some endogenous state variables are random.

Finally, Assumption 3 means that endogenous state variables have finite moments, which is normally satisfied in economic models. The assumption that the random shocks have a density function w is used for expositional convenience: it is sufficient to assume instead that random shocks have a distribution function that produces finite integrals in all basis functions considered.

To construct an expectation function, we choose the next-period reference point $Z(z,0)$. This choice is a matter of convenience. In our precomputation examples (as well as in many macroeconomic models), the stochastic process for an exogenous state variable is multiplicatively separable in the current state term and future disturbances, which makes the integral (74) independent of the current economy's state z, that is, $b_i'(z) = b_i'$ for all z. For example, in (22), the process is $Z(z,\varepsilon') = z^\rho \exp(\varepsilon')$ and, hence, we have $b_i'(z) = b_i \int_{\varepsilon'\in\Omega} \frac{z^\rho \exp(\varepsilon')}{z^\rho \exp(0)} w(\varepsilon')\, d\varepsilon' \equiv b_i'$ for all z. However, in general, the precomputed integral $b_i'(z)$ will depend on the current economy's state z, and we must construct a mapping $b_i'(z)$, either analytically or numerically.

5.2 *Discrete-state problems*

Let $x \in \mathbb{R}^{n_x}$ and $z \in \{z_1, \ldots, z_J\} \in \mathbb{R}^{n_z}$ be vectors of endogenous and exogenous state variables, respectively. We formulate a set of Assumptions 1′–3′ for the discrete-shock case that are parallel to Assumptions 1–3 for the continuous-shock case.

Assumption 1′. *For each state $\ell \in \{1, \ldots, J\}$, an approximating function is given by*

$$\mathcal{P}\big(x; b^\ell\big) = \sum_{i=0}^{n} b_i^\ell \psi_i(x), \tag{75}$$

where $b^\ell \equiv (b_0^\ell, \ldots, b_n^\ell) \in \mathbb{R}^{n+1}$; $\psi_i(x)$ is the ith basis function, by convention, $\psi_0(x)$ is equal to 1, *and $\psi_i : \mathbb{R}^{n_x} \to \mathbb{R}$ for $i = 1, \ldots, n$.*

Assumption 2′. *Next-period endogenous state variables x' are t-measurable.*

ASSUMPTION 3′. *Exogenous state variables follow a stochastic process that has a countable number of states* $z \in [z_1, \ldots, z_J]$ *and transition probabilities are given by* $\pi_{j\ell} = \Pr(z' = z_j | z = z_\ell)$, *where* $\ell \in \{1, \ldots, J\}$.

PROPOSITION 2. *Under Assumptions* 1′–3′, *we have*

$$E\big[\mathcal{P}\big(x'; b^j\big)|x, z_\ell\big] = \mathcal{P}\big(x'; \big(b^\ell\big)'\big), \tag{76}$$

where $(b^\ell)' \equiv ((b_0^\ell)', (b_1^\ell)', \ldots, (b_n^\ell)') \in \mathbb{R}^{n+1}$ *and the coefficients* $(b^\ell)'$ *and* b^j *are related by*

$$\big(b_i^\ell\big)' = \sum_{j=1}^{J} \pi_{j\ell} b_i^j, \quad i = 0, 1, \ldots, n. \tag{77}$$

The proof follows the derivations in (67).

In the case of discrete-state shocks, our generalizations go in two dimensions. First, we allow for any additively separable approximating family, in addition to the ordinary polynomial functions used in (65). Second, we allow for multiple endogenous and exogenous state variables. This means that if each shock $i \in \{1, \ldots, N\}$ has J_i states, then we need to approximate $J = J_1 \times \cdots \times J_N$ decision functions of exogenous state variables.[10] In this case, the number of exogenous states grows exponentially with the number of shocks. To alleviate the curse of dimensionality, one can use some nonproduct rules for selecting a smaller set of states that can approximate the process for multivariate shocks sufficiently well; see, for example, a cluster grid and epsilon distinguishable set techniques introduced in Maliar and Maliar (2015).

Finally, integrals can also be precomputed in models with discrete control variables or a mixture of discrete and continuous control variables. Indeed, precomputation results (10), (46), and (66) hold independently of whether control variables are continuous or discrete. Discrete controls are used in the recent literature on structural estimation in which dynamic stochastic models must be solved repeatedly a large number of times, and savings on cost from precomputation of expectation functions can be of value in this computationally intense class of problems.

6. CONCLUSION

A vast majority of the existing solution methods in the economics literature use approximating families of functions studied in the present paper. For such methods, we can precompute integrals in the stage of initialization and in effect, we can transform a stochastic problem into a deterministic problem. The technique of precomputation of integrals is very general and can be applied to essentially any set of equations that contains expectation functions. It works for both continuous- and discrete-state shocks, and it can be combined with other computational techniques used by the existing solution

[10] In the multivariate case, it is also possible to start from a continuous-state multivariate Markov process and to discretize this process into a Markov chain with a countable number of states by using the analysis of Tauchen (1986) and Tauchen and Hussey (1991).

methods, including a variety of solution domains, integration rules, fitting methods, and iterative schemes for finding unknown parameters of the approximating functions. Integrals can be constructed in a closed form for models with uni- or multivariate normally distributed shocks, which is the case of special interest to economics. In those cases in which integrals cannot be constructed analytically, we can precompute them numerically by applying very accurate computational methods since this is a one time fixed cost. In addition, precomputation of integrals is very simple to implement.

For small problems with few shocks that must be solved just once, precomputation of integrals is useful but not critical. Nevertheless, some interesting economic models in the recent literature may have dozens of exogenous shocks, including large-scale new Keynesian models used by central banks for projection and policy analysis, large-scale overlapping generation models, heterogeneous agents models, and climate change models; see Maliar and Maliar (2014) for a discussion and further examples of large-scale applications.[11] Furthermore, the literature on structural estimation must recompute a solution to dynamic economic models a large number of times under different parameter vectors. For these and other computationally intense applications, precomputation of integrals can be the only tractable alternative.

References

Aiyagari, R. (1994), "Uninsured idiosyncratic risk and aggregate saving." *Quarterly Journal of Economics*, 109 (3), 659–684. [851, 854, 860, 880, 883, 884, 885, 886]

Arellano, C., L. Maliar, S. Maliar, and V. Tsyrennikov (2016), "Envelope condition method with an application to default risk model." *Journal of Economic Dynamics and Control*, 69, 436–459. [854, 859, 862]

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. [852, 863]

Barillas, F. and J. Fernandez-Villaverde (2007), "A generalization of the endogenous grid method." *Journal of Economic Dynamics & Control*, 31, 2698–2712. [865]

Bewley, T. (1977), "The permanent income hypothesis: A theoretical formulation." *Journal of Economic Theory*, 16 (2), 252–292. [880]

Carroll, C. D. (1992), "The buffer-stock theory of saving: Some macroeconomic evidence." *Brookings Papers on Economic Activity*, 1992 (2), 61–156. [880]

Carroll, K. (2006), "The method of endogenous grid points for solving dynamic stochastic optimal problems." *Economics Letters*, 91, 312–320. [853, 862, 863, 864, 865]

[11] In particular, precomputation of integrals can significantly reduce the computational expense in the analysis of Lepetyuk, Maliar, and Maliar (2017), who construct a global nonlinear solution to a version of the Canadian Central Bank ToTEM model with 21 state variables in the presence of zero lower bound on the nominal interest rate.

Christiano, L. and D. Fisher (2000), "Algorithms for solving dynamic models with occasionally binding constraints." *Journal of Economic Dynamics and Control*, 24, 1179–1232. [852]

Deaton, A. (1991), "Saving and liquidity constraints." *Econometrica*, 59 (5), 1221–1248. [880]

Den Haan, W. (2010), "Comparison of solutions to the incomplete markets model with aggregate uncertainty." *Journal of Economic Dynamics and Control*, 34, 4–27. [852]

Den Haan, W. and A. Marcet (1990), "Solving the stochastic growth model by parameterized expectations." *Journal of Business and Economic Statistics*, 8, 31–34. [858, 866, 877]

Gaspar, J. and K. Judd (1997), "Solving large-scale rational-expectations models." *Macroeconomic Dynamics*, 1, 45–75. [852]

Geweke, J. (1996), "Monte Carlo simulation and numerical integration." In *Handbook of Computational Economics*, Vol. 1 (H. Amman, D. Kendrick, and J. Rust, eds.), Chapter 15, 733–800, Elsevier Science, Amsterdam. [874]

Huggett, M. (1993), "The risk-free rate in heterogeneous-agent incomplete-insurance economies." *Journal of Economic Dynamics and Control*, 17 (5–6), 953–969. [880]

Judd, K. (1992), "Projection methods for solving aggregate growth models." *Journal of Economic Theory*, 58, 410–452. [858, 866]

Judd, K. (1998), *Numerical Methods in Economics*. MIT Press, Cambridge, MA. [852, 853]

Judd, K., L. Maliar, and S. Maliar (2011), "Numerically stable and accurate stochastic simulation approaches for solving dynamic models." *Quantitative Economics*, 2, 173–210. [853, 860, 866, 871, 874, 876, 877, 879]

Judd, K. L., L. Maliar, S. Maliar, and R. Valero (2014), "Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain." *Journal of Economic Dynamics and Control*, 44 (C), 92–123. [853, 888]

Judd, K. L., L. Maliar, and S. Maliar (2017), "Lower bounds on approximation errors to numerical solutions of dynamic economic models." *Econometrica*, 85 (3), 991–1012. [852]

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. [852]

Krueger, D. and F. Kubler (2004), "Computing equilibrium in OLG models with production." *Journal of Economic Dynamics and Control*, 28, 1411–1436. [853]

Lepetyuk, V., L. Maliar, and S. Maliar (2017), "Should central banks worry about nonlinearities of their large-scale macroeconomic models?" Bank of Canada Staff Working Paper 2017-21. [890]

Maliar, L. and S. Maliar (2004), "Quasi-geometric discounting: A closed-form solution under the exponential utility function." *Bulletin of Economic Research*, 56 (2), 201–206. [860]

Maliar, L. and S. Maliar (2005a), "Solving nonlinear stochastic growth models: An algorithm iterating on value function by simulations." *Economics Letters*, 87, 135–140. [860]

Maliar, L. and S. Maliar (2005b), "Parameterized expectations algorithm: How to solve for labor easily." *Computational Economics*, 25, 269–274. [858, 869, 870]

Maliar, L. and S. Maliar (2006), "The neoclassical growth model with heterogeneous quasi-geometric consumers." *Journal of Money, Credit, and Banking*, 38 (3), 635–654. [883]

Maliar, L. and S. Maliar (2013), "Envelope condition method versus endogenous grid method for solving dynamic programming problems." *Economics Letters*, 120, 262–266. [853, 854, 859, 862, 863, 865, 870]

Maliar, L. and S. Maliar (2014), "Numerical methods for large scale dynamic economic models." In *Handbook of Computational Economics*, Vol. 3 (K. Schmedders and K. Judd, eds.), Chapter 7, 325–477, Elsevier Science, Amsterdam. [852, 865, 870, 877, 890]

Maliar, L. and S. Maliar (2015), "Merging simulation and projection approaches to solve high-dimensional problems with an application to a new Keynesian model." *Quantitative Economics*, 6, 1–47. [889]

Maliar, L., S. Maliar, and F. Valli (2010), "Solving the incomplete markets model with aggregate uncertainty using the Krusell–Smith algorithm." *Journal of Economic Dynamics and Control*, 34, 42–49. [886]

Maliar, S., L. Maliar, and K. Judd (2011), "Solving the multi-country real business cycle model using ergodic set methods." *Journal of Economic Dynamics & Control*, 35, 207–228. [871]

Marimon, R. and A. Scott (1999), *Computational Methods for Study of Dynamic Economies*. Oxford University Press, New York. [852]

Miranda, M. and P. Fackler (2002), *Applied Computational Economics and Finance*. MIT Press, Cambridge, MA. [852]

Rios-Rull, J. V. (1997), "Computing of equilibria in heterogeneous agent models." Federal Reserve Bank of Minneapolis Staff Report 231. [884]

Rust, J. (1996), "Numerical dynamic programming in economics." In *Handbook of Computational Economics*, Vol. 1 (H. Amman, D. Kendrick, and J. Rust, eds.), Chapter 14, 619–722, Elsevier Science, Amsterdam. [852, 874]

Rust, J. (2008), "Dynamic programming." In *The New Palgrave Dictionary of Economics* (S. Durlauf and L. Blume, eds.), Palgrave Macmillan, New York. [852]

Santos, M. (1999), "Numerical solution of dynamic economic models." In *Handbook of Macroeconomics* (J. Taylor and M. Woodford, eds.), 311–386, Elsevier Science, Amsterdam. [852]

Stachursky, J. (2009), *Economic Dynamics: Theory and Computation*. MIT Press, Cambridge, MA. [852]

Stroud, A. (1971), *Approximate Integration of Multiple Integrals*. Prentice Hall, Englewood Cliffs, NJ. [874]

Tauchen, G. (1986), "Finite state Markov chain approximations to univariate and vector autoregressions." *Economics Letters*, 20, 177–181. [884, 889]

Tauchen, G. and R. Hussey (1991), "Quadrature-based methods for obtaining approximate solutions to nonlinear asset pricing models." *Econometrica*, 59, 371–396. [884, 889]

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. [852]

Co-editor José-Víctor Ríos-Rull handled this manuscript.

Manuscript received 9 November, 2012; final version accepted 6 December, 2016; available online 2 June, 2017.

Supplement to "How to solve dynamic stochastic models computing expectations just once": Appendices

(*Quantitative Economics*, Vol. 8, No. 3, November 2017, 851–893)

Kenneth L. Judd
Hoover Institution, Stanford University

Lilia Maliar
Department of Economics, Stanford University

Serguei Maliar
Department of Economics, Santa Clara University

Inna Tsener
Department of Applied Economics, University of the Balearic Islands

Appendix A: Bellman and Euler equation algorithms for solving a growth model with inelastic labor supply

In Algorithms A1–A6, we provide a description of three Bellman and two Euler equation algorithms, which we use to solve the neoclassical stochastic growth model with inelastic labor supply described in Section 2.

Kenneth L. Judd: kennethjudd@mac.com
Lilia Maliar: maliarl@stanford.edu
Serguei Maliar: smaliar@scu.edu
Inna Tsener: inna.tcener@uib.es

DOI: 10.3982/QE329

Algorithm 1a. Value function iteration

Initialization.

a. Choose an approximating function $\widehat{V}(\cdot; b) \approx V$.

b. Choose integration nodes, ε_j, and weights, ω_j, $j = 1, \ldots, J$.

c. Construct a grid $\Gamma = \{k_m, z_m\}_{m=1}^M$.

d. Make an initial guess on $b^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of V on the grid.

For $m = 1, \ldots, M$:

a. Solve for k'_m satisfying

$u'((1-\delta)k_m + z_m f(k_m) - k'_m) = \beta \sum_{j=1}^J \omega_j \widehat{V}_1(k'_m, z_m^\rho \exp(\varepsilon_j); b^{(i)})$.

b. Find c_m satisfying

$c_m = (1-\delta)k_m + z_m f(k_m) - k'_m$.

c. Find value function on the grid

$\widehat{v}_m \equiv u(c_m) + \beta \sum_{j=1}^J \omega_j \widehat{V}(k'_m, z_m^\rho \exp(\varepsilon_j); b^{(i)})$.

Step 2. Computation of b that fits value function on the grid.

Run a regression to find $\widehat{b}$:

$\widehat{b} = \arg\min_b \sum_{m=1}^M \|\widehat{v}_m - \widehat{V}(k_m, z_m; b)\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$: end Step 2 if

$\frac{1}{M} \sum_{m=1}^M |\frac{(k'_m)^{(i)} - (k'_m)^{(i-1)}}{(k'_m)^{(i-1)}}| < 10^{-9}$.

b. Use damping with $\xi = 1$ to compute $b^{(i+1)} = (1-\xi)b^{(i)} + \xi\widehat{b}$.

Algorithm 1b. Value function iteration with precomputation

Initialization.

...

b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12).

...

Step 1. Computation of values of V on the grid.

At iteration i, for $m = 1, \ldots, M$:

a. Given $b^{(i)}$, find $b'^{(i)}$ from (11) and compute $\widehat{V}_1(k'_m, z_m^\rho; b'^{(i)})$.

b. Solve for k'_m satisfying

$u'((1-\delta)k_m + z_m f(k_m) - k'_m) = \beta \widehat{V}_1(k'_m, z_m^\rho; b'^{(i)})$.

...

d. Find value function on the grid

$\widehat{v}_m \equiv u(c_m) + \beta \widehat{V}(k'_m, z_m^\rho; b'^{(i)})$.

...

Algorithm 2a. Endogenous grid method

Initialization.

a. Choose an approximating function $\widehat{V}(\cdot; b) \approx V$.
b. Choose integration nodes, ε_j, and weights, ω_j, $j = 1, \ldots, J$.
c. Construct grid $\Gamma = \{k'_m, z_m\}_{m=1}^M$.
d. Make an initial guess on $b^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of V on the grid.

For $m = 1, \ldots, M$:
a. Compute $\widehat{W}(k'_m, z_m; b^{(i)}) \equiv \sum_{j=1}^J \omega_j \widehat{V}(k'_m, z_m^\rho \exp(\varepsilon_j); b^{(i)})$ and $\widehat{W}_1(k'_m, z_m; b^{(i)}) \equiv \sum_{j=1}^J \omega_j \widehat{V}_1(k'_m, z_m^\rho \exp(\varepsilon_j); b^{(i)})$.
b. Find $c_m = u'^{-1}[\beta \widehat{W}_1(k'_m, z_m; b^{(i)})]$.
c. Use a solver to find k_m satisfying $(1-\delta)k_m + z_m f(k_m) = c_m + k'_m$.
d. Find value function on the grid $\widehat{v}_m \equiv u(c_m) + \beta \widehat{W}(k'_m, z_m; b^{(i)})$.

Step 2. Computation of b that fits value function on the grid.

Run a regression to find $\widehat{b}$:
$\widehat{b} = \arg\min_b \sum_{m=1}^M \|\widehat{v}_m - \widehat{V}(k_m, z_m; b)\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$: end Step 2 if $\frac{1}{M}\sum_{m=1}^M |\frac{(k_m)^{(i)} - (k_m)^{(i-1)}}{(k_m)^{(i-1)}}| < 10^{-9}$.
b. Use damping with $\xi = 1$ to compute $b^{(i+1)} = (1-\xi)b^{(i)} + \xi\widehat{b}$.

Algorithm 2b. Endogenous grid method with precomputation

Initialization.

...
b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12).
...

Step 1. Computation of values of V on the grid.

At iteration i, for $m = 1, \ldots, M$:
a. Given $b^{(i)}$, find $b'^{(i)}$ from (11) and compute $\widehat{W}(k'_m, z_m^\rho; b'^{(i)})$ and $\widehat{W}_1(k'_m, z_m^\rho; b'^{(i)})$.
b. Find $c_m = u'^{-1}[\beta \widehat{W}_1(k'_m, z_m^\rho; b'^{(i)})]$.
...
d. Find value function on the grid $\widehat{v}_m \equiv u(c_m) + \beta \widehat{W}(k'_m, z_m^\rho; b'^{(i)})$.

...

Algorithm 3a. Envelope condition method

Initialization.

a. Choose an approximating function $\widehat{V}(\cdot;b)\approx V$.
b. Choose integration nodes, ε_j, and weights, ω_j, $j=1,\ldots,J$.
c. Construct grid $\Gamma=\{k_m,z_m\}_{m=1}^M$.
d. Make an initial guess on $b^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of V on the grid.

For $m=1,\ldots,M$:
a. Use $b^{(i)}$ to compute $\widehat{V}_1(k_m,z_m;b^{(i)})$.
b. Compute the corresponding values of c_m using
$c_m=u'^{-1}[\widehat{V}_1(k_m,z_m;b^{(i)})(1-\delta+z_mf'(k_m))^{-1}]$.
c. Find k'_m using
$k'_m=(1-\delta)k_m+z_mf(k_m)-c_m$.
d. Find value function on the grid
$\widehat{v}_m\equiv u(c_m)+\beta\sum_{j=1}^J\omega_j\widehat{V}(k'_m,z_m^\rho\exp(\varepsilon_j);b^{(i)})$.

Step 2. Computation of b that fits the value function on the grid.

Run a regression to find $\widehat{b}$:
$\widehat{b}=\arg\min_b\sum_{m=1}^M\|\widehat{v}_m-\widehat{V}(k_m,z_m;b)\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i\geq 2$: end Step 2 if
$\frac{1}{M}\sum_{m=1}^M|\frac{(k'_m)^{(i)}-(k'_m)^{(i-1)}}{(k'_m)^{(i-1)}}|<10^{-9}$.
b. Use damping with $\xi=1$ to compute $b^{(i+1)}=(1-\xi)b^{(i)}+\xi\widehat{b}$.

Algorithm 3b. Envelope condition method with precomputation

Initialization.

...
b. Precompute $\{\mathcal{I}_0,\ldots,\mathcal{I}_n\}$ using (12).
...

Step 1. Computation of values of V on the grid.

At iteration i, for $m=1,\ldots,M$:
a. Given $b^{(i)}$, find $b'^{(i)}$ from (11); use $b^{(i)}$ to compute $\widehat{V}_1(k_m,z_m;b^{(i)})$ and use $b'^{(i)}$ to compute $\widehat{V}(k'_m,z_m^\rho;b'^{(i)})$.
...
d. Find value function on the grid
$\widehat{v}_m\equiv u(c_m)+\beta\widehat{V}(k'_m,z_m^\rho;b'^{(i)})$.

...

Algorithm 4a. Euler equation algorithm parameterizing Q

Initialization.

a. Choose an approximating function $\widehat{Q}(\cdot; b) \approx Q$.
b. Choose integration nodes, ε_j, and weights, ω_j, $j = 1, \ldots, J$.
c. Construct grid $\Gamma = \{k_m, z_m\}_{m=1}^{M}$.
d. Make an initial guess on $b^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of Q on the grid.

For $m = 1, \ldots, M$:
a. Use $b^{(i)}$ to compute $\widehat{Q}(k_m, z_m; b^{(i)})$.
b. Find k'_m using
$k'_m = (1-\delta)k_m + z_m f(k_m) - u'^{-1}(\frac{\widehat{Q}(k_m, z_m; b^{(i)})}{1-\delta+z_m f(k_m)})$.
c. Find the values of q_m on the grid
$\widehat{q}_m \equiv \beta \sum_{j=1}^{J} \omega_j \widehat{Q}(k'_m, z_m^{\rho} \exp(\varepsilon_j); b^{(i)})[1 - \delta + zf'(k_m)]$.

Step 2. Computation of b that fits the Q function on the grid.

Run a regression to find $\widehat{b}$:
$\widehat{b} = \arg\min_b \sum_{m=1}^{M} \|\widehat{q}_m - \widehat{Q}(k_m, z_m; b)\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$: end Step 2 if
$\frac{1}{M} \sum_{m=1}^{M} |\frac{(k'_m)^{(i)} - (k'_m)^{(i-1)}}{(k'_m)^{(i-1)}}| < 10^{-9}$.
b. Use damping with $\xi = 1$ to compute $b^{(i+1)} = (1-\xi)b^{(i)} + \xi\widehat{b}$.

Algorithm 4b. Euler equation algorithm parameterizing Q with precomputation

Initialization.

...
b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12).
...

Step 1. Computation of values of Q on the grid.

At iteration i, for $m = 1, \ldots, M$:
a. Given $b^{(i)}$, find $b'^{(i)}$ from (11); compute $\widehat{Q}(k_m, z_m; b^{(i)})$ and $\widehat{Q}(k'_m, z_m^{\rho}; b'^{(i)})$.
...
c. Find the values of $\widehat{q}_m$ on the grid
$\widehat{q}_m \equiv \beta \widehat{Q}(k'_m, z_m^{\rho}; b'^{(i)})[1 - \delta + zf'(k_m)]$.

...

Algorithm 5a. Euler equation algorithm parameterizing Q and K

Initialization.

a. Choose approximating functions $\widehat{K}(\cdot; v) \approx K$ and $\widehat{Q}(\cdot; b) \approx Q$.

b. Choose integration nodes, ε_j, and weights, ω_j, $j = 1, \ldots, J$.

c. Construct grid $\Gamma = \{k_m, z_m\}_{m=1}^M$.

d. Make an initial guess on $v^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of Q on the grid.

For $m = 1, \ldots, M$:

a. Use $v^{(i)}$ to compute $\widehat{K}(k_m, z_m; v^{(i)})$.

b. Find c_m using

$c_m = (1-\delta)k_m + z_m f(k_m) - \widehat{K}(k_m, z_m; v^{(i)})$.

c. Find the values of q_m on the grid

$\widehat{q}_m \equiv u'(c_m)[1 - \delta + z_m f'(k_m)]$.

Step 2. Computation of v that fits the capital function on the grid.

a. Run a regression to find $\widehat{b}$:

$\widehat{b} = \arg\min_b \sum_{m=1}^M \|\widehat{q}_m - \widehat{Q}(k_m, z_m; b)\|$,

and compute $\widehat{Q}(k_m, z_m; \widehat{b})$.

b. Compute the values of next-period capital on the grid

$\widehat{k}'_m \equiv \beta \frac{\sum_{j=1}^J \omega_j \widehat{Q}(\widehat{K}(k_m, z_m; v^{(i)}), z_m^\rho \exp(\varepsilon_j); \widehat{b})}{\widehat{Q}(k_m, z_m; \widehat{b})} (1 - \delta + z f'(k_m)) \widehat{K}(k_m, z_m; v^{(i)})$.

c. Run a regression to find $\widehat{v}$:

$\widehat{v} = \arg\min_v \sum_{m=1}^M \|\widehat{k}'_m - \widehat{K}(k_m, z_m; v^{(i)})\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$: end Step 2 if

$\frac{1}{M} \sum_{m=1}^M |\frac{(k'_m)^{(i)} - (k'_m)^{(i-1)}}{(k'_m)^{(i-1)}}| < 10^{-9}$.

b. Use damping with $\xi = 0.15$ to compute $v^{(i+1)} = (1-\xi)v^{(i)} + \xi\widehat{v}$.

Algorithm 5b. Euler equation algorithm parameterizing Q and K with precomputation

Initialization.

...

b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12).

...

Step 2. Computation of v that fits the capital function on the grid.

a. Run a regression to find $\widehat{b}$,

$\widehat{b} = \arg\min_b \sum_{m=1}^{M} \|\widehat{q}_m - \widehat{Q}(k_m, z_m; b)\|$,

and compute $\widehat{Q}(k_m, z_m; \widehat{b})$.

Given $\widehat{b}$, find $\widehat{b}'$ from (11), and compute $\widehat{Q}(\widehat{K}(k_m, z_m; v^{(i)}), z_m^{\rho}; \widehat{b}')$.

b. Compute the value of next-period capital on the grid

$\widehat{k}'_m \equiv \beta \frac{\widehat{Q}(\widehat{K}(k_m, z_m; v^{(i)}), z_m^{\rho}; \widehat{b}')}{\widehat{Q}(k_m, z_m; \widehat{b})}(1 - \delta + z f'(k_m))\widehat{K}(k_m, z_m; v^{(i)})$.

...

Algorithm 6. Euler equation algorithm parameterizing K (not compatible with precomputation)

Initialization.

a. Choose approximating functions $\widehat{K}(\cdot; v) \approx K$.

...

Step 1. Computation of values of $\widehat{k}'$ on the grid.

For $m = 1, \ldots, M$:

a. Use $v^{(i)}$ to compute $\widehat{K}(k_m, z_m; v^{(i)})$ and

$k''_{m,j} = \widehat{K}(\widehat{K}(k_m, z_m; v^{(i)}), z_m^{\rho} \exp(\epsilon_j); v^{(i)})$, $j = 1, \ldots, J$.

b. Find $c'_{m,j}$ using

$c'_{m,j} = (1 - \delta)\widehat{K}(k_m, z_m; v^{(i)}) + z_m f(\widehat{K}(k_m, z_m; v^{(i)})) - k''_{m,j}$.

c. Find the values of c_m on the grid

$u'(c_m) = \beta \sum_{j=1}^{J} \omega_j u'(c'_{m,j})[1 - \delta + z_m^{\rho} \exp(\epsilon_j) f'(\widehat{K}(k_m, z_m; v^{(i)}))]$.

d. Find the values of k'_m on the grid

$\widehat{k}'_m = (1 - \delta)k_m + z_m f(k_m) - c_m$.

Step 2. Computation of v that fits the capital function on the grid.

a. Run a regression to find $\widehat{v}$:

$\widehat{v} = \arg\min_v \sum_{m=1}^{M} \|\widehat{k}'_m - \widehat{K}(k_m, z_m; v^{(i)})\|$.

...

Appendix B: ECM and Euler equation algorithms for solving a growth model with elastic labor supply

In Algorithms 7 and 8, we provide a description of ECM and Euler equation algorithms, which we use to solve the neoclassical stochastic growth model with valued leisure described in Section 2.

Algorithm 7a. Envelope condition method

Initialization.

a. Choose an approximating function $\widehat{V}(\cdot; b) \approx V$.
b. Choose integration nodes, ε_j, and weights, ω_j, $j = 1, \ldots, J$.
c. Construct grid $\Gamma = \{k_m, z_m\}_{m=1}^M$.
d. Make an initial guess on $b^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of V on the grid.

For $m = 1, \ldots, M$:
a. Use $b^{(i)}$ to compute $\widehat{V}_1(k_m, z_m; b^{(i)})$.
b. Solve for l_m that satisfies
$B(1-l_m)^{-\mu}[1-\delta+z_m f_1(k_m, l_m)] = V_1(k_m, z_m; b^{(i)}) z_m f_2(k_m, l_m)$.
c. Compute the corresponding values of c_m using
$c_m = u_1^{-1}[\frac{B(1-l_m)^{-\mu}}{z_m f_2(k_m, l_m)}]$.
d. Find k'_m using
$k'_m = (1-\delta)k_m + z_m f(k_m, l_m) - c_m$.
e. Find value function on the grid
$\widehat{v}_m \equiv u(c_m, l_m) + \beta \sum_{j=1}^J \omega_j \widehat{V}(k'_m, z_m^\rho \exp(\varepsilon_j); b^{(i)})$.

Step 2. Computation of b that fits the value function on the grid.

Run a regression to find $\widehat{b}$:
$\widehat{b} = \arg\min_b \sum_{m=1}^M \|\widehat{v}_m - \widehat{V}(k_m, z_m; b)\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$: end Step 2 if
$\frac{1}{M}\sum_{m=1}^M |\frac{(k'_m)^{(i)} - (k'_m)^{(i-1)}}{(k'_m)^{(i-1)}}| < 10^{-9}$.
b. Use damping with $\xi = 1$ to compute $b^{(i+1)} = (1-\xi)b^{(i)} + \xi\widehat{b}$.

Algorithm 7b. Envelope condition method with precomputation

Initialization.

...

b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12).

...

Step 1. Computation of values of V on the grid.

At iteration i, for $m = 1, \ldots, M$:

a. Given $b^{(i)}$, find $b'^{(i)}$ from (11); use $b^{(i)}$ to compute $\widehat{V}_1(k_m, z_m; b^{(i)})$ and use $b'^{(i)}$ to compute $\widehat{V}(k'_m, z^\rho_m; b'^{(i)})$.

...

e. Find value function on the grid $\widehat{v}_m \equiv u(c_m, l_m) + \beta \widehat{V}(k'_m, z^\rho_m; b'^{(i)})$.

...

Algorithm 8a. Euler equation algorithm parameterizing Q

Initialization.

a. Choose an approximating function $\widehat{Q}(\cdot; b) \approx Q$.

b. Choose integration nodes, ε_j, and weights, ω_j, $j = 1, \ldots, J$.

c. Construct grid $\Gamma = \{k_m, z_m\}_{m=1}^M$.

d. Make an initial guess on $b^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of Q on the grid.

For $m = 1, \ldots, M$:

a. Use $b^{(i)}$ to compute $\widehat{Q}(k_m, z_m; b^{(i)})$.

b. Solve for l_m that satisfies

$B(1 - l_m)^{-\mu}[1 - \delta + z_m f_1(k_m, l_m)] = \widehat{Q}(k_m, z_m; b^{(i)}) z_m f_2(k_m, l_m)$.

c. Compute the corresponding values of c_m using

$c_m = u_1^{-1}[\frac{B(1-l_m)^{-\mu}}{z_m f_2(k_m, l_m)}]$.

d. Find k'_m using

$k'_m = (1 - \delta)k_m + z_m f(k_m, l_m) - c_m$.

e. Find the values of q_m on the grid

$\widehat{q}_m \equiv \beta \sum_{j=1}^J \omega_j \widehat{Q}(k'_m, z_m^\rho \exp(\varepsilon_j); b^{(i)})[1 - \delta + z f_1(k_m, l_m)]$.

Step 2. Computation of b that fits the Q function on the grid.

Run a regression to find $\widehat{b}$:

$\widehat{b} = \arg\min_b \sum_{m=1}^M \|\widehat{q}_m - \widehat{Q}(k_m, z_m; b)\|$.

Step 3. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$: end Step 2 if

$\frac{1}{M} \sum_{m=1}^M |\frac{(k'_m)^{(i)} - (k'_m)^{(i-1)}}{(k'_m)^{(i-1)}}| < 10^{-9}$.

b. Use damping with $\xi = 1$ to compute $b^{(i+1)} = (1 - \xi)b^{(i)} + \xi\widehat{b}$.

Algorithm 8b. Euler equation algorithm parameterizing Q with precomputation

Initialization.

...

b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (12).

...

Step 1. Computation of values of Q on the grid.

At iteration i, for $m = 1, \ldots, M$:

a. Given $b^{(i)}$, find $b'^{(i)}$ from (11); compute $\widehat{Q}(k_m, z_m; b^{(i)})$ and $\widehat{Q}(k'_m, z^{\rho}_m; b'^{(i)})$.

...

e. Find the values of $\widehat{q}_m$ on the grid
$\widehat{q}_m \equiv \beta \widehat{Q}(k'_m, z^{\rho}_m; b'^{(i)})[1 - \delta + z f_2(k_m, l_m)]$.

...

Appendix C: Euler equation algorithm for solving the multicountry model

In this section, we describe the Euler equation methods that we use to analyze the multicountry model in Section 3.

Algorithm 9a. Euler equation algorithm parameterizing Q^h and K^h

Initialization.

a. Choose approximating functions $K^h(\cdot; v^h) \approx K^h$, $h = 1, \ldots, N$, and $Q^h(\cdot; b^h) \approx Q^h$.
b. Choose integration nodes, $\boldsymbol{\varepsilon}_j = (\varepsilon_j^1, \ldots, \varepsilon_j^N)$, and weights, ω_j, $j = 1, \ldots, J$.
c. Fix simulation length $T = 2000$ and $(\boldsymbol{k}_0, z_0) = (\mathbf{1}, \mathbf{1})$, where $\mathbf{1} \equiv (1, \ldots, 1) \in \mathbb{R}^N$.
d. Draw and fix a sequence of productivity levels $\{z_t\}_{t=1,\ldots,T}$ using (40).
e. Construct integration nodes, $z_{t+1,j} = (z_{t+1,j}^1, \ldots, z_{t+1,j}^N)$ with $z_{t+1,j}^h = (z_t^h)^\rho \exp(\varepsilon_j^h)$.
f. Make an initial guess on $(v^1)^{(1)}, \ldots, (v^h)^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, given $(v^1)^{(i)}, \ldots, (v^h)^{(i)}$:
Step 1. Computation of values of Q on the simulated points.
For $t = 1, \ldots, T$:
a. Use $k_{t+1}^h = \widehat{K}^h(\boldsymbol{k}_t, z_t; (v^h)^{(i)})$, $h = 1, \ldots, N$, to recursively calculate $\{\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$.
b. Compute $\{\boldsymbol{c}_t\}_{t=0,\ldots,T}$ satisfying $c_t^h = (\sum_{h=1}^N c_t^h)/N$.
c. Compute q_t^h from
$\widehat{q}_t^h \equiv u'(c_t^h)[1 - \delta + z_t^h f'(k_t^h)]$.

Step 2. Computation of v^h that fits the values of capital on the simulated points.
a. Run a regression to find $\widehat{b}^h$,
$\widehat{b}^h = \arg\min_{b^h} \sum_{m=1}^M \|\widehat{q}_t^h - \widehat{Q}^h(\boldsymbol{k}_t, z_t; b^h)\|$,
and compute $\widehat{Q}^h(\boldsymbol{k}_t, z_t; \widehat{b}^h)$.
b. Compute the values of next-period capital on the simulated points
$\widehat{k}_{t+1}^h \equiv \beta \frac{\sum_{j=1}^J \omega_j \widehat{Q}^h(\boldsymbol{k}_{t+1}, z_{t+1,j}; \widehat{b}^h)}{\widehat{Q}^h(\boldsymbol{k}_t, z_t; \widehat{b}^h)} [1 - \delta + z_t^h f'(k_t^h)] K^h(\boldsymbol{k}_t, z_t; v^h)$, $h = 1, \ldots, N$.
c. Run a regression to find $\widehat{v}^h$:
$\widehat{v}^h \equiv \arg\min_{v^h} \sum_{t=1}^T \|\widehat{k}_{t+1}^h - \widehat{K}^h(\boldsymbol{k}_t, z_t; (v^h)^{(i)})\|$.

Step 3. Convergence check and fixed-point iteration.
a. Check for convergence for $i \geq 2$; end Step 2 if
$\frac{1}{TN} \sum_{t=1}^T \sum_{h=1}^N |\frac{(k_{t+1}^h)^{(i)} - (k_{t+1}^h)^{(i-1)}}{(k_{t+1}^h)^{(i-1)}}| < 10^{-10}$.
b. Use damping with $\xi = 0.1$ to compute $(v^h)^{(i+1)} = (1 - \xi)(v^h)^{(i)} + \xi \widehat{v}^h$.

Algorithm 9b. Euler equation algorithm parameterizing Q^h and K^h with precomputation

Initialization.

…

b. Precompute $\{\mathcal{I}_0, \ldots, \mathcal{I}_n\}$ using (53).

…

Step 2. Computation of v^h that fits the values of capital on the simulated points.

a. Run a regression to find $\widehat{b}^h$,

$\widehat{b}^h = \arg\min_b \sum_{m=1}^{M} \|\widehat{q}_t^h - \widehat{Q}^h(\boldsymbol{k}_t, z_t; b^h)\|$,

and compute $\widehat{Q}^h(\boldsymbol{k}_t, z_t; \widehat{b}^h)$.

Given $\widehat{b}$, find $\widehat{b}'$ from (47), and compute $\widehat{Q}^h(\boldsymbol{k}_{t+1}, z_t^\rho; (\widehat{b}^h)')$.

b. Compute the values of next-period capital on the simulated points

$\widehat{k}_{t+1}^h \equiv \beta \frac{\widehat{Q}^h(\boldsymbol{k}_{t+1}, z_t^\rho; (\widehat{b}^h)')}{\widehat{Q}^h(\boldsymbol{k}_t, z_t; \widehat{b}^h)}[1 - \delta + z_t^h f'(k_t^h)]K^h(\boldsymbol{k}_t, z_t; v^h)$, $h = 1, \ldots, N$.

…

Algorithm 10. Euler equation algorithm parameterizing K^h (not compatible with precomputation)

Initialization.

a. Choose approximating functions $K^h(\cdot; v^h) \approx K^h$, $h = 1, \ldots, N$.

…

Step 1. Computation of values of $\widehat{k}_{t+1}^h$ on the simulated points.

For $t = 1, \ldots, T$:

a. Use $k_{t+1}^h = \widehat{K}^h(\boldsymbol{k}_t, z_t; (v^h)^{(i)})$, $h = 1, \ldots, N$, to recursively calculate $\{\boldsymbol{k}_{t+1}\}_{t=0,\ldots,T}$.

b. Compute $\{\boldsymbol{c}_t\}_{t=0,\ldots,T}$ satisfying $c_t^h = (\sum_{h=1}^{N} c_t^h)/N$.

c. Compute $k_{t+2,j}^h = \widehat{K}^h(\boldsymbol{k}_{t+1}, z_t^\rho \exp(\boldsymbol{\epsilon}_j); (v^h)^{(i)})$.

d. Find $\{\boldsymbol{c}_{t+1,j}\}_{t=0,\ldots,T}$ satisfying $c_{t+1,j}^h = (\sum_{h=1}^{N} c_{t+1,j}^h)/N$.

Step 2. Computation of v^h that fits the values of capital on the simulated points.

a. Compute the values of next-period capital on the simulated points

for $h = 1, \ldots, N$:

$\widehat{k}_{t+1}^h \equiv \beta \sum_{j=1}^{J} \omega_j u'(c_{t+1,j}^h)(u'(c_t^h))^{-1}[1 - \delta + (z_t^h)^\rho \exp(\epsilon_j^h) f'(k_{t+1}^h)]K^h(\boldsymbol{k}_t, z_t; v^h)$.

b. Run a regression to find $\widehat{v}^h$:

$\widehat{v}^h \equiv \arg\min_{v^h} \sum_{t=1}^{T} \|\widehat{k}_{t+1}^h - \widehat{K}^h(\boldsymbol{k}_t, z_t; (v^h)^{(i)})\|$.

…

TABLE C.1. Accuracy and cost of the Euler equation algorithm for the multicountry model: GSSA parameterizing K.

Polynomial Degree	Precomputation			M1			M2			GH(2)		
	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU	L_1	L_∞	CPU
1st	−2.77	−1.81	61	−2.77	−1.80	92	−2.77	−1.80	105	−2.77	−1.80	86
2nd	−3.88	−2.61	223	−3.88	−2.61	418	−3.88	−2.61	621	−3.88	−2.61	389
3rd	−4.94	−3.55	382	−4.95	−3.55	614	−4.95	−3.55	818	−4.95	−3.55	567
4th	−6.05	−4.68	574	−6.04	−4.67	840	−6.04	−4.67	1499	−6.04	−4.67	1241
5th	−7.15	−5.79	738	−7.15	−5.78	938	−7.15	−5.78	1875	−7.15	−5.78	1551
$N = 5$												
1st	−2.86	−1.94	75	−2.86	−1.93	184	−2.86	−1.93	687	−2.86	−1.93	396
2nd	−3.99	−2.81	319	−3.99	−2.82	958	−3.99	−2.82	3955	−3.99	−2.82	2456
3rd	−5.10	−3.84	2550	−5.13	−3.83	8893	−5.13	−3.83	7349	−5.13	−3.83	12,221
4th	−6.18	−4.89	7033	−6.31	−4.92	16,640	–	–	–	–	–	–
$N = 10$												
1st	−2.87	−1.89	125	−2.87	−1.89	515	−2.87	−1.89	4889	−2.87	−1.89	18,263
2nd	−4.00	−2.80	666	−4.01	−2.78	2681	−4.01	−2.78	23,610	–	–	–
3rd	−4.99	−3.88	14,837	−5.18	−3.92	14,232	–	–	–	–	–	–
$N = 20$												
1st	−3.12	−2.09	152	−3.12	−2.08	1803	−3.12	−2.08	32434	–	–	–
2nd	−4.36	−3.26	3303	−4.40	−3.32	13,204	–	–	–	–	–	–
$N = 30$												
1st	−3.15	−2.08	221	−3.16	−2.08	4688	–	–	–	–	–	–
2nd	−4.22	−3.22	13,543	–	–	–	–	–	–	–	–	–

Note: The main result columns correspond to variants of GSSA that evaluate integrals by using the precomputation method, the monomial integration methods with $2N$ and $2N^2 + 1$ nodes, and the Gauss Hermite quadrature method with two nodes, respectively; the statistics L_1 and L_∞ are, respectively, the average and maximum of absolute residuals across optimality condition and test points (in $\log 10$ units) on a stochastic simulation of 10,000 observations; CPU is the time necessary for computing a solution (in seconds).

Appendix D: Euler equation algorithm for solving the Aiyagari model

In this section, we describe a solution algorithm for Aiyagari's (1994) model analyzed in Section 4. We focus on the algorithm for solving the individual problem in which the precomputation results appear. The rest of the algorithm is standard and relies on stochastic simulation and a bisection technique as in Aiyagari (1994).

Algorithm 11a. Euler equation algorithm parameterizing Q and K

Initialization.

a. Given K, compute $R = \alpha K^{\alpha-1} - \delta$ and $W = (1-\alpha)(K)^{\alpha}$.
b. Choose an approximating function $\widehat{K}(\cdot; b^{\ell}) \approx K$ and $\widehat{Q}(\cdot; v^{\ell}) \approx Q$ for $\ell \in \{1, \ldots, J\}$.
c. Construct grid $\Gamma = \{k_m, z_m\}_{m=1}^{M}$.
d. Make an initial guess on $K(\cdot; b^{\ell})^{(1)}$.

Iterative cycle. Computation of a solution.

At iteration i, perform the following steps:

Step 1. Computation of values of K on the grid.

For $m = 1, \ldots, M$:
a. Find $c_{m,\ell}$ using
$c_{m,\ell} = W z_{\ell} + (1+R)k_m - \widehat{K}(\cdot; b^{\ell})$.
b. Compute $q_{m,\ell} = c_{m,\ell}^{-\gamma}(1+R)$.
c. Run a regression to find $\widehat{v}^{\ell}$ for $\ell \in \{1, \ldots, J\}$:
$\widehat{v}^{\ell} = \arg\min_{v^l} \sum_{m=1}^{M} \|\widehat{q}_{m,\ell} - \widehat{Q}(k_m; v)\|$.
d. Find $\widehat{k}'_{m,\ell}$ using
$\widehat{k}'_{m,\ell} = W z_{\ell} + (1+R)k_m - u'^{-1}[\beta \sum_j \pi_{jl} \widehat{Q}(k'_{m,\ell}; \widehat{v}^{j})]$.

Step 2. Convergence check and fixed-point iteration.

a. Check for convergence for $i \geq 2$; end Step 1 if
$\max_{m,l} |\frac{(c_{m,\ell})^{(i-1)} - (c_{m,\ell})^{(i)}}{(c_{m,\ell})^{(i-1)}}| < 10^{-10}$.
b. Use damping with $\xi = 0.5$ to compute
$(\widehat{K}(\cdot; b^{\ell}))^{(i+1)} = (1-\xi)(\widehat{K}(\cdot; b^{\ell}))^{(i)} + \xi(\widehat{k}'_{m,\ell})^{(i)}$.

Algorithm 11b. Euler equation algorithm parameterizing Q with precomputation

Initialization.

...

Iterative cycle. Computation of a solution.

Step 1. Computation of values of K on the grid.

At iteration i, for $m = 1, \ldots, M$:

...

d. Given $\widehat{v}^l$, $l = 1, \ldots, J$, find $(\widehat{v}^\ell)'$ from (67).

e. Find $\widehat{k}'_{m,\ell}$ using

$\widehat{k}'_{m,\ell} = Wz_\ell + (1+R)k_m - u'^{-1}[\beta \widehat{Q}(k'_{m,\ell}; (\widehat{v}^\ell)')]$.

...

Algorithm 12. Euler equation algorithm for Aiyagari's (1994) model parameterizing K (not compatible with precomputation)

Initialization.

...

b. Choose an approximating function $\widehat{K}(\cdot; b^\ell) \approx K$ for $\ell \in \{1, \ldots, J\}$.

...

Iterative cycle. Computation of a solution.

Step 1. Computation of values of K on the grid.

At iteration i, for $m = 1, \ldots, M$:

...

For $\ell = \{1, \ldots, J\}$:

b. Compute $k''_{m,j} = \widehat{K}(k'_{m,\ell}; b^j)$ for $j = \{1, \ldots, J\}$.

c. Find $c'_{m,j}$ for $j = \{1, \ldots, J\}$ using

$c'_{m,j} = Wz_j + (1+R)k'_{m,\ell} - k''_{m,j}$.

d. Find $\widehat{k}'_{m,\ell}$ using

$\widehat{k}'_{m,\ell} = Wz_\ell + (1+R)k_m - u'^{-1}[\beta \sum_j \pi_{jl}(c'_{m,j})^{-\gamma}[1+R]]$.

...

Co-editor José-Víctor Ríos-Rull handled this manuscript.

Manuscript received 9 November, 2012; final version accepted 6 December, 2016; available online 2 June, 2017.