Journal of Business & Economic Statistics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ubes20

Solving the Stochastic Growth Model by Policy-Function Iteration

Wilbur John Coleman II [a]
[a] Division of International Finance , Board of Governors of the Federal Reserve System , Washington , DC , 20551
Published online: 02 Jul 2012.

To cite this article: Wilbur John Coleman II (1990) Solving the Stochastic Growth Model by Policy-Function Iteration, Journal of Business & Economic Statistics, 8:1, 27-29, DOI: 10.1080/07350015.1990.10509769

To link to this article: http://dx.doi.org/10.1080/07350015.1990.10509769

Solving the Stochastic Growth Model by Policy-Function Iteration

Wilbur John Coleman II
Division of International Finance, Board of Governors of the Federal Reserve System, Washington, DC 20551

This article describes a computer algorithm that solves the stochastic growth model by iterating on a fixed-point equation in the decision rule determining consumption as a function of the state variables. This algorithm does not discretize the state space, but rather it preserves the continuous domain of the capital stock and the productivity shock. The main advantage of this algorithm is that it is based on a Euler equation and thus it has a straightforward generalization to dynamic economies that cannot be solved by a central planner, such as a non-Pareto optimal competitive economy.

KEY WORDS: Algorithm; Bilinear interpolation; Decision rule; Dynamic model; Quadrature.

1. INTRODUCTION

This article describes the mechanics of implementing an algorithm developed in Coleman (1987, 1989a) and described in general terms in Coleman (1989b) to the stochastic growth model. The basis of the algorithm is a fixed-point equation in the consumption function derived from the problem's Euler equation. In the references just cited, I prove that this fixed-point equation (for the version of the problem considered there) has a unique positive solution and that iterations based on this equation uniformly converge to this solution. For Pareto optimal economies that can be solved by a central planner, such as the stochastic growth model, this algorithm is equivalent to value-function iteration (although they may differ in their implementation and apparently in their rate of convergence). One key advantage of the method described here is that it has a straightforward extension to non-Pareto optimal economies with significant computational savings over attempts at extending the value-function approach in this direction.

2. THE FIXED-POINT EQUATION

The Euler equation for the optimal consumption function c can be written as

$$u'[c(k, \theta)] = \beta \int u'\{c[f(k, \theta) - c(k, \theta), \rho\theta + \varepsilon]\} \times f_1[f(k, \theta) - c(k, \theta), \rho\theta + \varepsilon]\, d\mu(\varepsilon),$$

where, for the particular example considered here, $u'(c) = c^{-\tau}$, $f(k, \theta) = e^{\theta}k^{\alpha} + k$, $f_1(k, \theta) = df(k, \theta)/dk$, and μ is the Normal$(0, \sigma^2)$ cdf. Based on this equation, define the function A, which maps consumption functions into consumption functions, by

$$u'[(Ac)(k, \theta)] = \beta \int u'\{c[f(k, \theta) - (Ac)(k, \theta), \rho\theta + \varepsilon]\} \times f_1[f(k, \theta) - (Ac)(k, \theta), \rho\theta + \varepsilon]\, d\mu(\varepsilon), \quad (1)$$

where the optimal consumption function is a fixed point $c = A(c)$. Note that $A(c)$ is defined pointwise and that $A(c)$ is an argument of c. The algorithm then simply constructs the sequence $\{c_n\}$, defined recursively by $c_{n+1} = A(c_n)$, where c_0 is some initial guess [if c is an increasing function of its first argument, then $A^n(c)$ is uniquely defined for every n]. Note that for the finite horizon stochastic growth model with the terminal consumption function fixed at c_0, c_n corresponds to the optimal consumption function n steps away from the terminal date.

To see that this method for solving the stochastic growth model is equivalent to value-function iteration, let $v(k, \theta)$ represent the value function and denote the standard contraction mapping in the value function by $v = T(v)$. Corresponding to the value function v is a policy function c and the derivative v_1 of the value function with respect to its first argument. Corresponding to T is thus a fixed-point equation $(v_1, c) = B(v_1, c)$, which can be written as $[B = (B_{v_1}, B_c)]$

$$u'[B_c(v_1, c)(k, \theta)] = \beta \int v_1\{f(k, \theta) - B_c(v_1, c)(k, \theta), \rho\theta + \varepsilon\}\, d\mu(\varepsilon)$$

and

$$B_{v_1}(v_1, c)(k, \theta) = u'[B_c(v_1, c)(k, \theta)]f_1(k, \theta).$$

By setting the initial $v_1(k, \theta) = u'[c(k, \theta)]f_1(k, \theta)$, the

sequence of consumption functions for the two algorithms are identical: $A^n(c) = B_c^n(v_1, c)$.

Clearly, more structure must be imposed on the problem to construct A (e.g., differentiability and concavity), and thus the value-function approach, which does not require this additional structure, is more general for solving dynamic programming problems. Being based on a Euler-equation representation of the solution though, A can easily handle equilibrium conditions for competitive economies. For example, equating the representative agent's capital with aggregate capital is done at the Euler-equation level, but not at the consumer's problem level of the value function. To construct the solution, the value-function approach must distinguish between these two types of capital and their respective laws of motion, even though they become the same quantities at the equilibrium. The policy-function-iteration approach described here can thus best be understood as a Euler-based extension to competitive economies of the value-function approach that agrees with that approach for centrally planned economies.

3. PROGRAMMING THE ALGORITHM

To implement the algorithm corresponding to (1) on a digital computer, a number of computational issues must be addressed, none of which has a uniformly best resolution. These issues are common to many problems, so I will not dwell on them too deeply for the particular implementation of the algorithm described here.

There are essentially two ways to transform the infinite dimensional problem corresponding to (1) to a finite dimensional problem. One way is to discretize the state space by allowing only a finite number of values for (k, θ). Bellman (1971, chap. 3) presented a discussion for the endogenous state variables, and Tauchen (1987) suggested a way to approximate the continuous-state-space Markov process by a finite-state-space one. The other way to transform the problem, which is the way I chose here, is to keep the domain intact but to approximate the consumption function c by an element in some finite dimensional set and to approximate the integration by a finite dimensional operator. There are many ways to do this, and I have had success with a piecewise bilinear approximation to $\ln c$ based on a rectangular grid of (k, θ) and a Hermite–Gauss quadrature rule for the integration with respect to the Normal density. [See Hildebrand (1956) for a discussion of numerical quadrature, which essentially imposes a discrete probability model on ε but not on θ.] To see that an interpolation routine is necessary, note that the first argument of c in (1) is $f(k, \theta) - (Ac)(k, \theta)$, which, since $(Ac)(k, \theta)$ is chosen such that (1) holds exactly at the set of grid points, must be allowed to take on arbitrary values. Moreover, for an arbitrary grid of θ and a fixed number of quadrature points for ε, in general the second argument of c, $\rho\theta + \varepsilon$, will not coincide with a grid point.

A remaining issue is a choice of a grid for (k, θ) from which to base the bilinear interpolation routine. Clearly, (k, θ) has, at the solution, a stationary joint distribution, so on the one hand a choice of a grid to, in some sense, best approximate this distribution would allow one to economize on grid points to study the behavior of sample time paths. Concentrating grid points around the mean, for example, would provide greater flexibility for the shape of the decision rules at a region that contains most of the sampled points. On the other hand, to study properties of an algorithm over the entire ergodic set, it seems that a more uniform grid is appropriate. For this reason, at this stage of the analysis I chose a uniform grid over $(\ln k, \theta)$ because it concentrates grid points towards the origin, a region most likely to cause problems.

For any point (k, θ), finding $(Ac)(k, \theta)$ that solves (1) is an easy matter, since the right side of (1) minus the left side is a strictly increasing function of $(Ac)(k, \theta)$. The variant of Newton's algorithm that I use is International Mathematical and Statistical Libraries' (1987) NEQNF, which uses the Levenberg–Marquardt algorithm with a finite difference approximation to the Jacobian. To ensure that guesses of $(Ac)(k, \theta)$ are feasible, let the root-finding routine guess $\ln\{(Ac)(k, \theta)/[f(k, \theta) - (Ac)(k, \theta)]\}$ and then invert this to find the implied guess of $(Ac)(k, \theta)$. Moreover, if a guess leads to a point outside of the grid, simply extend the bilinear segments of c at the boundary.

To summarize, to implement this algorithm you first choose a grid, initialize c_0 at each grid point, and then construct $c_1 = A(c_0)$ at each grid point such that (1) holds at each of these points. If the distance between c_0 and c_1, say $\sup|\ln c_1 - \ln c_0|$ over the grid points, is larger than some convergence parameter ε, replace c_0 by c_1 and iterate again (clearly logs are appropriate in the distance function to obtain the same degree of precision over the entire domain of c; that is, only percentage differences matter). The computer program that I use to solve the stochastic growth model is in the appendix to Coleman (1989b).

4. AN EXAMPLE

With the initial consumption function set to $c_0 = [1 - \alpha\beta/(1 - \beta + \alpha\beta)]f$ (which provides the correct average propensity to consume at the deterministic steady states) and for 50 uniformly spaced grid points of $\ln k$ from -4 to 4, 20 grid points of θ from $-.65$ to .65, nine points in the quadrature rule, $\beta = .95$, $\tau = .5$, $\alpha = .33$, $\sigma = .1$, $\rho = .95$, and a convergence parameter of $\varepsilon = .0001$, it took 110.92 seconds for the algorithm to converge on an Amdahl 5890-300 computer (38 million instructions per second) running VS–FORTRAN, version 2.3. For 50 grid points of $\ln k$, 20 grid points of θ, and three quadrature points, it took 54.22 seconds to converge, and for 25 grid points of $\ln k$, 9 grid points of θ, and three quadrature points, it

took 12.08 seconds. Although clearly it took different amounts of time to converge starting from different consumption functions, as long as the initial consumption function was feasible, positive, and an increasing function of the capital stock, the algorithm always converged.

5. GENERATING A TIME SERIES

With the decision rule c it is an easy matter to generate a sample time series of capital and consumption. Let $g(k, \theta) = f(k, \theta) - c(k, \theta)$ denote the equilibrium investment function, and let $\{\varepsilon_t\}$ correspond to a sequence of iid random variables drawn according to a Normal$(0, \sigma^2)$ distribution. Given any initial (k_0, θ_0), construct the remaining series $\{k_t, \theta_t\}$ recursively according to $k_{t+1} = g(k_t, \theta_t)$ and $\theta_{t+1} = \rho\theta_t + \varepsilon_{t+1}$. Generating a time series even as long as 100,000 will require only trivial amounts of computer time (it took .32 seconds to generate a sample time series of 2,000 observations).

6. CONCLUSION

The algorithm just described has apparently been successful in solving a rather wide variety of models. For example, in Coleman (1989a) the algorithm was used to solve the stochastic growth model with distortionary production taxes, and in Coleman (1987) the algorithm was used to solve a monetary growth model. The algorithm has also been used to solve a multicountry model of trade flows, a model of interest rates, and a model of capital and labor with various production lags. The success of this algorithm is probably due to its close association with value-function iteration, which is based on a contraction mapping. As mentioned previously, the algorithm exploits some additional structure that many problems have, and thus leads to a sharper characterization of the solution to dynamic programming problems, which is also a characterization of the solution to competitive equilibrium problems.

ACKNOWLEDGMENTS

The algorithm presented in this article was developed in my Ph.D. dissertation at the University of Chicago, for which Robert E. Lucas, Jr., Lars P. Hansen, and Robert M. Townsend provided many helpful comments.

[*Received December 1988.*]

REFERENCES

Bellman, Richard (1971), *Introduction to the Mathematical Theory of Control Processes* (Vol. 2), New York: Academic Press.

Coleman, Wilbur John, II (1987), "Money, Interest, and Capital," unpublished Ph.D. dissertation, University of Chicago, Dept. of Economics.

——— (1989a), "Equilibrium in a Production Economy With an Income Tax," unpublished manuscript, Federal Reserve Board, Washington, DC.

——— (1989b), "An Algorithm to Solve Dynamic Models," International Finance Discussion Working Paper 351, Federal Reserve Board, Washington, DC.

Hildebrand, F. B. (1956), *Introduction to Numerical Analysis*, New York: McGraw-Hill.

International Mathematical and Statistical Libraries, Inc. (1987), *IMSL Math/Library* (Version 1.0), Houston: Author.

Tauchen, George (1987), "Quadrature-Based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models," unpublished manuscript, Duke University, Dept. of Economics.