JOURNAL OF ECONOMIC THEORY 58, 410–452 (1992)

Projection Methods for Solving Aggregate Growth Models*

KENNETH L. JUDD

Hoover Institution, Stanford University, Stanford, California 94305 and National Bureau of Economic Research, 1055 Massachusetts Avenue, Cambridge, Massachusetts 02138

Received May 16, 1991; revised June 19, 1992

We describe a general numerical approach, the projection method, to solve operator equations which arise in economic models. Principles from numerical analysis are then used to develop efficient implementations of the projection method for solving aggregate growth models. Since any numerical approach will involve error, we derive error measures which are related to optimization errors by agents and argue that the numerical approximations can be viewed as equilibria with boundedly rational agents. The results are programs which run hundreds of times faster than competing methods in the literature while achieving high accuracy. *Journal of Economic Literature* Classification Numbers: C63, C68. © 1992 Academic Press, Inc.

1. INTRODUCTION

Economists are increasingly turning to numerical techniques for analyzing dynamic economic models. While numerical techniques have been used in dynamic economic analysis since Gustafson [14], the set of problems analyzed with numerical methods has expanded, and more effort has been expended on developing and evaluating algorithms (see, for example, Taylor and Uhlig [32]). While progress has been substantial, the numerical techniques have tended to be, or at least have appeared to be, problem specific. Also, economists have often ignored standard numerical methods in favor of idiosyncratic and *ad hoc* methods.

This paper presents a general approach to the numerical solution of dynamic economic problems with a focus on its application to simple

* This paper originally circulated as "Minimum Weighted Residual Methods for Solving Agregate Growth Models." The author gratefully acknowledges the comments of seminar participants at Northwestern University, Stanford University, the Santa Fe Institute, the comments and suggestions of two anonymous referees, and the research assistance of Pamela Chang. This work was supported by NSF Grants SES-8810935 and SES-9012128.

0022-0531/92 $5.00

aggregate growth problems. More specifically, we will discuss a class of techniques, called *projection* methods, which have been developed in the mathematical literature on numerical solutions to partial differential equations, and appear to be generally applicable to economic problems.[1] Many continuous-time economic problems can be formulated as partial differential equations, implying that projection methods would be applicable. For some economic applications, a discrete-time formulation may be preferred. We will show that the basic ideas of projection techniques can be adapted for discrete-time applications.

The greatest value of the projection approach is its robustness and flexibility. We will describe the general projection approach for solving problems and show that most of the techniques currently used by economists are also projection methods when viewed from the general perspective. With this common framework, we can discuss and compare many numerical methods. In particular, numerical analytic ideas will show why some methods outperform others and how to devise efficient algorithms.

The basic idea of projection techniques is to first express equilibrium as a zero of an operator, $\mathcal{N}: B_1 \to B_2$, where B_1 and B_2 are function spaces. That operator can be an ordinary differential equation, as in optimal control problems, a partial differential equation, as in continuous-time dynamic programming, or a more general functional equation, as in Euler equations expressing necessary conditions for recursive equilibria (as formulated in Prescott and Mehra [26]). Of course, space and time limitations make it impossible for computers to store and evaluate all the possible elements of B_1. To make the problem tractable, projection methods focus on a finite-dimensional subspace of candidates in B_1 which can be easily represented on a computer and is likely to contain elements "close" to the true solution. It may be difficult for the computer to compute $\mathcal{N}$, in which case we find a computable operator, $\hat{\mathcal{N}}$, which is "similar" to $\mathcal{N}$. Within the space of candidate solutions, we then find an element which is "almost" a zero of $\hat{\mathcal{N}}$.

While the basic idea is natural, there are many details. The key details are specifying the subspace within which we look for an approximate solution and its computer representation, defining what "close" and "almost" mean, and finding the approximate solution. By studying these details, we will see how to implement these ideas efficiently to solve numerically interesting dynamic nonlinear economic problems.

[1] Earlier versions of this paper used the term "minimum weighted residual" instead of "projection." The change was made since "projection" is one of the terms used in the mathematical literature to describe these methods (see, for example, Golub and Ortega [13]), projection operations are basic in econometrics, and "projection" is easier to pronounce.

This paper has two purposes. First, it lays out the general projection approach to solving dynamic economic problems. Second, we demonstrate the power, practicality, accuracy, and efficiency of particular implementations of the projection method for solving stylized discrete-time growth models. This is a useful focus since optimal growth problems are important, and several other authors have discussed the speed of their proposed solution methods for this problem in Taylor and Uhlig [32] and the accompanying papers. We will see that most of those methods fall within the general projection approach. Within this common framework we are able to discuss the relative merits of various implementations of the common underlying ideas. In particular, the algorithms developed in this paper run hundreds of times faster than the methods discussed in Taylor and Uhlig [32], a fact which can be explained by applying general numerical analytic principles within this projection framework.

Space limitations require a narrow focus, but from these examples it is clear that this method can be applied to a diverse collection of dynamic economic problems. Judd [18] shows how to use these methods to solve dynamic programming problems, general dynamic equilibrium with distortions, game-theoretic dynamic models in both discrete and continuous time, and even partially revealing rational expectations equilibria of asset markets with asymmetric information.

No reader will be surprised that there are techniques which will solve these problems. A few seconds of thought combined with brute force and a supercomputer will solve any problem discussed here or in Judd [18]. The challenge, of course, is to find techniques which can run quickly on computers currently available. There are two basic reasons why we want fast algorithms. First, we want algorithms which can be used for econometric estimation of the underlying nonlinear models.[2] For example, using the programs described below one could easily compute maximum likelihood estimates of simple macroeconomic models. Second, the only way to convincingly demonstrate comparative dynamic[3] or comparative static propositions would be to calculate hundreds or thousands of examples. Third, more efficient methods will be able to analyze more complex economic models.

The examples discussed below needed little computer power. To emphasize this, I have limited myself to using relatively slow equipment. In examples below, I give running times on a Compaq 386/20, running under DOS 3.1 and using a Weitek math coprocessor. The key fact is that these

[2] Rust [28] discusses the use of numerical solutions within maximum likelihood procedures.

[3] Danthine and Donaldson [10] is an early good example of the value of numerical techniques for studying comparative dynamics in growth model.

problems can be solved on computers sitting on the desks of many economists and available to most graduate students. Furthermore, I use standard, freely available software. All the programs were written in FORTRAN, but could be implemented in any programming language, such as BASIC or GAUSS.[4]

Section 2 describes the application of the projection method to a simple differential equation. This is done to illustrate how the approach and focus of projection methods differ from more conventional methods for solving operator equations. Section 3 then describes the projection method for operator equations on general function spaces. Section 4 uses basic numerical analysis techniques to apply the projection method to a simple deterministic optimal growth problem, and Section 5 accomplishes this for a stochastic optimal growth problem. Section 6 compares the algorithms in Section 5 to others which have been proposed, pointing out that all methods can be viewed as applications of the projection method, and that the slow speed of previous methods can be traced to inefficient choices. Section 7 discusses the convergence problem and Section 8 concludes.

This paper is not intended to be an exhaustive review of numerical and approximation techniques.[5] There are many other approaches which are also of potential values, including regular and singular perturbation techniques and finite difference schemes, each of which has its own particular strengths (see Judd [16]). One reason for examining projection techniques is that they are similar to standard econometric techniques; hence they will appear familiar to economists. Ultimately, one expects that the full range of approximation and numerical techniques will be valuable to economists with the best algorithms combining techniques.

2. A Simple Example

Before describing projection techniques in full generality, it is useful to examine how they would be used in a simple example. Seeing these techniques used in a familiar context help us understand the basic ideas behind projection techniques.

Suppose we want to solve the differential equation

$$y' = g(y, t), \tag{2.1}$$

[4] Upon request, I will provide the FORTRAN code by E-mail. My E-mail address is Judd @ Hoover.bitnet.

[5] I have tried to make this paper self-contained and accessible to readers not familiar with numerical analysis. I doubt that I have been totally successful. For more detailed discussion of the techniques I use, I refer the reader to numerical analysis books such as Atkinson [1], Golub and Ortega [13], and Judd [16].

together with the initial condition

$$y(0) = 1 \tag{2.2}$$

for $t \in [0, T]$. The most popular approaches for this problem are finite difference methods, such as the Euler and Runge–Kutta schemes, which convert differential equations into difference equations.[6] In this paper, however, we are interested in projection techniques and will illustrate them by applying the basic idea to this simple ordinary differential equation.

The first step is to change the way we view (2.1). We begin by defining the operator

$$Ly \equiv y' - g(y, t). \tag{2.3}$$

L maps $C^1[0, T]$ onto $C^0[0, T]$. The differential equation (2.1), (2.2) can be reformulated as finding a zero of an operator,

$$Ly = 0, \tag{2.4}$$

which satisfies (2.2).

The next step is to find a way to represent potential solutions on the computer. A natural set of possible solutions is $\{y \in C^1[0, T] \mid y(0) = 1\}$, but it cannot be represented on a computer. By the Stone–Weierstrass theorem, any $C^1[0, T]$ function can be expressed as the uniform limit of a sequence of polynomials. However, since our computer can only handle finite objects, projection methods find some finitely representable approximation, such as

$$\hat{y}(t; \mathbf{a}) \equiv 1 + \sum_{i=1}^{n} a_i t^i,$$

which "fits" (2.4) with "small" error. Note that the initial condition has been taken care of since $\hat{y}(0; \mathbf{a}) = 1$ for any choice of $\mathbf{a}$. We have now reduced the infinite-dimensional problem to a finite-dimensional one since the problem now is to find a "good" $\mathbf{a} \in R^n$.[7]

[6] For example, the Euler scheme fixes a step size Δt and solves the difference equation $y(t + \Delta t) = y(t) + g(y, t)\,\Delta t$; $y(0) = 1$ initiates the difference scheme which then continues for n steps, where n is the least integer such that $n\,\Delta x > T$.

[7] Whereas finite-difference methods focus on computing values of a solution at specified values of t, projection methods focus on computing coefficients of a decomposition of a solution. If I may borrow jargon from the time series literature, one can roughly think of finite-difference methods as state-space methods and projection methods as spectral methods. Indeed, an important class of projection methods are called spectral methods.

We next operationalize the idea of a good "fit." We define the residual function,

$$R(t;\mathbf{a}) \equiv L\hat{y}(t;\mathbf{a}) = \sum_{i=1}^{n} i a_i t^{i-1} - g\left(1 + \sum_{i=1}^{n} a_i t^i, t\right).$$

If the parameters $\mathbf{a}$ made $\hat{y}(t;\mathbf{a})$ a solution of (2.1), then $\mathbf{a}$ would make the residual function zero for all $t \in [0, T]$. Projection methods choose $\mathbf{a}$ so as to make $R(t;\mathbf{a})$, when viewed as a function of t, as close to the zero function as possible.

The *least-squares* approach defines the fit to be the L^2 norm of the residual function, and chooses $\mathbf{a}$ so as to minimize the "sum of squared residuals"; that is, we solve

$$\min_{\mathbf{a}} \int_0^T R(t;\mathbf{a})^2 \, dt.$$

The least-squares method is a direct implementation of the idea to make the error of the approximation small. In general, one could develop alternative implementations by using different measures of the "loss" due to a nonzero R.

Most projection techniques find a good-fitting approximation in a less direct fashion. These techniques begin with a simple observation: since the true solution would have a zero residual error function, the product of the residual function with any other function would be zero. Therefore, one way to find the n arguments of $\mathbf{a}$ is to fix n functions, $p_i(t)$, $i = 1, ..., n$, and choose $\mathbf{a}$ so that

$$\int_0^T p_i(t) \, R(t;\mathbf{a}) \, dt = 0, \qquad i = 1, ..., n.$$

There are obviously many ways to implement this idea with different specifications of the $p_i(t)$. For example, the *method of moments* implementation uses $p_i(t) = t^{i-1}$.

A somewhat different method is *collocation*. It finds an approximation which satisfies the differential equation at some chosen points. Specifically, we first choose a fixed set of n points $\{t_i\}_{i=1}^n$ in $[0, T]$, and then we choose $\mathbf{a}$ so that

$$R(t_i;\mathbf{a}) = 0, \qquad i = 1, ..., n,$$

which is a system of nonlinear equations in $\mathbf{a}$ if g is nonlinear.

In the interest of space, we do not report on the quality of the solutions here; see Judd [16] for a more complete discussion of an explicit example.

This simple example illustrates the basic approach of projection procedures applied to a familiar problem. First we express the problem as the zero of some operator. We then express the candidate approximations as finite sums of simple functions. The coefficients of these finite sums are then fixed by imposing conditions which would be satisfied by the exact solution. The projection method directly reduces this simple problem to a set of equations for **a**, a simple numerical problem. With this example in mind, we will now discuss the general projection method of solution and examine the variety of numerical procedures which it may involve.

3. General Algorithm

We now give some idea of the usefulness of the projection method by describing it in a more general context. One begins with an operator equation representation of the problem; that is, one reduces the economic problem to finding an operator $\mathcal{N}$ and a function f such that equilibrium is represented by the solution to

$$\mathcal{N}(f) = 0,$$

where $f: D \subset R^N \to R^M$, $\mathcal{N}: B_1 \to B_2$, and the B_i are function spaces. In our simple example above, $D = [0, T]$, $f: D \to R^1$, $\mathcal{N}(f) = (d/dt)(f(t)) - g(f(t), t)$, $B_1 = C^1[0, T]$, and $B_2 = C^0[0, T]$. A wide variety of economic problems can be so represented. For example, in optimal growth problems the domain D contains N state variables, the unknown function f is a vector of decision rules, and the operator $\mathcal{N}$ is a vector of M Euler conditions. Typically, $\mathcal{N}$ is a composition of algebraic operations, differential and integral operators, and functional compositions, and is frequently nonlinear.

We shall show how to implement the canonical projection technique in a step-by-step fashion. We first give an overview of the approach, then highlight the critical issues for each step and discuss how the steps interact.

The first step is to decide how to represent approximate solutions. One general way is to assume that our approximation, $\hat{f}$, is built up as a linear combination of simple functions.[8] We will also need a concept of when two functions are close. Therefore, the first step is to choose a basis and an appropriate concept of distance:

[8] Nonlinear combinations are also possible, but most current methods stay with linear combinations since linear approximation theory is a much more developed theory than nonlinear approximation theory.

Step 1. Choose bases, $\Phi_j = \{\varphi_i^j\}_{i=1}^{\infty}$, and inner products, $\langle \cdot,\cdot \rangle_j$, over B_j, $j = 1, 2$.

The basis over B_1 should be flexible, capable of yielding a good approximation for the solution, and the inner products should induce useful norms on the spaces. Next, we decide how many basis elements to use and how to implement $\mathcal{N}$:

Step 2. Choose a degree of approximation n for f, a computable approximation $\hat{\mathcal{N}}$ of $\mathcal{N}$, and a collection of n functions from B_2, $p_i: D \to R^M$, $i = 1, ..., n$.

The approximate solution will be $\hat{f} \equiv \sum_{i=1}^{n} a_i \phi_i(x)$. The convention is that the ϕ_i increase in "complexity" and "nonlinearity" as i increases and that the first n elements are used. The best choice of n cannot be determined *a priori*. Generally, the only "correct" choice is $n = \infty$. Larger n should yield better approximations, but one is most interested in the smallest n which yields an acceptable approximation. One initially begins with small n and increases n until some diagnostic indicates that little is gained by continuing. Similar issues arise in choosing $\hat{\mathcal{N}}$. Sometimes, as in Section 2, we can take $\hat{\mathcal{N}} = \mathcal{N}$. The p_i will be the projection directions used to determine **a**.

Step 1 lays down the topological structure of our approximation and Step 2 fixes the degrees of freedom of the approximation. Once we have made these basic decisions, we begin our search for an approximate solution to the problem. Since the true solution f satisfies $\mathcal{N}(f) = 0$, we will choose as our approximation some $\hat{f}$ which makes $\hat{\mathcal{N}}(\hat{f})$ "nearly" equal to the zero function. Since $\hat{f}$ is parameterized by **a**, the problem reduces to finding an **a** which makes $\hat{\mathcal{N}}(\hat{f})$ nearly zero. This search for **a** is the focus of Steps 3–5.

Step 3. For a guess **a**, compute the approximation, $\hat{f} \equiv \sum_{i=1}^{n} a_i \varphi_i(x)$, and the residual function,

$$R(x; \mathbf{a}) \equiv (\hat{\mathcal{N}}(\hat{f}))(x).$$

The first guess of **a** should reflect some initial knowledge about the solution. After the initial guess, further guesses are generated in Steps 4 and 5, where we see how we use the inner product, $\langle \cdot,\cdot \rangle_2$, to define what "near" means.

Step 4. For each guess of **a**, compute the n projections, $P_i(\cdot) \equiv \langle R(\cdot; \mathbf{a}), p_i(\cdot) \rangle_2$, $i = 1, ..., n$.

Step 5. By iterating over Steps 3 and 4, find **a** which sets the n projections equal to zero.

This general algorithm breaks the numerical problem into several distinct steps. It points out the many distinct techniques of numerical analysis which are important. First, in Steps 1 and 2 we choose the finite-dimensional space wherein we look for approximate solutions, hoping that within this set there is something "close" to the real solution. These steps require us to think seriously about approximation theory methods. Second, Step 4 will involve numerical integration if we cannot explicitly compute the integrals which define the projections. Third, Step 5 is a distinct numerical problem, involving the solution of a nonlinear set of simultaneous equations or the solution of a minimization problem. We shall now consider each of these numerical problems in isolation.

3.1. *Choice of Basis and Inner Product*

There are many criteria which the basis and inner product should satisfy. The full basis Φ_1 for the space of candidate solutions should be "rich"; in particular, it should be complete in B_1. We will generally use inner products of the form

$$\langle f(x), g(x) \rangle \equiv \int_D f(x)\, g(x)\, w(x)\, dx$$

for some weighting function $w(x) \geqslant 0$.

Computational considerations also play a role in choosing a basis. The φ_i should be simple to compute. They should be similar in size to avoid scaling problems.[9] While asymptotic results such as the Stone–Weierstrass theorem may lull one into accepting polynomial approximations, practical success requires a basis where only a few elements will do the job. This requires that the basis elements should "look something like" the solution. In particular, our discussion below shows that we should use smooth functions to approximate smooth functions. On the other hand, properties like monotonicity are not convenient since they are not preserved by linear combinations.

If few terms are to suffice, the individual terms should also be "different." In Section 3.4 we will see that orthogonality with respect to the inner product $\langle \cdot, \cdot \rangle_1$ is one sense in which they should differ. Nonorthogonal bases will reduce numerical accuracy just as collinear regressors enlarge confidence intervals in regression.

Examples of possible bases are numerous. First, one may consider the ordinary polynomials, $\{1, x, x^2, x^3, \ldots\}$. If B_1 is the set of bounded measurable functions on a compact set then the Stone–Weierstrass theorem

[9] Scaling problems are those problems which arise when numbers of vastly different magnitudes interact. For example, $10^{10} + 10^{-10} = 10^{10}$ on most machines.

assures us of their completeness in the L^1 norm. However, they may not be a good choice since they are very similar. For example, they are all monotonically increasing and positive on R^+ and they will not be orthogonal in any natural inner product on R^+. They will also vary a great deal in size, leading to scaling problems. This does not mean that we cannot use ordinary polynomials, just that it is preferable to use orthogonal collections. A generally useful choice is the Chebyshev polynomial family.[10]

Chebyshev Polynomials

Because of their usefulness, we will review the key properties of Chebyshev polynomials. They are defined over $[-1, 1]$ by the formula $T_n(x) \equiv \cos(n \arccos x)$. They are generated by the recursion scheme $T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$, which is initialized by $T_0(x) = 1$ and $T_1(x) = x$. The restriction to $[-1, 1]$ is inessential since one can define Chebyshev polynomials over any bounded interval by a linear transformation. One hint of their special properties is given by the identity $T_n(\cos\theta) = \cos n\theta$ which in itself is an orthogonal sequence on $[0, 2\pi]$. This identity ties Chebyshev polynomials to Fourier analysis (see Rivlin [27].)

The Chebyshev polynomials obey the continuous orthogonality relationship

$$\int_{-1}^{1} T_i(x)\, T_j(x)(1-x^2)^{-1/2}\, dx = 0, \qquad i \neq j.$$

Hence, Chebyshev polynomials are orthogonal on $[-1, 1]$ with respect to the inner product defined by the weighting function $(1-x^2)^{-1/2}$, which leads us to use this inner product when we use a Chebyshev polynomial basis. The ability of Chebyshev polynomials to approximate smooth functions is summarized in the following theorem (see Rivlin [27] for proofs):

CHEBYSHEV APPROXIMATION THEOREM. *Assume* $f \in C^r[-1, 1]$. *Let*

$$C_n(x) \equiv \frac{1}{2} c_0 + \sum_{j=1}^{n} c_j T_j(x),$$

[10] Nonpolynomial alternatives include various sequences of trigonometric and exponential functions. The choice depends on the range, D, computational demands, and the expected shape of a solution. In physics, trigonometric bases such as $\{1, \sin x, \sin 2x, \sin 3x, \ldots\}$ are often used since solutions are often periodic, allowing for Fourier series techniques. In economic problems, however, solutions are generally not periodic, and periodic approximations to nonperiodic functions require many terms, a fact implied by the Gibbs phenomenon.

where

$$c_j \equiv \frac{2}{\pi} \int_{-1}^{1} \frac{f(x)\, T_j(x)\, dx}{\sqrt{1-x^2}}.$$

Then there is a b such that, for all $n \geqslant 2$,

$$\|f - C_n\|_\infty \leqslant \frac{b \ln n}{n^r}.$$

Hence $C_n \to f$ *uniformly as* $n \to \infty$. *Furthermore, there is a constant c such that*

$$|c_j| \leqslant c/j^r, \qquad j \geqslant 1.$$

This theorem will help us assess the quality of our approximations since both the error and the coefficients eventually drop off rapidly for smooth functions. Also important for our purposes is the discrete orthogonality relationship

$$\sum_{l=1}^{n} T_i(z_l^n)\, T_j(z_l^n) = 0, \qquad i \neq j,$$

where we define the z_l^n, $l = 1, ..., n$, to be the zeroes of T_n, given by

$$z_l^n \equiv \cos\left(\frac{(2l-1)\pi}{2n}\right), \qquad l = 1, ..., n.$$

This discrete relationship leads us to consider discrete approximation methods. Suppose $f(x)$ is a continuous function and

$$c_j \equiv \frac{2}{n} \sum_{l=1}^{n} f(z_l^n)\, T_j(z_l^n), \qquad j = 0\ 1, ..., n-1.$$

Then the function

$$I_{n-1}^f(x) = -\frac{1}{2} c_0 + \sum_{k=1}^{n-1} c_k T_k(x)$$

agrees with $f(x)$ on the z_l^n, thereby being a polynomial of degree $n-1$ which interpolates $f(x)$ at the z_l^n. In this way, we can approximate a function by interpolation at the points z_l^n. We call I_{n-1}^f the *degree* $n-1$ *Chebyshev interpolant of f*.

One can interpolate at any collection of $n+1$ points. There are, however, strong reasons to choose I_n^f. Suppose that we want to find an nth degree

polynomial approximation, $p_n(x)$, which keeps the maximum error, $\|f-p_n\|_\infty$, small. The following theorem (see Rivlin [27]) indicates the advantages of I_n^f.

CHEBYSHEV INTERPOLATION THEOREM. *Suppose* $f \in C^k[a, b]$. *Then there is some* d_k *such that for all* n

$$\|f - I_n^f\|_\infty \leqslant \left(\frac{2}{\pi}\log(n+1)+2\right)\frac{d_k}{n^k}\|f^{(k)}\|_\infty.$$

Furthermore, interpolation at Chebyshev zeroes is optimal.

This theorem says that the Chebyshev interpolant converges to f rapidly as we use more Chebyshev zeroes. Convergence may seem to be an unremarkable property, but interpolation at uniformly spaced points does not necessarily converge as we use more points. Given these properties, Chebyshev methods are valuable whenever the approximated function is smooth.

Other Bases

Depending on the nature of the bases, projection methods are separated into *finite element* and *spectral* methods; both types have been used in economic applications. Spectral techniques use basis functions which are almost everywhere nonzero on D, such as Chebyshev polynomials. These basis functions are also C^∞, imposing (possibly undesirable) smoothness conditions on the approximate solutions.

On the other hand, finite-element techniques use basis functions which have small support; that is, each is zero except over a small portion of D and, at each point of D, all but a few of the basis elements are zero. For example, step function approximations on $[a, b]$ are generated by a basis of step functions, $\{\varphi_i : i = 1, ..., n\}$, where $h = (a-b)/n$ and

$$\varphi_i(x) = \begin{cases} 0, & a \leqslant x \leqslant a + (i-1)h \\ 1, & a + (i-1)\,h \leqslant x \leqslant a + ih \\ 0, & a + ih \leqslant x < b. \end{cases}$$

Piecewise linear approximations are generated by a basis of *tent functions*, that is, functions of the form, for $i = 0, ..., n$,

$$\varphi_i(x) = \begin{cases} 0, & a \leqslant x \leqslant a + (i-1)\,h \\ (x - (a + (i-1)h))/h, & a + (i-1)\,h \leqslant x \leqslant a + ih \\ 1 - (x - (a + (i-1)\,h))/h, & a + ih \leqslant x \leqslant a + (i+1)\,h \\ 0, & a + (i+1)\,h \leqslant x \leqslant b. \end{cases}$$

These are called tent functions since $\varphi_i(x)$ is zero to the right of $a+(i-1)h$, rises linearly to a peak at $a+ih$, and then falls back to zero at $a+(i+1)h$, and remains zero. More generally, one can compute piecewise degree k polynomial approximations which split $[a, b]$ into subintervals and equal a degree k polynomial on each subinterval. Hermite and spline interpolation methods (see Prenter [25]) are examples of piecewise cubic approximation methods.

Chebyshev interpolation is asymptotically superior to piecewise linear and piecewise constant approximations for $f \in C^3[a, b]$. By Taylor's theorem, on the interval $[x, x+h]$ the L^∞ error of any constant approximation to f is at best $f'(x)\, h/2 + \mathcal{O}(h^2)$, and the L^∞ error of a linear approximation is at best $f''(x)\, h^2/16 + \mathcal{O}(h^3)$. Therefore, the error of any piecewise constant approximation of f over $[a, b]$ to be $M_1 h/2 + o(h)$, where $M_1 \equiv \max_{x \in [a,b]} f'(x)$, and the error of any piecewise linear approximation is $M_2 h^2/16 + o(h^2)$, where $M_2 = \max_{x \in [a,b]} f''(x)$. Since h is inversely proportional to the number of basis elements needed to represent a piecewise constant or linear approximation, if we use n basis elements the error for a step function approximation can be $M_1/(2n)$, and the error for a piecewise linear approximation can be $M_2/(16n^2)$. Both are asymptotically inferior to Chebyshev interpolation error of I_n^f, which is of order $n^{-3} \ln n$. For $f \in C^k[a, b]$ and $k > 3$, the performance gap widens rapidly. Note that in this discussion, we are comparing worst-case error bounds, the typical approach taken in numerical analysis.

While these asymptotic results are strong, real-life computing also needs to watch the proportionality constants. In the Chebyshev interpolation theorem, the error bound on I_n^f for $f \in C^3$ is proportional to $\|f'''\|_\infty$, whereas the piecewise linear approximation error is proportional to $\|f''\|_\infty$. While the asymptotic error is $n^{-3} \ln n$, for small n the piecewise linear approximation is superior if $\|f'''\|_\infty$ is much larger than $\|f''\|_\infty$. Geometrically, this says that if the curvature of f changes rapidly, a piecewise linear approximation will initially do better than Chebyshev interpolation.

These considerations motivate the following rule: if high-order derivatives are not large and f is smooth, use Chebyshev polynomial basis; otherwise, use piecewise polynomial bases. In this paper, the examples are known to have very smooth solutions and we use a spectral approach with a Chebyshev polynomial basis.

Multidimensional Bases

Most interesting problems in economics involve more than one state variable—physical and human capital, capital stocks of competitors, wealth distribution, etc. *Tensor product* methods build multidimensional basis functions up from simple one-dimensional basis functions. If $\{\varphi_i(x)\}_{i=1}^\infty$ is

a basis for functions of one real variable, then the set of pairwise products, $\{\varphi_i(x)\,\varphi_j(y)\}_{i,j=1}^{\infty}$, is the tensor product basis for functions of two variables. To handle n-dimensional problems in general, one can take all the n-wise products and create the *n-fold tensor product* of a one-dimensional basis. One advantage of the tensor product approach is that if the one-dimensional basis is orthogonal in a norm, the tensor product is orthogonal in the product norm. The disadvantage is that the number of elements increases exponentially in the dimension.

There are many ways to form multidimensional bases and avoid the "curse of dimensionality." See Judd [16] for a discussion of hybrid perturbation–Galerkin methods and other alternatives and their applications to economics problems. We will explore *complete polynomial* bases, which grow only polynomially as the dimension increases. To motivate the complete polynomials, we recall Taylor's theorem for many dimensions:

TAYLOR'S THEOREM. *Suppose $f\colon R^n \to R^1$, and is C^{k+1}. Then for $x^0 \in R^n$,*

$$\begin{aligned} f(x) = f(x^0) &+ \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x^0)(x_i - x_i^0) \\ &+ \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i\,\partial x_j}(x^0)(x_i - x_i^0)(x_j - x_j^0) \\ &\vdots \\ &+ \frac{1}{k!}\sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} \frac{\partial^k f}{\partial x_{i_1} \cdots \partial x_{i_k}}(x^0)(x_{i_1} - x_{i_1}) \cdots (x_{i_k} - x_{i_k}^0) \\ &+ \mathcal{O}(\|x - x^0\|^{k+1}). \end{aligned}$$

Note the terms used in the kth degree Taylor series expansion. For $k = 1$, Taylor's theorem uses the linear functions $\mathscr{P}_1 \equiv \{1, x_1, x_2, ..., x_n\}$. For $k = 2$, Taylor's theorem uses

$$\mathscr{P}_2 \equiv \mathscr{P}_1 \cup \{x_1^2, ..., x_n^2, x_1x_2, x_1x_3, ..., x_{n-1}x_n\}.$$

$\mathscr{P}_2$ contains some product terms, but not all; for example, $x_1x_2x_3$ is not in $\mathscr{P}_2$. In general, the kth degree expansion uses functions in

$$\mathscr{P}_k \equiv \left\{ x_1^{i_1} \cdots x_n^{i_n} \,\middle|\, \sum_{l=1}^{n} i_l \leqslant k,\ 0 \leqslant i_1, ..., i_n \right\}.$$

The set $\mathscr{P}_k$ is the *complete set of polynomials of total degree k.*

Complete sets of polynomials are often superior to tensor products for multivariate approximation. The n-fold tensor product of $\{1, x, ..., x^k\}$ contains $(k+1)^n$ elements, far more than $\mathscr{P}_k$. For example, $\mathscr{P}_2$ contains

$1+n+n(n+1)/2$ elements compared to 3^n for the tensor product. Taylor's theorem tells us that many of the tensor product elements add little, saying that the elements of $\mathscr{P}_k$ will yield a kth-order approximation near x^0, and but that the n-fold tensor product of $\{1, x, ..., x^k\}$ can do no better than kth-order convergence since it does not contain all degree $k+1$ terms. This suggests that the complete polynomials will give us nearly as good an approximation as the tensor product with far fewer elements.

3.2. *Choice and Evaluation of Projection Conditions*

Projection techniques include a variety of special methods. Generally we use $\langle \cdot, \cdot \rangle_2$ to measure the "size" of the residual function, R. The general strategy is to find an $\mathbf{a}$ which makes R small. There are several ways to proceed.

First, we have the *least-squares* approach which chooses $\mathbf{a}$ so as to minimize the "weighted sum of squared residuals":

$$\min_{\mathbf{a}} \langle R(x; \mathbf{a}), R(x; \mathbf{a}) \rangle_2.$$

This replaces an infinite-dimensional operator equation with a nonlinear minimization problem in R^n. The standard difficulties may arise; for example, there may be local minima which are not global minima. However, there is no reason for these problems to arise more often here than in any other context, such as maximum likelihood estimation, where extremal problems are solved numerically.[11]

While the least-squares method is a direct approach to making the error of the approximation small, most projection techniques find approximations by fixing n projections and making the projection of the residual function in each of those n directions zero. Formally, these methods find $\mathbf{a}$ such that $\langle R, p_i \rangle_2 = 0$ for some specified collection of functions, p_i. Different choices of the p_i defines different implementations of the projection method.

The least-squares and projection implementations of projection ideas are similar since one way to solve a least-squares problem is to solve the nonlinear set of projection conditions generated by its first-order conditions, $\langle R, \partial R/\partial a_i \rangle_2 = 0$. Seeing the least-squares method expressed as a system of projection equations gives us some indication why other methods may be better. The projection directions in the least-squares case, the gradients of the residual function, could be highly correlated, thereby increasing the numerical difficulty of determining $\mathbf{a}$. Furthermore, the projection directions depend on the guess for $\mathbf{a}$. In other methods, we have tight control

[11] While I have not made a complete study, I have had little success in using the least-squares method to solve simple optimal growth models. Convergence appears to be very slow.

over the projection directions, an important consideration in nonlinear problems. Also, in economic problems we may have a preference for approximations which have zero projections in certain directions, such as the average error in an Euler equation. Most techniques will naturally include that condition, whereas it may be absent in a least-squares approach.

One such technique is the *Galerkin* method. In the Galerkin method we use the first n elements of the basis for the projection directions.[12] Therefore, $\mathbf{a}$ is chosen to solve the equations:

$$P_i(\mathbf{a}) \equiv \langle R(x; \mathbf{a}), \varphi_i(x) \rangle = 0, \qquad i = 1, ..., n.$$

Note that here we have reduced the problem of solving a functional equation to solving a set of nonlinear equations. In some cases the Galerkin projection equations are the first-order conditions to some minimization problem, in which case the Galerkin method is also called the *Rayleigh–Ritz* method. This is not as likely to happen in economics problems because of their inherent nonlinearities.

There are obviously many ways to implement the projection idea. Some of the other common ones are *method of moments*, *subdomain*, and *collocation*. If $D \subset R^1$, then the method of moments chooses the first n polynomials for the projection directions; i.e., we find $\mathbf{a}$ which solves the system

$$P_i(\mathbf{a}) \equiv \langle R(x; \mathbf{a}), x^{i-1} \rangle_2 = 0, \qquad i = 1, ..., n.$$

If D is of higher dimension, we project R against multivariate monomials. In the subdomain method, the idea is to find an approximation which is good on average on a collection of subsets which cover the whole domain. More specifically, we choose $\mathbf{a}$ so that

$$P_i(\mathbf{a}) \equiv \langle R(x; \mathbf{a}), I_{D_i} \rangle = 0, \qquad i = 1, ..., n,$$

where $\{D_i\}_{i=1}^n$ is a sequence of intervals covering D, and I_{D_i} is the indicator function for D_i. A collocation method takes n points from D, $\{x_i\}_{i=1}^n$, and chooses $\mathbf{a}$ to solve

$$R(x_i; \mathbf{a}) = 0, \qquad i = 1, ..., n.$$

This is a special case of the projection approach since $R(x_i; \mathbf{a})$ equals the projection of $R(x; \mathbf{a})$ against the Dirac delta function as x_i, $\langle R(x; \mathbf{a}), \delta(x - x_i) \rangle_2$.

[12] Formally, this requires that B_1 and B_2 have the same basis and inner product, a weak condition.

Orthogonal collocation chooses the collocation points in a special way. The chosen x_i are the zeroes of the nth basis element, where the basis elements are orthogonal with respect to the inner product. The Chebyshev interpolation theorem suggests its power. Suppose we have found an $\mathbf{a}$ such that $R(z_i^n; \mathbf{a}) = 0$, $i = 1, ..., n$, where the z_i^n are the n zeroes of T_n. As long as $R(x; \mathbf{a})$ is smooth in x, the Chebyshev interpolation theorem says that these zero conditions force $R(x; \mathbf{a})$ to be close to zero for all x and that these are the best possible points to use if we are to force $R(x; \mathbf{a})$ to be close to zero. Even after absorbing these considerations, it is not certain that even orthogonal collocation is a reliable method. Its performance turns out to be surprisingly good.

Choosing the projection conditions is a critical decision since the major computational task is the computation of those projections. The collocation method is fastest in this regard since it only uses the value of R at n points. More generally, the projections will involve integration. In some cases one may be able to explicitly perform the integration. This is generally possible for linear problems and possible for special nonlinear problems. However, our experience is that this will generally be impossible for nonlinear economic problems. We instead need to use numerical quadrature techniques to compute the integrals associated with evaluating $\langle \cdot, \cdot \rangle_2$. A typical quadrature formula approximates $\int_a^b f(x)\, g(x)\, dx$ with a finite sum $\sum_{i=1}^n w_i f(x_i)$, where the x_i are the quadrature nodes and the w_i are the weights. Since these formulas also evaluate R at just a finite number of points, quadrature-based projection techniques are essentially weighted collocation methods. The advantage of quadrature formulas over collocation is that information at more points is used to compute the approximation, hopefully yielding a more accurate approximation of the projections.

3.3. *Finding the Solution*

Step 5, which determines $\mathbf{a}$ by solving the projection conditions computed in Step 4, uses either a minimization algorithm (in the least-squares approach) or a nonlinear algebraic equation solver to solve the system $P(\mathbf{a}) = 0$. I have found Newton's method to be adequate for all examples below. Newton's method is the iteration scheme

$$\mathbf{a}^{k+1} = \mathbf{a}^k - \left(\frac{\partial P}{\partial \mathbf{a}} (\mathbf{a}^k) \right)^{-1} P(\mathbf{a}^k)$$

which converges quadratically for good initial guesses. I used the

FORTRAN program HYBRD[13] from the MINPACK collection. Newton's method is fine for these problems if the number of unknowns is small. For large problems, one must switch to other methods, such as conjugate gradient or homotopy methods, to avoid the high cost of inverting Jacobians. The examples below are also well behaved; for highly nonlinear problems, one would switch to homotopy methods.

A good initial guess is important, particularly when one uses Newton methods. Sometimes, there are special cases with a known solution, which in turn will be a good guess for the problem we want to solve. In general, one uses problem-specific ways to generate good initial guesses. However, if one has no idea about what would be a good initial guess, then one could use the least-squares approach to generate one.

3.4. *Coordination among Steps* 1–5: *The Importance of Conditioning*

We now see what is needed for efficiency. We need basis functions which are easy to evaluate since they will be frequently evaluated. The integration in Step 4 must be accurate but fast. This can be helped by using quadrature formulas which work well with the basis. Finally, the nonlinear equation solver in Step 5 needs to be efficient and should be able to use all the information arising from Step 4 calculations. Step 5 will typically use gradient information about the integrals of Step 4. It is therefore important to do those gradient calculations quickly, doing them analytically when possible.

A particularly important interaction is that between the formulation of $\mathcal{N}$, the choice of a basis and inner product, and the technique for solving the projection conditions. Newton-style methods for solving the system $P(\mathbf{a}) = 0$ will invert its Jacobian, $P_{\mathbf{a}}(\mathbf{a})$. This inversion makes the method sensitive to conditioning problems. The *spectral condition number*, defined to be the ratio of the largest and smallest (in magnitude) eigenvalues of a matrix, is a commonly used index of being nearly singular and indicates how sensitive matrix inversion is to error. If a Jacobian is nearly singular near the solution, the accuracy of the inversion will be poor due to round-off error and convergence will be slow. In particular, a condition number of 10^k tells you that an error of ε in specifying an inversion can yield an error of up to $10^k\varepsilon$ in the computed inverse; in particular, you can lose up to k significant digits when you solve for $\mathbf{a}$ in a Newton step.

We now see why an orthogonal basis is going to be important. If a basis is nearly collinear, then the rows of $P_{\mathbf{a}}(\mathbf{a})$ will likely be nearly collinear, $P_{\mathbf{a}}(\mathbf{a})$ will likely have a large condition number, and large errors will likely arise in computing its inverse. Bases with just the first six odinary polyno-

[13] HYBRD does not recompute the Jacobian on each iteration, instead trying an updating procedure. This greatly speeds up the calculations relative to a pure Newton method.

mials can easily generate Jacobians with condition numbers on the order of 10^{10}, in which case one has possibly lost almost all the significant digits on, say, a 13-digit machine, that is, a machine where the machine round-off error ε is 10^{-13}. By choosing a basis which is orthogonal with respect to the inner product used in defining $P(\mathbf{a})$, one reduces the chances of poor conditioning in the Jacobian of P.

The form chosen for $\mathcal{N}$ will also have a dramatic influence on conditioning, accuracy, and speed. If $\mathcal{N}$ is linear then $P(\mathbf{a})=0$ is a linear equation in $\mathbf{a}$, and Newton's method converges in just one step. In our economic problems, there are typically several economically equivalent operators which represent equilibrium, typically differing by nonlinear transformations. The more linear we can make $\mathcal{N}$, the better Newton's method will perform. Below we will use this "linearization" idea to find a good form for our problems.

Most methods used in numerical analysis of economic models fall within the general description above. We will see this below when we compare how various methods attack growth problems. The key fact is that the methods differ in their choices of basis, fitting criterion, and quadrature techniques. With the general method laid out, we will now report on a particularly important application to show its usefulness.

4. Discrete-Time Deterministic Optimal Growth

In the rest of this paper we examine optimal growth problems in discrete time and show how projection techniques can be adapted to calculate solutions. This will be valuable for four reasons. First, discrete time is commonly used in dynamic economic analysis. Second, in the deterministic case we will be able to do a thorough job in checking our solutions since other reliable, but slow, algorithms are available to deliver tight bounds on the true solution. Third, the stochastic case is one which has been studied by many others with various numerical techniques. In fact, one point we make below is that most of these procedures are really projection methods. By recognizing the common projection approach underlying these procedures, we can better understand their differences, particularly in accuracy and speed. We conjecture that the comparative performances of these various implementations of projection ideas in the discrete-time stochastic optimal growth problem is indicative of their relative value in other future problems.

Fourth, the application of projection methods to this class of mathematical problems is unusual. Projection methods are used extensively for solving ordinary and partial differential equations in the physical sciences; therefore, their value in solving continuous-time economic models is

assured. However, the functional equations describing discrete-time optimal growth problems do not fit either class. Therefore, it is of substantial technical interest to show that projection methods do a good job in solving the kind of discrete-time functional equations which commonly arise in economic analysis.

4.1 *Problem Formulation*

We first examine a deterministic growth problem. We want to choose consumption, c_t, to maximize

$$\sum_{t=0}^{\infty} \beta^t u(c_t) \tag{4.1}$$

and where capital obeys the law of motion

$$k_{t+1} = f(k_t) - c_t. \tag{4.2}$$

As shown in Bizer and Judd [4], to calculate the optimal consumption policy, $h(k)$, it is enough to focus on the Euler equation,

$$0 = u'(h(k)) - \beta u'(h(f(k) - h(k)))\, f'(f(k) - h(k)) \equiv (\mathcal{N}(h))(k). \tag{4.3}$$

We now describe the details of a projection approach to the problem. The domain D of our approximation will be $[k_m, k_M]$. Since the special properties of Chebyshev polynomials apply to their restriction to $[-1, 1]$, we need to apply the linear transformation $2(k - k_m)/(k_M - k_m) - 1$ to k to permit us to form Chebyshev polynomials on D. k_m and k_M are chosen so that the solution will have k confined to $[k_m, k_M]$. In particular, $[k_m, k_M]$ must contain the steady state, a point which we can determine before calculations begin. Therefore, our approximation to h is parametrically given by

$$\hat{h}(k; \mathbf{a}) = \sum_{i=1}^{n} a_i \psi_i(k),$$

where $\psi_i(k) \equiv T_{i-1}(2((k - k_m)/(k_M - k_m)) - 1)$ and n is the number of terms used.

In this problem, $\mathcal{N}$ is a simple operator using only arithmetic operations and composition. Therefore, we can take $\hat{\mathcal{N}} = \mathcal{N}$. Since h is continuous, we define $\mathcal{N}$ to have domain and range in $C^0[k_m, k_M]$; the only limitation on the domain is that u' and f' are defined in (4.3) at all $k \in D$. Hence, $B_1 = B_2 = C^0[k_m, k_M]$, the continuity of $\mathcal{N}$ in the L^∞ norm following

from the u, f, and $\hat{h}$ being C^1 in all their arguments. Given the Euler equation, (4.3), the residual function becomes

$$R(k;\mathbf{a}) = u'(h(k;\mathbf{a})) - \beta u'(h(f(k)-h(k;\mathbf{a});\mathbf{a}))\, f'(f(k)-h(k;\mathbf{a})) = \mathcal{N}(\hat{h}).$$

To compute $\mathbf{a}$, we can do one of several things. First, we consider orthogonal collocation. We choose n values of k, denoted by k_i, $i=1, ..., n$. We then choose $\mathbf{a}$ so that $R(k_i;\mathbf{a})=0$ for each i. Orthogonal collocation chooses the k_i to be the n zeroes of ψ_{n+1}, which are themselves linear transforms of the Chebyshev zeroes, z_i^n, defined above.

We will also implement the Galerkin method. Since we use Chebyshev polynomials as a basis, we will use the inner product

$$\langle h(k),\, g(k)\rangle \equiv \int_{k_m}^{k_M} h(k)\, g(k)\, w(k)\, dk,$$

where

$$w(k) \equiv \left(1 - \left(2\frac{k-k_m}{k_M-k_m} - 1\right)^2\right)^{-1/2}.$$

With this choice of inner product, the basis is orthogonal. The Galerkin method computes the n projections

$$P_i(\mathbf{a}) \equiv \int_{k_m}^{k_M} R(k;\mathbf{a})\, \psi_i(k)\, w(k)\, dk, \qquad i=1, ..., n,$$

and chooses $\mathbf{a}$ so that $P(\mathbf{a})=0$. Here the difficulty is that each $P_i(\mathbf{a})$ is an integral which needs to be computed numerically. The form of $w(k)$ implies the use of Gauss–Chebyshev quadrature. That is, we approximate $P_i(\mathbf{a})=0$ conditions with

$$\sum_{j=1}^{m} R(k_j;\mathbf{a})\, \psi_i(k_j) = 0$$

for some $m > n$, with the k_j being the m zeroes of ψ_{m+1}.

Results

We solved this model for the CRRA[14] utility functions, $u(c) = c^{1+\gamma}/(1+\gamma)$, and the Cobb–Douglas production function, $f(k) = Ak^{\alpha}$. The

[14] Actually, we deviated a little bit from CRRA. CRRA functions are not defined for negative c, but the Newton solver may examine values of $\mathbf{a}$ which imply negative c for some k. To deal with this, we pick a small c_ε, and for $c < c_\varepsilon$ we replace the CRRA utility function with a quadratic utility function which agrees with the CRRA utility function and its first two derivatives at c_ε. The resulting utility function is C^2, concave, and defined for all c. As long as c_ε is less than any equilibrium realization of c, this change has no effect on the solution.

constant A was chosen so that the steady state is $k = 1$. We always chose $\alpha = 0.25$. For several values of γ we solved the collocation system

$$0 = R(k_j; \mathbf{a}), \qquad i = 1, ..., n,$$

to arrive at our approximate solution $\hat{h}(k) = \sum_{i=1}^{n} a_i \psi_i(k)$. Galerkin procedures resulted in only trivial differences.

The Newton algorithm yielded solutions very quickly (a few seconds at most). For the initial guess, I used the consumption function which is linear in capital, goes through the steady state, and consumes zero at zero capital stock. This is a natural initial guess since it implies stable growth converging to the steady state. The Newton solver always converged with this initial guess. However, if one assumed a constant consumption function going through the deterministic steady state, the Newton solver often diverged. This is not too surprising since the unstable manifold is also a solution to the Euler equations.

4.2 *Accuracy Check*

When we have calculated our estimate of **a**, we would like to check if this procedure yields reliable approximations. The first check to make is to see if the coefficients look like they should according to the Chebyshev approximation theorem. Table I displays the result for $\gamma = -0.9$, $n = 2, 5, 9, 15$, $k_m = 0.333$, and $k_M = 1.667$. The coefficients behave as we expect. First, the computed a_k do decline rapidly in k. Second, they are insensitive to the choice of n. While these facts do not prove that the approximation is good, we would be uncomfortable if the high-order coefficients were not small or if the coefficient estimates were not stable, as we increase n.

Our second check, not usually available, is to compare the answer with another method known to be extremely accurate. If the capital stock could be only a finite number of values (i.e., "lumpy") then dynamic programming can solve for the equilibrium exactly. To this end, I computed the solution to our problem, assuming that k must be some multiple of 0.000001, where 1.0 is the steady state. I solved for the policy function over the range [0.5, 1.3], a total of 800,000 capital stocks. The main difficulty in this calculation is the need for space to store the policy function. Because this is a one-dimensional deterministic problem, time is no problem for this calculation, taking only a few minutes[15] on a VAX 8650, which is roughly three times as fast as a Compaq 386/20.

[15] I did *not* do value function iteration, a method which would have taken much longer. Instead I used the fact that we know the policy function at the steady state and that both the consumption and savings functions are monotonic to directly compute the policy function in one pass, beginning with the steady state.

TABLE I

Chebyshev Coefficients

	$\hat{a}_k$			
k	$n=2$	$n=5$	$n=9$	$n=15$
1	0.0589755899[a]	0.0600095844	0.0600137797	0.0600137922
2	0.0281934398	0.0284278730	0.0284329464	0.0284329804
3		−0.0114191783	−0.0113529374	−0.0113529464
4		0.0007725731	0.0006990930	0.0006988353
5		−0.0001616767	−0.0001633928	−0.0001634209
6			0.0000427201	0.0000430853
7			−0.0000123570	−0.0000122160
8			0.0000042498	0.0000036367
9			−0.0000011464	−0.0000011212
10				0.0000003557
11				−0.0000001147
12				0.0000000370
13				−0.0000000129
14				0.0000000052
15				−0.0000000015

[a] Each entry is the coefficient of the kth Chebyshev polynomial (over the interval [0.333, 1.667]) in the n-term approximation of the consumption policy function in (4.3) for the case discussed in Section 4.2.

We obtain some idea of the accuracy of the projection technique by comparing it with the presumably very accurate solution from the discretized problem. The results are shown in Table II. The first column (labelled γ) indicates γ, the second column (CAP) gives a range of values for k, the third column (PROD) gives output, Ak^{α}, and the fourth column gives the consumption choice as a function of capital computed by the discrete state space method. The remaining columns give the difference (the notation $a(-n)$ means $a10^{-n}$) between the discrete solution and the projection solution for $n = 10, 7, 5, 3$, where n is the number of Chebyshev polynomials included in the approximation. Note that the approximation of the aggregate consumption function for $n = 10$ disagrees with the discrete state space result by no more than one part in 10,000, an acceptable error for most purposes. These numerical experiments indicate that the projection method works well for the deterministic discrete-time optimal growth model, demonstrating its usefulness even for the nonstandard functional equations which arise often in such models.

Our analysis has focussed only on the Euler equation, ignoring the transversality condition at infinity. There are a continuum of solutions, but only one satisfies transversality. Therefore, any accuracy check should also check transversality. In this model, transversality is equivalent to stability;

TABLE II

Policy Function Errors

γ	CAP	PROD	CONS	$n=10$	$n=7$	$n=4$	$n=2$
−0.500	0.50	0.1253211	0.0911211	1(−7)[a]	1(−7)	−2(−7)	−2(−5)
	0.70	0.14001954	0.1185654	−3(−7)	−2(−7)	−1(−6)	1(−4)
	0.80	0.1465765	0.1318765	0	0	−3(−6)	1(−4)
	0.90	0.1524457	0.1449757	1(−7)	1(−7)	−3(−6)	1(−4)
	1.00	0.1578947	0.1578947	0	0	−1(−6)	9(−5)
	1.10	0.1629916	0.1706616	−1(−7)	−1(−7)	8(−7)	6(−5)
	1.30	0.1723252	0.195815	2(−7)	2(−7)	2(−6)	−5(−5)
−3.000	0.50	0.1253211	0.1147611	3(−7)	3(−7)	1(−6)	−1(−4)
	0.70	0.1401954	0.1335954	−3(−7)	−3(−7)	−1(−6)	1(−4)
	0.80	0.1465765	0.1421165	−2(−7)	−1(−7)	−5(−6)	2(−4)
	0.90	0.1524457	0.1501957	4(−7)	4(−6)	−5(−6)	2(−5)
	1.00	0.1578947	0.1578947	0	−1(−6)	−3(−6)	2(−5)
	1.10	0.1629916	0.1652816	−2(−7)	−2(−7)	2(−6)	9(−5)
	1.30	0.1723252	0.1792852	2(−7)	2(−7)	4(−6)	−1(−4)
−10.000	0.50	0.1253211	0.1214511	−2(−7)	−2(−7)	−2(−6)	−2(−4)
	0.70	0.1401954	0.1377454	−5(−7)	−5(−7)	2(−7)	1(−4)
	0.80	0.1465765	0.1449165	0	1(−7)	−3(−6)	2(−4)
	0.90	0.1524457	0.1516057	3(−7)	2(−7)	−4(−6)	2(−4)
	1.00	0.1578947	0.1578947	0	−1(−7)	−2(−6)	1(−4)
	1.10	0.1629916	0.1638516	−2(−7)	−2(−7)	2(−6)	7(−5)
	1.30	0.1723252	0.1749552	−1(−7)	−1(−7)	2(−6)	−1(−4)

[a] Each entry equals the nth degree approximation to the consumption policy function minus the 800,000 point discretization of (4.3). The notation $x(-n)$ denotes $x10^{-n}$.

hence we need only check that the law of motion implied by $\hat{h}$ is stable, as indeed it always turned out to be. Our choice of initial guess was critical since other guesses instead led to solutions with overaccumulation. In this deterministic problem one can avoid all of these problems by replacing one of the projection conditions with a condition forcing $\hat{h}$ to go through the deterministic steady state, a point which we can compute directly. In general, one needs some kind of "boundary" condition to eliminate extraneous solutions when solving operator equations. Exactly why that is not necessary for this problem is somewhat unclear, but almost surely linked to the fact that the steady state is a saddlepoint in the phase diagram.

5. Discrete-Time Stochastic Optimal Growth

We next turn to a stochastic optimal growth model. This example will show us how to handle multidimensional problems and the conditional

expectations which arise in stochastic dynamic problems. It will also force us to develop an alternative method to accuracy checking. In fact, we will develop an accuracy measure which measures both the size of the residual function and the extent to which agents are irrational if they follow the computed approximation, thereby relating numerical error to bounded rationality ideas.

We consider the problem

$$\max E\left\{\sum_{t=0}^{\infty} \beta^t u(\tilde{c}_t)\right\} \tag{5.1}$$

$$k_{t+1} = \theta_t f(k_t) - c_t, \tag{5.2}$$

where θ_t is a stationary AR(1) multiplicative productivity parameter. We assume that the productivity shock follows $\ln \theta_{t+1} = \rho \ln \theta_t + \varepsilon_{t+1}$ and that the $\varepsilon_t \sim N(0, \sigma^2)$ are independent. In this problem, both the beginning-of-period capital stock and the current value of θ are needed for a sufficient description of the state. Hence, the Euler equation[16] is

$$0 = u'(h(k,\theta)) - \beta E\{u'(h(\theta f(k) - h(k,\theta), \tilde{\theta}))\,\tilde{\theta} f'(\theta f(k) - h(k,\theta)) \mid \theta\}. \tag{5.3}$$

We could use (5.3) as our residual function. However, it could be highly nonlinear, particularly for highly concave utility functions. Our algorithm might do better if we make it more like a linear problem. To that end, we "linearize" (5.3) by rewriting it as

$$\begin{aligned} 0 = h(k,\theta) - (u')^{-1}\,(\beta E\{u'(h(\theta f(k) \\ - h(k,\theta), \tilde{\theta}))\,\tilde{\theta} f'(\theta f(k) - h(k,\theta)) \mid \theta\}) \equiv \mathcal{N}(h). \end{aligned} \tag{5.4}$$

Note that the RHS of (5.4) has two terms, one linear in $h(k,\theta)$, and the other is similar to a CRTS function of the next period's potential consumption values. Note also that the $(u')^{-1}$ operation in (5.4) will unwrap some of the nonlinearity arising from the u' operation inside the expectation, hopefully leaving us with a more linear problem.

[16] While we have concentrated on optimal growth problems, nowhere is the optimality of equilibria used. Euler equation methods have been extensively used in the numerical literature. Gustafson [14], Miranda and Helmburger [24], and Wright and Williams [33–35] use Euler equation methods to model various governmental interventions and distortions. Bizer and Judd [4] show that the equilibrium of a taxed economy can also be described by an Euler equation similar to (5.3). Klenow [19] applies projection methods to models with elastic labor supply and externalities, and Judd [16, 18] solves problems with incomplete asset markets and asymmetric information.

5.1. *Galerkin and Orthogonal Collocation Methods*

The procedure is similar to the deterministic case, but there are some extra twists due to the stochastic shocks. First, our approximation of the policy function is now given by the double sum

$$\hat{h}(k, \theta; \mathbf{a}) = \sum_{i=1}^{n_k} \sum_{j=1}^{n_\theta} a_{ij} \psi_{ij}(k, \theta),$$

where $\psi_{ij}(k, \theta) \equiv T_{i-1}(2((k - k_m)/(k_M - k_m)) - 1)\, T_{j-1}(2((\theta - \theta_m)/(\theta_M - \theta_m)) - 1)$. This tensor-product approach to two-dimensional approximation is reasonable here because the dimension is low and the tensor-product approach allows for a straightforward extension of the one-dimensional techniques to higher dimensions. Also note that we have added four parameters to our problem: k_M, k_m, θ_M, and θ_m. k_m and k_M are chosen so that k is usually confined to $[k_m, k_M]$, and similarly for θ_M and θ_m. Since θ is exogenous, this can be easily accomplished. If we truncate $\tilde{\varepsilon}$ so that $\tilde{\varepsilon} \in [\varepsilon_m, \varepsilon_M]$, we can take $\ln \theta_m = \varepsilon_m (1-\rho)^{-1}$ and $\ln \theta_M = \varepsilon_M (1-\rho)^{-1}$. Appropriate choices for k_m and k_M are more problematic and really cannot be made until after some experimentation finds a capture region. It is hoped that our solution is a good approximation on the rectangle $[k_m, k_M] \times [\theta_m, \theta_M]$.

Since tomorrow's log productivity level, $\ln \tilde{\theta}$, conditional on today's log productivity level, $\ln \theta$, is distributed as $\rho \ln \theta + \sigma Z$ for $Z \sim N(0, \sigma^2)$, (5.4) becomes

$$0 = \hat{h}(k, \theta; \mathbf{a}) - (u')^{-1} \left(\beta \int_{-\infty}^{\infty} I(k, \theta, \mathbf{a}, z) \frac{e^{-z^2/2}}{\sqrt{2}} dz \right), \tag{5.5}$$

where

$$I(k, \theta, \mathbf{a}, z) \equiv u'(\hat{h}(f(k)\theta - \hat{h}(k, \theta; \mathbf{a}), e^{\sigma z} \theta^\rho; \mathbf{a})) \times e^{\sigma z} \theta^\rho f'(\theta f(k) - \hat{h}(k, \theta; \mathbf{a}))\, \pi^{-1/2}.$$

In this problem $\mathcal{N}$ involves an integral which cannot generally be evaluated explicitly. In forming the residual function, we need to use an approximation $\hat{\mathcal{N}}$ of $\mathcal{N}$. The approximation will be to approximate the integral in (5.5), with a finite sum,

$$\int_{-\infty}^{\infty} I(k, \theta, \mathbf{a}, z) \frac{e^{-z^2/2}}{\sqrt{2}} dz \doteq \sum_{j=1}^{m_z} I(k, \theta, \mathbf{a}, \sqrt{2}\, z_j)\, w_j,$$

where w_j, z_j are Gauss-Hermite quadrature weights and points.[17] This numerical quadrature procedure approximates $\mathcal{N}$ with $\hat{\mathcal{N}}$. Since economic problems often involve such conditional expectations, numerical quadrature will often be necessary to evaluate these integrals, forcing us to replace $\mathcal{N}$ with $\hat{\mathcal{N}}$.

With $\hat{\mathcal{N}}$ in hand, we now define the residual function to be

$$R(k, \theta; \mathbf{a}) = \hat{h}(k, \theta; \mathbf{a}) - (u')^{-1}\left(\beta \sum_{j=1}^{m} I(k, \theta, \mathbf{a}, z_j)\, w_j\right) \equiv \hat{\mathcal{N}}(\hat{h}).$$

With this residual function, we can proceed as we did in the deterministic problem. The collocation method starts with choosing n_k capital stocks, $\{k_i\}_{i=1}^{n_k}$, and n_θ productivity levels, $\{\theta_i\}_{j=1}^{n_\theta}$, and then finding $\mathbf{a}$ so that $R(k_i, \theta_j; \mathbf{a}) = 0$ for all $i = 1, ..., n_k$ and $j = 1, ..., n_\theta$. The Galerkin approach forms the $n_k n_\theta$ projections

$$P_{ij}(\mathbf{a}) \equiv \int_{k_m}^{k_M} \int_{\theta_m}^{\theta_M} R(k, \theta; \mathbf{a})\, \psi_{ij}(k, \theta)\, d\theta\, dk$$

and chooses $\mathbf{a}$ so that $P_{ij}(\mathbf{a}) = 0$ for all i and j. Again, $P_{ij}(\mathbf{a})$ needs to be computed numerically. Since the ψ_{ij} are Chebyshev polynomials, we use Gauss–Chebyshev quadrature points, which are the zeroes of Chebyshev polynomials, over $k - \theta$ space. If we use m_k values of k and m_θ values of θ to compute the projections, we solve the system

$$\hat{P}_{ij}(\mathbf{a}) \equiv \sum_{l_k=1}^{m_k} \sum_{l_\theta=1}^{m_\theta} R(k_i, \theta_j; \mathbf{a})\, \psi_{ij}(k_{l_k}, \theta_{l_\theta}) = 0, \tag{5.6}$$

where

$$\begin{aligned} k_{l_k} &= k_m + \tfrac{1}{2}(k_M - k_m)(z_{l_k}^{m_k} + 1), && l_k = 1, ..., m_k, \\ \theta_{l_\theta} &= \theta_m + \tfrac{1}{2}(\theta_M - \theta_m)(z_{l_\theta}^{m_\theta} + 1), && l_\theta = 1, ..., m_\theta, \\ z_l^n &\equiv \cos\left(\frac{(2i-1)\pi}{2n}\right), && l = 1, ..., n. \end{aligned}$$

We have the nonlinear equation in the unknown coefficients, $\mathbf{a}$, which will determine our approximation, $\hat{h}(k, \theta)$.

While we focus on a problem with no growth, the method can be used to solve models which can be expressed in stationary "detrended" variables. For example, suppose output is $y = \theta^{1-\alpha} k^\alpha$, where $\ln \theta_{t+1} = \rho \ln \theta_t + \varepsilon_{t+1}$.

[17] The values of the quadrature weights and points are available in published tables, as in Judd [16]. Using general-purpose quadrature routines in IMSL or NAG would result in much slower performance.

If $\rho > 1$, we have trend productivity growth and both θ and k will grow. If $u'(c) = c^{\gamma}$, then the consumption ratio is stationary and consumption can be expressed as $c = \theta h(k/\theta)$, where h satisfies

$$\theta^{\gamma} h(k/\theta)^{\gamma} = \beta E\{\tilde{\theta}^{\gamma} \mid \theta\}\, E\{(h(k^{+}/\theta^{+}))^{\gamma}\,(1 + (k^{+}/\theta^{+})^{\alpha-1})\}$$

and $k^{+} \equiv k + \theta^{1-\alpha}k^{\alpha} - \theta h(k/\theta)$. This equation is of the same form as (5.3) above. Note that growth rate ρ can be folded into the discount factor β, implying that the problem with growth rate ρ and discount rate β is equivalent to the problem with growth rate zero and discount rate $\beta E\{(\tilde{\theta}/\theta)^{\gamma} \mid \theta\} = \beta e^{\rho - 1} E\{e^{\tilde{\varepsilon}}\}$. Through such detrending procedures, one can compute policy functions for economies with growth (see Klenow [19] for an example).

5.2. *Accuracy Checks—A Bounded Rationality Measure*

Once we have a candidate solution, we want to check its quality.[18] A direct procedure is to check how much, if at all, $\hat{\mathcal{N}}(\hat{h})$ differs from the zero function. First, we should understand what a deviation from zero means in economic terms. Consider (5.4). It is a difference between consumption at a capital stock k and productivity level θ and what that consumption would be if an optimizing agent knew that tomorrow he will use the consumption rule $\hat{h}$ and that personal and aggregate wealth will both be $\theta f(k) - h(k, \theta)$. Therefore, our residual equation applied to the approximate solution is the one-period optimization error in consumption terms. The function $E(k, \theta) \equiv R(k, \theta; \mathbf{a})/\hat{h}(k, \theta; \mathbf{a})$ yields a dimension-free quantity expressing that optimization error as a fraction of current consumption.

This approach to accuracy checking expresses the resulting errors in economic terms, essentially in terms of how irrational agents would be in using the approximate rule. If one found that this relative optimization error were about 0.1, then we would know that the approximation implies that agents make 10% errors in their period-to-period consumption decisions, a magnitude which few economists would find acceptable. However,

[18] Some might wonder just how accurate we need an approximation to be. In fact, Danthine *et al.* [9] have argued that the linear approximation computed by Magill [21] and McGratton [23], among others, is adequate for macroeconomic purposes. However, their tests concerned only a few economic variables such as consumption and output. In light of the results in Magil [21] (and, more generally, in Bensoussan [3] and Judd [17]), this is not surprising. The adequacy of the linear approximation is much less likely once one turns to other economic variables, such as risk premia, term structure of interest rates, and their correlations since these variables involve higher-order properties of tastes and technology, as documented in Judd [16]. Therefore, we attempt to find approximations which are as accurate as possible, given the limitations on computer time and space.

if this index were 0.000001, then the approximation implies that agents made only a \$ 1.00 mistake for every \$ 1,000,000 they spent. Few economists would seriously argue that real-world agents do better than this. While such an approximation, $\hat{h}$, may not be the mathematically exact equilibrium decision rule, it is hard to argue that it is unacceptable as a description of human behavior. In fact, many would argue that it is as compelling a description of behavior as the mathematical zero of the operator $\mathcal{N}$.

The philosophy behind this accuracy check is that we should find an ε such that our approximation is an ε-equilibrium. The advantage of this approach is that our approximation to an exact equilibrium becomes reinterpreted as an approximate equilibrium. The disadvantage of focusing on ε-equilibrium is the likely existence of an open set of such equilibria. However, as long as the problem is well conditioned, something which can be numerically checked, that set is likely to be small, and even negligible, for many purposes.

5.3. *Results for Galerkin and Orthogonal Collocation Methods*

With our approximation method specified and accuracy checks determined, we can now see just how fast we can compute our approximations and how accurate they are. To do so we make taste and technology specifications which bracket a wide range of empirically plausible values. Table III summarizes typical cases. We again assumed $u(c) = c^{\gamma+1}/(\gamma+1)$, and $f(k) = Ak^{\alpha}$, where A is chosen so that the deterministic steady state is $k = 1$. Throughout Table III, $\alpha = 0.25$, $k_m = 0.3$, and $k_M = 2.0$. I also chose $m_z = 8$, the eight-point, fifteenth-order accurate, Gauss–Hermite quadrature rule, to compute the conditional expectation. Table III lets σ and ρ vary; for each choice, θ_M is set equal to the long-run value of θ which would occur if $\varepsilon_t = 3\sigma$ for all t, and $\theta_m = 1/\theta_M$. It is extremely unlikely for θ to spend much, if any, time outside of $[\theta_m, \theta_M]$.

For the initial guess, I again used the consumption function, which is linear in capital, goes through the deterministic steady state, and consumes zero at capital stock. The Newton solver always converged with this initial guess.

Table III is composed of six blocks of error entries, each block corresponding to a particular choice of the four-tuple $(n_k, n_\theta, m_k, m_\theta)$. For example, the block headed by the four-tuple (2, 2, 2, 2) is the case where the approximate policy function is $a_1 + a_2 k + a_3 \theta + a_4 k\theta$; (2, 2, 2, 2) also indicates that we choose **a** so that the Euler equation fits exactly at the four zeroes of $\psi_{3,3}$, indicating an orthogonal collocation procedure; (10, 6, m_k, m_θ) corresponds to allowing k terms up to k^9, θ terms up to θ^5, and all possible pairwise products of those k and θ terms.

TABLE III

Euler Equation Errors

			$\log_{10}$ of					
γ	ρ	σ	$\|E\|_\infty$	$\|E\|_1$	$\|E_I\|_\infty$	$\|E\|_\infty$	$\|E\|_1$	$\|E_I\|_\infty$
		$(n_k, n_\theta, m_k, m_\theta)$:	(2, 2, 2, 2)			(4, 3, 4, 3)		
−15.00	0.80	0.01	−2.13	−2.80	−2.58	−3.00	−3.83	−3.70
−15.00	0.80	0.04	−1.89	−2.54	−2.28	−2.44	−2.87	−2.59
−15.00	0.30	0.01	−2.20	−2.82	−2.63	−3.05	−3.86	−3.82
−15.00	0.30	0.04	−2.13	−2.80	−2.58	−2.97	−3.83	−3.70
−0.10	0.80	0.01	−0.01	−1.22	−1.34	−1.68	−2.65	−2.70
−0.10	0.80	0.04	0.01	−1.19	−1.20	−1.48	−2.22	−1.89
−0.10	0.30	0.01	0.04	−1.22	−1.36	−1.67	−2.65	−2.74
−0.10	0.30	0.04	0.18	−1.22	−1.35	−1.63	−2.65	−2.74
			(7, 5, 7, 5)			(7, 5, 20, 12)		
−15.00	0.80	0.01	−4.28	−5.19	−5.00	−4.43	−5.18	−4.91
−15.00	0.80	0.04	−3.36	−4.00	−3.70	−3.30	−3.95	−3.67
−15.00	0.30	0.01	−4.37	−5.23	−5.10	−4.55	−5.22	−4.99
−15.00	0.30	0.04	−4.24	−5.19	−4.96	−4.38	−5.18	−4.87
−0.10	0.80	0.01	−3.40	−4.37	−4.35	−3.47	−4.39	−4.32
−0.10	0.80	0.04	−2.50	−3.22	−2.93	−2.60	−3.17	−2.91
−0.10	0.30	0.01	−3.44	−4.36	−4.36	−3.51	−4.39	−4.33
−0.10	0.30	0.04	−3.43	−4.37	−4.36	−3.49	−4.39	−4.33
			(10, 6, 10, 6)			(10, 6, 25, 15)		
−15.00	0.80	0.01	−5.48	−6.43	−6.19	−5.61	−6.42	−6.11
−15.00	0.80	0.04	−3.81	−4.38	−4.11	−3.88	−4.37	−4.11
−15.00	0.30	0.01	−5.66	−6.49	−6.31	−5.80	−6.49	−6.24
−15.00	0.30	0.04	−5.45	−6.43	−6.15	−5.57	−6.42	−6.08
−0.10	0.80	0.01	−5.09	−6.12	−5.94	−5.17	−6.15	−5.94
−0.10	0.80	0.04	−2.99	−3.68	−3.37	−3.09	−3.64	−3.38
−0.10	0.30	0.01	−5.22	−6.12	−6.04	−5.28	−6.14	−6.02
−0.10	0.30	0.04	−5.17	−6.12	−6.01	−5.23	−6.14	−5.99

To test for the quality of the candidate solution, we evaluate (5.5)[19] at a large number of (k, θ) combinations which themselves were not used in finding a solution. We defined the relative error at (k, θ) to be $E(k, \theta) \equiv R(k, \theta; a)/\hat{h}(k, \theta; \mathbf{a})$. The entries are the base 10 logarithm of various norms of $E(\cdot)$. Columns 4, 5, 7, and 8 report $\log_{10} \|E\|_\infty$ and

[19] The accuracy tests used the same eight-point Gauss–Hermite quadrature rule to evaluate the conditional expectations as used to compute **a**. A more severe accuracy test would be to use a higher-order quadrature rule, but on the occasions when that was done, there were no differences.

$\log_{10} \|E\|_1$. All of these norms were calculated by using 8000 grid points in $[k_m, k_M] \times [\theta_m, \theta_M]$, 100 in $[k_m, k_M]$, and 80 in $[\theta_m, \theta_M]$. Base 10 logs of these norms are natural measures for our exercise; $\log_{10} \|E\|_\infty$ is the maximum error we found and $\log_{10} \|E\|_1$ represents the average error. For example, an entry of -3 under the $\log_{10} \|E\|_\infty$ column says that a person with \$ 1000 of consumption makes *at most* a one-dollar error in current consumption in each period relative to the next period's consumption. Since solution paths concentrate near the center of D, we are particularly concerned about accuracy there. We define E_I to be E restricted to the inner rectangle $[k'_m, k'_M] \times [\theta'_m, \theta'_M]$, where $y' \equiv \frac{1}{2}(y+1)$, $y = k_m, k_M, \theta_m, \theta_M$. Columns 6 and 9 reports $\log_{10} \|E_I\|_\infty$.

There are several points to note. First, note that the errors are rather small. Even for the (2, 2, 2, 2) case, the errors are roughly one dollar per hundred, as long as the utility function is as concave as $\log c$. Second, as we allow the approximation to use more terms, the errors fall until, in the (10, 6, 10, 6) case, we often find optimization errors of less than one dollar per million. Third, the various norms of the residual function have very similar values, indicating that the errors are uniformly small. In particular, the similarity in values for the norms of E and E_I indicates that the solution is almost as good at the edges of the state space as in the middle.

Fourth, these methods are fast. The solutions in the (2, 2, 2, 2) case were solved in 0.2 to 0.4 s and, in the (4, 3, 4, 3) case, in 1.1 to 2 s. The slow parameterization throughout the table was $\gamma = -15.0$, $\rho = 0.8$, and $\sigma = 0.04$, which took 3 s for the (4, 3, 4, 3) case. The speed advantage of orthogonal collocation is demonstrated by the fact that the (7, 5, 7, 5) cases generally took 8 to 18 s, whereas the (7, 5, 20, 12) Galerkin cases took three times as long, which is expected since the projections were integrals using 240 points instead of 35. An intriguing exception was the slow parameterization which took nearly 2 min for collocation but only $1\frac{1}{2}$ min for the Galerkin. Apparently the extra information used by the Galerkin procedure helped the nonlinear equation solver to avoid bad directions; (10, 6, 10, 6) cases generally took 27 to 72 s, with the bad parameterization taking 100 s. The corresponding (10, 6, 20, 15) cases took roughly four times as long.

Fifth, note that the orthogonal collocation method does remarkably well, given the small amount of computation. This is indicated by the small optimization errors and the fact that the Galerkin procedures which use many more points achieved only slightly greater accuracy. In general, the collocation schemes yielded the most accuracy per unit of time.

Sixth, the dependence of speed and accuracy on parameters was clear. The algorithm was faster for less concave utility functions, less persistent productivity shocks, and smaller productivity shocks. The slowest parameterization, $\gamma = -15.0$, $\rho = 0.8$, and $\sigma = 0.04$, presents difficulties for

the algorithm since the policy function involves little saving on average and the range of θ and k is large. Since there is little central tendency in the true solution, the algorithm will have a tendency to examine candidates which are qualitatively very bad, such as policy rules which violate transversality and give little help towards finding a good approximation.

Accuracy was greatest for highly concave utility and for small productivity shocks with low persistence. The dependence on concavity is intuitive since highly concave utility implies that consumption is nearly equal to permanent income, whereas nearly linear utility implies a highly nonlinear consumption rule with very little consumption except near the deterministic steady state capital stock. With larger and more persistent shocks, the range of θ, $[\theta_m, \theta_M]$, grows, forcing the approximation to cover a larger area and making accuracy more difficult for a fixed number of degrees of freedom.

While the number of cases reported in Table III are small note that they cover a wider range of parameter values. I have also performed these same calculations for $\gamma = -10.0$, -7.0, -5.0, -4.0, -2.0, -0.9, and -0.5, $\sigma = 0.02$ and 0.03, and $\rho = 0.6$, 0.5, and 0.4, and several other choices of $(n_k, n_\theta, m_k, m_\theta)$. Since the patterns described above held over the whole collection of calculations, it suffices to report the cases in Table III.

Another way to check for accuracy is to see how the computed solution changes when we use higher-order and different quadrature schemes. Again, I found trivial sensitivity to these changes. For example, using $m_z = 4$ instead of 8 resulted in very few differences (Table III was unchanged) and cut the running time by almost half.

As with the deterministic case, we have ignored transversality considerations in our solution method. Again we should check that the solutions are stable, as they always were. In the stochastic case, we have no clear alternative which will assure convergence to a stable solution since we do not know *a priori* any point on the policy function. While solving operator equations without imposing boundary conditions is not proper procedure, it appears to be possible for these problems.

5.4. *Alternative Bases and Fitting Criterion*

Tables IV and V discuss the results when we attempt alternative implementations of the projection ideas. Each choice made above was motivated by some optimality or conditioning consideration. We will now see how important they were. Table IV reexamines some of the cases in Table III, using theoretically inferior methods. The pair of columns under G gives the $\log_{10} \|E\|_\infty$ error measure and running times when I used the procedure above. The pair of columns under P refers to $\log_{10} \|E\|_\infty$ and running times when I used ordinary polynomials instead of Chebyshev

TABLE IV

Alternative Implementations

γ	ρ	σ	G[a]		P[b]		U[c]		UP[d]	
			$n_k = 7,\ n_\theta = 5,\ m_k = 7,\ m_\theta = 5$							
−15.0	0.8	0.04	−3.18	1:15	−2.13	:40	−3.06	1:05	−2.19	:44
	0.3	0.01	−4.35	:11	−4.35	:52	−4.07	:08	−4.07	1:47
−0.9	0.8	0.04	−3.43	:05	−3.43	:19	−3.42	:08	−3.42	:39
	0.3	0.01	−4.03	:07	−4.03	:30	−3.76	:07	−3.76	1:10
−0.1	0.8	0.04	−2.50	:07	−2.50	:41	−2.52	:06	−2.52	:42
	0.3	0.01	−3.42	:08	−3.42	1:30	−3.18	:07	−3.18	:24
			$n_k = 10,\ n_\theta = 6,\ m_k = 25,\ m_\theta = 15$							
−15.0	0.08	0.04	−3.87	4:20	−3.90	24:44	−3.90	3:41	−3.36	42:15
	0.3	0.01	−5.68	2:19	−5.14	11:31	−5.49	2:14	−5.30	8:06
−0.9	0.8	0.04	−4.00	1:31	−4.00	5:17	−4.01	1:31	−4.01	5:02
	0.3	0.01	−5.40	1:23	−4.63	7:13	−5.25	1:20	−5.13	6:01
−0.1	0.8	0.04	−3.09	1:31	−3.09	9:16	−3.10	1:32	−3.07	12:01
	0.3	0.01	−5.27	1:32	−4.02	7:25	−5.09	1:27	−3.27	8:32

[a] Chebyshev polynomial basis, Chebyshev zeroes used in evaluating fit.
[b] Ordinary polynomial basis, Chebyshev zeroes used in evaluating fit.
[c] Chebyshev polynomial basis, uniform grid points.
[d] Ordinary polynomial basis, uniform grid points.

polynomials, but still fit the residual conditions at the Chebyshev zeroes. The results under G and P should be identical if we had infinite precision arithmetic and Newton's method always converged. In several cases, the results were the same, but the P times were far slower. P was faster in one case, but yielded an approximation with substantially larger error. The problem was that the Newton solver could not make any progress so it stopped early. The slower time and premature stopping are both reflections of the conditioning problems associated with ordinary polynomials.

The columns under U give the accuracy and running time when I used Chebyshev polynomials but used a uniform grid to compute the projections. Here G should do better since interpolation at a uniform grid is usually inferior to interpolation at Chebyshev zeroes. Accuracy is generally the same or worse, and running times are the same or slower. UP refers to using ordinary polynomials and uniform grid points. Here we have a substantial degradation in speed and/or accuracy.

As predicted by theory, the condition numbers of the Jacobian were strongly related to these performance indices. The cases in Table III always had Jacobians with condition numbers under 10^3 and usually of the order 10. The P cases in Table IV had condition numbers several orders of

magnitude greater, sometimes as great as 10^{20}. The U cases had condition numbers between 10^2 and 10^4, and the UP cases had condition numbers as large as those in the P cases. Our experiments also indicated that the condition number of the Jacobian at our initial guess was of the same order of magnitude as the condition number at the solution. This suggests a useful procedure. If the Jacobian's condition number is large at the initial guess then one should change the basis, fitting conditions, or something (see Section 5.6 below) to reduce the initial conditioning of the problem, whereas a low initial condition number is good evidence that the problem will be well behaved.

5.5. *Complete Polynomial versus Tensor Product Bases*

Table V demonstrates the value of using a complete polynomial basis. Again we report $\log_{10}\|E\|_\infty$ and, below it, the running time for a few cases. The parameter n is one more than the maximum exponent. For example, the $n=3$ case under "Tensor Product" refers to using the tensor product of quadratic polynomials in k and θ, where under "Complete Polynomials" it refers to using the basis $\{1, k, \theta, k^2, k\theta, \theta^2\}$. We see that the complete polynomial basis generally yields a lower quality fit; the exception occurs because the Newton solver had difficulty converging for the tensor product basis but not for the complete polynomial basis. However, the slightly lower accuracy for fixed n was achieved in much less time. Since the real objective is to find a method which achieves maximal accuracy for fixed time, we see from Table V that the complete polynomial basis generally

TABLE V

Tensor Product vs. Complete Polynomials[a]

			Tensor product			Complete polynomials[a]		
γ	ρ	σ	$n=3$	$n=6$	$n=10$	$n=3$	$n=6$	$n=10$
−15.0	0.8	0.04	−2.34[b]	−3.26	−3.48	−1.89	−3.10	−4.06
			:01[c]	:13	14:21	:03	:07	1:09
−0.9	0.3	0.10	−2.19	−3.60	−5.27	−2.14	−3.55	−5.22
			:01	:08	1:21	:01	:05	:32
−0.1	0.3	0.01	−1.00	−2.84	−5.21	−0.99	−2.83	−5.17
			:01	:08	1:24	:01	:05	:35

[a] The tensor product cases in this table used orthogonal collocation with $n_k = n_\theta = m_k = m_\theta = n$ to identify the n^2 free parameters. The complete polynomial cases used Galerkin projections to identify the $1+n+n(n+1)/2$ free parameters.

[b] $\log_{10}\|E\|_\infty$.

[c] Computation time expressed in minutes: seconds.

gives the most accuracy per unit time. Since this is so clear for a two-dimensional problem, one expects that there will be enormous gains from using complete bases when we move to higher dimensions.

5.6. *The Value of "Linearization"*

The final item to note is the importance of rewriting (5.3) as (5.4), and using (5.4) as our defining operator, $\mathcal{N}$. When I used (5.3) instead, the results were substantially inferior in terms of speed and accuracy, particularly when γ was large in magnitude. This is not surprising since I used a Newton solver for solving (5.5), and the Jacobian of (5.5) was better conditioned when R is taken from (5.4) than when it is taken from (5.3). While this may appear magical, the key fact is that (5.4) has two pieces, one linear in the unknown Chebyshev coefficients and the other "more linear" than (5.3). Again, the initial condition number was very good at predicting performance.

6. Comparisons with Alternative Methods

Numerical solutions of the stochastic growth model have been examined, particularly by Taylor and Uhlig [32] and their collaborators. This allows us to compare the methods used above with those alternatives. The important fact to keep in mind is that there is no one best method. The advantage of the projection method framework is that one can easily generate several different implementations, resulting in a menu of alternatives wherein we can make trade-offs among speed, accuracy, and reliability. In the discussion of the various alternatives explored in the literature, I will focus on those aspects which are most likely to help decide which method is best for other problems.

Most approximation techniques used in economics fit into the projection framework. Gustafson [14] used piecewise linear functions and fit them at a fixed set of points, basically a collocation method with a finite-element basis. This was also used in Bizer and Judd [4] to solve for dynamic general equilibrium with taxes and risk. They prove existence of equilibrium by monotone operator methods arising from the Euler equation. The existence proof was constructive and strongly suggests convergence of the numerical algorithm. Wright and Williams [35] have used ordinary polynomials.

These authors used "time iteration" based on the Euler equation; that is, they computed the time t solution from a time $t+1$ solution and iterated "backwards" through time. The disadvantage with that approach is the slow convergence common to algorithms which are (essentially) motivated

by dynamic programming. The differences in running time were substantial. Typical cases in Bizer and Judd [4] took a few minutes for the deterministic case, and several minutes for a stochastic case which allowed only two values for the productivity shock. The advantage of time iteration lies in its use of the operator's monotonicity properties, resulting in greater reliability compared to Newton's method. Furthermore, one never needs to worry about boundary conditions since the initial guess is essentially an initial condition in the implicit evolution equation, and, in these models, discounting makes the limiting solution invariant to the initial condition.

Coleman [8] also used time iteration, along with a tensor product of tent functions to approximate $h(\log k, \log \theta)$; that is, he divided $(\log k, \log \theta)$ space into equi-sized rectangles and assumed that the policy function is a linear combination of 1, $\log k$, $\log \theta$, and $(\log k)(\log \theta)$.[20] Such low-order approximations necessitate many grid points; Coleman [8] used a grid of 50 capital stocks and 20 productivity levels for a total of 1000 free parameters. Such a large number of free parameters results in a time-consuming computation; Coleman [8] reports using nearly 3 min on a 38 MIPS machine, corresponding to at least an hour on a 20 MHz 80386 personal computer.

Christiano [7], Baxter *et al.* [2], and Tauchen [31] solve a related model, where the capital stock is forced to remain on a finite grid of possible values. Generally, methods which discretize the state space are projection techniques which use step function bases, which, as we saw above, are inefficient ways to approximate a smooth investment policy function. The advantage of discretization is that the resulting model may be solved exactly by finite-state dynamic programming methods. However, reducing a continuous-state problem to a finite-state problem still involves an approximation error. Since it takes far more step functions than tent or Chebyshev functions to approximate most smooth functions, the discretized state space method, even when it can be used,[21] is likely to be slow.

Tauchen [31] discretized (k, θ) space by choosing 90 capital stocks so that $\log k$ is uniformly spaced, and 20 θ values to fit a 20-point Gauss–Hermite quadrature rule, for a total of 1800 points, and then treated the problem as a dynamic programming problem on this discrete space. While similar to the procedure above, note the difference in how the θ values were

[20] While this is a standard finite element scheme, it is an unusual choice of basis functions for this problem since all such functions are "saddles"; that is, they are convex (in logs) along one diagonal and concave in the other. A more standard finite element scheme for two-dimensional problems is to use piecewise planar functions over triangles (see Burnett [6]).

[21] Judd [16] contains an example of a partially revealing rational expectations problem which cannot be solved by discretizing the state space, but which can be approximated by more general projection methods.

chosen. In (5.6), there are two places where we chose values for θ. We first chose points at which $R(k, \theta; \mathbf{a})$ is evaluated. These choices were governed by the norm in use and, in our case, were the Gauss–Chebyshev quadrature points. Then, given such a θ, we evaluated the conditional expectation in (5.4), which requires Gauss–Hermite choices for $\tilde{\theta}$. The fact that the two integrals should use different quadrature rules yields a mixed set of θ values which are eventually used, but a set which is motivated by standard quadrature considerations. The other crucial difference between our procedure and Tauchen [31] is that we do not reduce the problem to a finite-state problem, instead using $h(k, \theta)$ at other values of k and θ. It should be noted, however, that Tauchen's excellent [30] numerical study of Lucas [38] used standard linear integral equation methods, which happen to be equivalent to a projection collocation technique. Therefore, the method we used above is the generalization of Tauchen [30] to optimal growth problems.

Our method differs from Sims [29] in that he changes the distribution of the errors so that the policy function is a particular functional form.

While the use of polynomial approximation is not new, other implementations have ignored orthogonality considerations. Some authors, such as Wright and Williams [35], experienced no conditioning problems with ordinary polynomials; this is not surprising since they used low degree polynomials, in which case the Jacobian almost surely has a low condition number. Our discussion of Table V shows how important orthogonality can be.

den Haan and Marcet [15] recognized that their basis is not orthogonal, but made inferences opposite from those made above. They approximated $\ln h(k, \theta)$ with polynomials in $\ln k$ and $\ln \theta$. When they examine the quadratic tensor product approximation, they note that "the terms $(\log k_{t-1})(\log \theta_t)$ and $(\log k_{t-1})^2$ are almost perfectly collinear with the others.... This is, in fact, a fortunate situation. It just means that these terms are redundant, and they can be dropped... without losing any predictive power." This paper comes to quite a different conclusion. Their logic led them to conclude that a four-term representation was as good as possible, whereas we showed that the accuracy can be improved by many orders of magnitude by going to more flexible representations.

This points out the difference between econometrics and numerical analysis. An econometrician has no control over his data, and if he has highly collinear data, then dropping some variables will not reduce predictive power. However, in numerical analysis, *we have control* over our data since we can choose where we check the equation. Numerical analysis is more akin to experimental design than regression. This fact is of critical difference. For example, if high-order basis elements are collinear with other elements, then they should be replaced with orthogonal high-order

elements, not, as suggested by econometric methodology, eliminated. This observation also applies to simulation, a procedure of important use in econometrics. Algorithms which heavily use simulation, such as extended path and parameterized expectations (Marcet [22]), are not likely to do as well in solving numerical problems since they surrender control over the selection of data to a random number generator. This is supported by the comparisons between our results and those reported in Taylor and Uhlig [32].

den Haan and Marcet [15] also differs from the procedures above in that they computed **a** via *functional iteration*; that is, they define the solution as a fixed point of some function, $g(a) = a$, and compute the fixed point as a limit of the sequence $a_{k+1} = g(a_k)$. This is another choice for implementing Step 5. As is typical for functional iteration, they use extrapolation to stabilize the sequence; that is, they compute

$$a_{k+1} = \lambda g(a_k) + (1 - \lambda)\, a_k$$

for some $\lambda \in (0, 1)$. Even after finding a λ which yields convergence, functional iteration procedures are at best linear in convergence.

Another important difference in den Haan and Marcet [15] lies in the residual computation. They simulate a dynamic path which results from a candidate approximation, calculating Euler equation errors along the simulated path, and use them to construct a norm for the error. This simulation approach has many undesirable properties relative to standard quadrature methods. First, to compensate for the low accuracy of Monte Carlo methods, long simulations must be run. den Haan and Marcet [15] report taking about 3 min on a Compaq 386/25 to solve for a solution with three free parameters.

Second, the points at which the errors are calculated depend on the guess for **a**. This endogeneity combined with the simulation aspects means that parameterized expectations evaluates the residual only at states which are frequently visited. This makes it inappropriate for many problems. For example, parametrized expectations cannot be applied to dynamic games since off-equilibrium path behavior is critical in the definition of subgame perfect equilibria. Also, parameterized expectations cannot be reliably used to compute changes in, say, tax policy, since the initial condition after a policy change is likely to be infrequently visited under the new policy, but must be accurately computed if one is to correctly compute the present value of the policy change. In contrast, the projection methods discussed above can be applied to computing tax policy changes (as in Bizer and Judd [4]) and to subgame-perfect equilibria of dynamic games (see Judd [18]).

The accuracy check used in Taylor and Uhlig [32] and developed in

den Haan and Marcet [15] differs from ours. The den Haan–Marcet procedure takes a candidate approximation, computes a simulated time series of consumption, capital, output date, and Euler equation errors, and then regresses the Euler equation errors on lagged variables. That regression should yield only zero coefficients if one had the true solution. They used the estimated coefficients to compute measures of inaccuracy. While this procedure is intuitive, its power is unclear. Klenow [19] has found that the den Haan–Marcet procedure failed to reject candidate solutions which had 1–2% consumption errors. He found that the den Haan–Marcet test was fooled by consumption errors which were large but had no discernible pattern. Our direct accuracy-checking procedure uncovers these optimization errors.

Table VI presents the comparable results for the cases examined in Taylor and Uhlig [32]. Each row specifies a choice of $(n_k, n_\theta, m_k, m_\theta)$ and the entries indicate how the error measures and computational time varied over the cases in Taylor and Uhlig [32]. The main difference is that k_m and k_M are fixed at 0.333 and 1.667, and θ_m and θ_M at 0.4 and 1.6, inde-

TABLE VI

Taylor–Uhlig Cases[a]

					Error range[b]		Time range[c]	
β	m_k	m_θ	n_k	n_θ	Min	Max	Min	Max
0.98	2	2	2	2	−2.96	−2.01	:00.15	.33
	4	3	4	3	−3.86	−3.09	:00.70	:03.00
	7	5	7	5	−5.01	−4.14	:0.48	:10
	7	5	20	12	−5.05	−4.08	:23	1:05
	10	6	10	6	−5.60	−4.65	:19	1:04
	10	6	25	15	−5.58	−4.80	1:06	3:26
	10	10	10	10	−6.64	−4.72	1:06	4:01
0.95	2	2	2	2	−2.52	−1.68	:00.15	:00.33
	4	3	4	3	−3.59	−2.80	:00.68	:01.31
	7	5	7	5	−4.91	−3.44	:05.1	:42
	7	5	20	12	−4.89	−3.38	:23	1:53
	10	6	10	6	−5.48	−4.00	:19	1:31
	10	6	25	15	−5.53	−4.35	1:05	3:31
	10	10	10	10	−6.65	−4.35	1:08	5:24

[a] The calculations assumed the parameter values $\alpha = 0.25$, and $\rho = 0.95$ and computed the policy function over $[k_m, k_M] \times [\theta_m, \theta_M]$, where $k_m = 0.33$, $k_M = 1.67$, $\theta_m = 0.4$, and $\theta_M = 1.6$.

[b] The error range is the range of values for all the error measures in Table III over all the cases in Taylor and Uhlig.

[c] The time range is expressed in minutes: seconds on a COMPAQ 386/20 with a Weitek coprocessor.

pendent of σ and ρ. Given the large and persistent shocks in these cases, these ranges for the state variables are considerably smaller than those which would be chosen by the procedures used in Table III. Also, Table VI covers cases which are substantially different than those in Table III. Despite this, the performance is unchanged. The comparisons between Table VI here and Table 15 in Taylor and Uhlig [32] are immediate. Our four-parameter approximation was always 200–500 times faster than the three-parameter approximation procedure in den Haan and Marcet [15] and yielded an approximation with global Euler errors of around 1%. The higher order approximation methods of Tauchen [31], Coleman [8], Christiano [7], and Baxter [2] attempt to be much more flexible, but were also orders of magnitude slower than the high-order approximations computed here.

The reasons for these speed differences are clear: the projection method here uses smooth functions to approximate the smooth solution, evaluates the critical Euler equation at optimally chosen points, and uses Newton's method, a quadratically convergent procedure, to solve for the unknown coefficients. While it is of little importance whether it takes several minutes or a few seconds to solve this particular problem, the theoretical considerations discussed above all indicate that the strategies used above will be even more valuable when we begin to examine high-dimension problems where speed differences will be of great importance.

7. Convergence Properties of Galerkin Methods

When using numerical procedures, it is desirable to know something concerning its errors. An important focus of theoretical numerical analysis is the derivation of bounds on errors. Two kinds of error results are desirable. First, it is desirable to derive an upper bound on the error for a given level of approximation. Second, if such upper bounds are not possible, it may still be valuable to know that the error goes to zero asymptotically, that is, as one lets the degree of approximation become arbitrarily large. The first kind of error information is rarely available. More typical in numerical algorithms for differential equations are asymptotic results. There has been little work on proving that the algorithms used by economists are asymptotically valid.

Fortunately, there are general theorems concerning the consistency of the Galerkin method. Recall that the Galerkin method takes projections of the residual function against the basis elements, and the integrals are theoretically exact. Zeidler [36, 37] proves consistency for the Galerkin method, assuming that the nonlinear operator $\mathcal{N}$ is monotone, coercive,

and satisfies a growth condition. Galerkin methods are quite natural for computational purposes since a common theoretical way to prove the existence of a solution to an operator is to prove the existence of a solution to an infinite collection of projection conditions. In fact, Zeidler shows that if these conditions are satisfied one simultaneously proves the existence of a (weak) solution and the consistency of the Galerkin method. Similarly, using degree theory, Krasnosel'skii and Zabreiko [20] demonstrate consistency for a more general set of projection methods (possibly including Galerkin methods which use numerical quadrature).

I have not shown that the operators used above satisfy the sufficiency conditions discussed in Zeidler, Krasnosel'skii and Zabreiko, and elsewhere. Even though it remains to be seen whether these theorems do cover our problems, they do indicate promising directions for theorems concerning projection methods for our economic problems.

8. Conclusions

We have shown that a general class of techniques from the numerical partial differential equations literature can be usefully applied and adapted to solve nonlinear economic growth problems. Despite the specificity of the applications discussed here, the general description makes clear the general usefulness of projection methods for economics; Judd [16, 18] discusses some of them. The Fletcher books [11, 12] also illustrate many useful extensions of projection techniques. Within the projection method framework, we were able to interpret the differences among solution techniques previously used in the economics literature. By applying standard numerical analytic principles we were able to develop algorithms over a hundred times faster than previous methods.

References

1. K. Atkinson, "An Introduction to Numerical Analysis," 2nd ed., Wiley, New York, 1989.
2. M. Baxter, M. J. Crucini, and K. G. Rouwenhorst, Solving the stochastic growth models by a discrete-state-space, euler-equation approach, *J. Bus. Econ. Stat.* **8** (1990), 19–21.
3. A. Bensoussan, "Perturbation Methods in Optimal Control," Wiley, New York, 1988.
4. D. S. Bizer and K. L. Judd, Uncertainty and taxation, *Amer. Econ. Rev.* **79** (1989), 331–336.
5. W. A. Brock and L. J. Mirman, Optimal economic growth and uncertainty: The discounted case, *J. Econ. Theory* **4** (1972), 479–513.
6. D. S. Burnett, "Finite Element Analysis," Addison–Wesley, Reading, MA, 1987.

7. L. J. CHRISTIANO, Solving the stochastic growth model by linear-quadratic approximation and by value-function iteration, *J. Bus. Econ. Stat.* **8** (1990), 23–26.
8. W. J. COLEMAN II, Solving the stochastic growth model by policy function iteration, *J. Bus. Econ. Stat.* **8** (1990), 27–29.
9. J. P. DANTHINE, J. B. DONALDSON, AND R. MEHRA, On some computational aspects of equilibrium business cycle theory, *J. Econ. Dynam. Control* **13** (1989), 449–470.
10. J. P. DANTHINE AND J. B. DONALDSON, Stochastic properties of fast vs. slow growing economies, *Econometrica* **49** (1981), 1007–1033.
11. C. A. J. FLETCHER, "Computational Galerkin Techniques," Springer-Verlag, New York, 1984.
12. C. A. J. FLETCHER, "Computational Techniques for Fluid Dynamics, Vols. I, II," Springer-Verlag, Berlin, 1988.
13. G. H. GOLUB AND J. M. ORTEGA, "Scientific Computing and Differential Equations: An Introduction to Numerical Methods," Academic Press, San Diego, 1992.
14. R. L. GUSTAFSON, "Carryover Levels for Grains: A Method for Determining Amounts That Are Optimal under Specified Conditions, USDA Technical Bulletin 1178," 1958.
15. W. J. DEN HAAN AND A. MARCET, Solving the stochastic growth model by parameterizing expectations, *J. Bus. Econ. Stat.* **8** (1990), 31–34.
16. K. L. JUDD, Numerical methods in economics, manuscript, Hoover Institution, 1991.
17. K. L. JUDD, Asymptotic methods in dynamic economic models, mimeo, Hoover Institution, October 1990.
18. K. L. JUDD, Minimum weighted residual methods for solving dynamic economic models, Hoover Institution, 1990.
19. P. J. KLENOW, "Externalities and Business Cycles," Ph.D. thesis, Department of Economics, Stanford University, June 1991.
20. M. A. KRASNOSEL'SKII AND P. ZABREIKO, "Geometrical Methods of Nonlinear Analysis," Springer-Verlag, New York/Berlin, 1984.
21. J. P. M. MAGILL, A local analysis of N-sector capital accumulation under uncertainty, *J. Econ. Theory* **15** (1977), 211–219.
22. A. MARCET, Solving nonlinear stochastic models by parametrizing expectations, mimeo, Carnegie–Mellon University, 1989.
23. E. R. MCGRATTON, Solving the stochastic growth model by linear-quadratic approximation, *J. Bus. Econ. Stat.* **8** (1990), 41–43.
24. M. J. MIRANDA AND P. G. HELMBURGER, The effects of commodity price stabilization programs, *Amer. Econ. Rev.* **78** (1988), 46–58.
25. P. M. PRENTER, "Splines and Variational Methods," Wiley, New York, 1989.
26. E. C. PRESCOTT AND R. MEHRA, Recursive competitive equilibrium: The case of homogeneous agents, *Econometrica* **48** (1980), 1365–1379.
27. T. J. RIVLIN, "Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory," Wiley–Interscience, New York, 1990.
28. J. RUST, Maximum likelihood estimation of discrete control processes, *SIAM J. Control Optim.* **26** (1988), 1006–1023.
29. C. A. SIMS, Solving the stochastic growth model by backsolving with a particular nonlinear form for the decision rule, *J. Bus. Econ. Stat.* **8** (1990), 45–47.
30. G. TAUCHEN, "Quadrature-Based Methods for Obtaining Approximate Solutions to the Integral Equations of Nonlinear Rational Expectations Models," mimeo, Duke University, 1986.
31. G. TAUCHEN, Solving the stochastic growth model by using quadrature methods and value-function iterations, *J. Bus. Econ. Stat.* **8** (1990), 49–51.
32. J. B. TAYLOR AND H. UHLIG, Solving nonlinear stochastic growth models: A comparison of alternative solution methods, *J. Bus. Econ. Stat.* **8** (1990), 1–18.

33. J. WILLIAMS AND B. WRIGHT, "Storage and Commodity Markets," Cambridge Univ. Press, Cambridge, UK, 1991.
34. J. WILLIAMS AND B. WRIGTH, The economic role of commodity storage, *Econ. J.* **92** (1982), 596–614.
35. J. WILLIAMS AND B. WRIGHT, The roles of public and private storage in managing oil import disruptions, *Bell J. Econ.* **13** (1982), 341–353.
36. E. ZEIDLER, "Nonlinear Functional Analysis and Its Applications: Volume I," Springer-Verlag, New York, 1986.
37. E. ZEIDLER, "Nonlinear Functional Analysis: Volume II," Springer-Verlag, New York, 1989.
38. R. E. LUCAS, JR., Asset prices in an exchange economy, *Econometrica* **46** (1978), 1429–1445.