Chapter 5

NUMERICAL SOLUTION OF DYNAMIC ECONOMIC MODELS*

MANUEL S. SANTOS

Department of Economics, University of Minnesota

Contents

Abstract 312
Keywords 312
1. Introduction 313
2. The model and preliminary considerations 314
3. Bellman's equation and differentiability of the value function 319
   3.1. Bellman's equation and the contraction property of the dynamic programming algorithm 320
   3.2. Differentiability of the value function 321
4. A numerical dynamic programming algorithm 324
   4.1. Formulation of the numerical algorithm 324
   4.2. Existence of numerical solutions and derivation of error bounds 326
   4.3. Stability of the numerical algorithm 328
   4.4. Numerical maximization 329
   4.5. Numerical integration 332
5. Extensions of the basic algorithm 334
   5.1. Multigrid methods 334
   5.2. Policy iteration 336
   5.3. Modified policy iteration 338
   5.4. Polynomial interpolation and spline functions 340
      5.4.1. Polynomial interpolation 340
      5.4.2. Spline functions 344
6. Numerical approximations of the Euler equation 345
   6.1. Numerical methods for approximating the Euler equation 347
   6.2. Accuracy based upon the Euler equation residuals 352
7. Some numerical experiments 355
   7.1. A one-sector deterministic growth model with leisure 355
   7.2. A one-sector chaotic growth model 362
   7.3. A one-sector stochastic growth model with leisure 364

* The author is grateful to Jerry Bona, Antonio Ladron de Guevara, Ken Judd, John Rust, John Taylor and Jesus Vigo for helpful discussions on this topic. Special thanks are due to Adrian Peralta-Alva for his devoted computational assistance.

Handbook of Macroeconomics, Volume 1, Edited by J.B. Taylor and M. Woodford

8. Quadratic approximations 368
9. Testing economic theories 375
10. A practical approach to computation 379
References 382

Abstract

This chapter is concerned with numerical simulation of dynamic economic models. We focus on some basic algorithms and assess their accuracy and stability properties. This analysis is useful for an optimal implementation and testing of these procedures, as well as to evaluate their performance. Several examples are provided in order to illustrate the functioning and efficiency of these algorithms.

Keywords

dynamic economic model, value function, policy function, Euler equation, numerical algorithm, numerical solution, approximation error

JEL classification: C61, C63, C68

1. Introduction

This chapter offers an overview of some important methods for simulating solutions of dynamic economic models with the aid of high-performance computers. The recent surge of research in this area has been impelled by current developments in computer processing, algorithm design, software and data storage. This progress has fostered the numerical analysis of a wide range of problems to limits beyond what one could possibly foresee a few years ago. Since advances in our computational capabilities are likely to continue, it is expected that numerical simulation of economic models will be an attractive and expanding research field.

A basic concern in science is understanding model predictions. Classical mathematical methods can help us derive basic qualitative properties of solutions such as existence, uniqueness and differentiability. But these methods usually fail to afford us with the specific information necessary to test a model. There are some well-known economic examples in which optimal decisions have an analytical representation or closed-form solution (e.g., models with linear decision rules, or with constant elasticities for consumption and saving). In these cases, optimal policies are generally derived from algebraic manipulations or analytical techniques, and computational methods are usually not needed. Such a state of affairs, however, is not the most common situation. Most dynamic economic models feature essential nonlinearities stemming from intra and intertemporal substitutions over non-constant margins. (These nonlinearities become more pronounced when uncertainty is present in the decision problem.) Digital computers are then the most plausible way to understand the behavior of a given model with a view toward its eventual testing. And one should expect that computational techniques will help to bridge the traditional gap between theoretical developments and empirical economic analysis.

Over the past decades, economic thinking has achieved levels of rigor and argumentation comparable to any other scientific discipline. The principles of axiomatization and mathematical logic are well rooted in economic theory. Also, empirical work has endorsed the underlying postulates of statistical analysis. If our main objective is to collect the fruits of this scientific endeavor, the same accepted practices should prevail for solving economic models.

A framework for carrying out and reporting numerical experiments is presented in Bona and Santos (1997). Our purpose here is to focus on the accuracy and stability properties of some algorithms currently used by economists, and evaluate their performance in the context of some growth models. Accuracy seems to be a minimal requirement for judging a numerical simulation. And once we have a theory of the error involved in a numerical approximation, we are in a better position to devise more efficient algorithms, and to test and debug the computer code. Stability is concerned with possible variations that numerical errors and misspecifications of parameter values may inflict on the computed solution. Unstable algorithms may lead to odd outcomes, and may considerably lessen the power of a numerical simulation in testing a particular theory. Our study of accuracy and stability properties will be

complemented with some numerical experiments where we discuss further aspects of the implementation and performance of these algorithms.

Computational tools have been applied to a wide variety of problems in economics and finance. Rather than providing a thorough review of these applications, the present work focuses on the analysis of some fundamental algorithms as applied to some simple growth models. Once the functioning of these algorithms is understood in this basic context, these same techniques should be of potential interest for solving other model economies, even though the assumptions of strong concavity and differentiability are fundamental to some of our results.

There are several survey papers on this topic, which to a certain extent may be complementary to the present one. Kehoe (1991) reviews the literature on static general equilibrium models, along with certain numerical methods for dynamic economies. Our paper is in the spirit of Taylor and Uhlig (1990), who describe several computational methods and evaluate their performance. In our case, we shall concentrate on fewer methods, place more emphasis on their accuracy and stability properties, and carry out alternative numerical tests. Marcet (1994) reexamines the literature on the so called parameterized expectations algorithm and presents a sample of its applications. This method computes the optimal law of motion from a direct approximation of the Euler equation. A variety of other methods that approximate the Euler equation are laid out in Judd (1992, 1996), who has advocated for the use of polynomial approximations with certain desired orthogonality properties. Variants of both Marcet's and Judd's procedures along with some other related algorithms are reevaluated in Christiano and Fisher (1994). Finally, Rust (1996) considers several numerical methods for solving dynamic programming programs, and analyzes their complexity properties. Complexity theory presents an integrated framework for assessing the efficiency of algorithms, although some of these asymptotic results may not be binding in simple applications.

2. The model and preliminary considerations

We begin our analysis with a stochastic, reduced-form model of economic growth in which the solution to the optimal planning problem may be interpreted as the equilibrium law of motion of a decentralized economy. Our framework is encompassed in the class of economies set out in Stokey and Lucas (1989). The reader is referred to this monograph for some basic definitions and technical points raised in the course of our discussion. The usefulness of this relatively abstract setting for carrying out numerical computations will be illustrated below with some simple examples.

Let $(K, \mathcal{K})$ and $(Z, \mathcal{Z})$ be measurable spaces, and let $(S, \mathcal{S}) = (K \times Z, \mathcal{K} \times \mathcal{Z})$ be the product space. The set K contains all possible values for the endogenous state variable, Z is the set of possible values for the exogenous shock, and S is the set of state values for the system. The evolution of the random component $\{z_t\}_{t \geq 0}$ is governed by a stochastic law defined by a function $\varphi : Z \times Z \to Z$

and an i.i.d. process $\{\varepsilon_t\}_{t\geqslant 1}$ where $z_t = \varphi(z_{t-1}, \varepsilon_t)$. It follows that the mapping φ induces a time-invariant transition function Q on $(Z, \mathcal{Z})$. Moreover, for each z_0 in Z one can define a probability measure $\mu^t(z_0, \cdot)$ on every t-fold product space $(Z^t, \mathcal{Z}^t) = (Z \times Z \times \cdots \times Z, \mathcal{Z} \times \mathcal{Z} \times \cdots \times \mathcal{Z})$ comprising all partial histories of the form $z^t = (z_1, \ldots, z_t)$.

The physical constraints of the economy are summarized by a given *feasible technology set,* $\Omega \subset K \times K \times Z$, which is the graph of a continuous correspondence, $\Gamma : K \times Z \to K$. The intertemporal objective is characterized by a *one-period return function* v on Ω and a *discount factor,* $0 < \beta < 1$. The optimization problem is to find a sequence of (measurable) functions, $\{\pi_t\}_{t=0}^{\infty}$, $\pi_t : Z^{t-1} \to K$, as a solution to

$$W(k_0, z_0) = \sup_{\{\pi_t\}_{t\geqslant 0}} \sum_{t=0}^{\infty} \beta^t \int_{Z^t} v(\pi_t, \pi_{t+1}, z_t)\, \mu^t(z_0, \mathrm{d}z^t)$$

$$\text{subject to } \begin{cases} (\pi_t, \pi_{t+1}, z_t) \in \Omega \\ z_{t+1} = \varphi(z_t, \varepsilon_{t+1}) \\ (k_0, z_0) \text{ fixed}, \quad \pi_0 = k_0, \quad 0 < \beta < 1, \text{ and } t = 0, 1, 2, \ldots. \end{cases} \tag{2.1}$$

As presently illustrated, this stylized framework comprises several basic macroeconomic models, and it is appropriate for computational purposes.

Example 2.1. *A one-sector deterministic growth model with leisure*: Consider the following simple dynamic optimization problem:

$$\max_{\{c_t, l_t, i_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t [\lambda \log c_t + (1 - \lambda) \log l_t]$$

$$\text{subject to } \begin{cases} c_t + i_t \leqslant A k_t^{\alpha} (1 - l_t)^{1-\alpha} \\ k_{t+1} = i_t + (1 - \delta) k_t \\ 0 < \beta < 1, \quad 0 < \lambda \leqslant 1, \quad A > 0, \\ 0 < \alpha < 1, \quad 0 \leqslant \delta \leqslant 1, \\ k_t, c_t \geqslant 0, \quad 0 \leqslant l_t \leqslant 1, \quad k_0 \text{ given}, \quad t = 0, 1, 2, \ldots. \end{cases} \tag{2.2}$$

At each time $t = 0, 1, 2, \ldots$, the economy produces a single good that can be either consumed, c_t, or invested, i_t, for the accumulation of physical capital, k_t. There is available a normalized unit of time that can be spent in leisure activities, l_t, or in the production sector, $(1 - l_t)$. The one-period utility function is logarithmic, and $0 < \lambda \leqslant 1$, is the weight of consumption. The one-period production function is Cobb–Douglas, $Ak^{\alpha}(1 - l)^{1-\alpha}$, characterized by parameters A and α, where $A > 0$ is a normalizing parameter (although for given measurement units it may represent the technology level), and $0 < \alpha < 1$ is interpreted as the income share of physical capital. Physical capital, k_t, is subject to a depreciation factor, $0 \leqslant \delta \leqslant 1$.

We observe the following asymmetry between variables c_t, l_t and k_t. At the beginning of each time $t \geqslant 0$, variable k_t has been already determined, and from such a value one can figure out the set of future technologically feasible plans $\{c_T, l_T, k_T\}_{T \geqslant t}$ for the economy. More specifically, k_t is a *state variable*. On the other hand, c_t and l_t can only affect the value of the current one-period utility, for given k_t and k_{t+1}. That is, c_t and l_t are *control variables*.

Let R_+ be the set of non-negative numbers. For all given feasible pairs (k, k'), define

$$v(k,k') = \max_{c,l} \lambda \log c + (1-\lambda)\log l$$
$$\text{subject to } Ak^{\alpha}(1-l)^{1-\alpha} + (1-\delta)k - c - k' \geqslant 0.$$

Let $c(k,k')$ and $l(k,k')$ represent the optimal choices for this one-period optimization problem. For every $k \geqslant 0$, let

$$\Gamma(k) = \{k' \in R_+ : Ak^{\alpha}\left(1 - l\left(k,k'\right)\right)^{1-\alpha} + (1-\delta)k - k' \geqslant 0\},$$

and let Ω denote the graph of the correspondence Γ. Let us now write the following optimization problem:

$$\max_{\{k_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t v(k_t, k_{t+1})$$
$$\text{subject to } \begin{cases} (k_t, k_{t+1}) \in \Omega \\ k_0 \text{ fixed}, \quad 0 < \beta < 1, \text{ and } t = 0, 1, 2, \ldots. \end{cases} \tag{2.3}$$

Then, both problems (2.2) and (2.3) contain the same set of optimal solutions. In other words, if $\{k_t\}_{t=0}^{\infty}$ is an optimal solution to problem (2.3), then $\{c_t(k_t, k_{t+1}), l_t(k_t, k_{t+1}), i_t(k_t, k_{t+1})\}_{t=0}^{\infty}$ is an optimal solution to problem (2.2), and vice versa.

Example 2.2. *A one-sector stochastic growth model with leisure*: Uncertainty is now introduced in this simple setting. The maximization problem is written as follows:

$$\max_{\{c_t, l_t, i_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t \left[\lambda \log c_t + (1-\lambda)\log l_t\right]$$
$$\text{subject to } \begin{cases} c_t + i_t = z_t A k_t^{\alpha}(1-l_t)^{1-\alpha} \\ k_{t+1} = i_t + (1-\delta)k_t \\ \log z_{t+1} = \rho \log z_t + \varepsilon_{t+1} \\ 0 < \beta < 1, \quad 0 < \lambda \leqslant 1, \quad A > 0, \\ 0 < \alpha < 1, \quad 0 \leqslant \delta \leqslant 1, \quad 0 \leqslant \rho < 1, \\ k_t, c_t \geqslant 0, \quad 0 \leqslant l_t \leqslant 1, \quad k_0 \text{ and } z_0 \text{ given}, t = 0, 1, 2, \ldots. \end{cases} \tag{2.4}$$

Here, random variable $z_t \geqslant 0$ enters the production function, and follows the law of motion $z_t = \varphi(z_{t-1}, \varepsilon_t)$, for $\varphi(z, \varepsilon) = z^{\rho} e^{\varepsilon}$. Analogously to the previous optimization

problem, one can derive optimal controls $c(k,k',z)$ and $l(k,k',z)$, and define the one-period return function $v(k,k',z)$ and the technological correspondence $\Gamma(k,z)$, so that the optimization problem can be expressed as in (2.1). The state variables are k and z, and the control variables are c and l. The objective function is now the discounted sum of expected utilities, and E_0 is the expectations operator at time 0.

There are certain variations of these simple examples worth considering. For instance, the one-period utility function may be of the form

$$\frac{\left(c^{\lambda}l^{1-\lambda}\right)^{1-\sigma}-1}{1-\sigma} \quad \text{for} \quad \sigma > 0.$$

Since this is a monotonic transformation of the previous one-period utility, the optimal control functions, $c(k,k',z)$ and $l(k,k',z)$, remain unchanged. Likewise, the derivation of the one-period return function $v(k,k',z)$, and the technological correspondence $\Gamma(k,z)$, proceed accordingly. Another interesting extension is to allow for an (endogenous) control variable that can influence the law of motion of stochastic variable z.

Example 2.3. *A one-sector stochastic growth model with endogenous uncertainty*: Consider the following dynamic optimization problem:

$$\max_{\{c_t,l_t,e_t,u_t,i_t\}_{t=0}^{\infty}} E_0\sum_{t=0}^{\infty}\beta^t\left[\lambda\log c_t+(1-\lambda)\log l_t\right]$$

$$\text{subject to}\begin{cases} c_t+i_t=z_tAk_t^{\alpha}e_t^{1-\alpha} \\ k_{t+1}=i_t+(1-\delta)k_t \\ \log z_{t+1}=\rho\log z_t+(1-u_t)\varepsilon_{t+1} \\ 0<\beta<1, \quad 0<\lambda\leqslant 1, \quad A>0, \\ 0<\alpha<1, \quad 0\leqslant\delta\leqslant 1, \quad 0\leqslant\rho<1, \\ k_t,c_t\geqslant 0, \quad 0\leqslant l_t+u_t+e_t\leqslant 1, \\ l_t\geqslant 0, \quad u_t\geqslant 0, \quad e_t\geqslant 0, \\ k_0 \text{ and } z_0 \text{ given}, \quad t=0,1,2,\ldots. \end{cases} \tag{2.5}$$

Here, the available unit of time can be allocated over three different margins: Leisure activities, l_t, working in the production sector, e_t, or time spent in reducing the variance of the random shock, u_t. One can find this kind of trade-offs in models with durable goods, human capital accumulation, or health care [cf. Becker (1993), Ladron de Guevara et al. (1997), Rust (1987)]. The reduced version of this optimization problem involves a return function, $v(k_t,k_{t+1},z_t,z_{t+1})$, and a technological correspondence, $\Gamma(k_t,z_t)$. Hence, variable z_{t+1} appears as an additional argument of function v.

Strictly speaking, model (2.5) cannot be embodied in the framework of our baseline optimization problem (2.1), since our optimization problem specifies an exogenous law of motion for variable z. Most of our results below, however, can be extended to a more general setting in which the stochastic process may be endogenously influenced by some control variables. As a matter of fact, our optimization problem (2.1) contemplates endogenous and exogenous state variables, and for convenience we have assumed an exogenous process for the stochastic law of motion.

One conclusion to be drawn from these examples is that the above modelization (2.1) is very useful for computational purposes. In most economic models, there are state and control variables. However, control variables can generally be calculated in terms of the state variables. Hence, recasting the model in reduced form – in terms of the state variables – will generally lower its computational burden.

For convenience of exposition, we now impose certain assumptions on the reduced form model (2.1). As one can infer from the above examples, these postulates may be derived from more primitive formulations, and hold in most standard macroeconomic applications.

Assumption 1: *The set $K \times Z \subset R^l \times R^m$ is compact, and for each fixed z the set $\Omega_z = \{(k,k') \mid (k,k',z) \in \Omega\}$ is convex.*

Assumption 2: *The mapping $v : \Omega \to R$ is continuous, and on the interior of its domain it is differentiable of class C^2 with bounded first- and second-order derivatives. Moreover, for all fixed z there exists some constant $\eta > 0$ such that $v(k,k',z) + \frac{1}{2}\eta \|k'\|^2$ is concave as a function on (k,k').*

Assumption 3: *For each interior point (k_0,z_0) in $K \times Z$ every optimal realization $\{k_t,z_t\}_{t\geqslant 0}$ has the property that $(k_t,k_{t+1},z_t) \in int(\Omega)$ for each $t \geqslant 0$.*

Assumption 4: *The function $\varphi : Z \times Z \to Z$ is continuous, and for each fixed ε, the mapping $\varphi(\cdot,\varepsilon)$ is C^2 and the derivative functions $D_1\varphi(z,\varepsilon)$ and $D_{11}\varphi(z,\varepsilon)$ are bounded and jointly continuous over all points (z,ε) in $Z \times Z$. Also, there are non-negative constants $0 \leqslant \rho < \beta^{-1/2}$ and $C \geqslant 0$ such that the first- and second-order partial derivatives of z_t with respect to z_0, $\partial z_t/\partial z_0$ and $\partial^2 z_t/\partial z_0^2$, have the property that*

$$\left\| \frac{\partial z_t}{\partial z_0} \right\| \leqslant C\rho^t \quad \text{and} \quad \left\| \frac{\partial^2 z_t}{\partial z_0^2} \right\| \leqslant C\rho^t \quad \text{for each } t > 0.$$

These assumptions are entirely standard [cf. Stokey and Lucas (1989), Ch. 9]. In Assumption 1, the compactness of the domain seems a natural restriction for the purposes of computing numerical solutions. And it should be observed that the convexity requirement on the technological correspondence precludes the existence of

increasing returns to scale. In Assumption 2, the norm $\|k'\|$ is the usual Euclidean norm. Hence, such hypothesis imposes a strong form of concavity on the second component of the utility function, and over compact sets the condition is weaker than the conventional postulate that the Hessian matrix $D^2 v_z(k, k')$ be negative definite[1] over all points (k, k', z) in Ω. The interiority condition asserted in Assumption 3 is necessary to establish subsequently the smoothness of optimal paths. As in the above examples, this assumption is satisfied in most models that feature Inada-type conditions [cf. Brock and Mirman (1972)], or when the domain is restricted to a certain absorbing region containing the asymptotic dynamics of the system. Regarding Assumption 4, note that $z_t = \varphi(\varphi(\cdots(\varphi(z_0, \varepsilon_1), \varepsilon_2)\cdots), \varepsilon_t)$. The assumption then requires the existence of some constants $0 \leqslant \rho < 1/\beta^{1/2}$ and $C \geqslant 0$ such that the matrix norms

$$\left\| \frac{\partial}{\partial z_0} \varphi(\varphi(\cdots(\varphi(z_0, \varepsilon_1), \varepsilon_2)\cdots), \varepsilon_t) \right\| \leqslant C\rho^t,$$

$$\left\| \frac{\partial^2}{\partial z_0^2} \varphi(\varphi(\cdots(\varphi(z_0, \varepsilon_1), \varepsilon_2)\cdots), \varepsilon_t) \right\| \leqslant C\rho^t$$

for every realization $(\varepsilon_1, \ldots, \varepsilon_t)$ and $t > 0$. Thus, this condition limits the asymptotic growth of the stochastic process, and it is satisfied in most models with bounded first- and second-order moments[2]. For the purposes of our analysis such condition would not be needed in situations where z is an endogenous variable [e.g., see the contrast between derivatives (3.3) and (3.4) below].

3. Bellman's equation and differentiability of the value function

In this section, we introduce the methodology of *dynamic programming*, and recall some qualitative properties of optimal solutions. These methods have proven very

[1] For functions v over a set $\Omega \subset R^l \times R^l \times R^m$, $Dv(k_0, k_1, z)$ will denote the (first-order) derivative of v evaluated at an interior point (k_0, k_1, z), and $D_i v(k_0, k_1, z)$, $i = 1, 2, 3$, will denote the partial derivative of v with respect to the ith component variable. Similarly, $D_{ij} v(k_0, k_1, z)$ will denote a second-order partial derivative of v with respect to the ith and jth components. Sometimes we will use the notation $D^2 v_z(k_0, k_1)$ to represent the Hessian matrix of the mapping $v(\cdot, \cdot, z)$, where z is held fixed.

[2] An additional, technical condition usually imposed in this stochastic framework is that the transition function Q on $(Z, \mathcal{Z})$ is weakly continuous or that it satisfies the *Feller property* [cf. Stokey and Lucas (1989), Ch. 9]. For the particular stochastic law of motion considered here, we can show that this technical assumption always holds. Thus, let f be a bounded continuous function on $K \times Z$. Let $\Psi(k, z) = \int f(k, z') Q(z, \mathrm{d}z')$, and assume that the sequence $\{(k^n, z^n)\}_{n \geqslant 0}$ converges to (k, z). Now, in order to show that $\Psi(k, z)$ is a continuous function, write the integral $\int f(k, z') Q(z, \mathrm{d}z')$ in the form $\int f(k, \varphi(z, \varepsilon))\, \mu(\mathrm{d}\varepsilon)$, and then apply the Lebesgue dominated convergence theorem. This alternative way of writing the integral will also be advantageous to compute certain derivatives, since in such a case it will not be necessary to differentiate over the transition function Q [cf. expressions (3.4) and (3.5) below].

useful for the construction of reliable numerical procedures. Hence, the design and implementation of efficient numerical algorithms provides an added stimulus for the study of analytical tools and for a reexamination of qualitative properties of optimal solutions such as existence, uniqueness, stability, and differentiability.

3.1. *Bellman's equation and the contraction property of the dynamic programming algorithm*

Under the above assumptions the value function $W(k_0, z_0)$, given in Equation (2.1), is well defined and jointly continuous [cf. Stokey and Lucas (1989)]. Moreover, for each fixed z_0 the mapping $W(\cdot, z_0)$ is concave, and satisfies the *Bellman equation*

$$W(k_0, z_0) = \max_{k_1} \; v(k_0, k_1, z_0) + \beta \int_Z W(k_1, z_1)\, Q(z_0, \mathrm{d}z_1) \qquad \text{subject to } (k_0, k_1, z_0) \in \Omega. \tag{3.1}$$

The optimal value $W(k_0, z_0)$ is attained at a unique point given by the policy function $k_1 = g(k_0, z_0)$. The policy function is also continuous. Furthermore, an iterated substitution on the right-hand side of (3.1) shows that the set of optimal contingency plans $\{k_t, z_t\}_{t \geqslant 0}$ is a Markov process determined by the optimal policy $k_{t+1} = g(k_t, z_t)$.

From a computational point of view, Bellman's equation is a functional equation, and function W is a fixed point of this equation[3]. The most common way to compute this fixed point is by the following recursive algorithm known as the *method of successive approximations* or *value-function iteration*. Let $\mathcal{W}$ be the space of bounded, continuous functions V on the state space $K \times Z$ endowed with the norm $\|V\| = \max_{(k,z) \in K \times Z} |V(k, z)|$. Define the (non-linear) operator $T : \mathcal{W} \to \mathcal{W}$, as

$$T(V)(k_0, z_0) = \max_{k_1} v(k_0, k_1, z_0) + \beta \int_Z V(k_1, z_1) Q(z_0, \mathrm{d}z_1) \qquad \text{subject to } (k_0, k_1, z_0) \in \Omega \tag{3.2}$$

for $V \in \mathcal{W}$. This functional mapping is known as the *dynamic programming operator*. Blackwell (1965) and Denardo (1967) first observed that T is a contractive application on $\mathcal{W}$ with modulus $0 < \beta < 1$, i.e. $\|TV_0 - TV_1\| \leqslant \beta \|V_0 - V_1\|$ for $V_0, V_1 \in \mathcal{W}$. It follows that W is the unique fixed point under T, and $\|W - V_n\| \leqslant \beta^n \|W - V_0\|$ for $V_n = T^n V_0$, where T^n denotes the n-times composition of T. Some simple examples can be constructed in which constant β is a tight upper bound. Hence, the method of successive approximations usually yields a linear rate of convergence to the fixed point W.

[3] It should be pointed out that certain regularity conditions are required for the existence of a fixed point for Bellman's equation. In stochastic problems, a main technical difficulty is the measurability of the value function [cf. Stokey and Lucas (1989), p. 253].

3.2. Differentiability of the value function

Under general assumptions, continuity of the value and policy functions can be established following the method of successive approximations. That is, by the theorem of the maximum [e.g. Stokey and Lucas (1989), p. 62], continuity is preserved at each iteration step, $V_n = T(V_{n-1})$ for $n \geqslant 1$, and the contraction property of T implies that convergence is uniform. This method, however, has not proved so effective for studying differentiability properties. For this purpose various tools of analysis have been introduced.

Invoking some basic properties of the subgradient of a concave function [cf. Benveniste and Scheinkman (1979)], one can show that at every interior point (k_0, z_0) function W is differentiable with respect to k, and the partial derivative is given by the familiar envelope condition

$$D_1 W(k_0, z_0) = D_1 v(k_0, k_1, z_0), \tag{3.3}$$

where k_1 is the optimal point. Moreover, a straightforward calculation allows us to check that

$$D_2 W(k_0, z_0) = \sum_{t=0}^{\infty} \beta^t \int_{Z^t} \left[D_3 v(k_t, k_{t+1}, z_t) \cdot \frac{\partial z_t}{\partial z_0} \right] \mu^t(z_0, \mathrm{d}z^t), \tag{3.4}$$

where again the right-hand side of Equation (3.4) is evaluated at the optimal contingency plan $\{k_t, z_t\}_{t \geqslant 0}$. Hence, W is a C^1 mapping. Therefore, at interior points the optimal policy $k_1 = g(k_0, z_0)$ can be characterized by the first-order condition

$$D_2 v(k_0, k_1, z_0) + \beta \int_Z D_1 W(k_1, z_1)\, Q(z_0, \mathrm{d}z_1) = 0. \tag{3.5}$$

Under the foregoing assumptions, it follows from a mere application of the implicit function theorem to (3.5) that if W is differentiable of class C^2 then g is C^1. The converse result is not so straightforward as Equation (3.4) involves an infinite series of partial derivatives.

In a discrete-time stochastic growth framework, the second-order differentiability of the value function has been studied by Blume, Easley and O'Hara (1982), Gallego (1993) and Santos and Vigo (1995). The assumptions imposed by Blume, Easley and O'Hara are fairly strong for the present context. Their method of proof builds on the idea that uncertainty should smooth out the optimization problem, since the integral operation in problem (2.1) can act as a convolution on function W. To guarantee this nice smoothness property of the integral these authors postulate certain separability and invertibility conditions over the stochastic process as well as smooth density functions. These assumptions would be fairly stringent for our purposes. Indeed, in the present model random variables may be discrete, or may not contain a smooth density, allowing thus for event trees or other commonly studied modelizations with uncertainty.

Gallego (1993) extends the method of analysis presented by Santos (1991, 1994). This approach was originally developed in deterministic growth models, but it turns out to extend in a simple manner to our stochastic framework. The idea here is to construct a "candidate" mapping for the second-order derivative of the value function that has nice continuity properties. This mapping is derived as the solution to a quadratic approximation of the original optimization problem. This associated problem is much easier to manipulate, and under the above assumptions, one simply checks that it defines the second-order derivative of W. Moreover, the quadratic optimization problem provides bounds for such derivative in terms of primitive data of the model, and it should be a benchmark for assessing the suitability of alternative quadratic approximations encountered in the literature.

Both Blume et al. (1982) and Gallego (1993) focus on the second-order differentiability of W on K. Santos and Vigo (1995) extend this analysis to allow for differentiability on Z. These are their main results:

Theorem 3.1. *Under Assumptions (1)–(4) the value function W is a C^2 mapping on $int(K \times Z)$.*

Corollary 3.2. *Under Assumptions (1)–(4) the policy function g is a C^1 mapping on $int(K \times Z)$.*

The derivatives of these functions can be computed from primitive elements of the model in the following recursive way [cf. *op. cit.*]. First, $D_{11}W(k_0, z_0)$ is determined by the associated quadratic optimization problem

$$\begin{aligned} & x_0 \cdot D_{11}W(k_0, z_0) \cdot x_0 \\ & \quad = \max_{\{\pi_t\}_{t\geqslant 0}} \sum_{t=0}^{\infty} \beta^t \int_{Z^t} \left[(\pi_t, \pi_{t+1}) \cdot D^2 v_{z_t}(k_t, k_{t+1}) \cdot (\pi_t, \pi_{t+1})\right] \mu^t(z_0, \mathrm{d}z^t) \qquad (3.6) \\ & \qquad \text{subject to } z_{t+1} = \varphi(z_t, \varepsilon_{t+1}), \quad t = 0, 1, 2, \ldots. \end{aligned}$$

Here the maximization proceeds over all measurable functions $\{\pi_t\}_{t=0}^{\infty}$, $\pi_t : Z^{t-1} \to R^l$, for $t \geqslant 1$ with $\pi_0 = x_0$ and z_0 fixed, the one-period objective $D^2 v_{z_t}(k_t, k_{t+1})$ is the Hessian matrix of the mapping $v(\cdot, \cdot, z_t)$ for given z_t, and $\{k_t, z_t\}_{t=0}^{\infty}$ is the optimal contingency plan to problem (2.1) for the initial value (k_0, z_0).

From this characterization, one readily proves that the optimal plan $\{\hat{\pi}_t\}_{t=0}^{\infty}$ to maximization problem (3.6) determines the derivative of the policy function with respect to k_0. That is, $\hat{\pi}_t = D_1 g^t(k_0, z_0) \cdot \pi_0$ for $t \geqslant 1$, where $D_1 g^t(k_0, z_0)$ denotes the derivative of the function $g(g(\cdots g(k_0, z_0)\cdots), z_{t-2}), z_{t-1})$ with respect to k_0 for every possible realization $(z_1, z_2, \ldots, z_{t-1})$. Given that $(x_0, 0, 0, 0, \ldots)$ is a feasible solution to maximization problem (3.6), we obtain that

$$\|D_{11}W(k_0, z_0)\| \leqslant \|D_{11}v(k_0, g(k_0, z_0), z_0)\| \leqslant L, \qquad (3.7)$$

where

$$L = \sup_{(k_0, k_1, z_0)\in\Omega} \|D_{11}v(k_0, g(k_0, z_0), z_0)\|.$$

Moreover, if $\{\widehat{\pi}_t\}_{t=0}^{\infty}$ is an optimal solution to problem (3.6) with $\|\widehat{\pi}_0\| = 1$, then in view of the asserted concavity of $v(k, k', z)$ we must have

$$\sum_{t=0}^{\infty} \beta^t \int_{Z^t} \left[\widehat{\pi}_{t+1} \cdot \widehat{\pi}_{t+1}\right] \mu^t(z_0, \mathrm{d}z^t) \leqslant \frac{L}{\eta}, \tag{3.8}$$

where $\widehat{\pi}_{t+1} \cdot \widehat{\pi}_{t+1}$ denotes the inner product multiplication at every possible value of the random vector $\widehat{\pi}_{t+1}$ and $\eta > 0$ is the lower estimate of the curvature of the return function, as specified in Assumption 2. Observe that condition (3.8) places an upper bound on the exponential growth factor of the derivative $\widehat{\pi}_t = D_1 g^t(k_0, z_0) \cdot \pi_0$. Indeed, (3.8) implies that

$$\sum_{t=0}^{\infty} \beta^t \int_{Z^t} \left\|D_1 g^{t+1}(k_0, z_0)\right\|^2 \mu(z_0, \mathrm{d}z^t) \leqslant \frac{L}{\eta}. \tag{3.9}$$

On the other hand, differentiation of $D_2 W(k_0, z_0)$ in (3.4) with respect to k_0 yields

$$\begin{aligned} D_{12} W(k_0, z_0)^T &= D_{21} W(k_0, z_0) \\ &= \sum_{t=0}^{\infty} \beta^t \int_{Z^t} \left[\left(\frac{\partial z_t}{\partial z_0}\right) \cdot \left(D_{31} v(k_t, k_{t+1}, z_t) \cdot D_1 g^t(k_0, z_0)\right.\right. \\ &\qquad \left.\left. + D_{32} v(k_t, k_{t+1}, z_t) \cdot D_1 g^{t+1}(k_0, z_0)\right)\right] \mu^t(z_0, \mathrm{d}z^t). \end{aligned}$$

Now, taking matrix norms we have

$$\begin{aligned} \left\|D_{12} W(k_0, z_0)^T\right\| &= \left\|D_{21} W(k_0, z_0)\right\| \\ &\leqslant \sum_{t=0}^{\infty} \beta^t \int_{Z^t} \left\|\frac{\partial z_t}{\partial z_0}\right\| \left[\left\|D_{31} v(k_t, k_{t+1}, z_t)\right\| \left\|D_1 g^t(k_0, z_0)\right\|\right. \\ &\qquad \left. + \left\|D_{32} v(k_t, k_{t+1}, z_t)\right\| \left\|D_1 g^{t+1}(k_0, z_0)\right\|\right] \mu^t(z_0, \mathrm{d}z^t) \\ &\leqslant \left[1 + \frac{2C(L/\eta)^{1/2}}{1 - \beta^{1/2}\rho}\right] G, \end{aligned} \tag{3.10}$$

where the last inequality follows from Assumption 4 and condition (3.9) for $G = \sup\{\|D_{31} v\|, \|D_{32} v\|\}$. Similar upper bounds can be obtained for $\|D_2 g\|$ and $\|D_{22} W\|$. Note from these computations that in general variable z will have a more pronounced effect on the second-order derivatives of the value function.

Much less is known about higher-order differentiability properties of the value and policy functions. In simple deterministic models, regular examples can be constructed where the value function is C^2 but fails to be C^3 [cf. Araujo (1991) and Santos

(1994)]. This lack of differentiability is closely related to certain non-independence (or *resonance*) conditions of the eigenvalues at a given unstable steady state. (For a pair of eigenvalues, λ_1 and λ_2, the resonance conditions amount to $\lambda_1^n = \lambda_2$ for some integer $n > 0$.) Such conditions are nevertheless pathological (i.e., not robust to small perturbations of the model), although further sources of non-differentiability may arise in multivariate models with complex behavior, or in stochastic frameworks.

In conclusion, there exists a reasonably general theory for first- and second-order derivatives of the value function, and such derivatives may be bounded from primitive elements of the model. Relatively little is known about existence and characterization of higher-order derivatives[4].

4. A numerical dynamic programming algorithm

Our purpose now is to formulate a numerical algorithm for computing functions W and g, and to study its accuracy properties. In the actual implementation of the algorithm, there are additional errors stemming from the numerical maximization and integration of these functions, and these errors may unfold over the iterative process. We shall present here a stability analysis of these approximations, along with a discussion of available methods for numerical maximization and integration. In subsequent sections we shall be concerned with possible extensions of this basic algorithm to speed up computations, and with its numerical implementation.

4.1. Formulation of the numerical algorithm

Our numerical algorithm is based on a discretized version of the method of successive approximations. The basic idea underlying this analysis is to restrict the set of functions $\mathcal{W}$ to a finite-dimensional domain so that the algorithm can be coded in a finite number of computer instructions. Our functions will be defined on the whole domain $K \times Z$ via piecewise affine interpolation of their values over a finite set of regular points. In the following section, we shall consider alternative interpolation schemes.

In the sequel we assume that the state space $S = K \times Z$ is a polyhedron. This does not entail much loss of generality for most economic applications. Let $\{S^j\}$ be a finite

[4] A thoughtful method to compute high-order derivatives of the value and policy functions is laid out in Judd (1996) and Gaspar and Judd (1997). This method is only operative for steady-state solutions, assuming the existence of such derivatives.

family of simplices[5] such that $\cup_j S^j = S$ and $\text{int}(S^i) \cap \text{int}(S^j) = \phi$ for every pair S^i, S^j. Define the *grid level* or *mesh size* as

$$h = \max_j \ \text{diam}\{S^j\}.$$

Let (k^j, z^j) be a generic vertex of the triangulation. Consider then the space of piecewise affine functions

$$\mathcal{W}^h = \left\{ V^h : S \to R \;\middle|\; \begin{array}{l} V^h \text{ is bounded, continuous and } DV^h \text{ is constant} \\ \text{in int}\,(S^j) \text{ for each } S^j \end{array} \right\}.$$

It follows that every function V^h in $\mathcal{W}^h$ is determined by the nodal values $V^h(k^j, z^j)$, for all vertex points (k^j, z^j). These nodal values uniquely define a piecewise affine function over the whole domain S, compatible with the given triangulation $\{S^j\}$. Also, $\mathcal{W}^h$ is a closed subspace of $\mathcal{W}$, equipped with the norm

$$\|V^h\| = \max_{(k,z) \in K \times Z} |V^h(k,z)| \quad \text{for} \quad V^h \in \mathcal{W}^h.$$

For a given triangulation $\{S^j\}$ with mesh size h, we now consider the following algorithm for *value function iteration*:

(i) *Initial step*: Select an accuracy level *TOLW* and an initial guess W_0^h.

(ii) *Value function evaluation*: Let

$$W_{n+1}^h(k_0^j, z_0^j) = \max_{k_1} \upsilon(k_0^j, k_1, z_0^j) + \beta \int_Z W_n^h(k_1, z_1)\, Q(z_0^j, \mathrm{d}z_1) \qquad (4.1)$$
$$\text{subject to } (k_0^j, k_1, z_0^j) \in \Omega \text{ for each vertex point } (k_0^j, z_0^j).$$

(iii) *End of iteration*: If $\|W_{n+1}^h - W_n^h\| \leqslant TOLW$, stop; else, increment n by 1, and return to step (ii).

For present purposes, it should be understood that the maximization and integration operations in algorithm (4.1) are performed exactly. As a matter of fact, in the actual implementation of the algorithm, one solves for the integral on the right-hand side of Equation (4.1), and then maximizes over k_1. These functional evaluations define a mapping $T^h : \mathcal{W} \to \mathcal{W}^h$ such that $W_{n+1}^h = T^h(W_n^h)$. The mapping T^h is a discretized version of the dynamic programming algorithm.

[5] A simplex S^j in R^l is the set of all convex combinations of $l+1$ given points. [cf. Rockafellar (1970)]. Thus, a simplex in R^1 is an interval, a simplex in R^2 is a triangle, and a simplex in R^3 is a tetrahedron.

Accuracy level *TOLW* is selected so as to provide a good approximation W^h_{n+1} to the fixed point W^h of the following functional equation

$$W^h(k_0^j, z_0^j) = \max_{k_1} \upsilon(k_0^j, k_1, z_0^j) + \beta \int_Z W^h(k_1, z_1)\, Q(z_0^j, \mathrm{d}z_1)$$
$$\text{subject to } (k_0^j, k_1, z_0^j) \in \Omega \text{ for each vertex point } (k_0^j, z_0^j). \tag{4.2}$$

This is the corresponding discretized version of Bellman's Equation (3.1), which is required to hold at the finite set of vertex points. Using the above iterative scheme, we now establish existence of the fixed point W^h and provide estimates for the approximation error involved in this discretization.

4.2. Existence of numerical solutions and derivation of error bounds

A basic topic in numerical analysis is existence and accuracy properties of numerical solutions. Relatively little is known, however, about the existence and accuracy properties of most numerical algorithms used by economists. Li (1993) and Marcet and Marshall (1994) are notable exceptions to this trend. Li provides a derivation of error bounds for a simple monetary model. Marcet and Marshall prove certain asymptotic properties of a parameterized expectations algorithm. Although Marcet and Marshall's results are comforting, these results cannot be generally invoked in numerical computations since they simply assert that the approximation scheme converges to the exact solution as the computational cost goes to infinity.

We now review some accuracy properties of our numerical model[6].

Lemma 4.1. *Under Assumptions (1)–(4), Equation (4.2) has a unique solution* W^h *in* $\mathcal{W}^h$.

Proof: The proof is the standard one. One immediately sees that T^h is a contraction mapping with modulus $0 < \beta < 1$. By a well known fixed-point theorem, equation (4.2) has a unique fixed point W^h in $\mathcal{W}^h$. □

It is worth noticing that the contraction property of the operator T^h implies that the sequence $\{W_n^h\}_{n\geq 0}$ generated by Equation (4.1) converges linearly to the fixed point W^h.

Lemma 4.2. *Let* W *be the value function in Equation (3.1). Let* $\gamma = \|D^2 W\|$. *Then, under Assumptions (1)–(4) it must hold that* $\|TW - T^h W\| \leq \frac{\gamma}{2} h^2$.

[6] The present analysis is taken from Santos and Vigo (1998). One can find related discretization procedures and derivation of error bounds for the computation of the value function [e.g. Bertsekas (1975), Chow and Tsitsiklis (1991), Falcone (1987), Fox (1973), Kitanidis and Foufoula-Georgiou (1987), and Whitt (1978, 1979)]. None of these latter papers, however, derives a quadratic order of convergence for the computed value function in an infinite-horizon optimization setting (as h goes to zero).

Observe that at each nodal value $T^h(W)(k^j, z^j) = W(k^j, z^j)$; moreover, $T^h W$ is piecewise affine, and function W is C^2. Then, the result follows from a standard application of Taylor's theorem [see Santos and Vigo (1998) for further details].

Theorem 4.3. *Let W be the fixed point of Equation (3.1) and W^h be the fixed point of Equation (4.2). Then, under Assumptions (1)–(4) it must hold that* $\|W - W^h\| \leqslant \frac{\gamma}{2(1-\beta)} h^2$.

Proof: Let T and T^h be as defined previously from Equations (3.2) and (4.1), respectively. We must then have,

$$\begin{aligned}\|W - W^h\| &= \|TW - T^h W^h\| \\ &\leqslant \|TW - T^h W\| + \|T^h W - T^h W^h\| \\ &\leqslant \|TW - T^h W\| + \beta \|W - W^h\|,\end{aligned}$$

where use is made in these computations of the triangle inequality and of Lemma 4.1. Therefore,

$$\|W - W^h\| \leqslant \frac{1}{1-\beta} \|TW - T^h W\|$$

Theorem 4.3 is now a direct consequence of Lemma 4.2. □

Inequalities (3.7)-(3.10) provide upper estimates for parameter $\gamma = \|D^2 W\|$. These estimates can be useful to bound the observed error in specific applications. It should be noted that constant $\gamma/2(1-\beta)$ becomes unbounded for $\beta = 1$. This singularity seems to be related to the fact that if the value of the approximation error in a single period may be up to $(\gamma/2)h^2$ (Lemma 4.2), then the cumulative error over the entire infinite horizon may extend up to $[\gamma/2(1-\beta)]h^2$.

An immediate consequence of Theorem 4.2 is the following useful result:

Corollary 4.4. *Let $g(k^j, z^j)$ be the optimal policy for the original value function W at a vertex point (k^j, z^j), and let $g^h(k^j, z^j)$ be the optimal policy for the approximate value function W^h at vertex point (k^j, z^j). Then,*

$$\|g(k^j, z^j) - g^h(k^j, z^j)\| \leqslant \left(\frac{2\gamma}{\eta(1-\beta)}\right)^{1/2} h \quad \textit{for all } (k^j, z^j).$$

It follows then from this analysis that for the computed value function the order of convergence is quadratic on h, whereas for the computed policy function the order of convergence is linear. As it is to be expected, a key assumption for the linear convergence of the optimal policy is a positive lower estimate η on the concavity of the return function as postulated in Assumption 2.

4.3. *Stability of the numerical algorithm*

Our results indicate that the numerical model has some desired accuracy properties. In particular applications, however, these computations are subject to round-off errors and inaccuracies due to numerical maximizations and integrations. These numerical errors may get propagated in unexpected ways over the iteration process. A numerical algorithm is said to be *unstable* if small computational errors produce large variations in the solution. Unstable algorithms should be avoided.

We now show that our algorithm has some desired stability properties. For a given triangulation $\{S^j\}$ with mesh size h, define a functional operator $T_\varepsilon^h : \mathcal{W} \to \mathcal{W}^h$ such that $|T_\varepsilon^h(V)(k^j, z^j) - T^h(V)(k^j, z^j)| \leqslant \varepsilon$, for all V in $\mathcal{W}$ and all vertex points (k^j, z^j), for fixed $\varepsilon > 0$. The interpretation is that under the operator T_ε the computational error is not greater than ε for all functions in the space $\mathcal{W}$. If such distance is preserved for all nodal values $V(k^j, z^j)$, it follows that the constructed piecewise linear interpolations of $\{T_\varepsilon^h(V)(k^j, z^j)\}$ and $\{T^h(V)(k^j, z^j)\}$ are also within an ε-distance over the whole domain S. In accordance with this interpretation, we postulate the following regularity conditions on functional operator T_ε^h:

(i) *Monotonicity*: $T_\varepsilon^h V' \geqslant T_\varepsilon^h V$ for $V' \geqslant V$;

(ii) *Discounting*: $T_\varepsilon^h(V + a) \leqslant T_\varepsilon^h V + \beta a$ for every V and every constant function a.

These properties are preserved under most standard numerical maximization and integration procedures. That is, condition (i) will hold if the maximization and integration schemes preserve monotonicity, and condition (ii) entails that adding a constant function to the maximizations and integrations results in an equivalent change for the corresponding solution to both operations. (Of course, inequality (ii) may be problematic in rounding off very small numbers.)

Under conditions (i) and (ii), functional operator T_ε^h is a contraction mapping on $\mathcal{W}$ with modulus β (cf. Lemma 4.1). Our next result bounds the distance between the fixed points of operators T^h and T_ε^h:

Theorem 4.5. *Let W be the fixed point of T, let W^h be the fixed point of T^h, and let W_ε^h be the fixed point of T_ε^h. Assume that T_ε^h satisfies conditions (i) and (ii). Then, under Assumptions (1)–(4), we have:*

$$(1)\ \|W^h - W_\varepsilon^h\| \leqslant \frac{\varepsilon}{1-\beta}; \qquad (2)\ \|W - W_\varepsilon^h\| \leqslant \frac{\gamma}{2(1-\beta)}h^2 + \frac{\varepsilon}{1-\beta}.$$

Part (1) can be established from the method of proof of Theorem 4.3. Part (2) is a consequence of the triangle inequality, using part (1) and Theorem 4.3. Again, the intuition for part (1) is that if $\varepsilon > 0$ is the possible computational error in each iteration, then $\varepsilon/(1-\beta)$ should be an upper bound for the cumulative error over the entire infinite horizon. Indeed, this estimate can be obtained from the recursion

$$|(\text{cumulative error})_t| \leqslant |(\text{current error})_t| + \beta\,|(\text{cumulative error})_{t-1}|.$$

The bounds established in Theorem 4.5 can be useful for an efficient design of the computer code. Thus, it would not be optimal to operate with a very fine grid

of points in cases where the approximation errors from maximization and integration are relatively large. An efficient implementation of the algorithm requires that these errors should be balanced. That is, $h^2/\varepsilon \cong 2/\gamma$. This benchmark value may be adjusted depending upon the computational cost of reducing each of these errors, and on further operational features of the algorithm discussed in subsequent sections.

Of course, the computational error ε stems from the maximization and integration approximations, and these individual errors should also be balanced. Routines for maximization and integration usually provide good estimates for these approximations. Moreover, it should be realized that if the maximization is carried out over a discrete set of grid points $\{k^j\}$ with mesh size h, then the additional error involved in this approximation is of order h^2, since the first-order derivative at a maximizer is equal to zero. On the other hand, if the integration is performed over a discretized space, as an approximation for an underlying continuous-valued random variable, then the additional error will depend on the integration scheme and the differentiability properties of the value function. [Observe that in general variable z has a more pronounced effect on the derivatives of the value function, cf. Equations (3.4) and (3.10).] Thus, one should make reasonable choices concerning discretizations of state spaces K and Z so that the involved approximation errors are of the same magnitude. It seems that commonly found computations which restrict the uncertainty space Z to very few states as compared to the space of capitals K [e.g., Christiano (1990) and Danthine, Donaldson and Mehra (1989)] may obtain more accurate approximations for the same computational cost by considering more balanced grids over the whole space $K \times Z$.

4.4. Numerical maximization

Methods for numerical optimization are covered in Gill, Murray and Wright (1981), Kahaner, Moler and Nash (1989), and Press et al. (1992). In addition, there are several professionally designed subroutines for the solution of various specialized problems, as well as the two all-purpose libraries NAG and IMSL. Here, we shall offer a brief introduction to these methods along with a discussion of some specific issues concerning the analysis of error and implementation of these numerical procedures.

As in classical mathematical analysis, a distinction is made in numerical maximization between smooth and non-smooth problems, global and local optimization, constrained and unconstrained solutions, and maximization in one and several dimensions. These properties not only dictate the nature of the techniques employed to tackle the optimization, but they also bear on practical computational considerations. Thus, a method for numerical maximization of non-smooth functions is generally inefficient for smooth problems. Likewise, a numerical method for maximization in several variables will not generally be suitable for one-dimensional problems.

Algorithms for numerical maximization of smooth functions usually search over the whole domain of feasible solutions, as opposed to restricting the search to a grid of prespecified points. The software usually indicates the tolerance level or *interval of*

uncertainty. Since at an interior maximum point the derivative of the function is equal to zero, it should be understood that if the computed maximum is at an ε-distance, then the approximation error in the functional value is of order ε^2. If the search for the maximum is restricted to a grid of prespecified points with mesh size h [e.g., Christiano (1990)], then the additional error incurred in this approximation would be of order h^2.

Although the search for a maximum over a grid of points is a very simple strategy associated with a relatively small approximation error, methods based on functional evaluations are not generally computationally efficient for smooth problems. There are faster, more powerful algorithms that take advantage of the information provided by the derivatives of the functions. In our case, our mappings are piecewise linear, and hence the gradient is defined at almost every point. Moreover, the curvature of these mappings can be bounded in a certain sense, since the first-order derivatives are determined by the envelope theorem [cf. equality (3.3)], and an upper estimate of the rate of change of these derivatives is the maximum value of the second-order derivative of the return function [cf. inequality (3.7) above, or Montrucchio (1987) for more general arguments]. Hence, our functions possess some smoothness properties, and have bounded curvature. Then, numerical maximization methods based on simple functional evaluations would generally be inefficient. Of course, smoothness can be obtained under higher-order interpolations; and if these interpolations preserve concavity, the optimization problem may be more tractable.

Another important consideration is the dimensionality of the optimization problem. Numerical methods for maximization in several dimensions are generally based on *one-line* maximizations. The choice of these directions is key for defining the search method. This choice, however, is trivial for unidimensional problems.

For univariate maximization, a typical initial step is to *bracket the maximum*. That is, in some very simple way one selects a triplet of points $a < b < c$ such that $f(b)$ is greater than both $f(a)$ and $f(c)$. (This choice guarantees the existence of a maximum inside the chosen interval; moreover, if the objective is concave such solution is the desired global maximum.) Once the maximum has been bracketed, then the searching process should exploit regularity properties of the function, using either smooth approximations or functional evaluations. There are also hybrid procedures that combine both types of information. This is the strategy followed by *Brent's* method [cf. Press et al. (1992), Sect. 10.2], and it seems suitable to our case where the functions have kinks at the vertex points, but at the same time preserve certain smoothness properties. The method proceeds along the following steps:

(a) *Smooth approximation*. The routine selects three given function values, and constructs a parabolic approximation. Then it quickly determines the maximum of the parabola. If this maximum point falls within certain limits (i.e., the maximum is *cooperative*), then this value is added in the next iteration for a subsequent parabolic approximation, until a desired level of accuracy is achieved. Convergence to the true maximum is of order 1.324.

(b) *Golden-section search.* If the parabolic approximation is not a reasonable one, then the routine switches to a more reliable but slower method called golden-section search. This procedure is analogous to the familiar method of *bisection* for finding the zeroes of a univariate function. Given at each stage a bracketing triplet of points, golden-section search tries a point that is a fraction 0.38197 into the largest of the two intervals from the central point of the triplet. With the four points now available, the procedure then selects a new bracketing triplet. Following this iterative process, the interval of search is eventually reduced at each stage by $1 - 0.38197 = 0.61803$, which corresponds to the rate of convergence of this method.

Brent's method falls into the class of so-called *safeguarded* procedures, which combine fast algorithms with slower, more reliable ones. Press et al. (1992, Sect. 10.3) discuss another method of this nature which seems appropriate for univariate concave optimization. The method proceeds as follows. Given a bracketing triplet of points $a < b < c$, one determines the direction of the derivative at the intermediate point, b. This information then defines the next interval of search, which would be either $[a, b]$ or $[b, c]$. The value of the derivatives of the two chosen points can then be used to produce another intermediate point by some root finding procedure such as the *secant* method. If this method yields values beyond certain limits, then one bisects the interval under consideration. Of course, in order to implement this latter safeguarded procedure, concavity and smoothness properties of the univariate problem are essential. In the unidimensional case, concavity is always preserved by piecewise linear interpolations. As for differentiability, one could compute for instance one-side derivatives, or else resort to higher-order interpolations preserving first-order differentiability.

Regarding multivariate optimization, there are also a host of algorithms, the usefulness of which will depend on the dimensionality, smoothness and concavity properties of the optimization problem. In recent years, there has been a considerable amount of attention devoted to numerical procedures on non-smooth optimization for both concave and non-concave objectives [e.g., see Bazaraa et al. (1993, Ch. 8), Hiriart-Urruti and Lemarechal (1993) and Shor (1985)]. A simple algorithm in this class of non-differentiable problems is the *downhill simplex method* (also called the *polytope algorithm).* For smooth optimization, two popular procedures are *quasi-Newton* and *conjugate-gradient* methods. These latter two methods can also be applied to non-smooth problems, making use of finite-difference approximations for the first-order derivatives.

In an n-dimensional space, the downhill simplex method considers $n + 1$ functional evaluations, say at points $x_0, x_1, \ldots, x_n$. These points can be visualized as the vertices of a simplex or polytope. Then, in the next step a new simplex is constructed by producing a new vertex that will replace the point with the worst functional evaluation. Depending on the new value, the polytope may further expand or contract in that direction. A new iteration then starts by replacing the worst point, and this iterative process goes on until a desired solution is attained. Under this procedure, the search for a maximum is not

usually guaranteed, but this seems to be a convenient way to find a maximum in cases where one cannot approximate the derivatives of the function.

Quasi-Newton methods derive estimates of the curvature of the function, without explicitly computing the second-order derivatives. Thus, each iteration starts at a point x_k with a matrix B_k which reflects second-order information, and which is supposed to be an approximation of the true Hessian if the function is sufficiently smooth. (At the initial stage one usually starts with B_0 equal to the identity matrix, and in such case the algorithm reduces to the *steepest descent* method.) Then, the *search direction*, p_k, is the solution to

$$B_k \cdot p_k = -g_k,$$

where g_k is the gradient vector. Subsequently, maximization is carried out in this direction, that is, on the line $x_k + \alpha p_k$. A choice of a number $\widehat{\alpha}$ defines a new point $x_{k+1} = x_k + \widehat{\alpha} p_k$, and completes the iteration. The Hessian estimate B_{k+1} is then updated following some standard methods [cf. Press et al. (1992), p. 420]. The whole process stops when the gradient g_k is sufficiently small.

Conjugate-gradient methods construct a sequence of searching directions which satisfy certain orthogonality and conjugacy conditions so as to improve at each step the search for a maximum. Conjugate-gradient methods do not require estimates or knowledge of the Hessian matrix. Hence, their applicability extends more naturally to large-scale problems.

4.5. Numerical integration

The integral operation in Equation (4.1) can be effected by standard numerical procedures. In general, numerical integration can be easily carried out in one dimension, and it becomes costly in several dimensions. Professionally designed software usually provides an estimate of the approximation error or tolerance, which in most cases can be adjusted.

An n-point *quadrature* formula, $\sum_{i=1}^{n} w_i f(x_i)$, is an estimate of a given integral, $I = \int_a^b f(x)\,\mathrm{d}x$, where the w_i and x_i are called *weights* and *nodes*. These values depend on a, b and n, but not on f. The difference $R_n = \int_a^b f(x)\,\mathrm{d}x - \sum_{i=1}^{n} w_i f(x_i)$ is called the *remainder* or error.

There are several well known quadrature rules, which under certain regularity conditions yield bounds for the approximation error. The *mid-point* rule takes $w = b - a$, and $x = \frac{1}{2}(b + a)$. The *trapezoidal* rule has weights $w_1 = w_2 = \frac{1}{2}(b - a)$, and nodes $x_1 = a, x_2 = b$. Each of these rules can be *compounded*. For example, let us divide the interval $[a, b]$ into N equally-sized panels, and let S_N be the integral value by applying the trapezoidal rule to each of these panels. Then, computations for S_N can be useful to calculate S_{2N}. The *three-point compounded Simpson's* rule can be defined as $S^N = \frac{4}{3}S_{2N} - \frac{1}{3}S_N$, and for sufficiently smooth functions the approximation

error under this latter rule is "fourth order" [cf. Kahaner et al. (1989) and Press et al. (1992)].

Compounding allows to attain higher accuracy using previous functional evaluations. Compounding is also very convenient to track numerically the approximation error. Theoretical error bounds are usually too conservative. Further, a quadrature rule is not useful unless there is some way to estimate the remainder R_n [Kahaner et al. (1989), p. 150].

Another basic family of integration rules is *Gaussian quadrature.* Here, the weights and the nodes are freely selected so that certain integrands can be more effectively approximated. *An n-point Gaussian quadrature* rule can integrate exactly every polynomial up to degree $2n - 1$; integration of polynomials of higher degree would generally entail an approximation error. Hence, Gaussian quadrature is very efficient for the integration of smooth, polynomial-like functions, but may not have a good performance for other types of integrands. More generally, Gaussian quadratures are constructed so as to integrate exactly polynomials times some weighting function, $\rho(x)$; that is, weights w_i and nodes x_i are chosen to satisfy $\int_a^b \rho(x)p(x)\,dx = \sum_{i=1}^n w_i p(x_i)$, where $p(x)$ is a polynomial. For the particular choice $\rho(x) = 1/\sqrt{1-x^2}$, the rule is termed *Gauss–Chebyshev integration,* and for $\rho(x) = 1$ the rule is termed *Gauss–Legendre integration.*

Compounding is not possible for Gaussian rules, since the nodes of an n-point rule are distinct from those of an m-point rule. (Only, if n and m are odd, the rules will have the mid-point in common.) There are, however, ways to estimate the error for Gaussian quadratures. Let $G_n = \sum_{i=1}^n w_i f(x_i)$ be an n-quadrature of polynomial degree $2n - 1$. Then, define

$$K_{2n+1} = \sum_{i=1}^{n} a_i f(x_i) + \sum_{j=1}^{n+1} b_j f(y_j).$$

Here, K_{2n+1} has $n + 1$ additional nodes, and different coefficients for all a_i, b_j. These values can be specified so that K_{2n+1} is of polynomial degree $3n + 1$. The two rules (G_n, K_{2n+1}) form a *Gauss–Kondrod* pair. The difference $|G_n - K_{2n+1}|$ is generally a fairly pessimistic error bound for the integral estimate K_{2n+1}. Gauss–Kondrod quadrature rules are usually regarded as very efficient methods for calculating integrals of smooth functions.

For double integrals, $\int_a^b \int_c^d f(x,y)\,dy\,dx$, an obvious procedure is to solve iteratively one-dimensional integrals. This is called a *product rule.* Here, the approximation error can be bounded from the estimates of the one-dimensional quadratures. In some cases, especially for integrals over many dimensions, it is optimal to acknowledge the multidimensional nature of the approximation problem, and resort to a more direct integration rule. This would be called a *non-product* rule. Sometimes, suitable transformations can be made to facilitate computations, but numerical integration of multiple integrals may become rather costly or infeasible. [Davis and Rabinowitz

(1984) and Stroud (1972) are still useful references.] An alternative route is *Monte-Carlo* integration, which is much easier to implement.

Monte-Carlo methods approximate the value of the integral from an average of functional evaluations over random points. Now, the error is actually a stochastic variable, and hence one can make some probabilistic inferences. Thus, if N is the number of sampled points, then the expected error goes to zero at a rate $N^{-1/2}$. This result holds under certain mild conditions on the integrand, without reference to the dimensionality.

Over the past three decades, there has been active research to improve these estimates of Monte-Carlo integration, using "quasi-random" methods or "low discrepancy points". The general idea of these deterministic methods (with some randomized extensions) is to sample more carefully the regions of integration, so that the error may exhibit on a worst-case basis a convergence rate close to N^{-1} [e.g., see Geweke (1996), Niederreiter (1992) and Press et al. (1992) for an introductory account to this theory, and Papageorgiou and Traub (1996), Paskov (1996), and Tan and Boyle (1997) for some numerical evaluations]. These latter results usually require some sort of Lipschitz continuity, and the constants involved in the orders of convergence may depend on the dimensionality of the domain.

5. Extensions of the basic algorithm

In this section we introduce some variations of the preceding algorithm which may accelerate the computation of the value and policy functions. There are two natural ways to modify the original method of successive approximations: (a) Introducing an alternative iteration process, or (b) appending a different interpolation procedure, under the same iteration structure. The accuracy properties of algorithms in class (a) remain unchanged, even though one may obtain considerable savings in computational time. The idea is that the method of successive approximations is slow, and there are other possible ways to compute the fixed point. The accuracy properties of algorithms in class (b) may change, although it should be recalled that higher-order approximants do not always yield higher accuracy, and these approximations may be more costly to implement.

A common feature of these extensions is that they have not been extensively used and tested in the economic literature; but some of them seem to be of potential interest, and may be quite useful in particular applications.

5.1. Multigrid methods

Multigrid methods have been widely applied for solving partial differential equations [cf. Stoer and Bulirsch (1993)]. In a natural way, these methods have been proposed by several authors for the solution of dynamic programs. Chow and Tsitsiklis (1991) have argued that the complexity of a multigrid algorithm is, in a certain sense, optimal.

Also, Santos and Vigo (1995) implement this algorithm for computing solutions of economic growth problems, and report substantial speedups with respect to the single grid method, especially for cases with fine grids or with high discount factors.

To motivate this approach, assume that in computing the value function the desired precision parameter h has been fixed, and there is not a good initial guess W_0 to start the process. If W^h is the fixed point of T^h, and W is the fixed point of T, then it follows from the contraction property of these operators that in the first iteration the approximation error is bounded by

$$\beta \left\| W^h - W_0 \right\| + Mh^2. \tag{5.1}$$

Likewise, in the nth iteration the approximation error is bounded by

$$\beta^n \left\| W^h - W_0 \right\| + Mh^2. \tag{5.2}$$

In these calculations, the distance between W_0 and the value function W has been decomposed in the following way. The term $\beta^n \left\| W^h - W_0 \right\|$ bounds the distance between the fixed point W^h and the nth iteration of W_0 under T^h for $n \geqslant 1$, whereas Mh^2 bounds the distance $\left\| W - W^h \right\|$ for $M = \gamma/[2(1-\beta)]$. This second component is fixed on h, whereas the first one decreases by the factor $0 < \beta < 1$. Hence, if our original guess is not close to W, initial reductions in the approximation error are relatively large (in absolute value) as compared to the second term Mh^2. Since these gains are basically related to the distance $\left\| W^h - W_0 \right\|$ and the contraction property of the operator T^h, a coarser grid may lead to similar results, even though it entails less computational cost. Thus, it may be beneficial to consider coarser grids in the early stages of the computational process.

Formally, the multigrid method proceeds as follows. Let $\left(\left\{S^{h_i}\right\}\right)_i$ be a sequence of triangulations, for $i = 0, 1, \ldots, n$. Let h_i be the mesh size of triangulation $\left\{S^{h_i}\right\}$. Suppose that $h_0 > h_1 > \cdots > h_i > \cdots > h_n$. Then, take an arbitrary initial guess W_0 and implement the above iterated method (4.1) under the coarsest partition $\left\{S^{h_0}\right\}$, to obtain a fixed point W^{h_0}. Next, choose W^{h_0} as the initial condition for computing W^{h_1}. And follow the same procedure for subsequent finer grids: Pick $W^{h_{i-1}}$ as the initial choice for computing W^{h_i}, for $i = 1, 2, 3, \ldots, n$.

For the same level of accuracy, this method may reduce the computational burden, since early computations are performed over coarser grids with less computational cost. Likewise, one can generally obtain reasonable initial guesses of the value and policy functions at a very low cost from numerical computations over coarse grids.

Our previous error analysis becomes useful for an optimal implementation of this method. As discussed in the preceding sections, inaccuracies of maximization and integration operations must be of order of magnitude h_i^2, where h_i is the grid level of triangulation $\left\{S^{h_i}\right\}$. Hence, in early stages of the iteration process these subroutines can be set up at a lower tolerance level to be effected more quickly. Furthermore, from early maximizations one can derive good guesses, so that the search for the maximum

can be restricted to smaller regions while proceeding with finer partitions. Another crucial issue is the timing of the grid change. A coarser grid is more cheaply effected, but the gain in the approximation error is also smaller; i.e., the right-hand term of expression (5.2) is larger, and the contraction property of the operator does not apply to that term. For an optimal grid change, it seems that an appropriate benchmark value is the gain achieved per work expended. Let us then estimate the approximation errors in expression (5.2), and then proceed with the corresponding calculations for a specific illustration. A good bound for the first term of expression (5.2) can be obtained from the contraction property of our operators; indeed,

$$\left\| W^h - W_n^h \right\| \leqslant \frac{\left\| W_{n+1}^h - W_n^h \right\|}{1-\beta}.$$

As for the term Mh^2, we need an estimate of the second-order derivative of the value function. This estimate may be calculated from our theoretical analysis of Section 3, although more operational bounds are usually obtained from computational experiments (cf. Section 7).

To discuss the issue under concern in a more precise manner, assume then that there is but one state variable, and the grid sizes $(h_i)_{i=0}^n$ follow the relation $h_i = h_0/2^i$, $i = 1, 2, \ldots, n$. Then, moving from mesh size h_i to mesh size h_{i+1} will roughly double the computational cost in each iteration, since there are twice as many vertex points. On the other hand, the additional gain in the approximation error is determined by the second term of expression (5.2), and for a grid change this term is estimated to be $\frac{3}{4}Mh_i^2$, since convergence is quadratic. Therefore, to benefit from the grid change, the gain in the approximation error must double, and this happens when the ratio $\left\| W^h - W_n^h \right\| / Mh^2 \simeq \frac{3}{4}$. For multidimensional problems, the optimal ratio would be smaller, as the computational cost increases exponentially.

5.2. *Policy iteration*[7]

The method of policy iteration is credited to Bellman (1955, 1957) and Howard (1960). [see Puterman and Brumelle (1979) for some early history.] In economics, the method has been successfully applied to some optimization problems involving large discrete state spaces [e.g., Rust (1987)]. There are, however, certain unresolved issues concerning the implementation of this procedure. From a theoretical point of view, the iterative scheme exhibits locally a quadratic convergence rate, and it has not been established an optimal radius or region of convergence. On the other hand, further computational work is needed to assess its performance in a wide range of economic applications.

In the context of our previous discretization procedure, for a given triangulation $\{S^j\}$ with mesh size h, we now consider the following iterative scheme for *policy iteration:*

[7] This presentation draws upon unpublished work of the author with John Rust.

(i) *Initial step:* Select an accuracy level *TOLW* and an initial guess W_0^h.

(ii) *Policy improvement step:* Find $g_n^h(k_0^j, z_0^j)$ that solves

$$B(W_n^h)(k_0^j, z_0^j) = -W_n^h(k_0^j, z_0^j) + \max_{k_1} \upsilon(k_0^j, k_1, z_0^j) + \beta \int_Z W_n^h(k_1, z')\, Q(z^j, \mathrm{d}z') \tag{5.3}$$

for each vertex point (k_0^j, z_0^j).

(iii) *Policy evaluation step:* Find $W_{n+1}^h(k_0^j, z_0^j)$ satisfying

$$W_{n+1}^h(k_0^j, z_0^j) = \upsilon(k_0^j, g_n^h(k_0^j, z_0^j), z^j) + \beta \int_Z W_{n+1}^h(g_n^h(k_0^j, z_0^j), z')\, Q(z^j, \mathrm{d}z') \tag{5.4}$$

for each vertex point (k_0^j, z_0^j).

(iv) *End of iteration:* If $\left\| W_{n+1}^h - W_n^h \right\| \leqslant TOLW$, stop; else, increment n by 1, and return to step (ii).

It should be understood that all functions derived from these iterations are piecewise linear, and compatible with the given triangulation $\{S^j\}$. Thus, B maps the space of continuous functions, $\mathcal{W}$, into the space of piecewise linear functions, $\mathcal{W}^h$. Of course, the fixed point of Equations (5.3)–(5.4) corresponds to the fixed point of Equation (4.2), and the existence of such a unique solution W^h has been established in Lemma 4.1. Therefore, the error bounds derived in Theorem 4.3 and Corollary 4.4 apply here.

Observe that step (ii) corresponds to a single iteration of the method of successive approximations. The innovation of policy iteration is in step (iii). For continuous random variables this step is generally non-trivial, since it involves the computation of a function W_n^h, which appears on both sides, and under an integral sign. In such situations, in order to facilitate calculations one may resort to a complete discretization of the space Z, with a further loss of accuracy[8].

To be more specific about step (iii), let us consider a return function $\upsilon(k_t, k_{t+1})$, with k_t, k_{t+1} in R_+. Assume that $\{k^i\}_{i=1}^N$ are the grid points. Then, for a piecewise linear function g_n^h, Equation (5.4) can be written as follows:

$$\upsilon_{g_n^h} = [I - \beta P_{g_n^h}] W_{n+1}^h, \tag{5.5}$$

where $\upsilon_{g_n^h}$ is an N-dimensional vector with elements $\upsilon(k^i, g_n^h(k^i))$, $i = 1, \ldots, N$; on the right-hand side, I is the $N \times N$ identity matrix, and $P_{g_n^h}$ is an $N \times N$ matrix generated by policy g_n^h in the following way: If $k_1 = g_n^h(k^i)$, for $i = 1, \ldots, N$, and $k_1 = \lambda k^{j-1} + (1 - \lambda) k^j$ for some j, then the ith row $P_{g_n^h}$ would be all zeroes, except for the $(j-1)$th and jth entries, which are equal to λ and $(1 - \lambda)$, respectively. As a result

[8] Technically, this is a Fredholm equation of the second kind. A natural approach for solving this problem is to use numerical integration and collocation [cf. Dahlquist and Bjorck (1974), pp. 396–397], and limit the search for the fixed point to a finite system of linear equations.

of this construction, each row of matrix $P_{g_n^h}$ is made up of non-negative elements that add up to unity. Hence, the matrix $[I - \beta P_{g_n^h}]$ is always invertible, for $0 < \beta < 1$.

An advantage of the discretization procedure of Section 4.1 is that under certain regularity conditions the (Frechet) derivative of operator B at W_n^h exists and it is given by $-[I - \beta P_{g_n^h}]$. Moreover, if $\{W_n^h\}_{n \geqslant 0}$ is a sequence generated by Equations (5.3)–(5.4) it follows from these equations that such sequence satisfies

$$W_{n+1}^h = W_n^h + \left[I - \beta P_{g_n^h}\right]^{-1} B(W_n^h). \tag{5.6}$$

Therefore, policy iteration is equivalent to Newton's method applied to operator B [cf. Puterman and Brumelle (1979)]. As is well known, Newton's method exhibits locally a quadratic rate of convergence; i.e. in a certain neighborhood of W^h there exists some constant $L > 0$ such that $\left\|W^h - W_{n+1}^h\right\| \leqslant L \left\|W^h - W_n^h\right\|^2$ for $\{W_n^h\}_{n \geqslant 0}$ generated by Equation (5.6). Moreover, the region of quadratic convergence and the complexity properties of this method are determined by the second-order derivative of B [or by the Lipschitz properties of the derivative of B, if second-order derivatives do not exist; e.g., Traub and Wozniakowski (1979)].

Observe that as we proceed to finer grids or partitions the numbers of rows and columns of matrix $P_{g_n^h}$ increase accordingly. Likewise, for the same perturbations of the function W^h (using as distance the sup norm), the values of these entries vary more rapidly for finer grids. Hence, the second-order differentiability properties of B cannot be bounded independently of the grid size. Indeed, such derivatives will get unbounded as h goes to zero. This is to be contrasted with the method of successive approximations where the rate of convergence is linear, bounded by factor β (i.e., $\left\|W^h - W_{n+1}^h\right\| \leqslant \beta \left\|W^h - W_n^h\right\|$ for all $n \geqslant 1$). Such bound is independent of the grid size, and of the distance from the fixed point.

Therefore, policy iteration exhibits quadratic convergence to the fixed point of the algorithm, at the cost of introducing the more complex computational step (5.4); quadratic convergence is local, and the region of quadratic convergence (as well as the constant involved in the order of convergence) may depend on the mesh size of the triangulation. Each iteration involves a matrix inversion; although such an operation may be achieved by some efficient procedures exploiting the structure of the problem, its computational cost increases exponentially with the size of the grid. These considerations lead us to think that policy iteration may not perform so well for very fine partitions of the state space. Some illustrative computations will be reported in Section 7.

5.3. Modified policy iteration

The method of modified policy iteration was originally discussed by Morton (1971). The main purpose is to speed convergence to the fixed point W^h, without incurring in the computational burden of policy iteration.

In the framework of our discretization procedure, for a given triangulation $\{S^j\}$ of mesh size h, the following iterative scheme will be considered for *modified policy iteration*:

(i) *Initial step*: Select an accuracy level *TOLW* and a function W_0^h.

(ii) *Policy improvement step:* Find $g_{n+1}^h(k_0^j, z_0^j)$ that solves

$$\max_{k_1} v(k_0^j, k_1, z_0^j) + \beta \int_Z W_n^h(k_1, z')\, Q(z_0^j, \mathrm{d}z') \tag{5.7}$$

for each vertex point (k_0^j, z_0^j).

(iii) *Policy evaluation step*: For a fixed integer $m \geqslant 1$, let

$$\begin{aligned} W_{n+1}^h(k_0^j, z_0^j) = \sum_{t=0}^{m-1} \beta^t \int_{Z^t} v(g_{n+1}^{ht}(k_0^j, z_0^j), g_{n+1}^{ht+1}(k_0^j, z_0^j), z_t)\, \mu^t(z_0^j, \mathrm{d}z^t) \\ + \beta^m \int_{Z^m} W_n(g_{n+1}^{hm}(k_0^j, z_0^j), z_m) \mu^m(z_0^j, \mathrm{d}z^m) \end{aligned} \tag{5.8}$$

for each vertex point (k_0^j, z_0^j).

(iv) *End of iteration*: If $\|W_{n+1}^h - W_n^h\| \leqslant TOLW$, stop; else, increment n by 1, and return to step (ii).

As in Section 3, the term g_n^{ht} refers to the composite function

$$g_n^h(g_n^h(\cdots g_n(k_0, z_0)\cdots), z_{t-2}), z_{t-1})$$

for every possible realization $(z_1, z_2, \ldots, z_{t-1})$. Hence, for $m > 1$ the right-hand side of Equation (5.8) involves the calculation of multidimensional integrals, which in some cases may be rather complex.

Observe that if $m = 1$ the iterative scheme reduces to the dynamic programming algorithm, and as m goes to infinity the method approaches policy iteration. For a related numerical framework, Puterman and Shin (1978) have established global convergence to the fixed point W^h, and an asymptotic, linear rate of convergence equal to β^m, i.e.,

$$\lim_{n\to\infty} \frac{\|W_{n+1}^h - W^h\|}{\|W_n^h - W^h\|} = \beta^m.$$

With respect to the dynamic programming algorithm, the above iterative process avoids the use of repetitive maximizations, which are generally fairly costly. Thus, the optimal m may be greater than 1 [see Christiano (1990), and Example 7.1 below]. Our experience is that modified policy iteration algorithms usually perform well in deterministic optimization problems or in cases where multiple integrals are easily calculated.

5.4. Polynomial interpolation and spline functions

For certain applications it may be more convenient to consider alternative approximation schemes such as piecewise multilinear functions over rectangular subdivisions, or higher-order interpolants (e.g., polynomials and splines). Under our previous assumptions, piecewise multilinear approximations would again yield an approximation error for the computed value function of order h^2. Moreover, for cases in which the value function W is higher-order differentiable, it is possible to obtain better orders of convergence under more appropriate approximation schemes. In particular, if the value function is C^k for $k \geqslant 3$, then it is also plausible to derive convergence of order k using higher-order interpolants.

As discussed in Section 3, fairly little is known about higher-order differentiability properties of the value and policy functions, and there are no known operational methods to bound these derivatives whenever they exist. In those cases, a more complex interpolant may not yield better accuracy, since the approximation error may depend on the size of the higher-order derivatives[9].

There are nevertheless certain situations where the use of higher-order interpolants may be advantageous. An obvious case is when for the family of models under consideration the value and policy functions are smooth with relatively small high-order derivatives. But even if these derivatives do not always exist, one may be led to believe that the functions are reasonably well behaved, and that smooth approximations will give good results. Additionally, if concavity is preserved, smooth approximations facilitate the application of more efficient numerical maximization subroutines, accelerating the computation process.

5.4.1. Polynomial interpolation

The use of polynomial approximation in dynamic programming dates back to Bellman, Kalaba and Kotkin (1963). These authors argue in favor of polynomial bases with certain orthogonality properties such as the *Chebyshev* and *Legendre* polynomials. Generally, the use of orthogonal bases facilitates the computations and leads to better accuracy.

The Chebyshev polynomial of degree m is denoted $T_m(x)$, and is defined by the relation

$$T_m(x) = \cos(m \arccos x), \quad m = 0, 1, 2, \ldots. \tag{5.9}$$

[9] In order to extend the analysis of Section 4 to higher-order interpolations, a technical problem is to establish the monotonicity of the discretized maximization operator T^h, asserted in Lemma 4.1. This property is not generally satisfied for higher-order interpolants [cf. Judd and Solnick (1997)].

Combining this definition with some trigonometric identities, we obtain for $m \geqslant 1$ the functional expressions

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ &\cdots \\ T_{m+1}(x) &= 2xT_m(x) - T_{m-1}(x). \end{aligned}$$

Each polynomial T_m has m zeroes in the interval $[-1, 1]$. These solutions are specified by the values

$$x_k = \cos\left(\frac{\pi(2k-1)}{2m}\right), \quad k = 1, 2, \ldots, m. \tag{5.10}$$

The location of these zeroes is such that $2^{-(m-1)}T_m$ is the polynomial of degree m with leading coefficient 1 which deviates least from zero over the interval $[-1, 1]$. Hence, these polynomials feature minimal oscillations and this is an attractive property for the purposes of interpolation. Another important property is the following discrete orthogonality relation. Let x_k $(k = 1, \ldots, m)$ be the m zeroes of $T_m(x)$. Then, for $i, j < m$,

$$\sum_{k=1}^{m} T_i(x_k)T_j(x_k) = \begin{cases} 0 & i \neq j, \\ m/2 & i = j \neq 0, \\ m & i = j = 0. \end{cases} \tag{5.11}$$

To illustrate the relevance of appropriately locating the nodes to minimize possible oscillations from polynomial interpolation, let us examine the following two examples. First, consider the function $f(x) = \sqrt{x}$, $x \in [0, 1]$. Functions of this form are frequently observed in economic models. Successive interpolations of this function will be taken at both equally spaced points and at the Chebyshev zeroes[10]. Table 1 reports the approximation error $e_m = \max_{x \in [0,1]} |f(x) - p_m(x)|$ for various polynomial degrees under both interpolation schemes. It can be observed that uniform convergence is insured for Chebyshev interpolation, but not for interpolation at equally spaced points. Indeed, under this latter interpolation, the error grows without bound.

[10] For a sequence of $m+1$ distinct points $\{x_i\}_{i=0}^m$, interpolation at the values $\{f(x_i)\}_{i=0}^m$ uniquely defines a polynomial $p_m(x)$ of degree m. For equally spaced points, x_0 and x_m would correspond to the extreme points of the interval. For interpolation at the Chebyshev zeroes the nodes are $x_i = \frac{1}{2}(y_i + 1)$ for all i, where $\{y_i\}_{i=0}^m$ are the zeroes of Chebyshev polynomial $T_{m+1}(y)$ defined on $[-1, 1]$. That is, in this latter case a change of units is needed, since the Chebyshev polynomials are defined on the interval $[-1, 1]$, and the function $f(x) = \sqrt{x}$ is restricted to the interval $[0, 1]$.

Table 1
Approximation errors for the function $f(x) = \sqrt{x}$, $x \in [0, 1]$[a]

Vertex points	Chebyshev interpolation	Point of max. error	Interpolation at equally spaced points	Point of max. error
10	5.01×10^{-2}	0.0	1.72×10^{-2}	5.55×10^{-2}
25	2.00×10^{-2}	0.0	5.86×10^{-3}	2.08×10^{-2}
50	1.00×10^{-2}	0.0	2.66×10^{-3}	1.02×10^{-2}
75	6.70×10^{-3}	0.0	2.59×10^{0}	6.75×10^{-3}
100	5.00×10^{-3}	0.0	4.28×10^{9}	5.05×10^{-3}
200	2.50×10^{-3}	0.0	4.69×10^{39}	9.97×10^{-1}
300	1.60×10^{-3}	0.0	3.72×10^{68}	9.98×10^{-1}
400	1.20×10^{-3}	0.0	9.50×10^{98}	1.25×10^{-3}
500	1.00×10^{-3}	0.0	1.36×10^{129}	9.98×10^{-1}

[a] Columns 2 and 4 report the approximation errors $e_m = \max_{x\in[0,1]} |f(x) - p_m(x)|$, with $p_m(x)$ the polynomial interpolant of degree m. Columns 3 and 5 report the point x in $[0, 1]$ where e_m attains the maximum value for each of these interpolants.

Another notorious example of non-convergence can be obtained from the function $f(x) = |x|$, $x \in [-1, 1]$. This function is non-differentiable at $x = 0$, and has constant derivatives at every other point. Again, simple functions of this form are commonly observed in economics. Non-differentiabilities, such as that of point $x = 0$, may arise from optimization at a boundary surface. In an analogous manner, Table 2 displays the approximation error e_m for polynomials of various degrees under both interpolation procedures. As in the previous example, only Chebyshev interpolation guarantees uniform convergence. As a matter of fact, it can be shown [cf. Natanson (1965), p. 30] that for interpolation at uniformly spaced points convergence occurs only at points $-1, 0, 1$, and not at any other point. (Convergence at the extreme points $-1, 1$ is guaranteed by construction.)

In the case of function $\sqrt{x}$ a main problem for polynomial interpolation is that the derivatives are unbounded at point $x = 0$, whereas for function $|x|$ the derivatives are not defined at point $x = 0$. Since the polynomial interpolant is jointly defined over the whole domain, sharp changes in the derivatives of the function may lead to large oscillations in the interpolant. These oscillations are somewhat minimized under Chebyshev interpolation, and in both of the above examples such interpolation procedure displays uniform convergence. There are, however, continuous functions for which Chebyshev interpolation may fail to converge uniformly. Although there is no known functional form with such a property, lack of convergence may be established by a constructive argument [cf. Natanson (1965), Ch. 2]. Indeed, from the construction of such a mapping one could actually show that the class of continuous functions for

Table 2
Approximation errors for the function $f(x) = |x|$, $x \in [-1, 1]$[a]

Vertex points	Chebyshev interpolation	Point of max. error	Interpolation at equally spaced points	Point of max. error
10	5.50×10^{-2}	-5.26×10^{-2}	7.47×10^{-2}	-5.55×10^{-17}
25	2.31×10^{-2}	-6.12×10^{-2}	5.83×10^{2}	9.58×10^{-1}
50	1.11×10^{-2}	-1.01×10^{-2}	4.77×10^{7}	-9.79×10^{-1}
75	7.73×10^{-3}	-2.01×10^{-2}	2.14×10^{16}	9.86×10^{-1}
100	5.58×10^{-3}	-5.02×10^{-3}	3.09×10^{21}	-9.89×10^{-1}
200	2.79×10^{-3}	-2.50×10^{-3}	2.35×10^{50}	-9.94×10^{-1}
300	1.86×10^{-3}	-1.66×10^{-3}	5.81×10^{79}	9.96×10^{-1}
400	1.39×10^{-3}	-1.25×10^{-3}	2.31×10^{109}	9.97×10^{-1}
500	1.11×10^{-3}	-1.00×10^{-3}	1.19×10^{139}	-9.97×10^{-1}

[a] Columns 2 and 4 report the approximation errors $e_m = \max_{x\in[-1,1]} |f(x) - p_m(x)|$, with $p_m(x)$ the polynomial interpolant of degree m. Columns 3 and 5 report the point x in $[-1, 1]$ where e_m attains the maximum value for each of these interpolants.

which Chebyshev interpolation may fail to converge uniformly is non-negligible in the metric space induced by the max norm.

For continuous functions, uniform convergence can be insured for a more fanciful, Hermitian interpolation at the Chebyshev nodes [cf. Rivlin (1990), Th. 1.3]. However, this procedure may become awkward for computational purposes, since there is no handy way to estimate the approximation error. For continuously differentiable functions, more operative error bounds are available. Thus, assume that f is a C^k function on $[-1, 1]$. Then, it can be shown [cf. Rivlin (1969, Theorems 1.5, 4.1, 4.5), Judd (1992)] that for Chebyshev interpolation,

$$e_m = \max_{-1\leqslant x\leqslant 1} |f(x) - p_m(x)| \leqslant M\frac{\log m}{m^k}\left\|f^k\right\| \tag{5.12}$$

for all $m > k$; here, M is a certain constant that depends on k, and $\left\|f^k\right\|$ is the maximum value of the kth-order derivative of f. As observed by Judd (1992), piecewise linear interpolation dominates asymptotically polynomial interpolation for $k \leqslant 2$, whereas for $k \geqslant 3$ polynomial interpolation exhibits a higher convergence order. Of course, in practical applications one should also take into account the constant terms involved in these error bounds. As discussed in Section 3, high-order derivatives of the policy function may grow without bound or may fail to exist. In those cases, polynomial interpolation may lead to detrimental results, and indeed many authors warn against its extensive use. The following excerpt is taken from Press et al. (1992), p. 101:

> Unless there is a solid evidence that the interpolation function is close in form to the true function, f, it is a good idea to be cautious about polynomial interpolation. We enthusiastically

> endorse interpolation with 3 or 4 points, we are perhaps tolerant of 5 or 6; but we rarely go higher than that unless there is quite rigorous monitoring of estimated errors.

A further problem with polynomial interpolation is that the functions may lose their original shape. Concavity and monotonicity are not usually preserved. These are key properties for numerical maximization and related operations. On the other hand, polynomials possess exact derivatives and integrals, and as simple, smooth functions they can allow for some other convenient manipulations.

Our commentary thus far has been limited to polynomial interpolation for functions of one variable. There is much less practical experience with multivariate polynomial interpolation, a topic surrounded by further technical difficulties [cf. Lorentz (1992), Xu (1996)]. One notorious problem is that for a given set of points and a proper polynomial subspace, the interpolant may not be uniquely defined. To obtain good results, either the functional domain or the location of the nodes must be restricted. A well-behaved family of multidimensional interpolants are those defined as products of monomials[11] (i.e., the so called *tensor products*), where many unidimensional arguments carry through. Indeed, for tensor products regular polynomial interpolation is uniquely defined; further, error bounds of the type (5.12) are also available, even though these estimates are somewhat diminished. For further details, see Hammerlin and Hoffmann (1991, Ch. 6).

5.4.2. Spline functions

Let $a = x_0 < x_1 < x_2 \ldots < x_n = b$ be an array of vertex points in the real line. Then, a spline function of *degree* k is a C^{k-1} mapping on $[a, b]$ that coincides at each internal $[x_i, x_{i+1}]$ with a polynomial of degree k. This definition generalizes to the multidimensional case by considering tensor products over unidimensional functions [cf. Schumaker (1981)]. A piecewise linear function would correspond to a spline of degree 1.

Splines combine in subtle ways benefits of polynomials and piecewise interpolation, since they allow for a tight control of the function, preserving the smoothness of the interpolant. More precisely,

(i) As piecewise functions, splines avoid the typical oscillating behavior of polynomial interpolation.
(ii) As higher-order interpolants, splines may exhibit better orders of convergence than piecewise linear functions.
(iii) As smooth approximations, splines permit the application of powerful numerical maximization methods.

[11] For instance, for bivariate interpolation in the (x, y)-plane, a polynomial $p_{nk}(x, y)$ would be defined at each point as $p_{nk}(x, y) = p_n(x)p_k(y)$, where $p_n(x)$ is a polynomial of degree n in x and $p_k(y)$ is a polynomial of degree k in y.

As already discussed, polynomials may have a poor performance for the approximation of certain functions. By focussing on a certain grid of points, splines may overcome this unnatural feature of polynomials. Piecewise linear approximations and splines are *finite element methods*, which allow for a local control of the approximation error. The main trade-off with respect to piecewise linear approximations is that splines may yield better orders of convergence, but generally require greater computational cost for determining the appropriate coefficients. *Cubic splines* and *B-splines* [cf. Schumaker (1981)] are two examples of higher-order interpolants that can be implemented at a relatively low computational cost.

An additional advantage of splines is that the resulting function is smooth, so that Newton-type methods – which rely on the computation of first- and second-order derivatives – can be applied for numerical maximization. Of course, for a successful implementation of a Newton-type method, the interpolation must preserve certain concavity properties, suggesting that the order of the interpolant cannot be too high. Indeed, there are certain quadratic splines that preserve the concavity of the approximation [cf. Schumaker (1983)] but it seems much harder to preserve concavity for splines of higher order. In those cases, it appears more practical to check concavity for some test functions [cf. Johnson et al. (1993)].

There has not been so much numerical experimentation in economics on the performance of splines. It seems, however, that these functions may be greatly beneficial for cases of fine grids and smooth optimal policies[12]. Some illustrative computations are provided in Section 7. Spline interpolation has been advocated by Johnson et al. (1993). These authors study a four-dimensional dynamic programming problem, and report reductions in CPU time over piecewise linear functions by factors 250–300, for a given level of accuracy. The gains stem from both faster numerical maximization and better accuracy properties of splines. For the numerical maximization, Johnson et al. (1993) apply a quasi-Newton algorithm under spline interpolations, and a polytope algorithm under piecewise linear interpolation. It appears that these gains are overestimated, since quasi-Newton methods can still be used for piecewise linear interpolants provided that these functions preserve some regularity properties (see the discussion in Section 4.4).

6. Numerical approximations of the Euler equation

Our main objective so far has been the computation of the value function from Bellman's equation (3.1). Since it does not seem plausible to compute this fixed point directly, several iterative algorithms have been considered that guarantee convergence to the fixed point. These algorithms are reliable, but they involve at each step costly

[12] Splines could still be effective in models lacking higher-order differentiability of the policy function, especially if such derivatives exist almost everywhere and are bounded.

numerical maximizations and integrations. In this section, we discuss alternative solution methods based on approximating the Euler equation. In general, these methods do not guarantee global convergence to a desired solution, but are sometimes very effective, since they approximate locally the fixed point at higher convergence rates. We shall also review some valuable results for testing the accuracy properties of these algorithms.

Our starting point is the following *Euler equation*:

$$D_2 v(k_0, k_1, z) + \beta \int_Z D_1 v(k_1, k_2, z') Q(z, \mathrm{d}z') = 0. \tag{6.1}$$

Under our previous assumptions of interiority and differentiability, this equation must be satisfied along every optimal orbit. Hence, this equation holds at all (k_0, z) such that $k_1 = g(k_0, z)$ and $k_2 = g(g(k_0, z), z')$. Moreover, under the present assumptions, function g is the unique solution to this functional equation [cf. Stokey and Lucas (1989, Ch. 4)].

Several discretization procedures are available to compute the optimal policy function g from Equation (6.1). In deterministic problems, there are well established methods for the solution of ordinary differential equations [e.g., see Gear (1971) and Lambert (1991), and Mulligan (1993) for a recent application to economic problems]. In order to extend this approach to our economic framework, the basic idea is to approximate the graph of the policy function as the set of solutions that satisfy at all times the second-order system of difference equations implicitly defined by Equation (6.1). Thus, let us assume that uncertainty is not present in the analysis, so that the Euler equation may be written in the following simple form:

$$D_2 v(k_0, g(k_0)) + \beta D_1 v(g(k_0), g^2(k_0)) = 0. \tag{6.2}$$

Suppose now that g has a unique, globally stable steady state, $k^* = g(k^*)$. Such stationary solution k^* can readily be calculated from Equation (6.2). Furthermore, the derivative of the policy function $Dg(k^*)$ can be determined from the tangent space of the stable manifold of the system. Hence, in a small neighborhood of k^*, function g may be approximated by its derivative $Dg(k^*)$. Once these functional values have been estimated for a given neighborhood, a global approximation can be obtained by iterating backwards on Equation (6.2). Indeed, computing the stable manifold of system (6.2) amounts to computing the graph of the policy function. Moreover, error estimates for all these approximations can be easily derived.

The approach just described allows us to compute directly the policy function, and avoids the explicit use of numerical maximizations, typical of the slower algorithms of the preceding sections. However, this computational procedure breaks down in deterministic models with complex dynamics or in the presence of uncertainty. Indeed, in a stochastic framework the stationary state would correspond to an invariant distribution, which generally cannot be well approximated locally by the derivative

of the policy function. Hence, we need alternative methods that can tackle the Euler equation in a more global way.

6.1. Numerical methods for approximating the Euler equation

Since numerical methods from ordinary difference equations are not directly applicable to stochastic models, an alternative approach is to compute function g directly from Equation (6.1) using a standard discretization procedure (e.g., using Newton's method over a finite-dimensional space of functions that approximate g). But given that function g is implicitly defined by such a non-linear equation, it seems that a more fruitful computational strategy would be to approximate directly the second term of Equation (6.1) – and consequently the first – as a function of (k,z); then, from this approximation we can compute function g. This approach avoids some of the non-linearities involved in attempting to compute the implicitly defined function g directly, and consequently it may lead to more operational numerical schemes. The family of methods following this approach is known as *parameterized expectations algorithms* [e.g., Wright and Williams (1984) for an early application, and den Haan and Marcet (1990), Judd (1992), and Christiano and Fisher (1994) for an account of recent developments]. This terminology may seem deceptive, since such algorithms can also be used for computing solutions of deterministic dynamic problems. Hence, "algorithms approximating the Euler equation" seems to be a more appropriate connotation.

We shall first outline a general method for carrying out these computations, and then focus on a simple algorithm due to Christiano and Fisher (1994). As with other algorithms in this family, there are potential problems concerning its implementation; in particular, a solution may not exist, there may be multiple solutions, the algorithm may fail to converge to a desired solution, and formal error estimates are not available.

Our description of a general framework for *approximating the Euler equation* proceeds as follows:

(Step 1) Select an n-dimensional space Ψ of real-valued, non-negative functions $\Psi_a(k,z)$; each function Ψ_a can be defined by a vector $a \in R^n$.

(Step 2) Compute a function $k' = g_a(k,z)$ from the condition

$$D_2 v(k,k',z) + \Psi_a(k,z) = 0. \tag{6.3}$$

(Step 3) Define a system of n-dimensional equations

$$\varphi_i(\Psi_a - m_a) = 0, \quad i = 1,2,\ldots,n, \tag{6.4}$$

where each φ_i is a real-valued mapping from the space of l-dimensional functions over (k,z), and m_a denotes the mapping

$$m_a(k,z) = \beta \int_Z D_1 v(g_a(k,z), g_a(g_a(k,z),z'),z')\, Q(z,\mathrm{d}z').$$

(Step 4) Find a solution a^* for system (6.4).

There are several issues involved in this numerical procedure. The main objective is to attain a certain level of accuracy for a reasonable computational cost. In step 1, one chooses a suitable n-dimensional space of functions Ψ. This space could be generated by the ordinary polynomials of degree $n-1$ [e.g., Marcet and Marshall (1994)], by polynomials with orthogonal bases [cf. Judd (1992)], or by piecewise linear functions [e.g., McGrattan (1996)], or by spline functions. Given an element Ψ_a, in step 2 we can compute from Equation (6.3) a function g_a. As already discussed, this indirect procedure for computing the policy function may simplify the solution method, since the first term in Equation (6.1) is now approximated by a known function Ψ_a, and hence all non-linearities are just embedded in mapping m_a [13]. Then, in step 3 we formulate an n-dimensional system of equations to determine the fixed point a^*, which may be calculated by some root-finding numerical procedure. The most powerful solution methods are those based upon Newton's method, involving the inversion of a certain Jacobian matrix; in those situations, one should check that the problem is *well-conditioned* [i.e., that the matrix to be inverted is not nearly singular; see Press et al. (1992)]. Newton's method ensures a quadratic order of convergence provided that the initial guess is close enough to the true solution.

Unless $\Psi_a(k,z)$ is a sufficiently good approximation, Equation (6.3) may not have a feasible solution. Indeed, $k' = g_a(k,z)$ is the maximizer for the optimization problem

$$\max_{k'} \upsilon(k,k',z) + \Psi_a(k,z) \cdot k'. \tag{6.5}$$

In view of the concavity of $\upsilon(k,k',z)$ on k', numerical maximization of problem (6.5) may be a reliable strategy for solving Equation (6.3). Alternatively, a solution to Equation (6.3) may be obtained by some root-finding method, and for multisector models a good initial guess is usually required in order to guarantee convergence to a desired solution. In models with one variable, computation of Equation (6.3) may just involve some simple algebraic operations.

Observe that the choice of the space Ψ limits the set of plausible conditions to be imposed in step 3, and consequently the available solution methods to be applied in step 4. Thus, simultaneous consideration of steps (1)–(4) is required for an optimal design of the numerical algorithm. Computation of the fixed point a^* may be a rather delicate problem, since function m_a could be highly non-linear, involving a conditional expectation; further, a solution may not exist, or there may be multiple solutions. Press et al. (1992, p. 372) share the following views on these issues:

[13] A common belief in favor of this computational approach is that the conditional expectation function $m(k,z) = \beta \int D_1 \upsilon(g(k,z), g(g(k,z),z'),z') Q(z,\mathrm{d}z')$ is smoother than other functions characterizing the optimal solution, such as the policy function $g(k,z)$. A straightforward application of the chain rule shows, however, that in general function $m(k,z)$ cannot be smoother than function $g(k,z)$. In a revised version of the original paper, Christiano and Fisher argue that, under an alternative approximation due to Wright and Williams (1984), function $m(k,z)$ may be smoother than $g(k,z)$ in some specific situations. This seems, however, a minor improvement; in addition, the Wright–Williams approach may lead to further complexities in the computation involved in Equation (6.3), as concavity is lost.

> We make an extreme but wholly defensible statement: There are *no* good, general methods for solving systems of more than one nonlinear equation. Furthermore, it is not hard to see why (very likely) there *never will be* any good, general methods. For problems in more than two dimensions, we need to find points mutually common to N unrelated zero-contour surfaces, each of dimension $N-1$. You see that root finding becomes virtually impossible without insight! You will almost always have to use additional information, specific to your particular problem, to answer such basic questions as, "Do I expect a unique solution?", and "Approximately where?" ...

We should then highlight some important differences with respect to the family of algorithms of preceding sections. Numerical dynamic programming may be slow and computationally costly. However, global convergence to a desired solution always holds, and the associated maximization and integration operations may be effected by relatively reliable procedures. In contrast, solving non-linear systems of equations is conceptually a much more complex numerical problem, although under a good initial candidate for the approximate solution convergence could be faster, assuming that the system is well-behaved (i.e., singularities do not unfold in the computational procedure). Given the inherent complexity of non-linear systems, we should not expect these solution methods to be operational for very large equations systems, or for discretizations involving a great number of vertex points.

As a representative element in the class of numerical methods approximating the Euler equations, we now present an algorithm put forth by Christiano and Fisher (1994). The finite-dimensional space Ψ will be generated by the Chebyshev polynomials. These polynomials satisfy certain desired orthogonality properties, which will facilitate the implementation of the numerical model, and allow for a more accurate approximation.

To describe the workings of the algorithm, let us confine ourselves to the simple framework of Equation (6.2), i.e. a deterministic setting in which k is a scalar variable. Each basis function Ψ_a is written in the form $\Psi_a = \exp(\sum_{j=0}^{n-1} a_j T_j(x))$, where $T_j(x)$ is the Chebyshev polynomial of degree j,

$$x = 2\frac{\log k - \log \underline{k}}{\log \bar{k} - \log \underline{k}} - 1,$$

and $[\underline{k}, \bar{k}]$ is the interval of feasible capitals[14]. The exponential function ensures that each basis function is non-negative; also, the new variable $x \in [-1, 1]$. We can then express k as a function of x, which in more compact form will be written as $k = \eta(x)$.

[14] An alternative parameterization is to let

$$x = 2\frac{k - \underline{k}}{\bar{k} - \underline{k}} - 1.$$

[If there are many state variables, tensor products can generate polynomial bases with orthogonal properties of the type (5.11), e.g. see Judd (1992).]

Step 2 requires computation of function g_a, and this is straightforward in some simple models [cf. Christiano and Fisher (1994)], since $\Psi_a(k)$ is equal to the inverse of the marginal utility of consumption. In step 3, the algorithm makes use of a *collocation* method so that condition (6.4) reduces to the following system of equations:

$$\exp\left(\sum_{j=0}^{n-1} a_j T_j(x_i)\right) - m_a(\eta(x_i)) = 0, \quad i = 1, 2, \ldots, n. \tag{6.6}$$

Here, x_i $(i = 1, 2, \ldots, n)$ are the zeroes of the nth-degree polynomial $T_n(x)$. Taking logs in Equation (6.6), and making use of the orthogonality conditions (5.11), we then obtain

$$a_j = \frac{\mu}{n} \sum_{i=1}^{n} T_j(x_i) \log m_a(\eta(x_i)), \qquad j = 0, 1, \ldots, n-1, \tag{6.7}$$

where $\mu = 2$ if $j \geqslant 1$, and $\mu = 1$ if $j = 0$. Observe that all non-linearities in system (6.7) appear on the right-hand side. This seemingly simple form may facilitate computation of a fixed point a^*, and such a simple structure stems from both the method of approximating the Euler equations and the orthogonality properties of the polynomials.

Let

$$\phi_j^n(a) = \frac{\mu}{n} \sum_{i=1}^{n} T_j(x_i) \log m_a(\eta(x_i)), \qquad j = 0, 1, \ldots, n-1.$$

Then, step 4 involves the computation of a fixed point $a^* = \phi^n(a^*)$; i.e., solving the equation system (6.6) or (6.7). This computation may be most efficiently effected via standard solution methods for non-linear equations. Christiano and Fisher (1994) resort to a Newton–Raphson algorithm, taking as initial guess the solution of an associated quadratic optimization problem. For more complex problems, homotopy methods may be more effective, or successive refinements of the algorithm, using as initial guess the solution of a coarser discretization (cf. Example 7.2, below). Alternatively, one could iterate on the map $a_{t+1} = \phi^n(a_t)$, and verify if the sequence converges to a fixed point, but unless we are sure of the stability properties of mapping ϕ^n, convergence here could be slower or even more problematic.

As illustrated by Christiano and Fisher (1994), the algorithm has been fast and accurate in some simple test cases. However, this good performance may not be observed in more complex applications, since computation of a fixed point a^* in Equation (6.7) is a rather delicate problem, and the indiscriminate use of polynomial interpolation may result in a poor approximation. As a matter of fact, there is relatively little theoretical work on the performance and asymptotic properties of this class of algorithms, and it seems that the following issues need further investigation:

(i) *Existence*: The non-linear system of equations characterizing a particular algorithm may not have a solution a^*, and hence we may be unable to produce a reasonable approximation.
(ii) *Multiplicity*: There may be multiple solutions.
(iii) *Computability*: Even if there is a unique solution, the system of non-linear equations may not be amenable to computation using standard numerical techniques.
(iv) *Accuracy*: There is no theory on error bounds or accuracy properties for this family of algorithms.

A related discussion on the existence of a fixed point a^* is contained in Marcet and Marshall (1994). These authors suggest the use of the Brouwer fixed-point theorem. To the best of our knowledge, at present there are neither rigorous proofs, nor well known counterexamples, regarding the existence of a fixed point a^*; moreover, it is hard to document if existence has been a serious operational issue.

Under our assumptions in Section 2, the previously discussed algorithm may generate multiple solutions. The multiplicity of solutions may signal the presence of singularities, which limit the range of methods for solving systems such as (6.7); however, multiplicity should not raise further logical concerns. If there are multiple solutions, one would be advised to select the one with smallest Euler equation residuals (cf. Theorem 6.1, below); else, if all solutions exhibit small residuals, then the corresponding policy functions cannot be far apart form each other.

Application of Newton-type methods for solving systems of non-linear equations involves inversions of matrices of derivatives, and these matrices may be singular (or nearly singular). Although Newton-type methods can attain quadratic orders of convergence to a fixed point a^*, collinearities may result in inaccurate solutions or in rather small regions of convergence. Such difficulties may especially arise in complex stochastic models, with costly numerical integrations, and where a given level of accuracy may require fine grids or polynomials of high degree.

Regarding accuracy, there is no formal derivation of error bounds for the class of algorithms studied in this section. In this analysis two types of accuracy results are pertinent:

(a) Accuracy of a given numerical scheme approximating the Euler equation.
(b) Derivation of error bounds for the policy and value functions using the Euler equation residuals.

Concerning point (a), in the absence of a good theory for bounding or estimating high-order derivatives of the policy function, it seems a difficult task to obtain tight error estimates for polynomial approximations, especially in stochastic models with several state variables. Likewise, stability properties [cf. Section 4.4] for algorithms using Newton's method will depend primarily on the conditioning number of the Jacobian matrix, and this has to be directed specifically to each particular application.

On the other hand, understanding how the size of the Euler equation residuals translates into approximation errors for the value and policy functions is not only essential to derive theoretical error bounds for the computed value and policy functions

under this family of algorithms, but it is also a key step for further implementational issues. For instance, evaluation of the residuals could allow us to assess the accuracy of competing numerical solutions, or the accuracy of a given numerical solution regardless of our confidence on the algorithm.

6.2. *Accuracy based upon the Euler equation residuals*

In practical applications, accuracy can be checked in simple test cases against analytical solutions, or against more reliable numerical methods. These indirect procedures may nevertheless be awkward or infeasible in some situations. That is, computing models with closed-form solutions is only illustrative of the performance of the algorithm in real applications, and the use of reliable numerical methods to test an algorithm involves further computational cost.

We shall present here a recent result that allows us to bound the approximation error for any arbitrary solution. This analysis is based upon the computation of the Euler equation residuals. Computation of the residuals is a relatively easy task which involves functional evaluations, and hence it can be effected for arbitrarily large samples of points at a reasonable computational cost.

In order to proceed more formally we need the following terminology. Let $\hat{g}$ be a measurable selection of the technology correspondence, Γ. Define $W_{\hat{g}}$ as

$$W_{\hat{g}}(k_0, z_0) = \sum_{t=0}^{\infty} \beta^t \int_{Z^t} v(\hat{g}^t(k_0, z_0), \hat{g}^{t+1}(k_0, z_0), z_t)\, \mu^t(z_0, \mathrm{d}z^t).$$

As before, $\hat{g}^t(k_0, z_0) = \hat{g}(\hat{g}(\cdots \hat{g}(k_0, z_0)\cdots), z_{t-2}), z_{t-1})$ for every possible realization $(z_1, z_2, \ldots, z_{t-1})$. The interpretation is that $\hat{g}$ is the computed policy function, and $W_{\hat{g}}$ is the resulting value function under the plan generated by $\hat{g}$. The following result applies to every numerical solution $\hat{g}$ independently of the algorithm under which $\hat{g}$ may have been secured.

Theorem 6.1 [*Santos (1999)*]. *Let* $\varepsilon > 0$. *Assume that*

$$\left\| D_2 v(k_0, \hat{g}(k_0, z_0), z_0) + \beta \int_Z D_1 v(\hat{g}(k_0, z_0), \hat{g}^2(k_0, z_0), z_1)\, Q(z_0, \mathrm{d}z_1) \right\| \leqslant \varepsilon \qquad (6.8)$$

for all (k_0, z_0). *Then, we have*

(i) $\left\| W - W_{\hat{g}} \right\| \leqslant \dfrac{2L}{\eta^2 \left(\frac{1}{\sqrt{\beta}} - 1 \right)^2 \left(1 - \sqrt{\beta} \right)^2} \varepsilon^2,$

(ii) $\left\| g - \hat{g} \right\| \leqslant \dfrac{2}{\eta \left(\frac{1}{\sqrt{\beta}} - 1 \right) \left(1 - \sqrt{\beta} \right)} \left(\dfrac{L}{\eta} \right)^{1/2} \varepsilon,$

for sufficiently small $\varepsilon > 0$.

In plain words, this theorem asserts that the approximation error of the policy function is of the same order of magnitude as that of the Euler equation residuals, ε,

whereas the approximation error of the value function is of order ε^2. Furthermore, the constants supporting these orders of convergence only depend on: (a) The discount factor, β; (b) the minimal curvature of the return function, η, as specified in Assumption 2; and (c) the norm of the second derivative of the return function, L, as specified in Equation (3.7). Sharper error bounds may be obtained by invoking further properties of the solution [cf. Santos (1999), Th. 3.3].

To understand the nature of this result, it may be instructive to focus on an analogous analysis of the finite-dimensional case. Thus, assume that $F : R^l \to R$ is a strongly concave, C^2 mapping. Let

$$x^* = \arg \max_{x \in R^l} F(x).$$

Then, the derivative

$$DF(x^*) = 0.$$

Moreover, by virtue of the strong concavity of this function,

$$\|DF(\hat{x})\| \leqslant \varepsilon \Rightarrow \|\hat{x} - x^*\| \leqslant N\varepsilon, \tag{6.9}$$

where the constant N is inversely related to the curvature of F, and so this estimate can be chosen independently of ε, for ε small enough. In addition, concavity implies that

$$\begin{aligned} F(x^*) - F(\hat{x}) &\leqslant DF(\hat{x}) \cdot (x^* - x) \\ &\leqslant \|DF(\hat{x})\| \, \|x^* - \hat{x}\| \\ &\leqslant N\varepsilon^2, \end{aligned} \tag{6.10}$$

where the last inequality follows from inequality (6.9). Therefore, in the finite-dimensional case, there exists a constant N such that

$$F(x^*) - F(\hat{x}) \leqslant N\varepsilon^2 \quad \text{and} \quad \|x - x^*\| \leqslant N\varepsilon. \tag{6.11}$$

Matters are not so simple in the infinite-horizon model, since asymptotically discounting brings down the curvature to zero, although it should be realized that the results established in Theorem 6.1 are weaker than those in Equation (6.11). That is, Theorem 6.1 provides a bound for the approximation error of the computed policy function $\widehat{k}_1 = \widehat{g}(k_0, z_0)$, but not for the entire orbit $\{\widehat{k}_t\}_{t=0}^{\infty}$. As is to be expected, we now have that the approximation error is not only influenced by the curvature of the return function v, but also by the discount factor, β.

To understand in a more precise sense the influence of the discount factor β on these estimates, note that the approximation error of the value function is bounded by the discounted sum of expected future deviations $\|\widehat{g}^t(k_0, z_0) - g^t(k_0, z_0)\|$ times

the maximum size of the Euler equation residuals [cf. inequality (6.10)]. Moreover, $\|\widehat{g}^t(k_0, z_0) - g^t(k_0, z_0)\|$ can be bounded iteratively for each $t \geqslant 1$ from the derivatives of the policy function $g(k_0, z_0)$, and such derivatives have an asymptotic exponential growth factor no greater than $1/\sqrt{\beta}$ [cf. Equation (3.9)]. Therefore, a higher β widens our estimate of the approximation error of the value function from the Euler equation residuals, and allows for a higher asymptotic growth of the derivatives of the policy function.

Accuracy checks based upon the size of the Euler equation residuals have been proposed by den Haan and Marcet (1994) and Judd (1992), even though none of these authors provide error bounds for the computed value and policy functions. These estimates can be easily obtained from Theorem 6.1, since the constants involved in those orders of convergence can be calculated from primitive data of the model, and sharper error estimates may be obtained by invoking further properties of the solution. Theoretical error bounds, however, are usually fairly pessimistic. Accordingly, one may consider that a main contribution of Theorem 6.1 is to establish asymptotic orders of convergence for the computed value and policy functions, and with these results now available one may proceed to estimate numerically the constants associated with those convergence orders [cf. Santos (1999)]. While not as convincing as a complete analysis, the information gathered from a numerical assessment of the error may give us a more precise idea of our approximation for the specific model under consideration.

The prevailing view in economics is that Euler equation residuals should be free from dimensionality or measurement units [cf. Judd (1992), p. 437]. Indeed, errors must be evaluated in light of all normalizations and scalings of the functions considered in the analysis, and unit-free measures of error are usually convenient and informative. It should be understood, however, that any measure or elasticity of this sort will not provide a complete picture of our approximation, since as shown previously the accuracy of the residuals is tied down to the curvature of the return function and to the discount factor. As will be argued in Section 9, numerical analysis offers an attractive framework for the study of economic models, and a crucial step in these computational experiments is to show that the errors involved are sufficiently small so as to make proper inferences about the true behavior of the economic model. Without further regard to the properties of the model, there is no hard and fast rule that will always ensure us of the validity of our computations.

As is to be expected, the bounds in Equation (6.10) and Theorem 6.1 are in a certain sense invariant to a rescaling of the variables, since the curvature of the return function varies accordingly. In other words, after a change of units a suitable modification of the tolerance allowed for the Euler residuals yields the same level of accuracy for the computed value and policy functions (cf. Santos (1999)]. Of course, regardless of the scale specified in a numerical exercise, accuracy should be fixed to a level such that despite all the numerical errors, the conclusions drawn from our computations remain essentially true for the original economic model.

7. Some numerical experiments

The preceding algorithms will now be applied to some simple growth models. Most of our discussion will focus on the accuracy and computational cost of these numerical approximations.

Our numerical computations were coded in standard FORTRAN 77, and run on a DEC ALPHA 2100 (with a dual processor, each component rated at 250 MHz), which in a double precision floating-point arithmetic allows for a sixteen-digit accuracy. Subject to this technological limit, our dynamic programming algorithm is in principle free from a fixed accuracy level. Accuracy is determined by the following parameter values:

h, mesh size;

TOLW, accuracy imposed in the iterative scheme: the program stops if for two consecutive value functions W_n^h and W_{n+1}^h the difference $\|W_n^h - W_{n+1}^h\| \leqslant TOLW$;

TOLI, accuracy attained in integration;

TOLM, accuracy attained in maximization.

For the PEA-collocation algorithm of Christiano and Fisher (1994), the accuracy level hinges on the degree of polynomial interpolation, and the degree of these interpolants is chosen so that the Euler equation residuals are sufficiently small.

7.1. A one-sector deterministic growth model with leisure

In our first set of experiments we shall attempt to compute the value and policy functions for the simple growth model of Example 2.1. For $\delta = 1$, these functions have analytical forms, and such forms will serve as benchmark for our numerical results.

For convenience, we again write the optimization problem

$$\max_{\{c_t, l_t, i_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t [\lambda \log c_t + (1-\lambda) \log l_t]$$

$$\text{subject to} \begin{cases} c_t + i_t = A k_t^{\alpha} (1-l_t)^{1-\alpha} \\ k_{t+1} = i_t + (1-\delta) k_t \\ 0 < \beta < 1, \quad 0 < \lambda \leqslant 1, \quad A > 0, \\ 0 < \alpha < 1, \quad 0 \leqslant \delta \leqslant 1 \\ k_t, c_t \geqslant 0, \quad 0 \leqslant l_t \leqslant 1, \quad k_0 \text{ given}, \quad t = 0, 1, 2, \ldots. \end{cases} \tag{7.1}$$

As is well known, for $\delta = 1$ the value function $W(k_0)$ takes the simple form $W(k_0) = B + C \ln k_0$, where B and C are constants such that $C = \lambda\alpha/(1-\alpha\beta)$. Likewise, the policy function $k_{t+1} = g(k_t)$ takes the form

$$k_{t+1} = \alpha\beta A k_t^{\alpha} (1-l_t)^{1-\alpha}, \quad \text{with} \quad l = \frac{(1-\lambda)(1-\alpha\beta)}{\lambda(1-\alpha) + (1-\lambda)(1-\alpha\beta)}.$$

It follows that the system has a unique steady state, $k^* > 0$, which is globally stable.

The existence of one state variable, k, and two controls, l and c, suggests that all numerical maximizations may be efficiently carried out with a unique choice variable. We then write the model in a more suitable form for our computations. The solution to the one-period maximization problem, is determined by the following system of equations:

$$\frac{\lambda}{c_0} = \mu, \qquad \frac{(1-\lambda)(1-l_0)^{\alpha}}{l_0 A k_0^{\alpha}(1-\alpha)} = \mu,$$

where $\mu > 0$ is the Lagrange multiplier. After some simple rearrangements, we obtain

$$c_0 = \frac{\lambda A k_0^{\alpha} l_0 (1-\alpha)}{(1-\lambda)(1-l_0)^{\alpha}}.$$

Likewise,

$$k_1 = A k_0^{\alpha}(1-l_0)^{-\alpha}\left[(1-l_0) - \frac{\lambda l_0(1-\alpha)}{1-\lambda}\right].$$

The iterative process $W_{n+1}^h = T^h(W_n^h)$ in (4.1) is then effected as follows:

$$\begin{aligned} W_{n+1}^h(k_0) = \max_{l_0} \lambda \log\left(\frac{\lambda A k_0^{\alpha} l_0(1-\alpha)}{(1-\lambda)(1-l_0)^{\alpha}}\right) + (1-\lambda)\log(l_0) \\ + \beta W_n^h\left(A k_0^{\alpha}(1-l_0)^{-\alpha}\left[(1-l_0) - \frac{\lambda l_0(1-\alpha)}{1-\lambda}\right]\right). \end{aligned} \tag{7.2}$$

Although Equation (7.2) may appear more cumbersome than the original formulation (7.1), this form will prove more appropriate for our computations as it only involves maximization in one variable.

We initially consider parameter values $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$. For such values the stationary state is $k^* = 1.9696$. For the purposes of this exercise, the domain of possible capitals, K, is restricted to the interval $[0.1, 10]$. Under these conditions it is then easy to check that Assumptions (1)–(3) are all satisfied.

Over the feasible interval of capital stocks, $[0.1, 10]$, we consider a uniform grid of points k^j with step size h. In this simple univariate case our interpolations yield concave, piecewise-linear functions. The maximization at vertex points k^j in Equation (4.1) is effected by Brent's algorithm [cf. Press et al. (1992)] with $TOLM = 10^{-8}$. Such a high precision should allow us to trace out the errors derived from other discretizations embedded in our algorithm. The computer program is

Table 3
Example 7.1. Computational method: dynamic programming algorithm with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
100	10^{-1}	91	3.81	5.31×10^{-2}	3.69×10^{-2}
1000	10^{-2}	181	73.46	5.76×10^{-3}	3.68×10^{-4}
10000	10^{-3}	271	1061.41	5.93×10^{-4}	3.80×10^{-6}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

instructed to stop once two consecutive value functions $W^h_{n+1} = T^h(W^h_n)$ satisfy the inequality

$$\left\| W^h_{n+1} - W^h_n \right\| \leqslant TOLW = \frac{h^2}{5}.$$

Since T^h is a contractive operator with modulus $0 < \beta < 1$, the fixed point $W^h = T^h(W^h)$ in Equation (4.2) should then lie within a distance

$$\left\| W^h - W^h_n \right\| \leqslant \frac{h^2}{5(1-\beta)}.$$

(As will be shown below, in this case constant $\frac{1}{5}$ balances roughly the truncation and approximation errors. Alternatively, this constant could be set in accordance with the estimates obtained at the end of our discussion of the multigrid algorithm in Section 5; in this example, both procedures yield similar values.)

We start this numerical exercise with $h = 10^{-1}$ and the initial condition $W_0 \equiv 0$. In computing the approximate value function W^h for $h = 10^{-1}$ the program stops after $\hat{n} = 91$ iterations with a reported CPU time of 3.81 seconds. We then proceed in the same manner with $h = 10^{-2}$ and $h = 10^{-3}$, taking as initial condition $W_0 = 0$. All the calculations are reported in Table 3, which includes the maximum observed error for the value and policy functions.

One can see that the constant stemming from our numerical computations associated with the quadratic convergence of the approximation error of the value function takes values around 3.8, whereas the corresponding constant for the linear convergence of the approximation error of policy function takes values around 0.59. Both functions converge as predicted by our error analysis. (The relatively small value for the constant associated with the error of the computed policy function seems to be due to the simple structure of the optimization problem.) The evolution of these errors suggests that most of the computational effort is spent in building up the value function for the infinite horizon optimization problem. Consequently, better initial guesses or more direct procedures for computing the value function may considerably lower the CPU time.

Table 4
Example 7.1. Upper estimates for the observed error[a]

Domain for k_0	M	$M + 1/[5(1-\beta)]$	$\widehat{M}$	$\widehat{M} + 1/[5(1-\beta)]$
[1, 10]	1.6740	5.6740	1.9456	5.9456
[0.5, 10]	6.6962	10.6962	7.7824	11.7824
[0.1, 10]	167.4051	171.4052	194.5609	198.5609

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

In order to compare these numerical estimates with our previous theoretical analysis we first decompose the approximation error for the value function as suggested in Equation (5.2). Thus,

$$|W(k) - W^h(k)| \leqslant Mh^2$$

is the error resulting from our numerical algorithm, and

$$|W^h(k) - W^h_{\hat{n}}(k)| \leqslant \frac{1}{5(1-\beta)} h^2$$

is the error resulting from stopping the iteration process in finite time. The right-hand side of this inequality is equal to $4h^2$. The constant $M = \gamma/[2(1-\beta)]$ from Theorem 4.3 depends on the maximum value of the second-order derivative of W, and such derivative gets unbounded at the origin. Since points near the origin are never visited, in particular applications one is more interested in computing the value and policy functions over a significative domain containing the asymptotic dynamics. This allows one to get more operational estimates for M. In Table 4 we list values for M over several alternative domains. For instance, the interval [1,10] almost includes the point $\frac{1}{2}k^*$, and here $M = 1.6740$. Therefore, in this case the upper estimate $[M + 1/5(1-\beta)]h^2$ from Equation (4.1) is equal to 5.6740, whereas the observed error is bounded by $4h^2$. Consequently, the observed error falls into the range imposed by our theoretical analysis. In situations where the value function does not feature a closed-form solution, one can use instead the alternative estimate $\widehat{M}$, derived from our upper bound (3.7) of the second-order derivative of W in terms of primitive data of the model. These values are also listed in Table 4. This difference between the observed error and our upper estimates is something to be expected in practice, since our results are meant to bound the error on a worst-case basis.

We also considered further variations of the preceding exercise that we now summarize:

(1) *Two-dimensional maximization*: In the spirit of problem (7.1), all iterations could alternatively involve a maximization over two variables (say l and k). For this case, our computations (see Table 5) show that the additional times reported are always beyond one-third of those in Table 3. The two-dimensional maximization was effected by the routine *min_con_gen_lin* of *IMSL*.

Table 5
Example 7.1. Computational method: dynamic programming algorithm with linear interpolation and with two-variable maximization[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
100	10^{-1}	91	11.19	5.31×10^{-2}	2.69×10^{-2}
1000	10^{-2}	181	222.20	5.76×10^{-3}	3.68×10^{-4}
10000	10^{-3}	271	3155.24	5.96×10^{-4}	3.80×10^{-6}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

Table 6
Example 7.1. Computational method: dynamic programming algorithm with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
100	10^{-1}	460	18.71	4.77×10^{-2}	1.980×10^{-1}
1000	10^{-2}	920	378.35	5.57×10^{-3}	1.949×10^{-3}
10000	10^{-3}	1379	5367.44	5.97×10^{-4}	1.900×10^{-5}

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

(2) *Discount factors β close to 1*: Our theoretical analysis suggests that as β approaches 1, the constants involved in the orders of convergence may become unbounded. For $\beta = 0.99$ and all the above parameter values, the constant of the approximation error of the value function goes up to 20, which is over a 5-fold increase with respect to the preceding figure. This is roughly the ratio, $(\frac{1}{1-0.99})/(\frac{1}{1-0.95}) = 5$, predicted by our error analysis, upscaled by a further increase in the second-order derivative of function W [i.e., $W(k) = B + C\log k$ with $C = \lambda\alpha/(1-\alpha\beta)$]. Hence, one should expect these constants to become unbounded as β approaches unity. Table 6 reports in an analogous way further information regarding this numerical experiment with $\beta = 0.99$.

(3) *Multigrid methods*: For multigrid methods, we have considered the following simple variation of the method of successive approximations: Start with $h_1 = 10^{-1}$, and take W^{h_1} as the initial guess for the iteration process with grid level $h_2 = 10^{-2}$. Then, take W^{h_2} as the initial value to start the iterations for grid level $h_3 = 10^{-3}$. This procedure leads to considerable speedups, and the CPU time gets down to one half (cf. Tables 6 and 7).

(4) *Higher-order approximations*: Instead of the space W^h of piecewise linear functions, one could focus on alternative finite-dimensional functional spaces involving higher-order interpolations or spline functions. The original numerical experiment in Table 3 is now replicated in Table 8, using cubic splines, and in Table 9, using shape-preserving splines (i.e., quadratic splines that preserve monotonicity and

Table 7
Example 7.1. Computational method: multigrid with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time
1000	10^{-2}	460	210.67
10000	10^{-3}	932	1802.19

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

Table 8
Example 7.1. Computational method: dynamic programming algorithm with cubic spline interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
100	10^{-1}	181	9.59	3.61×10^{-4}	6.13×10^{-5}
1000	10^{-2}	361	201.41	1.74×10^{-6}	3.45×10^{-8}
10000	10^{-3}	543	3630.06	1.74×10^{-6}	8.41×10^{-11}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

Table 9
Example 7.1. Computational method: dynamic programming algorithm with shape-preserving spline interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
100	10^{-1}	136	6.99	1.51×10^{-3}	3.65×10^{-3}
1000	10^{-2}	271	141.03	1.98×10^{-5}	3.59×10^{-6}
10000	10^{-3}	540	2951.07	1.88×10^{-6}	4.42×10^{-10}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

concavity). One can observe that cubic splines have roughly the same performance as shape-preserving splines since the functions to be approximated are relatively simple and the various grids considered are relatively fine. The possible negative effects on numerical maximization from the loss of concavity under cubic splines seem to be minor; furthermore, cubic splines yield better approximations, and have a lower implementation cost – as no effort is spent in checking concavity[15]. From Tables 3, 8 and 9, we can also observe that splines yield better accuracy results per work expended than linear interpolations. For instance, we can see that one thousand points under spline interpolation (second column in Table 8) lead to much better error estimates than ten thousand points under linear interpolations

[15] For cubic splines *TOLW* is set to $\frac{1}{5}h^4$, and for shape-preserving splines *TOLW* is set to $\frac{1}{5}h^3$.

Table 10
Example 7.1. Computational method: policy iteration with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. value of L	Max. error in W	Max. error in g
100	1.00×10^{-1}	6	0.51	131.7472	4.08×10^{-4}	5.9741×10^{-2}
300	3.31×10^{-2}	6	7.84	1318.7690	9.00×10^{-5}	2.9869×10^{-2}
750	1.32×10^{-2}	8	228.93	126.9136	6.00×10^{-6}	9.3287×10^{-3}
1000	9.90×10^{-3}	11	963.73	59943.0753	6.00×10^{-6}	5.7674×10^{-3}
3000	3.30×10^{-3}	9	28197.48	218854.7858	1.00×10^{-6}	2.2403×10^{-3}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

Table 11
Example 7.1. Computational method: modified policy iteration with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	m	Max. error in W
100	10^{-1}	6	0.5037	6.24×10^{-2}	65	9.42×10^{-4}
1000	10^{-2}	8	7.2365	4.32×10^{-3}	65	1.31×10^{-5}
10000	10^{-3}	12	146.30	3.95×10^{-4}	65	9.39×10^{-7}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

(third column in Table 3). Moreover, the computational cost under cubic splines interpolation is lower: the number of vertex points has been reduced by a factor of 10, whereas for each vertex point the time cost of spline interpolation (instead of piecewise linear interpolation) goes up by less than a factor of 2. Consequently, the outcome of these exercises is that splines outperform linear interpolations by factors of 5–50, and these gains may increase exponentially for finer grids.

(5) *Policy iteration:* As previously argued, policy iteration becomes less attractive for fine grids, since the region of quadratic convergence may get smaller and the computational burden associated with inverting large matrices increases considerably. These results are confirmed in Table 10. It can be seen that the constant

$$L = \frac{\|W - W_{n+1}\|}{\|W - W_n\|^2}$$

(associated with the quadratic order of convergence of the algorithm) increases with the mesh size of the grid, which leads to further iterations. However, for fine grids the major cost incurred seems to lie in the inversion of large matrices. Comparison of Tables 3 and 11 shows that policy iteration dominates dynamic programming for small and medium-sized grids, but it has a much worse performance for grids beyond 1000 points.

Table 12
Example 7.1. Computational method: modified policy iteration with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	m	Max. error in W
100	10^{-1}	8	1.63	2.76×10^{-2}	250	1.44×10^{-3}
1000	10^{-2}	9	20.26	3.81×10^{-3}	250	8.31×10^{-4}
10000	10^{-3}	13	319.63	4.13×10^{-4}	250	5.56×10^{-6}

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$ and $\delta = 1$.

(6) *Modified policy iteration*: Considerable gains were also attained with modified policy iteration (cf. Tables 3 and 6, and Tables 11 and 12); indeed, in some cases the iteration process stops ten times faster. For this simple example, we found that the optimal number of compositions on the policy function (i.e., number m in Section 5.3) was of the order $m = 65$ for $\beta = 0.95$, and $m = 250$ for $\beta = 0.99$. Of course, an optimal m is also bound to depend on the grid size, and further savings are obtained when modified policy iteration is combined with multigrid or with spline interpolation.

(7) *PEA-collocation:* The PEA-collocation method of Christiano and Fisher (1994) yields in this case the exact solution, for $x = (\log k - \log \underline{k})/(\log \bar{k} - \log \underline{k})$. Hence, this example does not seem to be an appropriate application of this method. To this end, we shall consider a chaotic law of motion in the next example.

(8) *Depreciation parameter,* $\delta < 1$: Under full depreciation, $\delta = 1$, the model has an analytical solution, and this becomes handy to compute exactly the approximation error. However, the optimal rule is rather simple, since variable l remains constant along the optimal solution. To make sure that this property was not affecting the performance of our algorithms, the above computational experiments were replicated for parameterizations $\delta = 0.05$ and $\delta = 0.10$, with no noticeable change in the reported CPU times. As expected, though, the computational error of the policy function went up in all cases. (This error was estimated numerically, considering as the true solution that of a sufficiently fine approximation.)

7.2. *A one-sector chaotic growth model*

In order to apply the PEA-collocation algorithm, we consider a deterministic example with a chaotic law of motion. Both the return function and the policy function are polynomials, and hence the PEA-collocation algorithm is expected to perform well.

The example is taken from Boldrin and Montrucchio (1986). Here, the graph of the technological correspondence Ω is the (two-dimensional) square $[0, 1] \times [0, 1]$. The return function $v : \Omega \to R$, is defined as

$$v(k_t, k_{t+1}) = 150k_t - 24k_t^2 + 4k_t k_{t+1}(1 - k_t) - 1.608k_{t+1} - 0.32848k_{t+1}^2 + 0.17152k_{t+1}^3 - 0.08576k_{t+1}^4,$$

Table 13
Chaotic growth model (Section 7.2). Computational method: dynamic programming algorithm with linear interpolation

Vertex points	Iterations	CPU time	Max. error in g	Max. Euler eq. residuals
100	5	0.1142	2.7438×10^{-3}	2.7726×10^{-3}
1000	6	0.9487	2.9135×10^{-4}	2.9091×10^{-4}
10000	7	10.8873	2.9134×10^{-5}	2.9134×10^{-5}
20000	7	21.7121	1.4603×10^{-5}	1.4537×10^{-5}

Table 14
Chaotic growth model (Section 7.2). Computational method: dynamic programming algorithm with shape-preserving spline interpolation

Vertex points	Iterations	CPU time	Max. error in g	Max. Euler eq. residuals
100	6	0.29	3.41×10^{-5}	3.27×10^{-5}
1000	7	3.26	4.38×10^{-7}	4.40×10^{-7}
10000	9	44.11	1.97×10^{-7}	1.77×10^{-7}
20000	10	108.41	1.92×10^{-7}	1.89×10^{-7}

Table 15
Chaotic growth model (Section 7.2). Computational method: PEA-collocation

Vertex points	CPU time	Max. error in g	Max. Euler eq. residuals
3	0.1112	4.5305×10^{-2}	4.6267×10^{-2}
5	0.4958	4.0453×10^{-2}	4.0704×10^{-2}
8	0.4089	9.4166×10^{-4}	9.4413×10^{-4}
12	1.2248	2.1193×10^{-4}	2.1155×10^{-4}
15	2.4517	1.8828×10^{-4}	1.8664×10^{-4}

and the discount factor, $\beta = 0.01072$. Under this specification, the policy function $g : [0,1] \rightarrow [0,1]$ is given by the chaotic map $4k(1-k)$ that goes from the unit interval into itself.

Computation of this model using our previous discretized dynamic programming algorithm is a relatively easy task, since the discount factor β is rather low; hence, the contraction property of the operator is unusually pronounced. Indeed, we can see from Tables 13 and 14 that it takes seven value-function iterations to reach accuracy levels for the policy function of order 10^{-4}, with reported CPU times less than 1 minute. As illustrated in Table 15, such accuracy levels can also be achieved with the PEA-

collocation method of Christiano and Fisher (1994) with polynomials of degree 14. Moreover, even in this example with small discounting, PEA-collocation seems to be faster[16].

In all these computations, the good performance of the PEA-collocation algorithm is partly due to the following successful implementation of the multigrid method: The algorithm is coded so that the computed solution for a given grid size is taken as the initial guess for the subsequent refinement. Thus, the fixed point of the 3-vertex-point grid is used as the initial guess for the computed solution of the 5-vertex-point grid, which in turn is the initial guess for the computed solution of the 8-vertex-point grid, and so forth. (Our guess for the initial 3-vertex-point grid was derived from a linearization of the Euler equation at the unique interior steady-state value.) This multigrid-type iterative procedure yields faster and more reliable outcomes than the simple PEA-collocation algorithm, since the computed solution from a previous grid is generally the best available starting point for the next round of computations.

In their last columns, Tables 13–15 report the maximum size of the Euler equation residuals associated with the computed policy function. From Theorem 6.1, the size of the residuals should be of the same order of magnitude as the error of the computed policy function. This pattern is actually confirmed in all the tables. Indeed, the constants associated with these orders of convergence are close to 1 in all three cases.

7.3. *A one-sector stochastic growth model with leisure*

We next consider the stochastic version of the growth model presented in Example 2.2 for a parameterization in which the value and policy functions retain exact analytical forms. The problem is written as

$$\max_{\{c_t, l_t, i_t\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t \left[\lambda \log c_t + (1-\lambda)\log l_t\right]$$

$$\text{subject to} \begin{cases} c_t + i_t = z_t A k_t^{\alpha}(1-l_t)^{1-\alpha} \\ k_{t+1} = i_t + (1-\delta)k_t \\ \log z_{t+1} = \rho \log z_t + \varepsilon_{t+1} \\ 0 < \beta < 1, \quad 0 < \lambda \leqslant 1, \quad A > 0, \\ 0 < \alpha < 1, \quad 0 \leqslant \delta \leqslant 1, \quad 0 \leqslant \rho < 1, \\ k_t, c_t \geqslant 0, \quad 0 \leqslant l_t \leqslant 1, \\ k_0 \text{ and } z_0 \text{ given}, \quad t = 0, 1, 2, \ldots. \end{cases} \tag{7.3}$$

[16] In the implementation of the algorithm, variable x was defined as $x = 2\frac{k-0}{1-0} - 1$. The alternative implementation discussed above, $x = 2\frac{\log k - \log \underline{k}}{\log \bar{k} - \log \underline{k}} - 1$, requires $\underline{k} > 0$.

where ε_t is an i.i.d. process with zero mean. For $\delta = 1$, the value function W has an analytical form given by

$$W(k_0, z_0) = B + C \ln k_0 + D \ln z_0,$$

where

$$C = \frac{\lambda\alpha}{1-\alpha\beta}, \qquad D = \frac{\lambda}{(1-\alpha\beta)(1-\rho\beta)}.$$

Also, as previously the optimal policy is a constant fraction of total production,

$$k_{t+1} = \alpha\beta z_t A k_t^{\alpha}(1-l_t)^{1-\alpha} \quad \text{with} \quad l = \frac{(1-\lambda)(1-\alpha\beta)}{\lambda(1-\alpha)+(1-\lambda)(1-\alpha\beta)}.$$

We fix parameter values, $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.90$. Also, we restrict the feasible domain so that $k \in [0.1, 10]$, $\varepsilon \in [-0.032, 0.032]$, and z is such that $\log z \in [-0.32, 0.32]$. The random process ε comes from a normal distribution, where the density has been rescaled in order to get a cumulative mass equal to unity. As in Prescott (1986) we assume a standard deviation $\sigma_\varepsilon = 0.008$. Observe then that the end-points of the domain of variable ε are four standard deviations away from the mean.

One can again check that under these restrictions Assumptions (1)–(4) are satisfied. Indeed, in this simple case one can show that the model has a globally stable invariant distribution. As the random shock has a small variance, all paths eventually fluctuate around the point $(k^*, \bar{z}) = (1.9696, 1)$, where k^* is the state value of the deterministic model and $\bar{z}$ is roughly the unconditional mean of the random process. Consequently, as in Section 7.1, to estimate the value M of Theorem 4.3 it is reasonable to restrict ourselves to a certain domain containing the ergodic set, such as

$$P = \{(x,z) | \tfrac{1}{2} \leqslant k \leqslant 10,\ e^{-0.32} \leqslant z \leqslant e^{0.32}\}.$$

This set is large enough to encompass most plausible economic applications using this framework, and here $Mh^2 = 64.9374h^2$. As in the preceding example, to this estimate we should add the other component of the observed error concerning the fact that the iteration process is stopped in finite time.

Over the feasible domain of state variables we set out a uniform grid of vertex points (k^j, z^j) with mesh size h. Our numerical procedure then follows the iterative process specified in Equation (4.2) with an initial value $W_0 \equiv 0$.

As in Section 7.1, the algorithm is written so that only unidimensional maximizations need to be considered. Thus, each iteration proceeds as follows

$$W_{n+1}^h(k_0, z_0) = \max_{l_0} \lambda \log\left(\frac{\lambda z_0 A k_0^{\alpha} l_0(1-\alpha)}{(1-\lambda)(1-l_0)^{\alpha}}\right) + (1-\lambda)\log l_0 + \beta \int_Z W_n^h\left(\left(z_0 A k_0^{\alpha}(1-l_0)^{-\alpha}\left[(1-l_0) - \frac{\lambda l_0(1-\alpha)}{1-\lambda}\right]\right), z_1\right) Q(z_0, \mathrm{d}z_1). \tag{7.4}$$

All the integrations have been carried out under the subroutines *qsimp* and *qtrap,* as specified in Press et al. (1992, Sect. 2.4), with $TOLI = 10^{-9}$. These subroutines follow

Table 16
Example 7.3. Computational method: dynamic programming algorithm with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
43×3	0.3872	10	1.44	1.12×10^{-1}	2.61
143×9	10^{-1}	57	87.49	3.63×10^{-2}	2.41×10^{-1}
500×33	0.0282	108	2198.95	1.06×10^{-2}	6.03×10^{-2}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

Table 17
Example 7.3. Computational method: dynamic programming algorithm with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
43×3	0.3872	25	3.98	1.27×10^{-1}	10.57
143×9	10^{-1}	286	445.89	3.68×10^{-2}	1.11
500×33	0.0282	550	9839.58	1.06×10^{-2}	1.56×10^{-1}

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

an N-stage refinement of an extended trapezoidal rule, and as argued in this treatise, they are fairly efficient for the integration of relatively simple problems.

Again, for univariate numerical maximizations we have employed Brent's algorithm with tolerance level $TOLM = 10^{-8}$. The iteration process in Equation (7.4) stops when

$$\|W_{n+1} - W_n\| \leqslant TOLW = h^2.$$

It should be noted that for the implementation of the algorithm (but not for the final calculation of the approximation errors) all interpolations need only be unidimensional. That is, although W_n in Equation (7.4) is defined over a grid in a two-dimensional space, a unidimensional interpolation over z allows us to compute the integral in Equation (7.4) for given k^j. Then, the integral values are interpolated over k to define a continuous objective for the univariate maximization.

Table 16 presents information on our numerical experiment for several values of h. It can be observed that the error term is always bounded by $24h^2$. Hence, the constant stemming from our computations is bounded above by 24, whereas our estimate of the observed error,

$$\begin{aligned} e_n^h(k,z) &= \left|W^h(k,z) - W_{\hat{n}}^h(k,z)\right| \\ &\leqslant \left|W(k,z) - W^h(k,z)\right| + \left|W^h(k,z) - W_{\hat{n}}^h(k,z)\right| \\ &\leqslant 64.397h^2 + 20h^2 \leqslant 84.397h^2. \end{aligned} \tag{7.5}$$

We again emphasize that this is a result to be expected in particular applications, since these estimates are by construction rough upper bounds of the maximum

Table 18
Example 7.3. Computational method: multigrid with linear interpolation[a]

Vertex points	Mesh size	Iterations	CPU time
143×9	10^{-1}	158	220.70
500×33	0.0282	341	6370.53

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

Table 19
Example 7.3. Computational method: dynamic programming algorithm with linear interpolation and with two-variable maximization[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
43×3	0.3872	10	2.69	1.12×10^{-1}	2.61
143×9	10^{-1}	57	160.47	3.63×10^{-2}	0.241
500×33	0.0282	108	4130.08	1.06×10^{-2}	6.03×10^{-2}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

approximation error over the entire infinite horizon, and include points fairly distant from the ergodic set.

As in the preceding example, we considered alternative numerical experiments for several values for β, and replicated the original computations under the multigrid method. Table 17 reports the corresponding numerical results for discount factor $\beta = 0.99$. As it is to be expected, the constant involved in the orders of convergence are about five times larger. Likewise, Table 18 replicates the computations of Table 17 under the multigrid method. It can be seen that the required CPU time gets down roughly to two thirds of that of the original experiment. These computational costs seem very reasonable, as compared to similar, rougher procedures of this basic problem [e.g., Christiano (1990), Coleman (1990) and Tauchen (1990)]. We have also carried out the numerical experiment under the original formulation (2.4) in a two-dimensional maximization framework. As shown in Table 19 the time load doubles in all three cases.

In view of the gains obtained by splines in the preceding examples, the value and policy functions of our stochastic model were also computed by the dynamic programming algorithm with spline interpolation[17], with significant savings in computing time (cf. Tables 20 and 21). Indeed, considering accuracy achieved per work expended, multigrid with spline interpolation outperforms the dynamic programming

[17] For fine grids, the best performance was observed in experiments with shape-preserving spline interpolation over variable k, and cubic spline interpolation over variable z.

Table 20
Example 7.3. Computational method: dynamic programming algorithm with spline interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
43×3	0.3872	28	7.02	1.92×10^{-2}	9.58×10^{-1}
143×9	10^{-1}	130	342.89	3.11×10^{-3}	4.99×10^{-3}
500×33	0.0282	234	8162.89	3.82×10^{-4}	2.42×10^{-5}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

Table 21
Example 7.3. Computational method: multigrid algorithm with spline interpolation[a]

Vertex points	Mesh size	Iterations	CPU time	Max. error in g	Max. error in W
143×9	10^{-1}	112	290.05	3.11×10^{-3}	4.82×10^{-3}
500×33	0.0282	170	5935.71	3.82×10^{-4}	2.46×10^{-5}

[a] Parameter values: $\beta = 0.95$, $\lambda = \frac{1}{3}$, $A = 10$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

algorithm with linear interpolation by factors of 100–1000 for the finest grids. Thus, using our computational facilities, it would have been practically unfeasible to get tolerance levels for the value function of order 10^{-6}, whereas it took less than three hours for the dynamic programming algorithm with cubic splines.

Finally, the PEA-collocation algorithm of Christiano and Fisher was not considered in this case, since for our parameterization with full depreciation this method yields the exact solution. This algorithm will be tested in the next section, where we attempt to solve an economy with more plausible depreciation values. As evaluated by the Euler equation residuals, in all our numerical experiments PEA-collocation is much faster than the dynamic programming algorithm, and can achieve high accuracy levels.

8. Quadratic approximations

An old and popular strategy for the simulation of solutions is to approximate the model with a one-period quadratic return function and linear feasibility constraints. This approach reduces substantially the computational complexity, since the Euler equation and the policy function are linear in the state variables. Thus, it is possible to compute quadratic models in many dimensions and this procedure is very convenient whenever the approximation is accurate. Computational algorithms for solving quadratic optimization problems have been recently surveyed in Amman (1996), and Anderson et al. (1996).

Several quadratic approximation methods have been proposed in the literature, and an important issue is the accuracy of these computations. For our original optimization

problem (2.1), a standard approach would be to take a second-order Taylor's expansion of the one-period return function $v(k_t, k_{t+1}, z_t)$ at a given point $(\bar{k}, \bar{k}, \bar{z})$, and then maximize over the new objective. A typical optimization problem would thus be expressed as follows:

$$\max_{\{x_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \int_{Z^t} (x_t, x_{t+1}, z_t) \cdot D^2 v(\bar{k}, \bar{k}, \bar{z}) \cdot (x_t, x_{t+1}, z_t)\, \mu^t(z_0, \mathrm{d}z^t) \tag{8.1}$$

$$\text{subject to } (x_0, z_0) \text{ fixed, and } t = 0, 1, 2, \ldots,$$

where $D^2 v(\bar{k}, \bar{k}, \bar{z})$ is the Hessian matrix of v at $(\bar{k}, \bar{k}, \bar{z})$, and μ^t is the probability law induced by the mapping φ. Under certain mild regularity conditions [e.g. Anderson et al. (1996)], an optimal solution $\{x_t^*\}$ exists for problem (8.1) and it is unique. Let $x_1 = \eta(x_0, z_0)$ be the optimal policy for the quadratic optimization problem. Then, for $k_0 = \bar{k} + x_0$, the computed policy function $\hat{g}$ for our original optimization problem (2.1) is derived from $k_1 = \hat{g}(k_0, z_0) = \bar{k} + \eta(x_0, z_0)$.

Of course, this approximation is only supposed to be accurate for small perturbations around the vector $(\bar{k}, \bar{k}, \bar{z})$, usually assumed to be the steady state of a deterministic version of the model. But even for such small perturbations, the following problems may arise:

(i) *Biased estimates for the first-order moments*: In simple situations in which problem (2.1) generates a globally stable invariant distribution for the set of optimal solutions, the first-order sample moments of k and z are not necessarily equal to those of the deterministic steady state [e.g., see Christiano (1990) and den Haan and Marcet (1994) for some illustrative computations, and Becker and Zilcha (1997) for an analytic example]. Mean-preserving perturbations of the stochastic innovation ε may lead to shifts in first-order moments from either non-linearities in the function φ or from economic maximizing behavior, since agents may want to carry over a larger stock of capital to insure against unexpected shocks.

(ii) *Biased estimates for the slope*: Function η is not the best linear approximation at point $(\bar{k}, \bar{z})$ of the policy function g for optimization problem (2.1). As pointed out in Section 3, the derivative $D_1 g(k, z)$ is determined by the quadratic optimization problem (3.6); such optimization problem is slightly different and harder to compute, since variable z appears in a fundamental non-linear way. [See Gaspar and Judd (1997) for an alternative discussion on the computation of these derivatives.] Hence, for stochastic shocks with a large variance, optimization problems (3.6) and (8.1) may yield different solutions, and function η may not be an appropriate estimate of the derivative of g.

An alternative quadratic approximation can be obtained from a second-order Taylor's expansion of the return function over the log values of k and z. In such a case, the optimal policy of the constructed optimization problem [e.g., see Christiano (1990) and King, Plosser and Rebelo (1988)] yields the exact solution for the stochastic model with full physical capital depreciation in Equation (7.4). Hence, this optimization problem is expected to generate higher sample moments, ameliorating thus the loss

Table 22
Example 7.3. Standard deviations, $\sigma(i)$, and correlation coefficients, corr(i,j), for $i,j = k,c,u,y$[a]

Method	$\sigma(y)$	$\sigma(i)$	$\sigma(c)$	$\sigma(u)$	corr(w,u)	corr(c,y)
Exact solution	0.3524	0.1186	0.2337	0.0	1.0	1.0
Dynamic programming	0.3524	0.1186	0.2337	0.0	1.0	1.0
Quadratic approximation	0.3523	0.1185	0.2337	0.0	1.0	0.9999

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 17$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

of accuracy discussed in point (i) regarding the linear quadratic model. But the log-linear approximation does not provide an exact solution of the derivative of the policy function [cf. optimization problem (3.6)], and so it may also lead to inaccurate results, even for small shocks.

There have been several accuracy tests for quadratic approximations in the standard stochastic growth model [e.g., Christiano (1990), Danthine, Donaldson and Mehra (1989), Dotsey and Mao (1992)]. This is a topic of particular concern in the real business cycle literature, where the stochastic innovation is usually calibrated with a relatively small variance. It is then expected that the quadratic approximation would mimic reasonably well the invariant distribution of the non-linear solution. However, for the aforementioned accuracy tests the authors restrict the law of motion z to a discrete stochastic chain with three states. Such discretization cannot be generally considered as a good approximation of the underlying non-linear model. Therefore, to conduct these comparisons, more accurate simulations of the original non-linear model are needed.

We shall present here further numerical evidence from the more accurate computational procedures developed in the preceding sections. To understand the nature of these approximations, we begin our analysis with a parameterization with an analytic solution. Thus, Table 22 reports the standard deviations and correlation coefficients for physical capital, k, consumption, c, work, $u = 1 - l$, output, $y = zAk^{\alpha}u^{1-\alpha}$, and labor productivity, $w = (1-\alpha)zAk^{\alpha}u^{-\alpha}$, for the log-linear model computed in Section 7.3 with $\beta = 0.99$. These sample moments have been obtained from 10 000 draws of the i.i.d. random process $\{\varepsilon_t\}$ that enabled us to construct a random path for state variable z. The solution for the quadratic approximation is derived from optimization problem (8.1), for the deterministic steady-state $(\bar{k},\bar{k},\bar{z}) = (4.669, 4.669, 1)$. The solution of the dynamic programming algorithm is derived from the multigrid method with spline interpolation, where the finest grid contains 1000×70 vertex points evenly spread over the domain $K \times Z = [2.5, 7.1711] \times [e^{-0.32}, e^{+0.32}]$.

From Table 22 we observe that the computed moments are relatively accurate for both the quadratic approximation and the numerical dynamic programming algorithm. The most significant deviations are observed in the standard deviations of output, $\sigma(y)$, and investment $\sigma(i)$, where the quadratic approximation yields slightly smaller

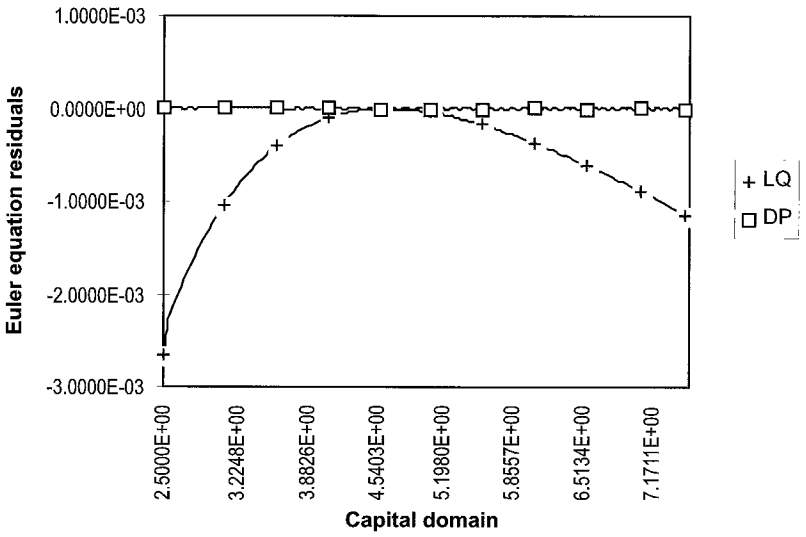

Fig. 1. Example 7.3. Euler equation residuals for the dynamic programming algorithm (DP) and quadratic approximation (LQ) policy functions for $z = 1$. Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 17$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$. In the deterministic case, (i.e., for $\sigma_\varepsilon = 0$), this parameterization yields a steady-state value $k^* = 4.669$.

estimates. However, for the chosen calibration of parameter values with $\sigma_\varepsilon = 0.008$, these differences are not substantial.

To gain further understanding of these results, Figures 1–3 plot the Euler equation residuals and the computed policy functions over the capital domain [2.5, 7.1711], for $z = 1$ fixed. In this example, the Euler equation residuals are small, and so are the approximation errors of the policy function. For the quadratic approximation, these errors are of order 10^{-3} for a sizeable portion of the domain.[18] This approximation is nearly exact near the steady-state value, $k^* = 4.669$, but it becomes less accurate as we deviate from the central point. Similar patterns and quantitative estimates are observed for alternative feasible values for z. Let $g(k,z) = \alpha\beta zAk^\alpha(1 - \bar{l})$ be the true policy function, g^{DP} the policy function derived from the dynamic programming algorithm, and g^{LQ} the solution derived from the quadratic approximation. By virtue of the concavity of g in k, the linear policy g^{LQ} overestimates the true policy for both positive and negative deviations from the steady-state value k^* (cf. Figures 2 and 3). Moreover, the concavity of g also implies that this difference grows faster for negative deviations from k^* (i.e., for low values of k). Consequently, the quadratic model overestimates output fluctuations for large values of k (i.e., for $k > k^*$) and underestimates output fluctuations for small values of k. These countervailing effects

[18] In all the figures, a number mEn means $m \times 10^n$.

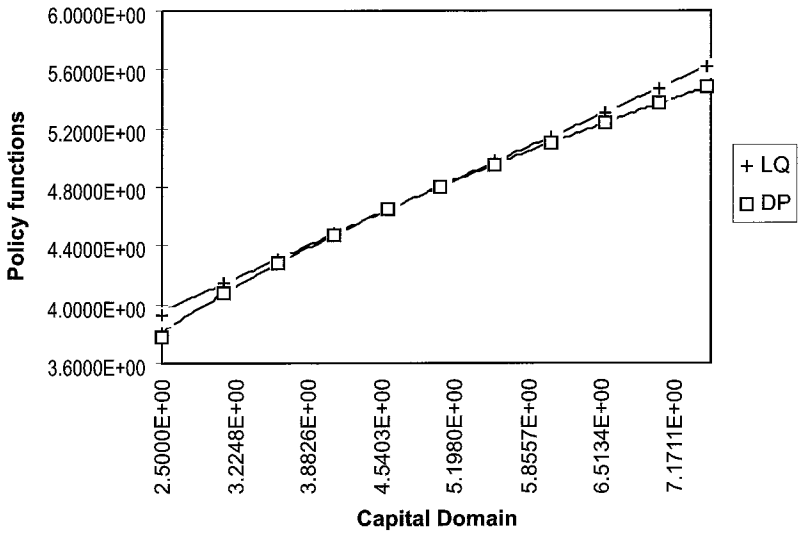

Fig. 2. Example 7.3. Policy functions for the dynamic programming algorithm (DP) and quadratic approximation (LQ) for $z = 1$. Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 17$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$. In the deterministic case, (i.e., for $\sigma_\varepsilon = 0$), this parameterization yields a steady-state value $k^* = 4.669$.

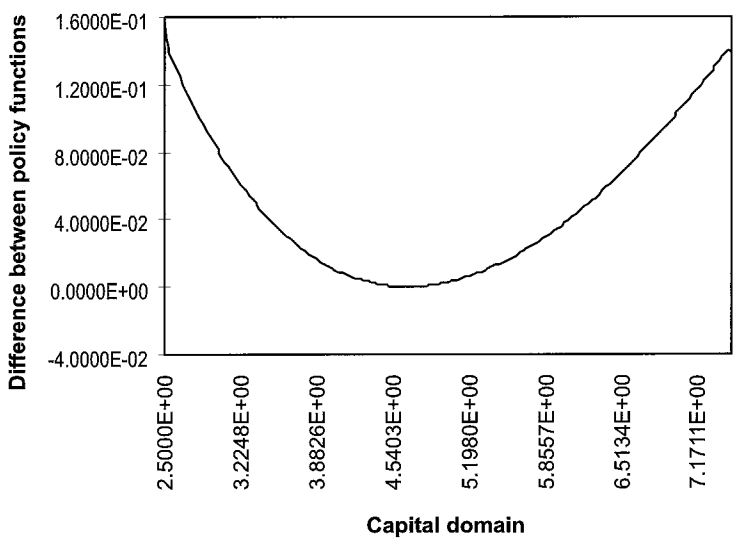

Fig. 3. Example 7.3. Difference between the computed policy functions, $g^{LQ} - g^{DP}$, for $z = 1$. Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 17$, $\alpha = 0.34$, $\delta = 1$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$. In the deterministic case (i.e., for $\sigma_\varepsilon = 0$), this parameterization yields a steady-state value $k^* = 4.669$.

Table 23
Example 7.3. Standard deviations, $\sigma(i)$, and correlation coefficients, corr(i,j), for $i,j = k,c,u,y$ [a]

Method	$\sigma(y)$	$\sigma(i)$	$\sigma(c)$	$\sigma(u)$	corr(w,u)	corr(c,y)
PEA-collocation	0.0283	0.0170	0.0142	0.0034	0.5436	0.8867
Dynamic programming	0.0287	0.0174	0.0142	0.0035	0.5498	0.8838
Quadratic approximation	0.0280	0.0167	0.0141	0.0033	0.5507	0.8917

[a] Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 1.1$, $\alpha = 0.34$, $\delta = 0.05$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$.

balance each other and lead to a similar estimate for $\sigma(y)$. Since the approximation error is more pronounced for small values of k, quadratic approximations would then have a tendency to underestimate $\sigma(y)$.

Finally, Table 23 and Figures 4 and 5 replicate the above computations for a more realistic parameterization with depreciation factor $\delta = 0.05$. This calibration of the model does not possess a close-form solution, and consequently we can only report computations corresponding to our numerical methods – quadratic approximation, PEA-collocation and the dynamic programming algorithm. Concerning the accuracy

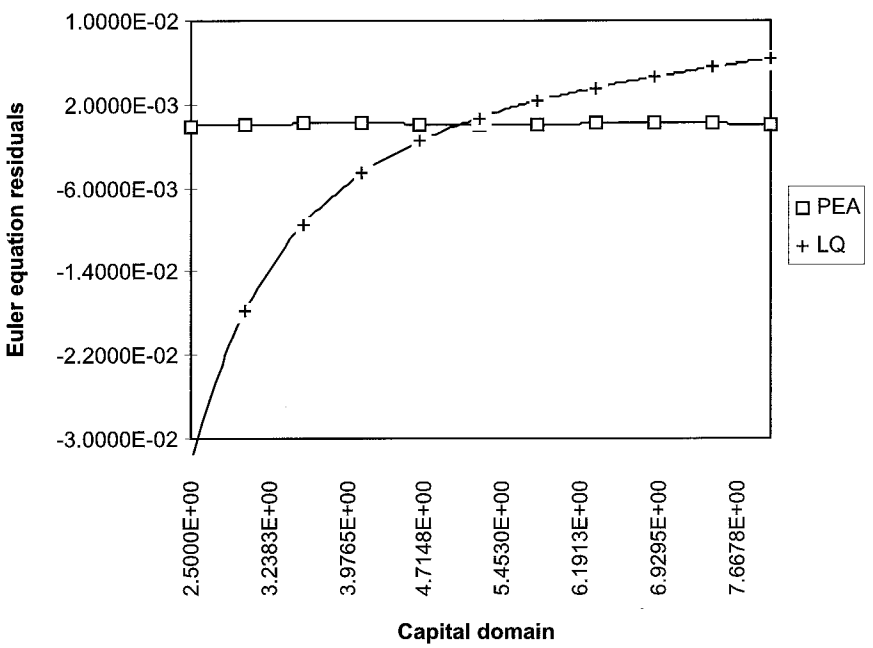

Fig. 4. Example 7.3. Euler equation residuals for PEA and quadratic approximation (LQ) policy functions for $z = 1$. Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 1.1$, $\alpha = 0.34$, $\delta = 0.05$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$. In the deterministic case, this parameterization yields a steady-state value $k^* = 5.0294$.

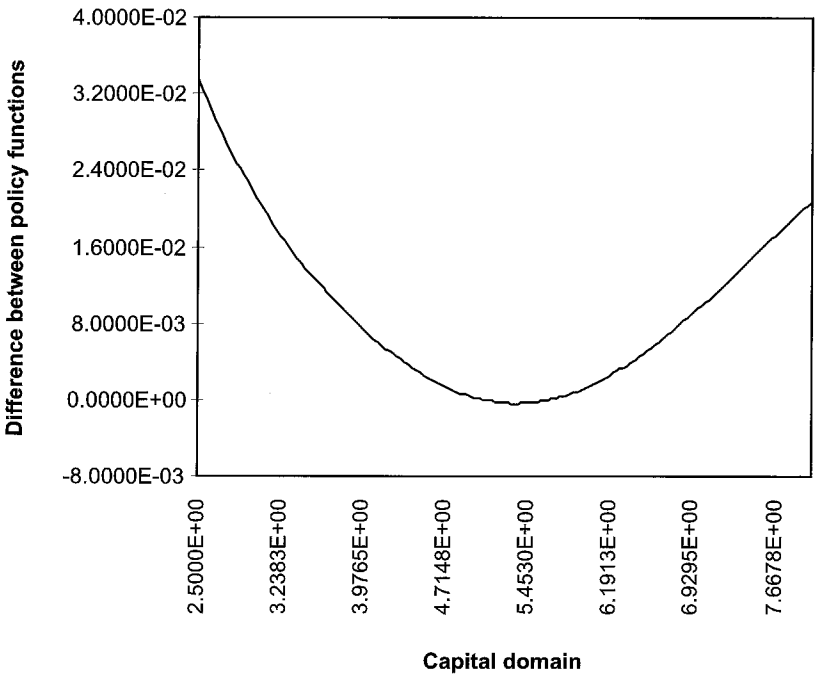

Fig. 5. Example 7.3. Difference between the computed policy functions, $g^{LQ} - g^{DP}$, for $z = 1$. Parameter values: $\beta = 0.99$, $\lambda = \frac{1}{3}$, $A = 1.1$, $\alpha = 0.34$, $\delta = 0.05$, $\rho = 0.9$ and $\sigma_\varepsilon = 0.008$. In the deterministic case (i.e., for $\sigma_\varepsilon = 0$), this parameterization yields a steady-state value $k^* = 5.0294$.

of quadratic approximations, there are no substantial variations with respect to the preceding experiment. As before, this approximation yields good estimates for the second-order moments, and the most significant differences are downward biases for the standard deviation of both output and investment, which are nevertheless fairly small. Also, Euler equation residuals and approximation errors for the linear policy function are of order 10^{-3} in a significant region. This approximation becomes increasingly less accurate as we deviate from the steady-state solution, especially for small values of k.

In this experiment, we should stress the good performance of PEA-collocation, since an 8×5 vertex-point grid achieves Euler equation residuals of order 10^{-6} for all points in the domain (cf. Figure 4). Obtaining residuals of this order of magnitude under the dynamic programming algorithm requires considerably more grid points, and a computational effort of the order of a thousand times higher. One should bear in mind that in all these computations, the standard deviation of the innovation is very small, i.e., $\sigma_\varepsilon = 0.008$. Peralta-Alva and Santos (1998) explore in detail the sensitivity of quadratic approximations to variations in the parameters β, δ, ρ and σ_ε. The most sizeable deviations regarding second-order moments are observed for variations in parameters ρ and σ_ε, although substantial changes in other parameter values may also have non-negligible effects.

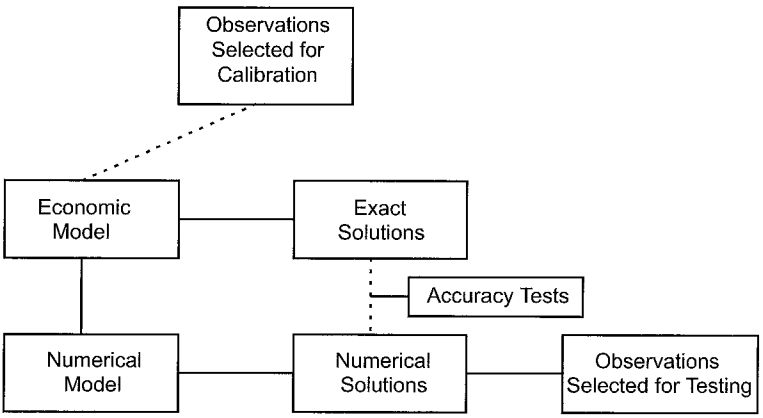

Fig. 6. Testing economic theories.

9. Testing economic theories

In this section, we focus on certain basic issues concerning the process of validation of economic models, where computer simulations must play a prominent role. Our aim is to provide a useful framework for the analysis and evaluation of numerical procedures.

As a result of some conjecture or proposed theory, we consider that a mathematical model has been constructed, with the ultimate goal of rationalizing an economic situation, phenomenon or activity. As is well understood, in general exact solutions are not readily available. Hence, for the purpose of testing an economic model one has to follow the indirect route of computing suitable approximations. To gain confidence in these computations, one needs to weigh a wide range of sources of error that go from the construction of the model to the formal testing of the solutions. A framework for this discussion is outlined in Figure 6, which highlights some major steps in the process of scientific inquiry.

In order to perform quantitative experiments, a model must be calibrated. That is, specific functional forms and corresponding parameter values must be stipulated so as to derive definite predictions. Considering that models are artificial constructs, it does not seem plausible to assign these values independently of the underlying postulates. Thus, a parameter such as the elasticity of intertemporal substitution for aggregate consumption can be difficult to identify in the data. But even if it is identified, such parameter may proxy inherent simplifications in the model such as the absence of a government sector, the absence of leisure, or of home production, or of some other observable or non-observable (i.e., not easy to measure) components not explicitly modelled. Besides, from a conceptual perspective this parameter may be attached a different meaning in environments with bounded rationality or with alternative mechanisms for forming expectations.

Evidence from panel data and microeconomic studies may help to pin down parameter values, but this evidence is not conclusive on most occasions. Thus, Browning, Hansen and Heckman (ch. 8, this volume) observe that estimates from microeconomic studies are not readily transportable to macroeconomic models. Likewise, estimating a subset of parameter values from some independent observations may lead to biases and inconsistencies in the ulterior calibration of the remaining ones [cf. Gregory and Smith (1993) and Canova and Ortega (1996)].

In view of all these complexities, a more operative strategy for calibrating a model is simply to select a set of observations where all model parameters may be jointly estimated. For instance, a common practice nowadays is to calibrate a business cycle model from some facts related to growth theory [e.g., Christiano and Eichenbaum (1992) and Cooley and Prescott (1995)] without focussing so much attention on microeconomic studies. Of course, alternative sets of observations selected for calibration may lead to different parameter values, and the ability of a certain range of parameter values to account for a wide group of well established observations may enhance our confidence in the model chosen. All these calibrations may be associated with standard errors stemming from uncertainties present in the data [cf. Christiano and Eichenbaum (1992)].

The approach just described presents a subtle difference with respect to what would be called a naive interpretation of the positivist (or falsificationist) view, often associated with the writings of Friedman (1953) and Popper (1965). Under these latter methodological programs, a model could in principle be tested without a previous confrontation to an independent or unrelated set of data. In such circumstances, functional forms and parameter values could be chosen so as to insure a *best fit* for the data or sets of data to be accounted for, without further regard to other seemingly unrelated events that the theory was not initially purported to explain.

After an economic model has been specified, one can proceed to its further analysis. Some basic properties such as existence of solutions, existence of stationary paths or invariant distributions, monotonicity, differentiability, uniqueness and stability, may be explored by classical mathematical methods. But most often these tools fail to provide us with the quantitative information necessary to test a model. In such circumstances, numerical approximations may be needed to deepen our understanding of the model's predictions.

The numerical model is not usually aimed to rationalize an economic situation. This model is generally an algorithmic device, to be implemented in a computer with a view toward the simulation of the true behavior of the original economic model. Therefore, all errors involved in these approximations should be made sufficiently small so that our inferences based upon the numerical simulations remain basically true for the exact solution. (Accuracy tests are designed to bound the error of the approximate solution.) It should be emphasized that there is no universal criterion or hard and fast rule for the approximation error that can be valid for all applications. The appropriate size of the error will depend on further considerations such as the purpose of the investigation, the sensitivity of the solution to initial conditions and parameter values,

and the conditions under which a model is tested. In other words, our ability to make good inferences from approximate solutions is not unambiguously determined by the size of the approximation error.

Since few mathematical models yield closed-form solutions, one may be tempted to specify from the start a numerical model (i.e., a collection of rules which could be coded as a finite set of computer instructions), avoiding formulation of an abstract economic model. This point of view may seem very appealing[19], but some practical considerations work in favor of formulating an abstract model. First of all, there is a well developed mathematical theory concerning regular spaces of functions over continuum quantities, which has no counterpart for discrete variables. Indeed, our most powerful mathematical tools apply to locally linear domains, with continuous or differentiable functions. Second, computations introduce approximations and round-off errors, and it becomes tedious to verify certain properties such as existence, stability, monotonicity of solutions or the constancy of a given elasticity. Validation of these results via numerical computations would require a full sampling of the state and parameter spaces, and this is not usually the most effective way to establish a given property of the solution. Thus, in most situations it is not plausible to support our entire analysis on a numerical model, and a more abstract framework is necessary. Computer simulations, though, may help us make further theoretical progress, since these computations may give way to reasonable conjectures that can be subsequently examined by mathematical analysis or by further numerical experiments.

Finally, predictions and properties of solutions of the economic model should be contrasted with the underlying real-world situation. In some simple cases, these critical comparisons may be effected without resort to a formal analysis (e.g., a model may be dismissed if its predictions about the rates of interest are very distant from the observed data). In more subtle situations, statistical techniques are usually needed.

Some recent work has focussed attention on testing the output of computer simulations [cf. Canova and Ortega (1996), Christiano and Eichenbaum (1992), Gregory and Smith (1993), Kim and Pagan (1995), and references therein.] For setting up an appropriate framework for model testing, one should realize that data sets and computer simulations are subjected to sources of uncertainty of a different nature. Data sets are characterized by measurement and sampling errors, whereas simulations involve approximations of the original model along with further numerical errors. While measurement errors may resemble round-off errors and machine failures, sampling errors may be avoided in model simulations. Indeed, the moments of an invariant distribution of a model can be accurately reproduced by resorting to arbitrarily large sample paths or by theoretical analysis, although these model statistics may not be univocally defined in the presence of multiple equilibria or multiple

[19] After all, one is ultimately interested in quantitative assessments; moreover, the numerical model may often offer a more faithful approximation of the real world situation, where some features and quantities are also discrete.

invariant distributions, and are subject to the uncertainty stemming from parameter calibrations[20].

Consequently, for evaluating the performance of a model system we require a detailed study of both the economic and numerical models, together with an analysis of the available data. Each element in this chain has its own specificities, which may give rise to different types of errors that must be accounted for in the process of testing a particular theory or conjecture. Let us briefly discuss some of the major components:

(a) *Sensitivity of solutions to initial conditions*: Small changes in the state variables may lead to fundamentally different predictions, and such pathological behavior may appear in both theoretical and numerical models, especially in environments with chaotic dynamics, multiple steady states or invariant distributions, or indeterminate equilibria. These are instances in which there are no definite predictions, and one should investigate how these instabilities unravel in the numerical model and in the data analysis. But even if solutions are reasonably well-behaved, a distinction should be made between steady-state behavior and transitional dynamics. Often, it is useful to compute the speed of convergence to a given steady state or invariant distribution.

(b) *Uncertainty of the calibrated parameters*: A model may have definite predictions regarding steady-state behavior and transitional dynamics, but such predictions are subjected to the uncertainty stemming from the parameter space. For a given set of parameter values, arbitrarily good estimates of unconditional moments of the model's invariant distributions may generally be obtained by large sets of simulations. These moments depend on parameter values, and correspondingly follow the probability law induced by the parameter space.

(c) *Rounding and chopping errors*: Computer arithmetic is not exact, and hence calculations are subject to errors. For rounding errors, it is reasonable to presume that they are normally distributed. Furthermore, their order of magnitude is small, in most cases around 10^{-15}. But care should be exercised, since in the presence of instabilities these errors may grow over the iterative scheme in many unexpected ways. Their cumulative effect over the computed solution is usually more sizeable.

(d) *Approximation errors*: These are the errors involved in the discretization of the theoretical model. Again, their cumulative effect over the computed solution may be considerably larger than that of the functions being approximated (cf. Lemma 4.2 and Theorem 4.3). Approximation errors are often impaired by systematic components, and hence their statistical modelization may become problematic. For instance, the numerical algorithm presented in Section 4 always underestimates the value function. Systematic biases may also affect the curvature of the interpolants.

[20] A further issue in model testing is to determine if the data selected correspond to a steady-state situation, or if transitional dynamics are playing an important role in the analysis.

(e) *Sampling errors*: As we only observe a limited number of realizations of a random variable, it is well understood that our inferences are subject to sampling error. Statistical theory can help us gauge this error for both small and large data samples.

(f) *Measurement errors*: This is also a classical topic of statistical theory. It is well known that imperfect measurement may bias the sample moments of our estimates.

Several recent attempts at analyzing computer simulations [see the recent surveys by Canova and Ortega (1996), and Kim and Pagan (1995)] contemplate some of these sources of error. For instance, Christiano and Eichenbaum (1992) advance an statistical framework for testing second-order moments assuming that uncertainty stems from calibrated parameter values and sample observations. As other alternative approaches, the implicit postulate for the purpose at hand is that some other errors are so small that can be safely ignored. But, in cases where sensitivity of solutions to initial conditions and approximation errors become relevant considerations, a formal analysis of these influences is needed before embedding them into a statistical framework. For the same significance level, approximation errors may change or widen the critical region, or lessen the power of a test.

If errors are of different orders of magnitude, an obvious step is to lessen the influence of the most critical ones. Approximation or numerical errors can generally be reduced at the expense of more computational effort, and the gains from reducing these errors should be evaluated against the incurred cost. Of course, these considerations apply for all other errors, but often their costs are prohibitive.

Summarizing, this section started with the basic idea that numerical analysis is a useful tool in the study of economic models, and our purpose has been to highlight several sources of error to be accounted for in testing an economic theory. Numerical analysis yields approximate solutions. Thus, to make proper inferences about an economic model, the approximation error must be sufficiently small. This error must be evaluated in conjunction with further properties of both the theoretical and numerical models, and the statistical properties of the data. It does not seem plausible to derive a universal measure of error – or any other purely econometric statistic – that can yield definite answers in all situations. And the benefits derived from a better approximation have to be balanced against the additional computational effort.

10. A practical approach to computation

This chapter has reviewed several numerical techniques for solving economic models. Our analysis has been restricted to a family of standard growth models in which optimal solutions may be decentralized as competitive allocations. This equivalence between optimal solutions and competitive allocations will break down in the presence of externalities, incompleteness of financial markets, taxation, public expenditure, money, and other frictions or governmental interventions. Additionally, there are

alternative frameworks that have not been considered here, such as overlapping generations economies or models based on game theoretical assumptions. The array of methods presented in this paper should nevertheless be of potential interest for solving these other modelizations.

Indeed, one should expect that in the near future a good share of research efforts will leave the frictionless economic framework considered in this paper and aim at a rigorous simulation of more complex economies. For a good start to some of these topics, the reader is referred to the recent monograph edited by Cooley (1995). Chapters 1–4, 7 and 12 of that volume focus especially on computational issues. For further related work and extensions, the following is a very partial list of theoretical and applied papers on computation: (a) Economics with heterogeneous agents and borrowing constraints: Castañeda, Diaz-Gimenez and Rios-Rull (1997), Huggett and Ventura (1997), Krieger (1996), Krusell and Smith (1995), and Rios-Rull (1997); (b) Economies with taxes: Bizer and Judd (1989), Chari and Kehoe (ch. 26, this volume), Coleman (1991), and Jones, Manuelli and Rossi (1993); (c) Suboptimal equilibria: Baxter (1991), and Greenwood and Huffman (1995); (d) Monetary economies: Cooley and Hansen (1989), Giovannini and Labadie (1991), and Lucas and Stokey (1987); (e) Finance models: Duffie (1996, ch. 11), Boyle, Broadie and Glasserman (1997), and Heaton and Lucas (1996); (f) Overlapping generations economies: Auerbach and Kotlikoff (1987), Kehoe and Levine (1985), and Kehoe, Levine, Mas-Colell and Woodford (1991); (g) Asymmetric information: Prescott (1997), and Phelan and Townsend (1991); (h) Game theory: McKelvey and McLennan (1996), and Dutta and Sundaram (1993).

Given the ample variety of economic models and computational techniques, a researcher will most often be faced with the choice of an appropriate numerical method. Our purpose now is to outline a set of preliminary steps which may prove useful in the selection and implementation of a numerical procedure.

The first basic principle to bear in mind is that there is no numerical method that will perform better in all situations. Hence, the choice will generally be complex, and the numerical procedure must be suited to the analytic nature of the problem under consideration. At this stage, a theoretical study of the model is most helpful. Qualitative properties of optimal solutions, such as existence or differentiability, should shed light on the error stemming from different approximations. The existence and stability properties of steady states or invariant distributions will help determine the mesh size or order of the approximant, as well as the most suitable restriction of the domain. In addition, a theoretical analysis of the model should provide valuable clues in the computation process, such as the choice of an initial guess for the solution, the efficient manipulation of state and control variables to simplify the model, or the appropriate subroutines for integration, maximization, and related operations. Also, it is useful to undertake a theoretical analysis of the numerical model, and examine differences between the dynamic behavior of these solutions and their continuum analogues.

A second point to be stressed is that for smooth, concave models involving one or two state variables there are generally reliable algorithms that can compute the solution

in reasonable time at a desired level of accuracy. As already stressed, subroutines for integration and maximization in one dimension are fairly efficient. Moreover, technological developments will facilitate the application of reliable methods in a near future. It seems then that the use of less rigorous or less reliable approximation procedures becomes more attractive for more difficult, time consuming computational problems. Although quadratic approximations and methods approximating the Euler equation performed remarkably well for all models considered in this paper, it should be realized that in most cases these models can be solved via the discretized dynamic programming algorithm combined with spline interpolation. And an efficient use of the multigrid algorithm (and to a lesser extent, with policy and modified policy iteration) may help minimize the computing time.

There are, however, computational problems in which the use of reliable methods becomes awkward or infeasible. These models may simply lack concavity, interiority of solutions, or smoothness, or involve several state variables. Techniques for the computation of non-smooth or discrete problems are generally less powerful, and sometimes model-specific; hence, at this general level of discussion, it seems difficult to offer specific guidelines. On the other hand, for the computation of large-scale models, we suggest the following sequential procedure:

(i) Quadratic approximations;
(ii) Globally convergent numerical methods in which accuracy can be controlled;
(iii) Faster computational procedures, which may lack global convergence or a formal derivation of error bounds.

If the model contains a globally stable steady state or invariant distribution, then it seems natural to start with a quadratic approximation. To assess the accuracy of this approach it may be helpful to calculate the (exact) derivatives of the value and policy functions, or simply check the Euler equation residuals. From these residuals, one may be able to estimate approximation errors for the value and policy functions (cf. Section 6.2). The quadratic model may also provide a good initial guess of the solution for the remaining, more sophisticated computational methods.

Reliable algorithms, which are amenable to a progressive control of the approximation error, can also be useful in the computation of large-scale dynamic problems. In these cases, one may resort to coarse grids with higher-order approximants. (For instance, Johnson et al. (1993) solve a four-dimensional dynamic model with a discretized version of the dynamic programming algorithm with spline interpolation.) Comparisons of outcomes with progressively finer grids – or against simple test cases with closed-form solutions – may serve to appraise numerically the constants involved in the approximation errors. Likewise, a close examination of the Euler equation residuals may allow us to evaluate the accuracy of these methods. Reliable algorithms should also provide reasonable initial guesses for the computation of a model under faster numerical methods.

Sometimes, the only feasible route for solving a model is via faster algorithms, which may lack global convergence or are not easily amenable to a formal error analysis. The efficient design of such computational procedures often involves

a combined application of standard techniques from numerical analysis in ways suggested by a previous analysis of error for reliable algorithms. There are also some subtle issues concerning implementation of these procedures. As suggested in Section 6, one should check for existence and uniqueness of solutions; moreover, application of Newton-type methods to find zeroes of non-linear equations requires that the system be locally well-conditioned. For the practical operation of the algorithm, it may be helpful to start with an initial candidate from (i) or (ii), and then compute successively finer approximants, taking as initial guess in each step the solution obtained from the previous approximation. Comparisons of the outcomes and coefficients obtained from successive approximants may shed light on the stability and accuracy of the numerical method. Moreover, an evaluation of the Euler equation residuals may be the most effective way to estimate numerically the approximation error [cf. Santos (1999)].

References

Amman, H.M. (1996), "Numerical methods for linear-quadratic models", in: H.M. Amman, D.A. Kendrick and J. Rust, eds., Handbook of Computational Economics (Elsevier, Amsterdam) 588–618.

Anderson, E.W., L.P. Hansen, E.R. McGrattan and T.J. Sargent (1996), "Mechanics of forming and estimating dynamic linear economies", in: H.M. Amman, D.A. Kendrick and J. Rust, eds., Handbook of Computational Economics (Elsevier, Amsterdam) 173–252.

Araujo, A. (1991), "The once but not twice differentiability of the policy function", Econometrica 59:1383–1391.

Auerbach, A.J., and L.J. Kotlikoff (1987), Dynamic Fiscal Policy (Cambridge University Press, Cambridge).

Baxter, M. (1991), "Approximating suboptimal dynamic equilibria: an Euler equation approach", Journal of Monetary Economics 27:173–200.

Bazaraa, M.S., D.H. Sherali and C.M. Shetty (1993), Nonlinear Programming (Wiley, New York).

Becker, G.S. (1993), "Nobel lecture: The economic way of looking at behavior", Journal of Political Economy 101:385–409.

Becker, R., and I. Zilcha (1997), "Stationary Ramsey equilibrium under uncertainty", Journal of Economic Theory 75(1):122–140.

Bellman, R. (1955), "Functional equations in the theory of dynamic programming V. Positivity and quasilinearity", Proceedings of the National Academy of Sciences, USA 41:743–746.

Bellman, R. (1957), Dynamic Programming (Princeton University Press, Princeton, NJ).

Bellman, R., R. Kalaba and B. Kotkin (1963), "Polynomial approximation – a new computational technique in dynamic programming: Allocation processes", Mathematics of Computation 17:155–161.

Benveniste, L.M., and J.A. Scheinkman (1979), "On the differentiability of the value function in dynamic models of economics", Econometrica 47:727–732.

Bertsekas, D.P. (1975), "Convergence of discretization procedures in dynamic programing", IEEE Transactions on Automatic Control 20:415–419.

Bizer, D., and K. Judd (1989), "Taxation and uncertainty", American Economic Review 79:331–336.

Blackwell, D. (1965), "Discounted dynamic programming", Annals of Mathematical Statics 36:226–235.

Blume, L.E., D. Easley and M. O'Hara (1982), "Characterization of optimal plans for stochastic dynamic programs", Journal of Economic Theory 28:221–234.

Boldrin, M., and L. Montrucchio (1986), "On the indeterminacy of capital accumulation paths", Journal of Economic Theory 40:26–39.

Bona, J.L., and M.S. Santos (1997), "On the role of computation in economic theory", Journal of Economic Theory 72:241–281.

Boyle, P., M. Broadie and P. Glasserman (1997), "Monte Carlo methods for security pricing", Journal of Economic Dynamics and Control 21:1267–1321.

Brock, W.A., and L.J. Mirman (1972), "Optimal economic growth and uncertainty: the discounted case", Journal of Economic Theory 4:479–513.

Canova, F., and E. Ortega (1996), "Testing calibrated general equilibrium models", mimeograph (Department of Economics, Universitat Pompeu Fabra, Barcelona, Spain).

Castañeda, A., J. Diaz-Gimenez and J.-V. Rios-Rull (1997), "Unemployment spells, cyclically moving factor shares and income distribution dynamics", mimeograph (Federal Reserve Bank of Minneapolis).

Chow, C.-S., and J.N. Tsitsiklis (1991), "An optimal one-way multigrid algorithm for discrete-time stochastic control", IEEE Transactions on Automatic Control 36:898–914.

Christiano, L.J. (1990), "Linear-quadratic approximation and value-function iteration: a comparison", Journal of Business and Economic Statistics 8:99–113.

Christiano, L.J., and M. Eichenbaum (1992), "Current real-business-cycle theories and aggregate labor-market fluctuations", American Economic Review 82:430–450.

Christiano, L.J., and J. Fisher (1994), "Algorithms for solving dynamic models with occasionally binding constraints", manuscript (University of Western Ontario).

Coleman, W.J. (1990), "Solving the stochastic growth model by policy-function iteration", Journal of Business and Economic Statistics 8:27–29.

Coleman, W.J. (1991), "Equilibrium in a production economy with income taxes", Econometrica 59: 1091–1104.

Cooley, T.F. (1995), Frontiers of Business Cycle Research (Princeton University Press, Princeton, NJ).

Cooley, T.F., and G.D. Hansen (1989), "The inflation tax in a real business cycle model", American Economic Review 79:733–748.

Cooley, T.F., and E.C. Prescott (1995), "Economic growth and business cycles", in: T.F. Cooley, ed., Frontiers of Business Cycle Research (Princeton University Press, Princeton, NJ) 39–65.

Dahlquist, G., and A. Bjorck (1974), Numerical Methods (Prentice-Hall, Englewood Cliffs, NJ).

Danthine, J.-P., J.B. Donaldson and R. Mehra (1989), "On some computational aspects of equilibrium business cycle theory", Journal of Economic Dynamics and Control 13:449–470.

Davis, P.J., and P. Rabinowitz (1984), Methods of Numerical Integration (Academic Press, New York).

den Haan, W.J., and A. Marcet (1990), "Solving the stochastic growth model by parameterizing expectations", Journal of Business and Economic Statistics 8:31–34.

den Haan, W.J., and A. Marcet (1994), "Accuracy in simulations", Review of Economic Studies 61:3–17.

Denardo, E.V. (1967), "Contraction mappings in the theory underlying dynamic programming", SIAM Review 9:165–177.

Dotsey, M., and C.S. Mao (1992), "How well do linear approximation methods work?", Journal of Monetary Economics 29:25–58.

Duffie, D. (1996), Dynamic Asset Pricing Theory (Princeton University Press, Princeton, NJ).

Dutta, P.K., and R.K. Sundaram (1993), "Markovian games and their applications I: Theory", mimeograph (Columbia University).

Falcone, M. (1987), "A numerical approach to the infinite horizon problem of deterministic control theory", Applied Mathematics and Optimization 15:1–13.

Fox, B.L. (1973), "Discretizing dynamic programs", Journal of Optimization Theory and Applications 11:28–234.

Friedman, M. (1953), "The methodology of positive economics", in: M. Friedman, ed., Essays in Positive Economics (Chicago University Press, Chicago, IL).

Gallego, A.M. (1993), "On the differentiability of the value function in stochastic growth models", manuscript (Universidad de Alicante).

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

Gear, C.W. (1971), Numerical Initial Value Problems and Ordinary Differential Equations (Prentice Hall, Englewood Cliffs, NJ).

Geweke, J. (1996), "Monte Carlo simulation and numerical integration", in: H.M. Amman, D.A. Kendrick and J. Rust, eds., Handbook of Computational Economics (Elsevier, Amsterdam) 731–800.

Gill, P.E., W. Murray and M.H. Wright (1981), Practical Optimization (Academic Press, New York).

Giovannini, A., and P. Labadie (1991), "Asset prices and interest rates in cash-in-advance models", Journal of Political Economy 99:1215–1251.

Greenwood, J., and G.W. Huffman (1995), "On the existence of nonoptimal equilibria in dynamic stochastic economies", Journal of Economic Theory 65:611–623.

Gregory, A.W., and G.W. Smith (1993), "Statistical aspects of calibration in macroeconomics", in: G.S. Maddala, C.R. Rao and H.D. Vinod, eds., Handbook of Statistics, vol. 11 (Elsevier, Amsterdam) 703–719.

Hammerlin, G., and K.-H. Hoffmann (1991), Numerical Mathematics (Springer, Berlin).

Heaton, J., and D.J. Lucas (1996), "Evaluating the effects of incomplete markets on risk sharing and asset pricing", Journal of Political Economy 104:443–487.

Hiriart-Urruti, J.B., and C. Lemarechal (1993), Convex Analysis and Minimization Algorithms II: Advanced theory and bundle methods (Springer, Berlin).

Howard, R. (1960), Dynamic Programming and Markov Processes (MIT Press, Cambridge, MA).

Huggett, M., and G. Ventura (1997), "Understanding why high income households save more than low income households", mimeograph (Centro de Investigacion Economica, ITAM, Mexico City).

Johnson, S.A., J.R. Stedinger, C.A. Shoemaker, Y. Li and J.A. Tejada-Guibert (1993), "Numerical solution of continuous-state dynamic programs using linear and spline interpolation", Operations Research 41(3):484–500.

Jones, L.E., R.E. Manuelli and P.E. Rossi (1993), "Optimal taxation in models of endogenous growth", Journal of Political Economy 101:485–517.

Judd, K.L. (1992), "Projection methods for solving aggregate growth models", Journal of Economic Theory 58:410–452.

Judd, K.L. (1996), "Approximation, perturbation, and projection methods in economic analysis", in: H.M. Amman, D.A. Kendrick and J. Rust, eds., Handbook of Computational Economics (Elsevier, Amsterdam) 511–585.

Judd, K.L., and A. Solnick (1997), "Numerical dynamic programing with shape-preserving splines", mimeograph (Stanford University).

Kahaner, D., C. Moler and S. Nash (1989), Numerical Methods and Software (PTR Prentice Hall, Englewood Cliffs, NJ).

Kehoe, T.J. (1991), "Computation and multiplicity of equilibrium", in: W. Hildenbrand and H. Sonnenschein, eds., Handbook of Mathematical Economics (North-Holland, Amsterdam) 2049–2143.

Kehoe, T.J., and D.K. Levine (1985), "Comparative statics and perfect foresight", Econometrica 53: 433–454.

Kehoe, T.J., D.K. Levine, A. Mas-Colell and M. Woodford (1991), "Gross substitutibility in large square economies", Journal of Economic Theory 54:1–25.

Kim, K., and A. Pagan (1995), "The econometric analysis of calibrated macroeconomic models", in: H. Pesaran and M. Wickens, eds., Handbook of Applied Econometrics, vol. 1 (Blackwell Press, London) 356–390.

King, R.G., C.I. Plosser and S.T. Rebelo (1988), "Production, growht and business cycles. I. The basic neoclassical model", Journal of Monetary Economics 21:195–232.

Kitanidis, P.K., and E. Foufoula-Georgiou (1987), "Error analysis of conventional discrete and gradient dynamic programming", Water Resources Research 23:845–848.

Krieger, S. (1996), "The general equilibrium dynamics of investment, scrapping and reorganization", mimeograph (University of Chicago).
Krusell, P., and A.A. Smith (1995), "Income and wealth heterogeneity in the macroeconomy", mimeograph (University of Pennsylvania).
Ladron de Guevara, A., S. Ortigueira and M.S. Santos (1997), "Equilibrium dynamics in two-sector models of endogenous growth", Journal of Economic Dynamics and Control 21:115–143.
Lambert, J.D. (1991), Numerical Methods for Ordinary Differential Systems (Wiley, New York).
Li, J.X. (1993), Essays in mathematical economics and economic theory, Ph.D. dissertation (Department of Economics, Cornell University).
Lorentz, A.L. (1992), Multivariate Birkhoff Interpolation (Springer, New York).
Lucas, R.E., and N.L. Stokey (1987), "Money and interest in a cash-in-advance economy", Econometrica 55:491–514.
Marcet, A. (1994), "Simulation analysis of dynamic stochastic models: application to theory and estimation", in: C.A. Sims, ed., Advances in Econometrics, Sixth World Congress, vol. II (Cambridge University Press, Cambridge) 91–118.
Marcet, A., and D.A. Marshall (1994), "Solving non-linear rational expectations models by parameterized expectations", manuscript (Universitat Pompeu Fabra, Barcelona, Spain).
McGrattan, E.R. (1996), "Solving the stochastic growth model with a finite element method", Journal of Economic Dynamics and Control 20:19–42.
McKelvey, R.D., and A. McLennan (1996), "Computation of equilibria in finite games", in: H.M. Amman, D.A. Kendrick and J. Rust, eds., Handbook of Computational Economics (Elsevier, Amsterdam) 87–142.
Montrucchio, L. (1987), "Lipschitz continuous policy functions for strongly concave optimization problems", Journal of Mathematical Economics 16:259–273.
Morton, T.E. (1971), "On the asymptotic convergence rate of cost differences for Markovian decison processes", Operations Research 19:244–248.
Mulligan, C.B. (1993), "Computing transitional dynamics in recursive growth models: the method of progressive paths", manuscript (University of Chicago).
Natanson, I.P. (1965), Constructive Function Theory, vol. 3 (Frederic Ungar Publishing Company, New York).
Niederreiter, H. (1992), Random Number Generation and Quasi-Monte Carlo Methods (SIAM, Philadelphia, PA).
Papageorgiou, A., and J.F. Traub (1996), "New results on deterministic pricing of financial derivatives", working paper no. 9606–040 (Santa Fe Institute).
Paskov, S.H. (1996), "New methodologies for valuing securities", in: S. Pliska and M. Dempster, eds., Mathematics of Derivative Securities (Isaac Newton Institute, Cambridge).
Peralta-Alva, A., and M.S. Santos (1998), "Accuracy of quadratic approximations in stochastic models of economic growth", mimeograph (University of Minnesota).
Phelan, C., and R.M. Townsend (1991), "Computing multi-period information and constrained optima", Review of Economic Studies 59:853–881.
Popper, K. (1965), The Logic of Scientific Discovery (Harper Torchbooks, New York).
Prescott, E.C. (1986), "Theory ahead of business cycle measurement", Quarterly Review, Federal Reserve Bank of Minneapolis 10(4):9–22.
Prescott, E.S. (1997), "Computing private information problems with dynamic programming methods", mimeograph (Federal Reserve Bank of Richmond).
Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery (1992), Numerical Recipes in FORTRAN. The Art of Scientific Computing (Cambridge University Press, Cambridge).
Puterman, M.L., and S.L. Brumelle (1979), "On the convergence of policy iteration in stationary dynamic programing", Mathematics of Operations Research 4:60–69.
Puterman, M.L., and M.C. Shin (1978), "Modified policy iteration algorithms for discounted Markov decision problems", Management Science 24:1127–1137.

Rios-Rull, J.-V. (1997), "Computation of equilibria in heterogeneous agent models", Staff Report no. 231 (Federal Reserve Bank of Minneapolis).
Rivlin, T.J. (1969), An Introduction to the Approximation of Functions (Dover Publications, New York).
Rivlin, T.J. (1990), Chebyshev Polynomials (Wiley, New York).
Rockafellar, R.T. (1970), Convex Analysis (Princeton University Press, Princeton, NJ).
Rust, J. (1987), "Optimal replacement of GMC bus engines: an empirical model of Harold Zurcher", Econometrica 55:999–1033.
Rust, J. (1996), "Numerical dynamic programming in economics", in: H.M. Amman, D.A. Kendrick and J. Rust, eds., Handbook of Computational Economics (Elsevier, Amsterdam) 619–729.
Santos, M.S. (1991), "Smoothness of the policy function in discrete-time economic models", Econometrica 59:1365–1382.
Santos, M.S. (1994), "Smooth dynamics and computation in models of economic growth", Journal of Economic Dynamics and Control 18:879–895.
Santos, M.S. (1999), "Accuracy of numerical solutions using the Euler equation residuals", Econometrica, forthcoming.
Santos, M.S., and J. Vigo (1995), "Accuracy estimates for a numerical approach to stochastic growth models", Discussion paper no. 107, IEM (Federal Reserve Bank of Minneapolis).
Santos, M.S., and J. Vigo (1998), "Analysis of a numerical dynamic programming algorithm applied to economic models", Econometrica 66:409–426.
Schumaker, L.L. (1981), Spline Functions: Basic Theory (Wiley/Interscience, New York).
Schumaker, L.L. (1983), "On shape preserving quadratic spline interpolation", SIAM Journal of Numerical Analysis 20(4):854–864.
Shor, N.Z. (1985), Minimization methods for nondifferentiable functions (Springer, Berlin).
Stoer, J., and R. Bulirsch (1993), Introduction to Numerical Analysis (Springer, New York).
Stokey, N.L., and R.E. Lucas (1989), Recursive Methods in Economic Dynamics (Harvard University Press, Cambridge, MA).
Stroud, A.H. (1972), Approximate Calculations of Multiple Integrals (Prentice Hall, Englewood Cliffs, NJ).
Tan, K.S., and P.P. Boyle (1997), "Applications of scramble low discrepancy sequences to the valuation of complex securities", mimeograph (University of Waterloo).
Tauchen, G. (1990), "Solving the stochastic growth model by using quadrature methods and value-function iterations", Journal of Business and Economic Statistics 8:49–51.
Taylor, J.B., and H. Uhlig (1990), "Solving non-linear stochastic growth models: a comparison of alternative solution methods", Journal of Business and Economic Statistics 8:1–18.
Traub, J.F., and H. Wozniakowski (1979), "Convergence and complexity of Newton iteration for operator equations", Journal of the Association of Computing Machinery 26:250–258.
Whitt, W. (1978), "Approximations of dynamic programs, I", Mathematics of Operations Research 3:231–243.
Whitt, W. (1979), "Approximations of dynamic programs, II", Mathematics of Operations Research 4:179–185.
Wright, B.D., and J.C. Williams (1984), "The welfare effects of the introduction of storage", Quarterly Journal of Economics 99:169–182.
Xu, Y. (1996), "Lagrange interpolation on Chebyshev points of two variables", Journal of Approximation Theory 87:220–238.