Federal Reserve Bank of Minneapolis
Research Department Staff Report 164/JV

Solving the Stochastic Growth Model with a Finite Element Method

Ellen R. McGrattan*

Federal Reserve Bank of Minneapolis

ABSTRACT

Since it is the dominant paradigm of the business cycle and growth literatures, the stochastic growth model has been used to test the performance of alternative numerical methods. In this paper I apply the finite element method to this model. I show that the method is easy to apply and that, for examples such as the stochastic growth model, it gives accurate solutions within a second or two on a desktop computer. I also show how inequality constraints can be handled by redefining the optimization problem with penalty functions.

*This article is based in part on conversations with Jeff Eischen at North Carolina State University. I thank Graham Candler, Tom Sargent, Martie Starr, and three anonymous referees for helpful comments and the National Science Foundation for its support under grant SES-9108758. The views expressed herein are those of the author and not necessarily those of the Federal Reserve Bank of Minneapolis or the Federal Reserve System.

1. Introduction

For many applications, economists must rely on numerical methods to compute equilibria in dynamic models. If the economic decision variables are linear (or approximately linear) functions of the state variables, then typically the computational task is small (as in Kydland and Prescott, 1982). But if the problem is not suited to linear approximations, then the computational task can be significant (as in Braun and McGrattan, 1993). In this paper, I describe a numerical method that can be applied in such cases.

Since the dominant paradigm of the business cycle and growth literatures has been the stochastic growth model, most practitioners have focused on it when testing their numerical methods. Taylor and Uhlig (1990), for example, compare a variety of different algorithms for computing equilibria in the stochastic growth model.[1] They find that none of these algorithms perform well in all respects, and they illustrate, through a battery of tests, the need for better, less computer-intensive methods.

Here, I apply a method that is widely used in engineering applications such as structural analysis and aerodynamic design to compute the equilibrium of a growth model. This method, called the *finite element method*, is an algorithm for solving functional equations and, for certain problems, is both fast and accurate. Using the Taylor and Uhlig (1990) application, I demonstrate that the finite element method works extremely well when applied to a case with an analytical solution. For cases without such a solution, I show that the method yields decision functions similar to discretized dynamic programming in a fraction of the computing time. Finally, I show that the method can also be applied to problems with inequality constraints. In a growth example, inequality constraints arise from the assumption that investment is positive for all realizations of the capital stock and the shock to technology. Even for a constrained problem, the finite element method performs well.

In their study of alternative methods for solving the stochastic growth model, Taylor and Uhlig (1990) focus on the tradeoff between speed and accuracy. A relevant metric that they do not consider is the storage requirement of the algorithm. For dynamic programs with many state variables, what has been called the *curse of dimensionality* renders many

[1] Computational methods were the subject of a conference held at the Federal Reserve Bank of Minneapolis in 1988. The result of the meeting was a collection of papers, which includes Taylor and Uhlig (1990), in volume 8 of the *Journal of Business and Economic Statistics*.

methods infeasible. In this regard, the finite element method offers several advantages over many of the algorithms that Taylor and Uhlig (1990) analyze. With the finite element method, the first step in solving the functional equation is to subdivide the domain of the state space into nonintersecting subdomains called *elements*. The domain is subdivided because the method relies on fitting low-order polynomials on subdomains of the state space rather than high-order polynomials on the entire state space. The result is a system of equations that is sparse. Furthermore, as the dimensionality of the problem increases, higher order functions can be used where needed, with fewer grid points. Or, adaptive grid techniques can be used to better resolve the grid in regions of the state space where nonlinearities occur.

In section 2 of this paper, I describe one- and two-dimensional versions of the stochastic growth model. In section 3, I describe the finite element method in detail. In section 4, I apply the method to three specific examples. For the first, I choose a parameterization of the model that allows for analytical solutions. This example serves as a test case for evaluating the accuracy of the algorithm. The second example is a case considered by Taylor and Uhlig (1990). In the third example, I impose non-negativity constraints on investment which bind in some regions of the state space. I make some concluding remarks in section 5.

2. The stochastic growth model

The stochastic growth model assumes that output in period t can be allocated either to current consumption c_t or to current investment i_t. The consumption-savings decision is assumed to be optimal in that the preferences of households are maximized. The preferences are given by

$$E\left[\sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\tau}-1}{1-\tau} \,\middle|\, k_{-1}\right], \qquad 0<\beta<1,\ \tau>0, \tag{1}$$

where k_t is the capital stock at t and k_{-1} is known. The maximization of (1) is done subject to the feasibility constraints,

$$c_t + k_t - (1-\delta)k_{t-1} = \theta_t k_{t-1}^{\alpha}, \qquad 0<\alpha<1,\ 0\le\delta\le 1, \tag{2}$$

and to the non-negativity constraints $c_t \ge 0$, $k_t \ge 0$, for all $t \ge 0$; assume that the process for the level of technology θ_t is given. The term $k_t-(1-\delta)k_{t-1}$ is investment in period t.

Note that δ of the stock depreciates between periods $t-1$ and t. The term $\theta_t k_{t-1}^{\alpha}$ is the output produced with k_{t-1} units of the capital stock if the level of technology is θ_t.

In the examples of this paper, I use one of two Markovian specifications for the technology process. The first is a Markov chain with transition probabilities $\Pi_{i,j} = Pr[\theta_t = \theta(j)|\theta_{t-1} = \theta(i)]$, $1 \leq i, j \leq I$. The second specification is the continuous-valued autoregressive process:

$$\ln \theta_t = \rho \ln \theta_{t-1} + \varepsilon_t, \qquad -1 < \rho < 1, \tag{3}$$

where ε_t is a serially uncorrelated, normally distributed random variable with mean zero and variance σ^2. I consider both cases in order to illustrate how the finite element method can be applied in one- and two-dimensional problems. Although there are two state variables for both specifications of θ_t, the case of the Markov chain involves only one continuous-valued state; therefore, I approximate I one-dimensional functions.

If the technology shock is drawn from a Markov chain, then the first-order conditions for the optimization problem imply that the following condition must hold for all feasible capital stocks k and all $i \in \{1, 2, \ldots, I\}$:

$$R(k, i; c) = c(k, i)^{-\tau} - \beta \sum_{j=1}^{I} \Pi_{i,j}\, c(\tilde{k}, j)^{-\tau} (\alpha \theta(j) \tilde{k}^{\alpha-1} + 1 - \delta) = 0, \tag{4}$$

where $\tilde{k} = \theta(i)k^{\alpha} + (1-\delta)k - c(k, i)$ and $c(0, i) = 0$. Let $\Omega = [0, \bar{k}]$ where $\bar{k}$ is the maximum capital stock that can be sustained. The problem is, therefore, to find the function $c(k, i)$ that satisfies eq. (4) for all $k \in \Omega$ and all $i \in \{1, \ldots, I\}$.[2]

If the technology shock is an autoregressive process, the first-order conditions for the optimization problem imply that the following condition must hold for all $(k, z) \in \Omega$,

$$R(k, z; c) = c(k, z)^{-\tau} - \frac{\beta}{\sqrt{\pi}} \int_{-\infty}^{\infty} c(\tilde{k}, \tilde{z})^{-\tau} \left(\alpha \tilde{k}^{\alpha-1} \sqrt{\frac{1+\tilde{z}}{1-\tilde{z}}} + 1 - \delta \right) e^{-\nu^2}\, d\nu = 0, \tag{5}$$

where

$$\tilde{k} = k^{\alpha} \sqrt{(1+z)/(1-z)} + (1-\delta)k - c(k, z),$$
$$\tilde{z} = \tanh(\rho \tanh^{-1}(z) + \sqrt{2}\sigma\nu),$$

[2] Stokey and Lucas (1989) discuss the assumptions needed to show that the Euler equations plus a transversality condition are sufficient conditions for optimality. An alternative approach to that taken here is to construct a finite element approximation of the value function which solves Bellman's functional equation.

$c(0,z) = 0$, ν is distributed normally with mean zero and variance 1/2, and $\Omega = [0, \bar{k}] \times [-1, 1]$. Notice that before specifying this Euler equation, I made two transformations. First, I set $z = \tanh(\ln(\theta))$ since z has a compact support – namely $[-1, 1]$ – while θ does not. Second, I set $\nu = \varepsilon/(\sqrt{2}\sigma)$. Here ν has a density function equal to $\exp(-\nu^2)$ which is in a more convenient form for applying a Gauss-Hermite quadrature rule than the density function of ε. The application of the quadrature rule implies that

$$R(k, z; c) \simeq c(k, z)^{-\tau} - \frac{\beta}{\sqrt{\pi}} \sum_{l=1}^{m_\nu} c(\tilde{k}, \tilde{z}_l)^{-\tau} \left(\alpha \tilde{k}^{\alpha-1} \sqrt{\frac{1+\tilde{z}_l}{1-\tilde{z}_l}} + 1 - \delta \right) \omega_l, \tag{6}$$

where $\tilde{z}_l = \tanh(\rho \tanh^{-1}(z) + \sqrt{2}\sigma\nu_l)$ and ν_l, ω_l, $l = 1, \ldots, m_\nu$ are the abscissas and weights for an m_ν-point quadrature rule. (For the quadrature formulas, see Press, et. al. (1986).)

For both specifications, the goal is to find approximate solutions to the functional equation $R(x, y; c) = 0$.

3. The solution method

The task at hand is to construct an approximate solution to the Euler equation of section 2. To do that, first I divide the state space Ω into nonoverlapping subdomains called *finite elements* or just *elements*. Then I construct approximate solutions on the elements. Over each element, the approximate consumption function is represented as a linear combination of interpolation functions (e.g., low-order polynomials). The approximate solution on Ω is found by piecing the local approximations together; hence, the approximate solution on Ω is a piecewise function. I choose the approximation that sets a weighted integral of the Euler equation residual equal to zero. Below, I describe the details of this approximation for the one- and two-dimensional versions of the growth model.

First consider the case in which the technology shock is an I-dimensional Markov chain. In this case, the solution to the Euler equation is given by I functions defined on $\Omega = [0, \bar{k}]$, where $\bar{k}$ is the maximum capital stock. Each element is an interval on $[0, \bar{k}]$. Since no elements overlap, the discretization of the domain is achieved by simply partitioning $[0, \bar{k}]$. The approximate solution on an element is a linear combination of interpolation functions, and the approximate solution over the entire domain Ω is obtained by connecting the local approximations. Let c^h be the approximate solution (where h

denotes the maximum element diameter). Then c^h can be represented as follows:

$$c^h(k,i) = \sum_{a=1}^{A} c_a^i N_a(k), \tag{7}$$

where the $N_a(k)$ are the interpolation functions and the c_a^i, $i = 1, \ldots, I$, $a = 1, \ldots, A$, are constants. The $N_a(k)$ are also known as *basis* or *shape functions*. To distinguish the finite element method from traditional variational methods, I must specify the interpolation functions. The interpolation functions for the finite element method are nonzero only on a small number of the elements and are defined in a piecewise manner. For example, if the consumption function is approximated by a piecewise linear function, then the basis functions are defined as follows:

$$N_a(k) = \begin{cases} \frac{k-k_{a-1}}{k_a-k_{a-1}}, & k_{a-1} \le k \le k_a \\ \frac{k_{a+1}-k}{k_{a+1}-k_a}, & k_a \le k \le k_{a+1} \\ 0, & \text{elsewhere} \end{cases} \tag{8}$$

and

$$c^h(k,i) = \left(\frac{k_{a+1}-k}{k_{a+1}-k_a}\right) c_a^i + \left(\frac{k-k_a}{k_{a+1}-k_a}\right) c_{a+1}^i, \qquad \text{for } k \in [k_a, k_{a+1}], \tag{9}$$

with $c_a^i = c^h(k_a, i)$ and $c_{a+1}^i = c^h(k_{a+1}, i)$. In eq. (8), $N_a(k)$ has the shape of a tent which peaks at $k = k_a$ and is only nonzero on elements $[k_{a-1}, k_a]$ and $[k_a, k_{a+1}]$. At its peak, $N_a(k) = 1$; therefore, the coefficients c_a^i and c_{a+1}^i in (9) are the values of consumption at the endpoints of the element $[k_a, k_{a+1}]$. Using the definition in (8), I can show that $N_{a+1}(k)$ has the shape of a tent that peaks at $k = k_{a+1}$. The left side of the $N_{a+1}(k)$ tent sits on top of the right side of the $N_a(k)$ tent, and both are nonzero on the element $[k_a, k_{a+1}]$. In fact, they are the only basis functions that are nonzero on the element $[k_a, k_{a+1}]$. Therefore, only these two functions appear in eq. (9). In general, the basis functions are complete polynomials that are continuous over the element.

In most cases, it is convenient to consider the approximation at the element level and then to assemble the local approximations into the global approximation. To accomplish both the assembly of the elements and the derivation of local approximations, I use points on the element called *nodes*. I select the position and number of the nodes in a particular way. Nodes that appear on the boundary of an element are the points at which the element

is connected to neighboring elements. Nodes that appear on the interior of an element are used in derivation of the local approximation. For example, if the approximation on element $[k_a, k_{a+1}]$ is linear, then two points are needed to describe the function. I place two nodes at the endpoints of the element (i.e., at k_a and k_{a+1}) in order to form a connection to other elements. No additional nodes are needed on the element to describe the approximation. If the approximation is quadratic, three points are needed to uniquely determine the local approximation. Two nodes are placed at the element endpoints, and the third is placed somewhere on the interior of the element.

The position or number of nodes may not be the same on each element. However, with a simple transformation from global coordinates to local coordinates, I can construct approximations for typical elements on Ω. Assume, for example, that the approximation on each element is linear, but that elements have different lengths. Construction of the approximation on any element can be standardized by introducing notation that is local rather than global. In other words, with the transformation $\xi : [k_a, k_{a+1}] \to [-1, 1]$ where $\xi(k) = (2k - k_a - k_{a+1})/(k_{a+1} - k_a)$, the consumption function over the element can be written in terms of local coordinates, e.g.,

$$c_e^h(\xi, i) = \tfrac{1}{2}(1 - \xi)\, c_{1,e}^i + \tfrac{1}{2}(1 + \xi)\, c_{2,e}^i, \tag{10}$$

for $\xi \in [-1, 1]$. The subscript e on the function c_e^h and the coefficients $c_{1,e}^i$ and $c_{2,e}^i$ denotes the element number. Note, however, that there is a simple mapping between local and global nodal values, i.e., $c_a^i = c_{1,e}^i$ and $c_{a+1}^i = c_{2,e}^i$. The functions $\frac{1}{2}(1 - \xi)$ and $\frac{1}{2}(1 + \xi)$ can be thought of as the local interpolation functions – equivalent to the functions in (9) but written in terms of the local variable ξ. In fig. 1(a), I display these functions. In fig. 1(b), I display the interpolation functions when the approximation over some element is quadratic. As in the linear case, the approximation for consumption is a weighted sum of the interpolation functions; i.e.,

$$c_e^h(\xi, i) = \tfrac{1}{2}\xi(\xi - 1)\, c_{1,e}^i + (1 - \xi^2)\, c_{2,e}^i + \tfrac{1}{2}\xi(\xi + 1)\, c_{3,e}^i. \tag{11}$$

Note that there are three nodes on this element. Thus, there are three unknowns, $c_{1,e}^i$, $c_{2,e}^i$, and $c_{3,e}^i$ – which are the values of consumption at the three nodes in fig. 1(b).

Now that I have a systematic approach to constructing the approximation, I turn to the computation of the unknowns which are the values of consumption at the nodal points.

Because c^h is approximate, $R(k,i;c^h)$ is not necessarily equal to zero for all $k \in \Omega$ and $i \in \{1,\ldots,I\}$. I assume that the approximate solution satisfies the weighted integral

$$\sum_{i=1}^{I} \int_0^{\bar{k}} w(k,i) R(k,i;c^h)\, dk = 0, \tag{12}$$

with $c^h(0,i) = 0$, $i = 1,\ldots,I$, where R is defined in eq. (4) and $w(k,i)$ is the *weighting function*. The weighting function is also assumed to be a linear combination of the basis functions used to approximate c^h; i.e.,[3]

$$w(k,i) = \sum_{a=1}^{A} w_a^i N_a(k), \tag{13}$$

where $N_a(k)$, $a = 1,\ldots,A$, are the same basis functions used to approximate the consumption functions and w_a^i, $a = 1,\ldots,A$, $i = 1,\ldots,I$, are constants. If $c^h(0,i) = 0$, I must impose $c_1^i = 0$ and $w_1^i = 0$ for all i, where global node 1 is the node at $k = 0$.

If the weights in eq. (13) are substituted into the weak form of the problem given by (12), the result is

$$\sum_{i=1}^{I} \sum_{a=2}^{A} w_a^i \Big\{ \sum_{j=1}^{A-1} \int_{k_j}^{k_{j+1}} N_a(k) R(k,i;c^h)\, dk \Big\} = 0. \tag{14}$$

Notice that integration is done element by element. Assuming that eq. (14) holds for any arbitrary weights, each term in brackets must be equal to zero; i.e.,

$$\sum_{j=1}^{A-1} \int_{k_j}^{k_{j+1}} N_a(k) R(k,i;c^h)\, dk = 0, \qquad a = 2,\ldots,A,\ i = 1,\ldots,I. \tag{15}$$

The system of equations in (15) is a system of $(A-1) \times I$ equations that depend on the coefficients of the approximate solution at all (global) nodes, e.g., $\vec{c} = [\vec{c}^1, \vec{c}^2, \ldots, \vec{c}^I]'$, $\vec{c}^i = [0, c_2^i, \ldots, c_A^i]'$. I solve these equations to determine the finite element approximation for the consumption function. Denote the system in (15) by $H(\vec{c}) = 0$.

Note several things here. First, each equation in (15) has only a small number of terms because $N_a(k)$ is zero on most of the elements. Second, as is clear from eq. (15), the computations are done element by element. Thus, the distinction between local and global

[3] This is the Galerkin or Bubnov-Galerkin implementation of the finite element method.

approximations comes into play when the system of equations is computed. The local variables are useful because the calculation of the Euler residual is the same for elements with the same geometry; thus, calculations on a typical element can be standardized. Finally, because the integration is being done element by element, the quadrature rule used need not involve many points. For example, if a Gauss-Legendre quadrature rule is applied, the integral in eq. (15) can be replaced with a weighted sum; e.g.,

$$\int_{k_j}^{k_{j+1}} N_a(k)R(k,i;c^h)\,dk \simeq \sum_{l=1}^{m_j} N_a(x_l)R(x_l,i;c^h)\,\omega_l, \tag{16}$$

where x_l and ω_l, $l = 1,\ldots,m_j$, are the Gauss-Legendre abscissas and weights for the interval $[k_j, k_{j+1}]$. If the interval is small, I can set m_j, $j = 1,\ldots,A-1$, to be a small integer and achieve good accuracy.

If I use a Newton-Raphson algorithm to find the vector $\vec{c}$ which satisfies the nonlinear system of equations $H(\vec{c}) = 0$, then I choose some initial guess – say, $\vec{c}_0$ – and iterate as follows:

$$\vec{c}_{\ell+1} = \vec{c}_\ell - J(\vec{c}_\ell)^{-1}H(\vec{c}_\ell), \tag{17}$$

where $\vec{c}_\ell$ is the guess of $\vec{c}$ at iteration ℓ and J is the Jacobian matrix of H. The (i,j) element of J is the derivative of the ith equation in H with respect to the jth element of $\vec{c}$. The most time-consuming part of the algorithm is the inversion of the Jacobian. For most examples, however, $J(\vec{c})$ is sparse. To see this, consider a case with elements of equal length and linear interpolation functions. In this case, a typical term in the system of equations in (15) is

$$\int_{k_{a-1}}^{k_a} (k - k_{a-1})R(k,i;c^h)\,dk + \int_{k_a}^{k_{a+1}} (k_{a+1} - k)R(k,i;c^h)\,dk = 0, \tag{18}$$

assuming that the length of each element is equal to 1. If the residual R on elements $[k_{a-1}, k_a]$ and $[k_a, k_{a+1}]$ depends only on consumption values in the neighborhood of k_a, then the Jacobian matrix J in (17) will only have nonzero elements on or near the diagonal. Thus, the Jacobian matrix can be stored in compressed form for more efficient storage, and direct or iterative methods can be used if the system is very large. Furthermore, if the application has many state variables, then the sparseness of J alleviates the curse of

dimensionality problem.[4]

Consider next the two-dimensional version of the stochastic growth model with $z \in [-1, 1]$ and $R(k, z; c)$ defined in eq. (5). For the examples that appear in later sections, I assume that the domain Ω is subdivided into smaller nonoverlapping rectangles and that linear and quadratic interpolation functions are used to construct the local approximations.[5] The linear interpolation functions in two dimensions are tensor products of the one-dimensional linear basis functions. In fig. 1(c), I display a typical bilinear interpolation function. The consumption function on the element displayed would actually be a weighted sum of four such interpolation functions; i.e.,

$$\begin{aligned} c_e^h(\xi, \eta) = {} & \tfrac{1}{4}(1-\xi)(1-\eta)\, c_{1,e} + \tfrac{1}{4}(1+\xi)(1-\eta)\, c_{2,e} \\ & + \tfrac{1}{4}(1+\xi)(1+\eta)\, c_{3,e} + \tfrac{1}{4}(1-\xi)(1+\eta)\, c_{4,e}. \end{aligned} \tag{19}$$

This is the local representation found by mapping a typical element – say, $[k_a, k_{a+1}] \times [z_b, z_{b+1}]$ – to the biunit square. Because I have chosen to use rectangular elements, the mappings between global and local variables are $\xi(k) = (2k - k_a - k_{a+1})/(k_{a+1} - k_a)$ and $\eta(z) = (2z - z_b - z_{b+1})/(z_{b+1} - z_b)$. If these expressions for $\xi(k)$ and $\eta(z)$ are substituted into the formula in (19), then the global formulation for consumption is obtained; e.g.,

$$c^h(k, z) = \sum_a c_a N_a(k, z), \tag{20}$$

where $N_a(k, z)$ has the shape of a pyramid which peaks at global node a, is nonzero on the four elements that connect at node a, and is zero everywhere else. Here, as in the one-dimensional case, the interpolation functions are equal to 1 at the nodes of the element. The nodes in the bilinear case are the four corners of the element. Thus, all four interpolation functions have the same shape as the function in fig. 1(c).

---

[4] Judd (1992) applies spectral methods to the stochastic growth model. Like the finite element method, spectral methods are used to solve functional equations such as eq. (5). The main difference is the function space from which the approximations are chosen. Spectral methods use shape functions which are almost everywhere nonzero on Ω. Typically these functions are polynomials. The finite element method uses shape functions which are nonzero on only small subdomains of Ω. This difference leads to two important advantages for the finite element method over spectral methods. First, the finite element method can handle problems with large high-order derivatives because high-order polynomials can be used in regions where large gradients occur and low-order polynomials can be used elsewhere. Second, the finite element approximations lead to sparse systems of equations.

[5] For a description of alternative element specifications (e.g., quadrilaterals or triangles) and alternative bases (e.g., cubic functions), see Hughes (1987) or Reddy (1993).

To attain a more accurate approximation, one can increase the number of elements while retaining linear basis functions or use higher-order polynomials. Consider, for example, quadratic functions in two dimensions. One simple way to construct these functions would be to take the product of one-dimensional quadratic polynomials. A unique set of coefficients for the polynomial requires that there be nine nodes and, hence, nine interpolation functions. The approximation on a typical element is given by

$$\begin{aligned} c_e^h(\xi,\eta) = {} & \tfrac{1}{4}\xi(\xi-1)\eta(\eta-1)\,c_{1,e} + \tfrac{1}{4}\xi(\xi+1)\eta(\eta-1)\,c_{2,e} + \tfrac{1}{4}\xi(\xi+1)\eta(\eta+1)\,c_{3,e} \\ & + \tfrac{1}{4}\xi(\xi-1)\eta(\eta+1)\,c_{4,e} + \tfrac{1}{2}(1-\xi^2)\eta(\eta-1)\,c_{5,e} + \tfrac{1}{2}\xi(\xi+1)(1-\eta^2)\,c_{6,e} \\ & + \tfrac{1}{2}(1-\xi^2)\eta(\eta+1)\,c_{7,e} + \tfrac{1}{2}\xi(\xi-1)(1-\eta^2)\,c_{8,e} + (1-\xi^2)(1-\eta^2)\,c_{9,e}. \end{aligned} \tag{21}$$

The interpolation functions in eq. (21) are *Lagrange polynomials* and the nine-node elements on which they are defined are *Lagrange elements*. In fig. 1(d), I display interpolation functions for a corner node, a midside node, and the middle node. The element has four corner nodes (i.e., nodes 1, 2, 3, 4), four midside nodes (i.e., nodes 5, 6, 7, 8), and one middle node (i.e., node 9).

Since internal nodes are not needed for connecting the element to its neighbors, an alternative quadratic approximation can be constructed on the rectangular element by placing nodes only on the boundary of the element. The element in this case is called a *serendipity element*. To construct the approximation, I set the jth interpolation function equal to 1 at node j and 0 at all other nodes, $j = 1, \ldots, 8$. The result is an approximation of the form

$$\begin{aligned} c_e^h(\xi,\eta) = {} & \tfrac{1}{4}(1-\xi)(1-\eta)(-1-\xi-\eta)\,c_{1,e} + \tfrac{1}{4}(1+\xi)(1-\eta)(-1+\xi-\eta)\,c_{2,e} \\ & + \tfrac{1}{4}(1+\xi)(1+\eta)(-1+\xi+\eta)\,c_{3,e} + \tfrac{1}{4}(1-\xi)(1+\eta)(-1-\xi+\eta)\,c_{4,e} \\ & + \tfrac{1}{2}(1-\xi^2)(1-\eta)\,c_{5,e} + \tfrac{1}{2}(1+\xi)(1-\eta^2)\,c_{6,e} + \tfrac{1}{2}(1-\xi^2)(1+\eta)\,c_{7,e} \\ & + \tfrac{1}{2}(1-\xi)(1-\eta^2)\,c_{8,e}. \end{aligned} \tag{22}$$

Notice that the polynomial of the Lagrange quadratic rectangular element (in (21)) is a complete polynomial of degree 2 that also includes two third-order terms $\xi\eta^2$ and $\xi^2\eta$, and a fourth-order term $\xi^2\eta^2$. The polynomial on the serendipity element (in (22)) is a complete polynomial of degree 2 that includes two third-order terms $\xi\eta^2$ and $\xi^2\eta$, but no

fourth-order terms. In fig. 1(e), I display interpolation functions for a typical corner node and a typical midside node of the serendipity element. The element has four corner nodes (i.e., nodes 1, 2, 3, 4) and four midside nodes (i.e., nodes 5, 6, 7, 8).

The system of equations that are solved to obtain the finite element approximation in the two-dimensional case are constructed using the same steps as in the one-dimensional case. The approximation is chosen to satisfy the weak form of the problem,

$$\int_{-1}^{1}\int_{0}^{\bar{k}} w(k,z)R(k,z;c^h)\,dk\,dz = 0, \tag{23}$$

where $w(k,z)$ is a weighting function that satisfies $w(0,z) = 0$ for all $z \in [-1,1]$. If the weak form of the problem is to be satisfied for an arbitrary weighting function, then the problem is to find constants c_a of the approximation in (20) that satisfy

$$\int_{-1}^{1}\int_{0}^{\bar{k}} N_a(k,z)R(k,z;c^h)\,dk\,dz = 0, \qquad \text{for all } a \in \mathcal{A}. \tag{24}$$

The set $\mathcal{A}$ includes all nodes except those at the $k = 0$ boundary since $c(0,z) = 0$. As before, these equations can be simplified using the fact that the interpolation functions are piecewise functions. In particular, the system can be written as

$$\sum_{e=1}^{n_e}\int_{\Omega_e} N_a(k,z)R(k,z;c^h)\,dk\,dz = 0, \qquad \text{for all } a \in \mathcal{A}, \tag{25}$$

where Ω_e is the domain of element e and n_e are the total number of elements. As in the one-dimensional case, it is convenient to first construct local approximations for the residual and then assemble the matrices. In other words, first I decide on the discretization of the domain Ω; then I choose the set of interpolation functions. With these functions, I can construct the approximation for consumption and, hence, the weighted residual on a typical element. I then assemble the equations for all elements.

Let $\vec{c}$ be the vector with elements c_a, $a \in \mathcal{A}$, and denote the system of equations in (25) by $H(\vec{c}) = 0$. As before, I use a Newton-Raphson algorithm to compute the vector of unknowns and, if need be, exploit the sparseness of the Jacobian matrix.

In the appendix, I describe the main steps of the algorithm in more detail.[6] In specifying the algorithm for the growth example, I assume that certain parameters are

[6] The appendix and codes written in FORTRAN77 are available from the author upon request to erm@ellen.mpls.frb.fed.us.

given. If the technology shock is a Markov chain, then the following inputs must be provided: β, τ, α, δ, $\theta(i), i = 1, \ldots, I$, Π, $\bar{k}$, a partition on $[0, \bar{k}]$, and the number of quadrature points for the integrals in eq. (15) (m_e, $e = 1, \ldots, n_e$). If the technology shock is an autoregressive process, then the following inputs must be provided: β, τ, α, δ, ρ, σ, $\bar{k}$, a partition on $[0, \bar{k}]$, a partition on $[-1, 1]$, the number of quadrature points for the integral in eq. (5) (m_ν), and the number of quadrature points in the k and z directions for the integrals in eq. (25). In addition to these sets of inputs, I need to specify an initial guess for the consumption function at all nodal points.

Here, I have considered only one- and two-dimensional problems. For many economic applications, however, the number of dimensions is three or higher and the computing and storage requirements are large. For this reason, the finite element method is an attractive alternative for applied economists. As the number of dimensions increase, more complicated interpolation functions can be used to obtain the same level of approximation with fewer grid points. Furthermore, the fact that the system of equations to be solved is sparse implies that the storage requirement will be less than that of other methods. To alleviate the larger computational demand required of higher-dimensional problems, there have been efforts made to implement the finite element method on vector and parallel architectures (as in Hanson 1991 and Chung, Hanson, and Xu 1992). The reduced storage requirement, together with vector or parallel computations, allow for the efficient solution of larger problems.

4. Examples

In this section, I illustrate the performance of the finite element method with three specific parameterizations. The first is a test case, for which a solution is known. The second is an example studied by Taylor and Uhlig (1990). The third is an example with binding inequality constraints.

4.1. A test case

If I assume that the capital stock fully depreciates each period (i.e., $\delta = 1$) and that the utility function is logarithmic (i.e., $\tau = 1$), then I can obtain an analytical solution to

the functional equation in (5), namely,

$$c(k,z) = (1-\beta\alpha)k^{\alpha}\sqrt{\frac{1+z}{1-z}}. \tag{26}$$

In this economy, the level of the capital stock tends to $(\beta\alpha)^{1/(1-\alpha)}$ for small values of σ.

To obtain the finite element approximation of the consumption function, I need to specify the model and algorithmic parameters. Assume that the process for technology is given by eq. (3) and let $\beta = 0.95$, $\alpha = 0.33$, $\rho = 0.95$, and $\sigma = 0.1$. For the discretization of the state space, I first apply the method of Tauchen and Hussey (1991) to construct a Markov chain analogue of (3). With values for $\ln\theta$, I can back out values for z. I use these values as the midpoints of the intervals in the partition of z. For the partition on capital, I choose an unevenly spaced grid with more points near the origin than at the upper bound. In particular, I apply the following recursive formula:

$$k_{j+2} = k_{j+1} + \Delta e^{aj}, \quad j = 0,\ldots,m-2,\ k_1 = 0, \tag{27}$$

for specific values of Δ and m. The value of a is determined by the terminal condition $k_m = \bar{k}$.

For this example, I compute a Markov chain with six discrete values for θ assuming that $\rho = 0.95$ and $\sigma = 0.1$. To get values for z, I use the transformation $\tanh(\ln(\theta))$. Placing these points at the midpoints of six intervals, I construct the following partition for z: [-0.391, -0.250, -0.123, 0.0, 0.123, 0.250, 0.391]. Notice that I have ignored very large and very small technology shocks. I do this to avoid putting elements in regions of the state space that are rarely observed.[7] For the upper bound on capital, I set $\bar{k} = 1.51^{\frac{1}{1-\alpha}}$ or 1.85 which is the maximum sustainable capital stock when $\theta = 1.51$ (and $z = 0.391$).

To illustrate how the approximation changes as I change the grids and interpolation functions, I compute finite element approximations for six different cases. In Table 1, I report the approximation errors, the computer processing times, and the storage requirements for these cases. Two measures of the approximation error are reported. The sup-norm is the maximum absolute difference between the exact solution and the approximation (e.g., $\max_{x\in\Omega}|c(x)-c^h(x)|$) and the L_2-norm is defined by $(\int_\Omega |c(x)-c^h(x)|^2 dx)^{1/2}$.

[7] For states outside of the chosen domain, I use the basis functions of the nearest element to evaluate the function.

Two computation times are reported: the seconds required for each Newton-Raphson iteration and the total processing time.[8] Finally, I report the fraction of elements in the Jacobian matrix J that are equal to zero as a measure of the storage requirement.

The meshes of Table 1 are characterized by four parameters. The first parameter is n_l which is the number of local nodes per element. If n_l is equal to 4, the interpolants are linear. (See fig. 1(c).) If n_l is equal to 9 the interpolants are quadratic and Lagrange polynomials are used. (See fig. 1(d).) If n_l is equal to 8, the serendipity quadrilateral elements are used. (See fig. 1(e).) In the fifth case, I use a mixture of element types. The second parameter is n_g which is the number of global nodes. This number includes nodes on the $k = 0$ boundary. The third parameter is n_e which is the number of elements. The fourth parameter characterizing the mesh is h, the maximum element diameter.

For all cases in Table 1, I set the number of quadrature points on each element equal to 9, i.e., 3 points for integration with respect to the capital stock and 3 points for integration with respect to the technology shock. For integration over ν, I set the number of quadrature points, m_ν, equal to 10. In all cases, the initial guess for the consumption function is one-half of total output, i.e., $\frac{1}{2}\theta k^\alpha$. This guess is motivated by the sample average of the ratio of consumption to gross national product in the data.[9] I set c_a equal to 0 at all nodes a on the $k = 0$ boundary. I assume that the iterations in (17) are converged when $||\vec{c}_{\ell+1} - \vec{c}_\ell|| < 10^{-7}$ where $||\vec{x}||$ is the vector norm equal to $(\sum_{i=1}^n x_i^2)^{\frac{1}{2}}/n$.

The first mesh of Table 1 is the coarsest. Only 4 points are used for the partition of z, namely [-0.391, -0.123, 0.123, 0.391]. The partition of k has 7 points which are found by applying (27) with $\Delta = 0.01$ and $a = 0.947$. Thus, there are 18 (or 3×6) rectangular subdomains on Ω. The largest interval in the partition of the capital stock has length 1.14. The largest interval in the z-partition has length 0.268. Therefore, the maximum element diameter is 1.17. For this example, the error in the sup-norm is 0.082 and the error in the L_2-norm is 0.0082. The absolute distance between the approximation and the exact solution is largest near the $k = 0$ boundary since the gradient there is infinite. The

[8] All computation times reported in Table 1 are based on runs of FORTRAN77 code on a Silicon Graphics Indigo R-4400/150MHz. This machine is estimated to run double precision Linpack routines at a rate of 25 million floating point operations per second.

[9] Another reasonable starting point is a linear approximation around the steady state for capital. The results are the same for this initial guess.

differences near the origin also have a large affect on the L_2-norm error. In terms of the processing time, only 0.029 seconds are required per iteration. Convergence is achieved in four steps of the Newton-Raphson algorithm. Therefore, the total processing time is approximately 0.12 seconds. The final statistic reported is the fraction of elements in the Jacobian matrix J that are equal to zero. For this case, 44% of the elements are zero.

Now consider two refinements of the first mesh. For the first refinement, I take each element and subdivide it into four smaller rectangles. The results for this case are given in the second row of Table 1. The partition for z used in this case is [-0.391, -0.250, -0.123, 0.0, 0.123, 0.391] and the partition for k is found by applying (27) with $\Delta = 0.00384$ and $a = 0.473$ (or $m = 13$). This choice of partitions implies that all of the nodes for the first mesh appear in this mesh as well. Linear basis functions are used again in this mesh so that the number of local nodes n_l is still 4. The number of global nodes n_g is the product of the partition lengths (i.e., 13×7). Because each element was subdivided into four smaller elements, the number of elements is four times that of the first case. The maximum diameter is smaller than that of first case ($h = 0.71$ versus $h = 1.17$). Notice that the sup-norm error falls by 28% and the L_2 error falls by 32%. When applying the finite element method to solve a continuous-time stochastic dynamic program, Chung, Hanson, and Xu (1992) show that the accuracy of the method is of order h^q in the L_2-norm if the interpolating function has degree $q - 1$. Thus, for linear functions, their estimate implies that the rate of convergence of the finite element solution is 2, i.e., that $\log_{10}(||e||)$ is equal to $2\log_{10}(h)$ plus a constant, where $||e||$ is the L_2-norm error. The first two cases of Table 1 are consistent with a rate of 2.27, which is close to the theoretical estimate. The cost of the increase in accuracy is an increase in computational time. However, at 0.184 seconds per iteration, the cost is still very small. Furthermore, there is a large gain in terms of storage. The fraction of elements in the Jacobian matrix equal to zero is 69%.

For the second refinement, I keep the discretization the same as in the first case, but use quadratic rather than linear interpolants. The third and fourth rows of Table 1 have the results for the Lagrange and serendipity elements, respectively. Notice that the number of global nodes n_g for the serendipity elements is equal to the number of global nodes for the Lagrange elements less the number of elements. This is due to the fact that the middle nodes have been omitted on the serendipity elements. Because, the element domains are

the same in the two quadratic cases, the parameters n_e and h are the same. Interestingly, the errors for the two cases with quadratic shape functions are the same to two significant digits. The error in the sup-norm is 0.055 and the error in the L_2-norm is 0.0022. There are slight gains in computing time with the serendipity elements (0.071 versus 0.079 seconds) but more storage is required (57% versus 53% zeros). Relative to the 72-element linear case, both give lower approximation errors and computing times. However, the Jacobian matrices in the quadratic cases are less sparse.

An alternative approach is to use quadratic elements only where needed. In the fifth row of Table 1, the results are reported for a mesh with both linear and quadratic shape functions. For the six elements in the region with $z \in$ [-0.391,-0.123], I use linear shape functions. For the six elements in the region with $z \in$ [-0.123,0.123], five-node *transitional* elements are used.[10] And in the region with $z \in$ [0.123,0.391], eight-node serendipity elements are used. This case is intermediate to the first which has only linear elements and the fourth which has only serendipity elements. Notice that the estimates of the approximation error are significantly lower than that of the first case. There is no improvement in computational time over the fourth case because a general algorithm is used. The algorithm allows for any "hybrid" case between all elements having 4 nodes and all elements having 9 nodes. Improvements in speed can be achieved if the specific shape functions are written directly into the computer code.

The last row of Table 1 is a further refinement of the mesh with linear elements. I added this case to illustrate the potential gains in sparseness that can be achieved as the mesh is refined. In this case, twenty-five points are used in the partition for the capital stock and seven points are used in the partition for the technology shock. The partition for z is the same one used for the mesh with 72 linear elements. The partition for k is found by applying (27) with $\Delta = 0.0017$ and $a = 0.237$. Notice that there is a reduction in the approximation errors over the coarse mesh, e.g., 0.0015 versus 0.0082 in the L_2-norm. However, Chung, Hanson, and Xu's (1992) estimate of the rate of convergence of the finite element solution would imply an L_2 error closer to 0.0009. The fact that the solution here is not achieving as high of a rate is due in part to the inclusion of the $k = 0$ boundary when calculating the solution and the approximation errors. In other words, if the partition on

[10] See Hughes for formulas of the shape functions on transitional elements.

capital stocks is given by $[\underline{k}, \bar{k}]$, for $\underline{k} > 0$, then the rates of convergence approach the theoretical rates as $\underline{k}$ is increased. A cost of the increase in accuracy is an increase in the computation time; the total computation time takes 2 seconds which is considerably longer than that of the coarse grid. But in this case, 82% of the elements in the Jacobian matrix are zero. The fraction of zeros in the matrix is close to double that of the coarse grid. In problems with many state variables, where the order of the Jacobian matrix is much larger than 175, this factor plays an important role.

In fig. 2, I plot the exact solution, the finite element approximation with 18 Lagrange elements (row 3 of Table 1), the finite approximation with 18 elements and linear interpolants (row 1 of Table 1) and the finite approximation with 72 elements and linear interpolants (rwo 2 of Table 1). The function is displayed for only one value of z; it is displayed for the value at the upper bound of the z-partition (i.e., $z = 0.391$). Notice that even the solution for the coarse mesh is hard to distinguish from the exact solution. And this is the worst case. The approximation at values of z near its steady state are closer to the exact solution.

4.2. An example from Taylor and Uhlig (1990)

In the examples of Taylor and Uhlig (1990), the rate of depreciation is equal to zero (i.e., $\delta = 0$) and no analytical solutions exist. In this section, I consider their 'case 2' which has $\beta = 0.95$, $\tau = 1.5$, $\alpha = 0.33$, $\delta = 0$, $\rho = 0.95$, and $\sigma = 0.1$. Taylor and Uhlig (1990) focus on the decision rules for $k \in [5, 25]$ and $\theta \in [0.4, 1.6]$. To compute the finite element approximation, I use $\Omega = [0, 35] \times [0.3, 1.7]$ so that the points of interest are not on the boundary of the domain. For the capital stock partition, I apply (27) with $\Delta = 0.1$ and $a = 0.225$; there are 21 points in this partition. For the technology shock z, I use an evenly spaced 11-point grid on [tanh(ln(0.3)),tanh(ln(1.7))]. Therefore, the mesh has 231 global nodes and 50 elements.

Each rectangular element is assumed to have nine nodes (as in fig. 1(d)) and the interpolation functions are Lagrange polynomials. I set the number of quadrature points for integration in each element equal to 9. For integration over ν, I set the number of quadrature points equal to 10. For the initial consumption function, I use $0.14(\theta k^{\alpha} + k - \delta k)$. This initial guess assumes that a constant fraction of output is used for consumption and a constant fraction is used for the purchase of new capital. The fraction 0.14 is chosen

because it implies that the correct marginal propensity to consume when σ is small. I assume that the iterations in (17) are converged when $||\vec{c}_{\ell+1} - \vec{c}_\ell|| < 10^{-7}$ where $||\vec{x}||$ is the vector norm equal to $(\sum_{i=1}^n x_i^2)^{\frac{1}{2}}/n$.

In fig. 3, I plot both the finite element approximation and the approximation obtained by solving Bellman's equation directly, i.e., by finding v that satisfies

$$v(k,z) = \max_{c\geq 0}\left\{\frac{c^{1-\tau}-1}{1-\tau} + \frac{\beta}{\sqrt{\pi}}\int_{-\infty}^{\infty} v(\tilde{k},\tilde{z})e^{-\nu^2}\,d\nu\right\} \tag{28}$$

for all $(k,z) \in \Omega$, where $\tilde{k}$ and $\tilde{z}$ are defined in (5).[11] Given the value function v, I can derive the optimal consumption function by solving the maximization problem on the right hand side of eq. (28). To compute the value function in (28), an evenly spaced grid with 2^{14} points is used for the capital stocks with $\bar{k} = 60$. The same partition of z is used for the finite element calculations and the value-function iterations.

Each curve in fig. 3 is a finite element approximation for consumption as a function of capital and some fixed level of the technology shock. I display the finite element solution for $k \in [0,26]$ and $\theta = 0.4$, 0.7, 1.0, 1.3, and 1.6. I also display the discretized dynamic programming solution for the points reported in Taylor and Uhlig (1990). These points are marked '+' in fig. 3. Notice that each of the finite element solutions coincide with the solution found from the method of value-function iteration. The total computation time required to compute the finite element approximation is 4.75 seconds (on the SGI R-4400). Eight iterations of the Newton-Raphson algorithm are required for convergence. Therefore, the per-iteration computation time is approximately 0.59 seconds which is a small fraction of the time required for discretized dynamic programming.[12]

4.3. An example with inequality constraints

If there are inequality constraints that bind for certain values of the capital stock and the technology shock, then the algorithm as described in section 3 will not enforce the constraints. Suppose for example, that investment cannot fall below zero.[13] Then the

[11] See Christiano (1990) for details on the method of value-function iteration.

[12] The value-function iterations take approximately 370 seconds per iteration on a Cray-C90.

[13] Braun and McGrattan (1993) study a version of this model with large fiscal shocks like those observed in World War II. For some of their parameterizations, investment falls below zero. Aiyagari and McGrattan (1994) have also considered versions of this model in which individuals face idiosyncratic shocks and liquidity constraints that bind.

solution to the problem of section 2 must satisfy

$$c_t \leq \theta_t k_{t-1}^{\alpha}, \qquad t \geq 0. \tag{29}$$

Can we modify the problem or the algorithm so that the solution satisfies eq. (29)?

The approach that I take here is to modify the problem. In particular, I replace the objective function in eq. (1) by

$$E\left[\sum_{t=0}^{\infty} \beta^t \left\{ \frac{c_t^{1-\tau}-1}{1-\tau} + \frac{1}{3}\gamma \min(\theta_t k_{t-1}^{\alpha} - c_t, 0)^3 \right], \qquad 0 < \beta < 1,\ \tau > 0. \tag{30}$$

Notice that I have included a penalty function of the form $\min(x,0)^3$ which is equal to 0 for values of x greater than 0 and x^3 for negative values of x. (For more details on penalty functions, see Fletcher 1987.) If the constraint is violated and investment is negative, then there is a loss in utility. The larger is consumption relative to income, the larger is the penalty.

The size of the penalty is determined by the value of the parameter γ. To compute the optimal decision function, I solve a sequence of optimization problems, each indexed by a different penalty parameter. One approach is to choose a sequence $\gamma^{(j)}$, such as {1, 10, 10^2, 10^3, ...}, which has $\gamma^{(j)} \to \infty$; the finite element approximation is calculated for each $\gamma^{(j)}$ until the constraints in (29) are approximately satisfied. If too many approximations need to be calculated, then this method is inefficient. Also, the numerical optimization of (30) becomes increasingly difficult as as $\gamma^{(j)}$ approaches infinity. A shortcut method often used in practice is to pick only one or two largish values for γ.

Another approach to this problem is advocated by Christiano and Fisher (1994). They solve the Kuhn-Tucker conditions of the original problem. To make their method tractable, Christiano and Fisher (1994) assume that at some level of the capital stock, k^*, which depends on the value of the technology shock, the constraint is binding, and it is binding at all levels of the stock above k^*. For the particular case that they consider, the technology shock is a serially uncorrelated process that takes on only two values. The constraint binds only when the technology shock is low and, hence, k^* is a point. However, in problems with many levels of technology (possibly a continuum), it is difficult to keep track of the regions where the constraints bind. With penalty functions, there is no need

to impose a priori assumptions about where the constraints bind. Furthermore, with penalty functions there is no need to calculate the Lagrange multipliers associated with the constraints in (29).

For an example, consider the following parameterization of the model. Let $\beta = 0.99$, $\tau = 1$, $\alpha = 0.3$, and $\delta = 0.025$, with θ given by a 10-state Markov chain. The Markov chain is assumed to be the discrete analogue to the process in eq. (3) with $\rho = 0.9$ and $\sigma^2 = 0.005$. To derive the discrete values for $\ln(\theta)$ and the transition probabilities, I apply the method of Tauchen and Hussey (1991). The partition that I use for the capital stocks has 60 points and is found by applying (27) with $\Delta = 0.001$ and $a = 0.690$ for the first 11 points (with $k_m = 1$) and by using an evenly spaced grid between $k = 2$ and $k = 50$ for the remaining 49 points. For the initial consumption function, I use $0.0843(\theta k^\alpha + k - \delta k)$. This initial guess assumes that a constant fraction of output is used for consumption and a constant fraction is used for the purchase of new capital. The fraction 0.0843 yields the correct marginal propensity to consume when σ is small.

In fig. 4, I plot the finite element approximation for the investment function (i.e., $i(k,z) = k^\alpha\sqrt{(1+z)/(1-z)} - c(k,z)$) for the constrained problem. Two steps are taken to obtain this solution: $\gamma^{(1)} = 10$ and $\gamma^{(2)} = 1000$. The per-iteration computation time is 4.7 seconds and a total of 11 iterations were necessary for convergence (i.e., $||\vec{c}_{\ell+1} - \vec{c}_\ell|| < 10^{-7}$) for each value of γ in the sequence. Therefore, the total computation cost is 103 seconds. Much of this time was devoted to inverting the 600×600 Jacobian matrix 22 times. In fig. 4, I also plot the approximation obtained by solving Bellman's equation directly, i.e., by finding v that satisfies

$$v(k,i) = \max_{(1-\delta)k \le \tilde{k} \le \theta(i)k^\alpha} \{(\theta(i)k^\alpha + (1-\delta)k - \tilde{k})^{1-\tau}/(1-\tau) + \sum_{j=1}^{I} \Pi_{i,j} v(\tilde{k}, j)\} \qquad (31)$$

for all $k \in [0,50]$ and all $i \in \{1,\ldots,10\}$. Given the value function v, I can derive the optimal investment function by solving the maximization problem on the right hand side of eq. (31). To compute the value function in (31), an evenly spaced grid with 2^{14} points is used for the capital stocks. In the figure, I only display the points used in the finite element mesh. Notice that the finite element approximation is hard to distinguish from the solution using value-function iterations even where the constraints bind. Notice also that the constraints bind for five levels of technology. For this particular example, I did not

know a priori where the constraints would bind and, therefore, did not use this information when constructing the finite element approximation.

5. Conclusion

This paper describes the finite element method by applying it to several examples. I show that the method is easy to apply and, for examples such as the stochastic growth model, gives accurate solutions within a second or two on a desktop computer. I also show how inequality constraints can be handled by redefining the optimization problem with penalty functions.

References

Aiyagari, S. R. and E. R. McGrattan, 1994, The optimal quantity of debt, Manuscript (Federal Reserve Bank of Minneapolis, Minneapolis, MN).

Braun, R. A. and E. R. McGrattan, 1993, The macroeconomics of war and peace, in: O. Blanchard and S. Fischer, eds., NBER Macroeconomics Annual 1993 (MIT Press, Cambridge MA), 197-247.

Christiano, L. J., 1990, Solving the stochastic growth model by linear-quadratic approximation and by value-function iteration, Journal of Business and Economic Statistics 8, 23-26.

Christiano, L. J. and J. D. Fisher, 1994, Algorithms for solving dynamic models with occasionally binding constraints, Manuscript (Northwestern University, Evanston, IL).

Chung, S. L., F. B. Hanson, and H. H. Xu, 1992, Parallel stochastic dynamic programming: finite element methods, Linear Algebra and its Applications 172, 197-218.

Fletcher, R., 1987, Practical methods of optimization (Wiley, Chichester, England).

Hanson, F. B., 1991, Computational stochastic dynamic programming on a vector multiprocessor, IEEE Transactions on Automatic Control 36, 507-511.

Hughes, T. J. R., 1987, The finite element method: linear static and dynamic finite element analysis (Prentice-Hall, Englewood Cliffs, NJ).

Judd, K. L., 1992, Projection methods for solving aggregate growth models, Journal of Economic Theory 58, 410-452.

Kydland, F. E. and E. C. Prescott., 1982, Time to build and aggregate fluctuations, Econometrica 50, 1345-1370.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, 1986, Numerical recipes: The art of scientific computing (Cambridge University Press, Cambridge, England).

Reddy, J. N., 1993, An introduction to the finite element method (McGraw-Hill, New York, NY).

Stokey, N. L. and R. E. Lucas, Jr., 1989, Recursive methods in economic dynamics (Harvard University Press, Cambridge, MA).

Tauchen, G. and R. Hussey, 1991, Quadrature-based methods for obtaining approximate solutions to nonlinear asset pricing models, Econometrica 59, 371-396.

Taylor, J. B. and H. Uhlig, 1990, Solving nonlinear stochastic growth models: A comparison of alternative solution methods, Journal of Business and Economic Statistics 8, 1-17.

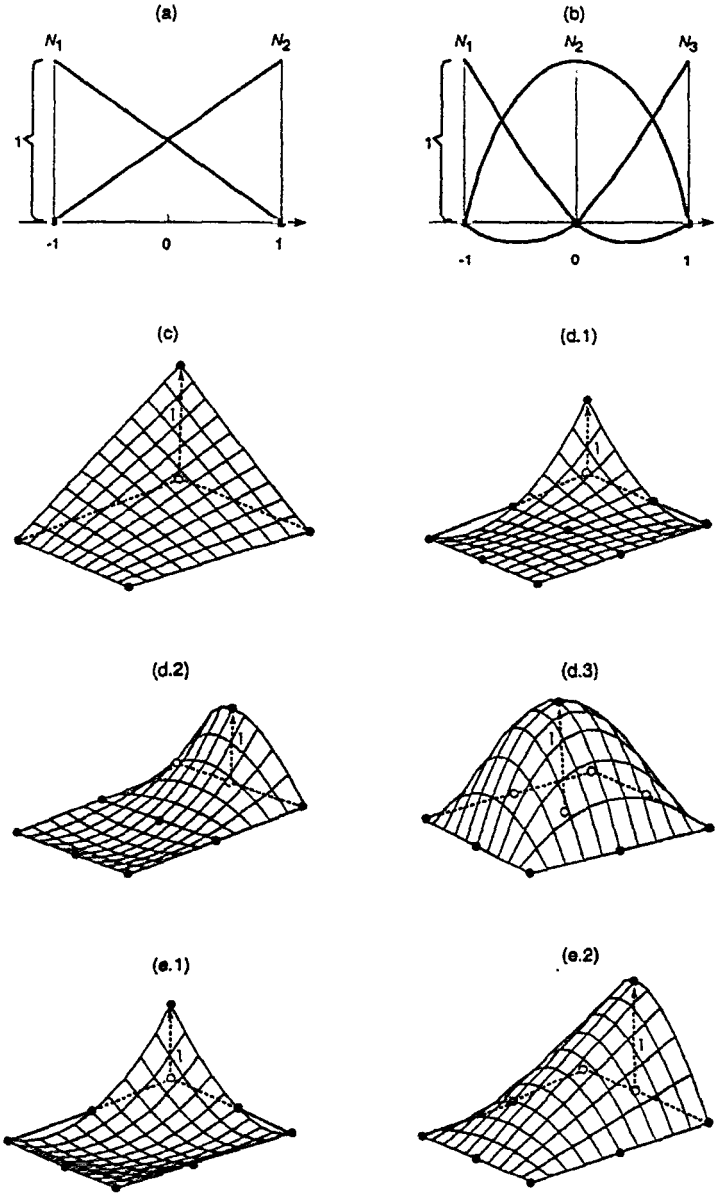

Figure 1. Typical interpolation functions for 1-dimensional (a, b) and 2-dimensional (c – e) elements with (a) 2 nodes, (b) 3 nodes, (c) 4 nodes, (d) 9 nodes, and (e) 8 nodes.

Table 1. Comparison of errors, running time, and storage for test case with alternative meshes.

Mesh Parameters †				Errors		CPU Seconds		Fraction
n_l	n_g	n_e	h	sup-norm	L_2-norm	1 iteration	total	of zeros
4	28	18	1.17	0.082	0.0082	0.029	0.12	0.44
4	91	72	0.71	0.059	0.0026	0.184	0.74	0.69
9	91	18	1.17	0.055	0.0022	0.079	0.39	0.57
8	73	18	1.17	0.055	0.0022	0.071	0.36	0.53
4-8	47	18	1.17	0.057	0.0044	0.073	0.37	0.50
4	175	144	0.42	0.045	0.0015	0.513	2.04	0.82

† n_l is the number of local nodes, n_g is the number of global nodes, n_e is the number of elements, and h is the maximum element diameter.

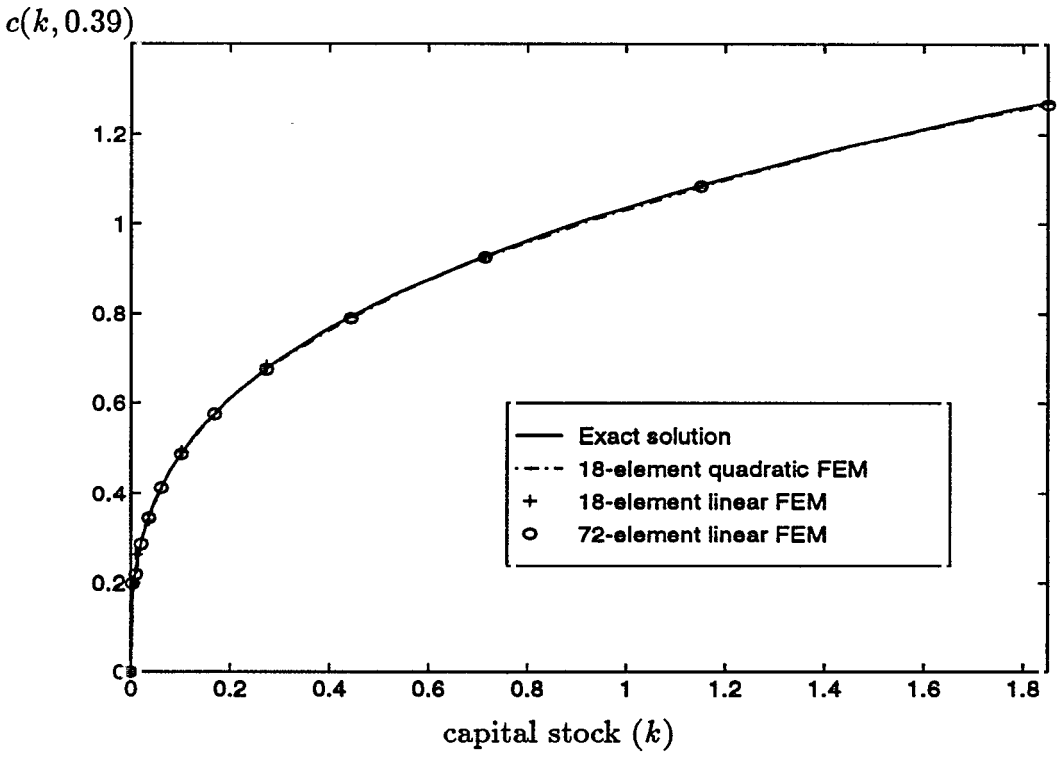

Figure 2. Consumption function for the test case, with $z = 0.39$.

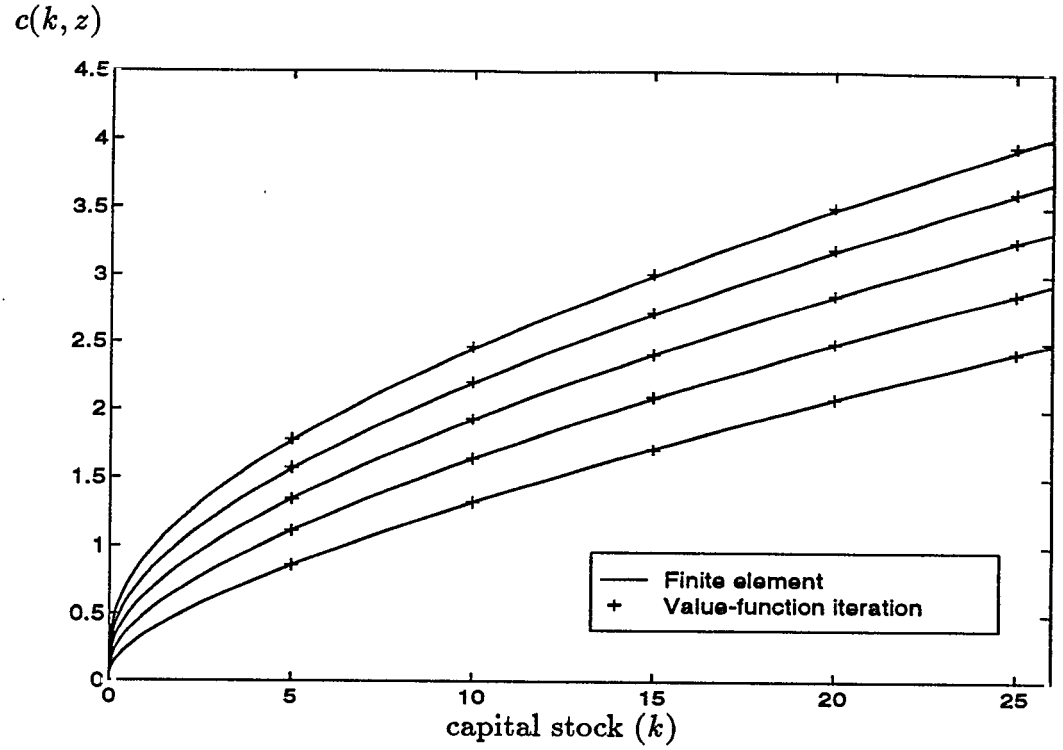

Figure 3. Consumption function for the Taylor-Uhlig example, with θ=0.4, 0.7, 1.0, 1.3, 1.6, and $z = \tanh(\ln(\theta))$.

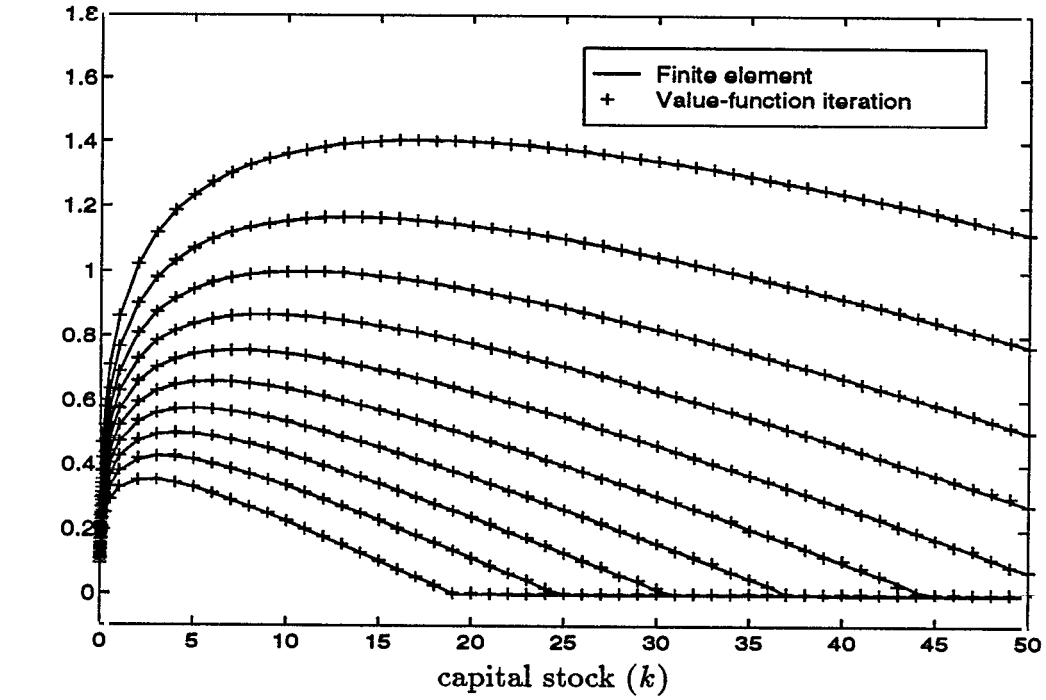

Figure 4. Investment function for the example with inequality constraints binding.