Chapter 9

Discrete Time Continuous State Dynamic Models: Methods

This chapter discusses numerical methods for solving discrete time continuous state dynamic economic models, with particular emphasis on Markov decision and rational expectations models.

Continuous state dynamic models give rise to functional equations whose unknowns are functions defined on closed convex subsets of Euclidean space. For example, the unknown of a Bellman equation

$$V(s) = \max_{x \in X(s)} \{f(s,x) + \delta E_\epsilon V(g(s,x,\epsilon))\}, \qquad s \in S,$$

is the value function $V(\cdot)$. The unknown of the Euler conditions

$$0 = f_x(s,x(s)) + \delta E_\epsilon \left[\lambda(g(s,x(s),\epsilon)) \cdot g_x(s,x(s),\epsilon)\right], \qquad s \in S,$$

$$\lambda(s) = f_s(s,x(s)) + \delta E_\epsilon \left[\lambda(g(s,x(s),\epsilon)) \cdot g_s(s,x(s),\epsilon)\right], \qquad s \in S,$$

are the shadow price and policy functions $\lambda(\cdot)$ and $x(\cdot)$. And the unknown of a rational expectations intertemporal equilibrium condition

$$f(s,x(s),Ex(g(s,x(s),\epsilon))), \qquad s \in S,$$

is the response function $x(\cdot)$. In most applications, these functional equations do not have known closed form analytic solutions and can only be solved only numerically.

For over three decades, Economists have been solving continuous state dynamic economic models numerically, relying mostly on linear-quadratic approximation and space discretization methods. However, for many economic applications these numerical techniques are inadequate. Linear quadratic approximation forces the analyst to impose conditions, such as a linear state transition or unbounded actions, that in many economic applications are unrealistic. Although linear quadratic approximation can make a dynamic model easier to solve, it will often yield economic predictions that are unreliable, if not completely useless. Space discretization, in which the continuous state model is reformulated as a discrete state model, is also problematic. Space discretization generates approximate solutions that lack the smoothness and differentiability of the true solution. Yet in many applications, it is the derivative, which yields information about shadow prices or marginal benefits and costs, which is the object of central interest to the economic analyst. Space discretization can also be computationally inefficient, making it difficult to solve models, particularly high-dimensional models, to an acceptable degree of accuracy in a reasonable amount of time.

Over the same thirty year period, there have also been significant advancements in the theory and methods of numerical analysis and many computational techniques for solving nonlinear functional and differential equations have been developed and improved. Concurrent with these developments was the enthusiastic adoption of numerical analysis methods by engineers and physical scientists to address the complex dynamic models encountered in their disciplines. Among the many methods developed to solve functional equations, Galerkin and finite difference methods have proven extremely powerful for solving dynamic models.

Galerkin methods can be straightforwardly adapted to discrete time dynamic models encountered in economics and finance. Of the various versions of the Galerkin techniques, the collocation method stands out as the singly most powerful technique for solving discrete time continuous state dynamic economic models. The collocation method is highly flexible. It may be used to solve discrete and continuous choice Markov decision models and rational expectations models. Bounds and general constraints on variables can also be handled easily.

The collocation method is relatively easy to understand because it involves only the combination of elementary numerical integration, approximation, and rootfinding methods. Specifically, the collocation method employs the following general strategy for solving the functional equations underlying dynamic economic models:

- Approximate the unknown function (value, shadow price, or response function) using a finite linear combination of n known basis functions.

- Require the function approximant to satisfy the functional equation (Bellman, Euler, or intertemporal equilibrium equation) at only n prescribed points of the domain.

This strategy replaces the functional equation with a finite-dimensional nonlinear equation that can be solved using basic nonlinear equation techniques. In most applications, the global approximation error associated with the solution function approximant can be reduced to an arbitrary prescribed tolerance by increasing the number of basis functions and nodes.

The collocation method is a solution strategy rather than a specific technique. Upon electing to use the collocation method, the analyst still faces a number of numerical modeling decisions. For example, the analyst must choose the basis function and collocation nodes. Numerical approximation theory offers guidance here, suggesting a Chebychev polynomial basis coupled with Chebychev collocation nodes, or a spline basis coupled with equally spaced nodes will often be good choices. Also, the analyst must chose an algorithm for solving the resulting nonlinear equation. Nonlinear equation theory offers numerous choices, including Newton, quasi-Newton, and function iteration schemes. A careful analyst will often try a variety of basis-node combination, and may employ more than one iterative scheme in order to assure the robustness of the results.

Although the collocation method is general in its applicability, the details of implementation vary with the functional equation being solved. Below, the collocation method is developed in greater detail for Bellman equations, Euler conditions, and rational expectations equilibrium conditions.

9.1 Traditional Solution Methods

Before introducing collocation methods for continuous state Markov decision models, let us briefly examine the two numerical techniques that have been used most often by economists over the past thirty years to compute approximate solutions to such models: space discretization and linear-quadratic approximation.

9.1.1 Space Discretization

One way to compute an approximate solution for a continuous state Markov decision problem is to solve a discrete Markov decision problem that closely resembles it. To "discretize" a continuous state Markov decision problem, one partitions the state and action spaces S into finitely many regions, $S_1, S_2, \ldots, S_n$. If the action space X is also continuous, it too is partitioned into finitely many regions $X_1, X_2, \ldots, X_m$. Once the space and action spaces have been partitioned, the analyst selects representative elements, $s_i \in S_i$ and $x_j \in X_j$, from each region. The nodes serve as the state and action spaces of the discrete problem. The transition probabilities of the discrete problem are computed by integrating with respect to the density of the random shock:

$$P(s_{i'}|s_i, x_j) = \Pr[g(s_i, x_j, \epsilon) \in S_{i'}].$$

When the state and action spaces are intervals, say, $S = [s_{min}, s_{max}]$ and $X = [x_{min}, x_{max}]$, it is often easiest to partition the spaces so that the nodes are equally-spaced and the first and final nodes correspond to the endpoints of the intervals. Specifically, we set $s_i = s_{min} + (i-1)w_s$ and $x_j = x_{min} + (j-1)w_x$, for $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, m$, where $w_s = (s_{max} - s_{min})/(n-1)$ and $w_x = (x_{max} - x_{min})/(m-1)$. In this instance the transition probabilities are given by

$$P(s_{i'}|s_i, x_j) = \Pr[s_{i'} - w_s/2 \leq g(s_i, x_j, \epsilon) \leq s_{i'} + w_s/2].$$

9.1.2 Linear-Quadratic Approximation

Because linear quadratic control models have finite-dimensional solutions and can be directly solved using nonlinear equation methods, many analysts compute approximate solutions to more general Markov decision models using the method of linear quadratic approximation. Linear quadratic approximation calls for all constraints of the original problem to be discarded, for its reward function to be replaced with a second-order quadratic approximation, and for its transition function to be replaced with a first-order linear approximation. The solution of the linear quadratic control problem is taken as an approximate solution for the more original problem.

The first step in constructing a linear-quadratic approximation is to form the linear and quadratic approximant of the transition and reward functions, respectively. Typically, the approximants are formed, respectively, by taking the first and second order Taylor series approximant about the certainty equivalent steady state. One finds the certainty equivalent steady state state $\bar{s}$, optimal action $\bar{x}$, and shadow price $\bar{\lambda}$ by solving the nonlinear equation system

$$f_x(\bar{s},\bar{x}) + \delta\bar{\lambda}g_x(\bar{s},\bar{x},\bar{\epsilon}) = 0$$

$$\bar{\lambda} = f_s(\bar{s},\bar{x}) + \delta\bar{\lambda}g_s(\bar{s},\bar{x},\bar{\epsilon})$$

$$\bar{s} = g(\bar{s},\bar{x},\bar{\epsilon}).$$

Here, $\bar{\epsilon}$ denotes the mean of ϵ, and f_s, f_x, f_{ss}, f_{sx}, f_{xx}, g_s, and g_x, represent partial derivatives of the reward and state transition functions. The nonlinear equation system may be solved using standard nonlinear equation methods.

Once the certainty equivalent steady-state has been computed, the Taylor series approximations of the reward and transition functions are formed as follows:

$$\begin{aligned} f(s,x) \;\approx\; & f_0 + f_s'(s-\bar{s}) + f_x'(x-\bar{x}) + 0.5(x-\bar{x})'f_{xx}(x-\bar{x}) \\ & +0.5(s-\bar{s})'f_{ss}(s-\bar{s}) + (s-\bar{s})'f_{sx}(x-\bar{x}) \end{aligned}$$

$$g(s,x,\epsilon) \approx \bar{s} + g_s(s-\bar{s}) + g_x(x-\bar{x}).$$

Here, f_0 represents the value of the reward function at the certainty equivalent steady state, and f_s, f_x, f_{ss}, f_{sx}, f_{xx}, g_s, and g_x, represent partial derivatives of the reward and state transition functions evaluated at the certainty equivalent steady state.

Using the results from section 8.3, one can derive expressions for the optimal shadow price and policy function for the approximating linear quadratic optimization problem. Exploiting properties of the derivative of the reward and state transition functions at the steady state, these expressions may be simplified to

$$\lambda(s) = \bar{\lambda} + \Lambda_s(s-\bar{s})$$

$$x(s) = \bar{x} + X_s(s-\bar{s})$$

where

$$\begin{aligned} \Lambda_s \;=\; & -[\delta g_s'\Lambda_s g_x + f_{sx}][\delta g_x'\Lambda_s g_x + f_{xx}']^{-1}[\delta g_x'\Lambda_s g_s + f_{sx}'] \\ & +\delta g_s'\Lambda_s g_s + f_{ss} \\ X_s \;=\; & -[\delta g_x'\Lambda_s g_x + f_{xx}']^{-1}[\delta g_x'\Lambda_s g_s + f_{sx}'] \end{aligned}$$

These first equation can be easily solved for Λ_s using function iteration. Once Λ_s has been computed, X_s can be computed directly without using a function iteration technique. If the problem has one dimensional state and action spaces, and if $f_{ss}f_{xx} = f_{sx}^2$, a condition often encountered in economic problems, then the slope of the shadow price function may be computed analytically as follows:

$$\Lambda_s = [f_{ss}g_x^2 - 2f_{ss}f_{xx}g_sg_x + f_{xx}g_s^2 - f_{xx}/\delta]/g_x^2$$

For example, consider stochastic optimal growth problem of the preceding chapter under the assumption that social utility is a function $u(c) = c^{1-\alpha}/(1-\alpha)$ of current consumption and next period's certainty equivalent production is a function $f(x) = x^\beta$ of current investment. After some algebraic simplification of the Euler condition, the certainty equivalent steady-state state, action, and shadow price may be computed in sequence as follows:

$$\bar{x} = \left(\frac{1-\delta\gamma}{\delta\beta}\right)^{\frac{1}{\beta-1}}$$

$$\bar{s} = \gamma\bar{x} + \bar{x}^\beta$$

$$\bar{\lambda} = (\bar{s} - \bar{x})^{-\alpha}.$$

Given these results, and the above relations, the shadow price and optimal policy function approximant are thus:

$$\lambda(s) = \bar{\lambda} - (1-\delta)\alpha(\bar{s}-\bar{x})^{-\alpha-1}(s-\bar{s})$$

$$x(s) = \bar{x} + \delta(s - \bar{s}).$$

More specifically, if $\alpha = 0.2$, $\beta = 0.5$, $\gamma = 0.9$, and $\delta = 0.9$, then

$$\lambda(s) = 0.8884 - 0.0098(s - 7.4169)$$

$$x(s) = 5.6094 + 0.9000(s - 7.4169).$$

As another example, consider the renewable resource problem of the preceding chapter under the assumptions that $p(x) = x^{-\gamma}$, $c(x) = kx$, and $g(s-x) = \alpha(s-x) - 0.5\beta(s-x)^2$. To solve the renewable resource model by linear quadratic approximation one first computes the certainty equivalent steady state state, action, and shadow price in sequence:

$$\bar{s} = \frac{\alpha^2 - \delta^{-2}}{2\beta}$$

$$\bar{x} = \bar{s} - \frac{\delta\alpha - 1}{\delta\beta}$$

$$\bar{\lambda} = \bar{x}^{-\gamma} - k.$$

Using the results above, it then follows that the shadow price and optimal policy function approximant are:

$$\lambda(s) = \bar{\lambda} - \frac{1-\delta}{\gamma(\bar{x})^{1+\gamma}}(s - \bar{s})$$

$$x(s) = \bar{x} + (1-\delta)(s - \bar{s}).$$

More specifically, if $\gamma = 0.5$, $\alpha = 4$, $\beta = 1$, $k = 0.2$, and $\delta = 0.9$, then

$$\lambda(s) = 0.2717 - 0.0053(s - 7.3827)$$

$$x(s) = 4.4938 + 0.1000(s - 7.3827).$$

9.2 Bellman Equation Collocation Methods

Consider Bellman's equation for an infinite horizon discrete time continuous state dynamic decision problem:

$$V(s) = \max_{x \in X(s)} \{f(s,x) + \delta E_\epsilon V(g(s,x,\epsilon))\} \qquad s \in S.$$

To compute an approximate solution to Bellman's equation via collocation, one employs the following strategy: First, one approximates the unknown value function V using a linear combination of known basis functions $\phi_1, \phi_2, \ldots, \phi_n$ whose coefficients $c_1, c_2, \ldots, c_n$ are to be determined:

$$V(s) \approx \sum_{j=1}^{n} c_j \phi_j(s).$$

Second, the basis function coefficients $c_1, c_2, \ldots, c_n$ are fixed by requiring the approximant to satisfy Bellman's equation, not at all possible states, but rather at n states $s_1, s_2, \ldots, s_n$, called the collocation nodes. Many collocation node and basis function schemes are available to the analyst, including Chebychev polynomials and nodes, and spline functions and uniform nodes.

The collocation strategy replaces the Bellman functional equation with a system of n nonlinear equations in n unknowns. Specifically, to compute the approximate solution to the Bellman equation, or more precisely, to compute the n basis coefficients $c_1, c_2, \ldots, c_n$ in the basis representation of the approximant, one solves the n *collocation equations*:

$$\sum_j c_j \phi_j(s_i) = \max_{x \in X(s_i)} \{ f(s_i, x) + \delta E_\epsilon \sum_{j=1}^{n} c_j \phi_j(g(s, x, \epsilon)) \} \qquad i = 1, 2, \ldots, n.$$

The collocation equation can be more compactly written in the form

$$\Phi c = v(c).$$

Here, Φ, the *collocation matrix*, is the n by n matrix whose typical ij^{th} element is the j^{th} basis function evaluated at the i^{th} collocation node

$$\Phi_{ij} = \phi_j(s_i)$$

and v, the *conditional value function*, is a function from $\Re^n$ to $\Re^n$ whose typical i^{th} element is

$$v_i(c) = \max_{x \in X(s_i)} \{ f(s_i, x) + \delta E_\epsilon \sum_{j=1}^{n} c_j \phi_j(g(s_i, x, \epsilon)) \}.$$

The conditional value function gives the maximum value obtained when solving the optimization problem embedded in Bellman's equation at each collocation node, taking the current coefficient vector c as given.

In principle, the collocation equation may be solved using any nonlinear equation solution method. For example, one could write the collocation equation in the equivalent fixed-point form $c = \Phi^{-1} v(c)$ and use function iteration, which employs the iterative update rule

$$c \leftarrow \Phi^{-1} v(c).$$

At each function iteration, the conditional value $v_i(c)$ must be computed at every i, that is, the optimization problem embedded in Bellman's equation must be solved at every collocation node s_i, taking the current coefficient vector c as fixed. The coefficient vector is then updated by premultiplying the vector of the updated optimal values obtained by the inverse of the collocation matrix.

Alternatively, one may write the collocation equation as a rootfinding problem $\Phi c - v(c) = 0$ and solve for c using Newton's method. Newton's method uses the iterative updating rule

$$c \leftarrow c - [\Phi - v'(c)]^{-1}[\Phi c - v(c)].$$

were $v'(c)$ is the n by n Jacobian of the conditional value function v at c. The typical element of v' may be computed by applying the Envelope Theorem to the optimization problem that defines $v_i(c)$:

$$v'_{ij}c = \frac{\partial v_i}{\partial c_j}(c) = \delta E_\epsilon \phi_j(g(s_i, x_i, \epsilon))$$

where x_i is the optimal argument in the maximization problem in the definition of $v_i(c)$ above. As a variant to Newton's method one could also employ a quasi-Newton method to solve the collocation equation. Both Newton and quasi-Newton methods, like function iteration, require that the optimization problem embedded in Bellman's equation to be solved for every collocation node with each iteration. The Newton method has the additional requirement of computing the derivative of v at c. Computing the derivative, however, comes at only a small additional cost because most of the effort required to compute the derivative comes from solving the optimization problem embedded in Bellman's equation, at task that must be performed regardless of the solution method used.

Of course, in any collocation scheme, if the model is stochastic and not deterministic, one must handle the expectation operation in a numerically feasible way. Based on numerical analysis theory and practice, a Gaussian quadrature scheme is strongly recommended in collocation schemes. When using a Gaussian quadrature scheme, the continuous random variable ϵ in the state transition function is replaced with a discrete approximant, say, one that assumes values $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$ with probabilities $w_1, w_2, \ldots, w_m$, respectively. In this instance, the conditional value function v takes the form

$$v_i(c) = \max_{x \in X(s_i)} \{f(s_i, x) + \delta \sum_{k=1}^{m} \sum_{j=1}^{n} w_k c_j \phi_j(g(s_i, x, \epsilon_k))\}.$$

and its Jacobian takes the form

$$v'_{ij}c = \delta \sum_{k=1}^{m} w_k \phi_j(g(s_i, x_i, \epsilon_k)).$$

The critical step in numerically implementing the collocation method to solve Bellman's equation is to evaluate the conditional value function $v(c)$ and its Jacobian. This, in turn, requires the analyst to write a subroutine to solve the maximization problem embedded in Bellman's equation. For reasons that will be made clear shortly, one should write a subroutine that solves the optimization problem embedded in Bellman's equation, not just at the collocation nodes, but any arbitrary vector of states. More specifically, given an n-degree interpolation scheme selected by the analyst, the subroutine should solve the optimization problem

$$\max_{x \in X(s_i)} \{f(s_i, x) + \delta E_\epsilon \sum_{j=1}^{n} c_j \phi_j(g(s_i, x, \epsilon))\}.$$

for every element of an arbitrary vector s of state nodes and any n-vector c of basis coefficients. The subroutine should also return the optimal policy at each of the states and the derivatives with respect to the basis coefficients.

The specific form of the maximization routine written by the analyst will differ according to the dimensionality of the state and action spaces and whether the action space is discrete or continuous. For the moment, however, assume that such a subroutine has been written. In its minimal form, the subroutine will have the following calling sequence:

```
[v,vc] = vmax(c,s).
```

Here, on input, s is an m-vector of states, and c is an n-vector of basis coefficients of the current value function approximant. On output, v is the m-vector of optimal values obtained by solving the optimization embedded in Bellman's equation for each state in the vector s, x is the m-vector of optimal policies for each of these optimization problems, and vc is the m-by-n vector of partial derivatives of the values with respect to the basis coefficients.

Given the optimization routine, implementation of a collocation scheme to solve Bellman's equation in Matlab is straightforward. First, the analyst must chose an underlying interpolation scheme by specifying the basis functions and collocation nodes to be used—we will assume a Chebychev approximation scheme for the purposes of illustration. More specifically, the analyst must specify the lower bound of the state interval `smin`, the upper bound of the state interval `smax`, the degree of interpolation `n`, and form the collocation nodes

```
s = nodecheb(n,smin,smax);
```

Given the collocation nodes, one then forms the interpolation matrix by evaluating the basis functions at the collocation nodes:

```
phi = basecheb(s,n,smin,smax);
```

After choosing the Gaussian nodes and weights for the random shock, if the model is stochastic, one may implement either a function iteration or Newton iteration scheme. To implement the function iteration scheme one initializes the basis function coefficients by setting them equal to zero

```
c = zeros(n,1);
```

or by making some other good guess, if readily available. The basic structure of function iteration takes the form:

```
for it=1:maxit
   cold = c;
   v = vmax(c,s);
   c = phi\v;
   change = norm(c-cold);
   if change<tol, break, end;
end
```

Here, `tol` and `maxit` are iteration control parameters set by the analyst. The basic structure of Newton iteration takes the form:

```
for it=1:maxit
   cold = c;
   [v,vc] = vmax(c,s);
   c = cold - [phi-vc]\[phi*c-v];
   change = norm(c-cold);
   if change<tol, break, end;
end
```

Once convergence has apparently been achieved, the analyst must perform two essential diagnostic checks. First, since interpolants may provide inaccurate approximations when evaluated outside the interpolation interval, one must check to ensure that all possible state transitions from the collocation nodes remain within the interpolation interval. This can be done easily as follows:

```
g = [];
for k=1:m;
   g = [g gfunc(s,x,e(k))];
end
if min(min(g))<smin, disp('Warning:  reduce smin'), end;
if max(max(g))>smax, disp('Warning:  increase smax'), end;
```

Here, `gfunc` is an user supplied routine that evaluates the state transition function at an arbitrary vector of states, actions, and shocks.

Next, one must check to see that value function approximant solves Bellman's equation to an acceptable degree of accuracy over the entire approximation interval. Since, by construction, the approximant generated by the solving the collocation equation must solve Bellman's equation exactly at the collocation nodes, this amounts to checking the approximation error at non node points. The easiest way to do this is to plot, over a fine grid spanning the interpolation interval, the residual between the values obtained from the approximant and the values obtained by directly solving the optimization problem embedded in Bellman's equation. For example, if 50 Chebychev collocation nodes are use to interpolate the value function, the approximation residual could be checked at 500 equally spaced nodes as follows:

```
nplot = 500;
splot = nodeunif(nplot,smin,smax);
resid = vmax(c,splot) - basecheb(splot,n,smin,smax)*c;
plot(splot,resid)
```

If the residual appears to be reasonably small throughout the entire approximation interval, the computed value function approximant is accepted, otherwise it is rejected and a new one is computed using either more collocation nodes or an alternative interpolation scheme. Notice how `vmax` was evaluated at states that are not collocation nodes—this is why `vmax` was constructed to accept an arbitrary vector of states, not just the collocation nodes.

9.2.1 Example: Infinite Put Option

The simplest continuous state Markov decision model encountered in practice involve binary choice. Dynamic binary choice models come in various form, including the optimal stopping problem and the optimal replacement problem. Consider the optimal stopping problem. At each point in time t the agent is offered a one-time reward $f(s_t)$ that depends on the state of some purely exogenous stochastic economic process s_t. The agent must then decide whether to accept the offer, receiving the reward, or decline the offer, forgoing the reward and waiting another period for a better reward. Assuming that the economic process s_t is a continuous state Markov process with transition function $s_{t+1} = g(s_t, \epsilon_{t+1})$, the agent's decision problem is captured by the Bellman equation

$$V(s) = \max\{f(s), \delta E_\epsilon V(g(s,\epsilon))\}, \qquad s \in S,$$

To solve this Bellman equation numerically by collocation, one first uses Gaussian quadrature methods to replace the shock ϵ with a discrete random variable, say, one that assumes values $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$ with probabilities $w_1, w_2, \ldots, w_m$, respectively. If the transition function g is monotonic in ϵ, say, increasing in ϵ, then one can easily compute a minimum and maximum state for the value function interpolation interval by solving the two fixed point problems

$$s_{min} = g(s_{min}, \epsilon_{min})$$

$$s_{max} = g(s_{max}, \epsilon_{max})$$

These two state values define an interval $I = [s_{min}, s_{max}]$ with the property that $g(s, \epsilon_j) \in I$ for all j whenever $s \in I$. That is, given the shock discretization, the interval will not be extrapolated by the numerical collocation routine if the collocation nodes are chosen within the interval.

To compute an approximate solution to Bellman's equation via collocation, one employs the following strategy: One approximates the unknown value function V using a linear combination of known basis functions $\phi_1, \phi_2, \ldots, \phi_n$ defined on I, whose basis coefficients $c_1, c_2, \ldots, c_n$ are to be determined:

$$V(s) \approx \sum_{j=1}^{n} c_j \phi_j(s).$$

One then fixes the basis function coefficients $c_1, c_2, \ldots, c_n$ by requiring the approximant to satisfy Bellman's equation at n collocation nodes $s_1, s_2, \ldots, s_n$ in I. Specifically, one solves the nonlinear vector collocation equation

$$\Phi c = v(c)$$

where Φ is the interpolation matrix associated with the underlying basis-node interpolation scheme and

$$v_i(c) = \max_{x \in X(s_i)} \{f(s_i), \delta \sum_{k=1}^{m} \sum_{j=1}^{n} c_j \phi_j(g(s_i, \epsilon_k))\}.$$

To solve the collocation equation via Newton's method further requires one to compute the Jacobian of v, which is given by

$$v'_{ij}(c) = \frac{\partial v_i}{\partial c_j}(c) = \begin{cases} 0 & v_i(c) > f(s_i) \\ \delta \sum_{k=1}^{m} \phi_j(g(s_i, \epsilon_k)) & \text{otherwise} \end{cases}$$

The Bellman equation for an infinite put option is solved via collocation in the Matlab routine Opt1.m supplied with these lecture notes.

9.2.2 Example: Stochastic Optimal Growth

Consider the problem of numerically solving the stochastic optimal growth problem of the preceding chapter under the assumption that $u(c) = c^{1-\alpha}/(1-\alpha)$, $f(x) = x^{\beta}$, and $\log(\epsilon_t)$ is i.i.d Normal$(0, \sigma^2)$ with $\alpha = 0.2$, $\beta = 0.5$, $\gamma = 0.9$, and $\delta = 0.9$.

To solve the optimal growth model using collocation, one selects a series of n basis functions ϕ_j and n collocation nodes s_i, and writes an approximation

$$V(s) \approx \sum_{j=1}^{n} c_j \phi_j(s).$$

The unknown vector of basis coefficients c is computed by solving the collocation equation

$$\Phi c = v(c)$$

where Φ is the interpolation matrix constructed by evaluating the basis functions at the collocation nodes and

$$v_i(c) = \max_{0 \le x \le s_i} \{(s_i - x)^{1-\alpha}/(1-\alpha) + \delta E_{\epsilon} \sum_{j=1}^{n} c_j \phi_j(\gamma x + \epsilon x^{\beta})\}.$$

To solve the collocation equation via Newton's method further requires one to compute the Jacobian of v, which is given by

$$v'_{ij}(c) = \frac{\partial v_i}{\partial c_j}(c) = \delta E_\epsilon \phi_j(\gamma x_i + \epsilon x_i^\beta)$$

where x_i solves the optimization problem above. To compute the expectation, one uses Gaussian quadrature scheme to replace the stochastic shock with a discrete approximant, say, one that assumes values $\epsilon_1, \epsilon_2, \ldots, \epsilon_m$ with probabilities $w_1, w_2, \ldots, w_m$, respectively.

The Bellman equation of the stochastic optimal growth model is solved via collocation in the Matlab routine Growval.m supplied with these lecture notes.

9.2.3 Example: Renewable Resource Problem

Consider the renewable resource problem under the assumptions that $p(x) = x^{-\gamma}$, $c(x) = kx$, and $g(s, x) = \alpha(s - x) - 0.5\beta(s - x)^2$ where $\gamma = 0.5$, $\alpha = 4$, $\beta = 1$, $k = 0.2$, and $\delta = 0.9$.

To solve the renewable resource problem using collocation, one selects a series of n basis functions ϕ_j and n collocation nodes s_i, and writes an approximation

$$V(s) \approx \sum_{j=1}^{n} c_j \phi_j(s).$$

The unknown vector of basis coefficients c is computed by solving the collocation equation

$$\Phi c = v(c)$$

where Φ is the interpolation matrix constructed by evaluating the basis functions at the collocation nodes and

$$v_i(c) = \max_{0 \le x \le s_i} \{x^{1-\gamma}/(1-\gamma) - kx + \delta E_\epsilon \sum_{j=1}^{n} c_j \phi_j(\alpha(s_i - x) - 0.5\beta(s_i - x)^2)\}.$$

To solve the collocation equation via Newton's method further requires one to compute the Jacobian of v, which is given by

$$v'_{ij}(c) = \frac{\partial v_i}{\partial c_j}(c) = \delta E_\epsilon \phi_j(\alpha(s_i - x_i) - 0.5\beta(s_i - x_i)^2)$$

where x_i solves the maximization problem above.

The Bellman equation of the renewable resource problem is solved via collocation in the Matlab routine Renrval.m supplied with these lecture notes.

9.2.4 Example: Nonrenewable Resource Problem

Consider the mine management problem under the assumption that $c(s,x) = x^2/(s+\beta)$ where $\alpha = 1$, $\beta = 10$, and $\delta = 0.9$.

To solve the nonrenewable resource problem using collocation, one selects a series of n basis functions ϕ_j and n collocation nodes s_i, and writes an approximation

$$V(s) \approx \sum_{j=1}^{n} c_j \phi_j(s).$$

The unknown vector of basis coefficients c is computed by solving the collocation equation

$$\Phi c = v(c)$$

where Φ is the interpolation matrix constructed by evaluating the basis functions at the collocation nodes and

$$v_i(c) = \max_{0 \le x \le s_i} \{\alpha x - x^2/(s+\beta) + \delta E_\epsilon \sum_{j=1}^{n} c_j \phi_j(s_i - x)\}.$$

To solve the collocation equation via Newton's method further requires one to compute the Jacobian of v, which is given by

$$v'_{ij}(c) = \frac{\partial v_i}{\partial c_j}(c) = \delta E_\epsilon \phi_j(s_i - x_i)\}$$

where x_i solves the maximization problem above.

The Bellman equation of the nonrenewable resource problem is solved via collocation in the Matlab routine Mineval.m supplied with these lecture notes.

9.3 Solving Euler Equations via Collocation

Euler equation methods call for solving the first-order Euler equilibrium conditions of the continuous-space decision problem for the unknown shadow price function λ. Since I did not cover collocation for Euler equations in depth in class, I will not expect you to know them for the exam.

9.3.1 Example: Stochastic Optimal Growth

9.3.2 Example: Renewable Resource Problem

9.3.3 Example: Nonrenewable Resource Problem

9.4 Solving Rational Expectation Models via Collocation

I did not get to cover these methods in sufficient depth in class. I will not exam you on them.

9.4.1 Example: Commodity Storage

9.4.2 Example: Asset Pricing Model

9.5 Comparison of Solution Methods

In developing a numerical approximation strategy for solving Bellman's equation, one pursues a series of multiple, sometimes conflicting goals. First, the algorithm should offer a high degree of accuracy for a minimal computational effort. Second, the algorithm should be capable of yielding arbitrary accuracy, given sufficient computational effort. Third, the algorithm should yields answers with minimal convergence problems. Fourth, it should be possible to code the algorithm relatively quickly with limited chances for programmer error.

Space discretization has some major advantages for computing approximate solutions to continuous-space dynamic decision problems. The biggest advantage to space discretization is that it is easy to implement. In particular, the optimization problem embedded in Bellman's equation is solved by complete enumeration, which is easy to code and numerically stable. Also, constraints are easily handled by the complete enumeration algorithm. Each time a new action is examined, one simply tests whether the action satisfies the constraint, and rejects it if it fails to do so. Finally, space discretization can provide an arbitrarily accurate approximation by increasing the number of state nodes.

Space discretization, however, has several major disadvantages. The biggest disadvantage is that complete enumeration is extremely slow. Complete enumeration mindlessly examines all possible actions, ignoring the derivative information that would otherwise help to find the optimal action. Another drawback to space discretization is that it uses discontinuous step functions to approximate the value and policy functions. The approximate optimal solution generated by space discretization will not possess the smoothness and curvature properties of the true optimal solution. Finally, because the states and actions are forced to coincide with specified nodes, the accuracy afforded by space discretization will be limited by the coarseness of the state and action space grids.

Linear-quadratic approximation is perhaps the method easiest to implement. The solution to the approximating problem is a linear function whose coefficients can be derived analytically using the methods discussed in section (*). Alternatively, the coefficients can easily be computed numerically using a successive approximation scheme that is typically free of convergence problems.

Linear-quadratic approximation, however, has some severe shortcomings. The basic problem with linear-quadratic approximation is that it relies on Taylor series approximations that are accurate only in the vicinity of the steady-state, and then only if the process is deterministic or nearly so. Linear-quadratic approximation will yield poor results if random shocks repeatedly throw the state variable far from the steady-state and if the reward and state transition functions are not accurately approximated by second- and first-degree polynomials over their entire domains. Linear-quadratic approximation will yield especially poor approximations if the true optimal process is likely to encounter any inequality and nonnegativity constraints, which must be discarded in passing to a linear-quadratic approximation.

Collocation methods address many of the shortcomings of linear-quadratic approximation and space discretization methods. Unlike linear-quadratic approximation, collocation methods employ global, rather than local, function approximation schemes and, unlike space discretization, they approximate the solution using a smooth, not discontinuous, function. Chebychev collocation methods, in particular, are motivated by the Wieirstrass polynomial approximation theorem, which asserts that a smooth function can be approximated to any level of accuracy using a polynomial of sufficiently high degree. A second important advantage to collocation methods is that they may employ rootfinding or optimization that exploit derivative information. A differentiable approach can help pinpoint the equilibrium solution at each state node faster and more accurately than the complete enumeration scheme of discrete dynamic programming.

The collocation method replaces the inherently infinite-dimensional functional equation problem with a finite-dimensional nonlinear equation problem that can be solved using standard nonlinear equation methods. The accuracy afforded by the computed approximant will depend on a number of factors, most notably the number of basis functions and collocation nodes n. The greater the degree of approximation n, the more accurate the resulting approximant, but the more expensive is its computation. For this reason choosing a good set of basis functions and collocation nodes is critical for achieving computational efficiency. Approximation theory suggests that Chebychev polynomials basis functions and Chebychev collocation points will often make superior choices, provided the solution to the functional equation is relatively smooth. Otherwise, linear or cubic basic splines with equally spaced collocation nodes may provide better approximation.

In using collocation schemes, one might be tempted to choose equally spaced points and to represent the interpolating polynomial as the linear combination of the standard monomials. However, as seen in Chapter 3, uniform node polynomial interpolation can yield extremely poor global approximations and can produce explosive approximation error. Also, computing the monomial coefficients of an interpolating polynomial is an ill-conditioned process that is highly vulnerable to rounding error and convergence failure.

Numerical analysis theory suggest that the Chebychev interpolation nodes and Chebychev polynomials are nearly optimal choices for forming polynomial interpolants. Accuracy and efficiency with Chebychev nodes and polynomials are guaranteed by Chebychev polynomial approximation theorem, which asserts that, for a given degree, the best approximating polynomial is the one that interpolates the function at the Chebychev nodes. The theorem also asserts that such approximation error will tend to disappear if the degree of approximation is increased. Also, using this combination of nodes and basis polynomials will ensure that the interpolating matrix will be orthogonal. Thus, computing the coefficients c_j of the interpolating polynomial will be faster and numerically more stable than for other polynomial bases.

Chebychev collocation, however, is not without its disadvantages. First, polynomial interpolants can behave strangely outside the range of interpolation and should be extrapolated with extreme caution. Even when state variable bounds for the model solution are known, states outside the bounds can easily be generated in the early stages of the solution algorithm, leading to convergence problems. Also, polynomial interpolants can behave strangely in the vicinity of nondifferentiabilities in the function being interpolated. In particular, interpolating polynomials can fail to preserve monotonicity properties near such points, undermining the rootfinding algorithm used to compute the equilibrium at each state node. Finally, inequality constraints, such as nonnegativity constraints, require the use of special methods for solving nonlinear complementarity problems.

Table 1 gives the execution time and approximation error associated with four solution schemes, including uniform polynomial and Chebychev collocation, as applied to the commodity storage model examined in section (*). Approximation error is defined as the maximum absolute difference between the "true" price function and the approximant at points spaced 0.001 units apart over the approximation interval $[0.5, 2.0]$. Execution times are based on the successive approximation algorithm implemented on an 80486 50 megahertz Gateway 2000 personal microcomputer.

The superiority of the Chebychev collocation for solving the storage model is evident from table 1. The accuracy afforded by Chebychev collocation exceeded that of space discretization by several orders of magnitude. For example, the accuracy achieved by space discretization in nearly five minutes of computation was easily achieved by Chebychev collocation in less than one-tenth of a second. In the same amount of time, the linear-quadratic approximation method afforded an approximation that was three orders of magnitude worse than that afforded by Chebychev collocation. The approximation afforded by linear-quadratic approximation, moreover, was not subject to improvement by raising the degree of the approximation, which is fixed. Finally, as seen in table 1, when using uniform node, monomial collocation, the approximation error actually increased as the number of nodes doubled from 10 to 20; the algorithm, moreover, would not converge for more than 23 nodes. The example thus illustrates once again the inconsistency and instability of uniform node monomial interpolation.

Method	Number of Nodes	Execution Time (seconds)	Maximum Absolute Error
Chebychev Polynomial Collocation	10	0.1	4.7E−02
	20	0.4	1.1E−02
	30	0.7	2.7E−03
	40	1.1	5.9E−04
	50	1.6	3.3E−04
	100	5.8	3.1E−06
	150	12.5	2.3E−08
Uniform Polynomial Collocation	10	0.1	1.4E−01
	20	0.3	1.7E+00
	30	N.A.	N.A.
Space Discretization	10	2.0	4.5E+00
	20	7.5	1.7E+00
	30	16.9	8.6E−01
	40	31.0	5.3E−01
	50	32.3	3.5E−01
	100	124.6	9.7E−02
	150	292.2	4.5E−02
L-Q Approximation		0.1	2.8E+01

Table 9.1: Execution Times and Approximation Error for Selected Continuous-Space Approximation Methods