Numerical Dynamic Programming with Shape-Preserving Splines

Kenneth L. Judd
Hoover Institution

Andrew Solnick
Stanford University

1994 [PRELIMINARY AND INCOMPLETE]

Abstract. Parametric approximations of the value function are a critical feature of the value function iteration method for solving dynamic programming problems with continuous states. Simple approximation methods such as polynomial or spline interpolation may cause value function iteration to diverge. We show that shape-preserving splines can avoid divergence problems while producing a smooth approximation to the value function.

Dynamic programming is a basic tool of dynamic economic analysis, allowing economists to examine a wide variety of economic problems. Theoretical properties are well-understood (see Bertsekas (1976), and Bertsekas and Shreve (1978)), and there have been numerous applications in economic theory; see Lucas and Stokey (1987), Sargent (1987), and Rust(1993, 1994) for recent reviews of this literature.

There has also been extensive work in the operations research literature to develop numerical solution methods for dynamic programming. The first methods were designed for finite-state problems, and reduced to simple matrix computations. Initially, continuous-state problems were solved with finite-state approximations; unfortunately, that approach often results in impractically large finite-state Markov problems. Dantzig et al. (1974) advocated a multilinear approximation approach. Beginning with Bellman et al. (1963), polynomial approximations have been studied, and yield methods which, according to approximation theory, are potentially far faster for smooth problems. Daniel (1976) and Johnson et al.(1993) have demonstrated the advantages of splines for dynamic programming.

Despite these facts, state discretization has continued to be used extensively in economic applications. One justification which has been offered (see Rust, 1993, 1994) is that convergence theorems are available for state discretization methods, whereas few corresponding results exist for polynomial methods. Indeed, we will show below that there can be no convergence theorems for the most promising approaches by providing examples where these methods diverge wildly.

The task then is to find polynomial methods for continuous-state problems which maintain the many advantages of smooth approximation methods but also lead to convergent algorithms for dynamic programming. In this paper we develop two efficient approaches. First, we show that there are two critical properties which are each sufficient for the convergence of approximation methods: *monotonicity in the data* and *shape-preservation*.

These convergent schemes may be quite conservative in that they take a long time to converge. Nonconvergent schemes may still be valuable if we can compute upper bounds on their accuracy. For example, in nonlinear equations, Newton's method is used far more widely than the globally convergent Scarf algorithm and related homotopy methods. This is quite acceptable because we can check whether the final approximation is nearly a solution. We show that approximation methods which are not always convergent can be excellent under certain conditions; in the context of a common example, we display the critical properties of a problem which are critical in determining whether the less reliable schemes will work.

We will first review basic dynamic programming methods. We then describe the canonical continuous approximation scheme for numerical dynamic programming. We then discuss approaches which provide convergent algorithms, and approaches which are stable, both of which revolve around monotone approximations, and shape-preserving approximation methods. The paper concludes with some applications to simple wealth accumulation models, both concave and nonconcave.

1. Dynamic Programming Problems

We first define the general dynamic programming problem. In dynamic programming, we are concerned with controlling a dynamic process which takes on several possible states, can be influenced by the application of controls, and yields a stream of state- and control-dependent payoffs.

The current flow of payoffs to the controller is $v(u,x,t)$ in period t if $x \in X$ is the beginning-of-period state and X the set of states, and the control u is applied in period t. There may be state-contingent constraints on the controls; let $D(x,t)$ be the nonempty set of controls which are feasible in state x at time t. The state in period $t+1$ depends on the period t state and action in a possibly stochastic fashion; let $F(A;x,u,t)$ be the probability that $x_{t+1} \in A \subset X$ if the period t state is x and the period t control is u. We assume that $F(\cdot;x,u,t)$ describes a probability measure over X for each state x, control u, and time t.

The objective of the controller is to maximize expected total returns,

$$E\left\{\sum_{t=0}^{T} v(x_t,u_t,t) + W(x_T)\right\}$$

where $W(x)$ is the terminal valuation. We define the value function, $V(x,t)$, to be greatest possible total payoff from time t to T if the time t state is x; formally,

$$V(x,t) \equiv \sup_{\mathcal{U}} E\left\{\sum_{s=t}^{T} v(x_s,u_s s) + W(x_T)\right\}$$

where $\mathcal{U}$ is the set of all feasible strategies. The value function satisfies the Bellman equation

$$V(x,t) = \sup_{u \in D(x,t)} v(u,x,t) + \beta E\{\ V\left(x^+,t+1\right)|x,u)\}$$

where x^+ is the next period's state. If it exists, the optimal policy function, $U(x,t)$, achieves the value $V(x,t)$ and solves

$$U(x,t)\ \in \arg\max_{u \in D(x,t)}\ v(u,x,t) + \beta E\{\ V\left(x^+,t+1\right)|x,u\}$$

The value function always exists since it is a supremum. The existence of U is not assured in general, but, by definition of supremum, there is a policy function which uniformly comes within ϵ of achieving the value function for any $\epsilon > 0$.

The infinite horizon, autonomous problem is frequently used to analyze long-run economic problems. In such problems, the current flow of payoffs to the controller is $v(u,x)$ per period if the current state is x and the current control is u. Let $D(x)$ be the nonempty set of controls which are feasible in state x. The state tomorrow depends on the

current state and action in a possibly stochastic fashion; let $F(A; x, u)$ be the probability distribution of tomorrow's state if the current state is x and the current control is u.

The objective of the controller is to maximize discounted expected returns

$$E\left\{\sum_{t=0}^{\infty}\beta^t v(x_t, u_t)\right\}$$

where β is the discount rate. The value function is defined to be

$$V(x) \equiv \sup_{\mathcal{U}} E\left\{\beta^t \sum_{t=0}^{\infty} v(x_t, u_t)\right\}$$

where $\mathcal{U}$ is the set of all feasible strategies. The value function satisfies the Bellman equation

$$V(x) = \max_{u\in D(x)} \; v(u,x) + \beta E\{\, V\left(x^+\right)|x,u)\} \equiv (TV)(x)$$

and the policy function, $U(x)$, solves

$$U(x) \;\in \arg\max_{u\in D(x)} \; v(u,x) + \beta E\{\, V\left(x^+\right)|x,u\}$$

Under mild conditions, $V(x)$ exists by the contraction mapping theorem.

2. A Simple Dynamic Programming Problem

It will be convenient to use a familiar simple problem as an example in our discussions of various computation methods. Consider the optimal accumulation problem

$$\begin{aligned}
&\max_{c_t} E\left\{\sum_{t=0}^{\infty}\beta^t u(c_t)\right\} \\
&k_{t+1} = f(k_t, \theta_t) - c_t \\
&\theta_{t+1} = g(\theta_t, \epsilon_{t+1}) \\
&k_t \geq 0
\end{aligned}$$

where c_t is consumption in period t, $u(c)$ is the utility function at each date, k_t is the wealth at the beginning of period t, and $f(k,\theta)$ is the gross income in a period which begins with wealth k and productivity parameter θ. The disturbance ϵ is i.i.d. We assume that consumption occurs after income has been observed.

This problem is a common one. It includes both the Brock-Mirman stochastic growth model and the Imrohoroglu precautionary saving model. In the first case $f(k,\theta)$ is a stochastic, aggregate production function and in the second $f(k,\theta)$ is the stochastic, serially correlated wage plus interest income on savings, k.

The dynamic programming formulation for the general problem is

$$\begin{aligned}
V(k,\theta) &= \max_{k^+} \; u(f(k,\theta) - k^+) + \beta E\left\{V(k^+, \theta^+) \mid \theta\right\} \\
&\equiv (TV)(k,\theta)
\end{aligned}$$

where θ^+ is next period's productivity factor. We will assume that u is concave, but will not always assume that f is concave. We will assume that f_k is small for large k so that both c and k is confined to compact, but large, intervals; this is necessary for the conventional contraction mapping arguments to apply.

3. BASIC SOLUTION METHODS

We next discuss the two basic ideal solution methods used to solve dynamic programming problems. We emphasize the term "ideal" since while they are constructive methods they are also purely abstract mathematical methods which ignore the practical problems of implementation on the computer. In fact, they cannot be implemented on the computer since they are infinite-dimensional. This distinction is often blurred in discussions of these methods, but this paper shows that it is dangerous to ignore this. After discussing the abstract methods, we then discuss methods which can be implemented.

3.1. Value Function Iteration. One simple approach for computing V is called value function iteration and is motivated by the contraction properties of the Bellman equation. Ideally, value function iteration computes the sequence

$$V^{n+1} = TV^n \ .$$

By the Contraction Mapping Theorem, V^n will converge to the infinite horizon value function where for any initial guess V^0. The sequence of control rules, U^n, will also converge to the optimal control rule.

This is a pure mathematical construction, not a true numerical algorithm. There are two practical problems which prevent us from achieving the ideal sequence. First, since the limit can never be achieved, in practice one iterates until the successive V^n and U^n change little. Second, the mapping T is a functional, that is, a map which takes a function and creates a new function. Computers cannot represent arbitrary functions. Therefore, we must approximate both V and T in some fashion in order to implement value function iteration.

The other popular approach is the policy function iteration method, also called the Howard improvement algorithm. Since we do not use it in our examples below, we do not discuss it here. However, the methods we examine below can be applied with policy function iteration.

3.2. Continuous Problems: Discretization Methods. In many economic applications, the state and control variables are both naturally continuous. One way to find an approximate solution to such problems is to specify a finite-state problem which is "similar" to the continuous problem, and solve the finite problem using the methods above. In this section, we shall examine the details of this approach.

In general, this is accomplished by replacing the continuous distribution F with a discrete distribution. The discretization approach chooses a finite set of states, X, and defines a discrete-state dynamic programming problem. The specification of state and controls must be such that the problem always remains on the grid X. The typical deterministic problem becomes

$$V(x_i) = \max_{u \in D(x_i), g(x_i,u) \in X} \pi(u, x_i) + \beta V\left(g(x_i, u)\right) \equiv (T_X V)(x_i)$$

where our choice of u is constrained by $D(x)$ and by the requirement that the future state belong to the grid X. Applying value function iteration to discrete problems is straightforward. In such problems, we are only solving for a finite number of unknown scalars, the $V(x_i)$.

Our simple example can be discretized in a simple way by defining the control to be the future state. This results in the Bellman equation

$$V_{n+1}(k) = \max_{k^+} \; u(f(k) - k^+) + \beta V_n\left(k^+\right)$$

3.3. Weaknesses of Finite-State Approximations. We will next discuss the problems with finite-state dynamic programming algorithms and possible avenues for improvement.

The first obvious problem is that a good approximation is likely to require a large number of discrete states. This problem will become particularly bad when one uses the discretization approach to solve multidimensional problems.

The other problem with finite-state approximations is that they do not generate as much information as is available in the original problem. A value function iteration will generate only information about the level of the value function at the points in the grid X.

However, when we solve our simple example problem,

$$V_{n+1}(k) = \max_{c} \; u(c) + \beta V_n\left(f(k) - c\right)$$

we not only get $V_{n+1}(k)$ but also, by envelope theorem and first-order condition

$$V'_{n+1}(k) = u'(c^*)$$

which is easily calculated once optimum consumption, c^*, is known.

This derivative information is actually more important for policy function approximation than is level information since c and $V'_{n+1}(k)$ are related by the first-order condition

$$u'(c) = \beta V'_{n+1}(f(k) - c)$$

Therefore, the quality of the approximation of policy function is only as good as the approximation of $V'(k)$, which can be orders of magnitude worse that the approximation of $V(k)$ if we only use level information.

Since the marginal value $V'_{n+1}(k)$ at the k's used is produced by the maximization step, it should be used if possible. Below we shall study methods which use this information.

3.4. Approximation Methods. A key problem in dynamic programming with continuous states is to find a way to approximate the value function. Typically, we will know the value function at a finite collection of points and must make a guess about its value elsewhere. In this section, we review the many methods which we can use to approximate value functions, using different kinds of data.

A key problem in dynamic programming with continuous states is to find a way to approximate the value function. Typically, we will know the value function at a finite collection of points and must make a guess about its value elsewhere. In this section, we review the many methods which we can use to approximate value functions, using different kinds of data.

L^p Approximation. finds a "nice" function g which is "close to" a given function f in the sense of a L^p norm. Interpolation and approximation are similar except that the data in interpolation is a finite set of points whereas in approximation our input is a function.

Interpolation is any method which takes information at a finite set of points, X, and finds a function which satisfies that information at the points X. The information consists of level, slope, possibly shape information about a curve. Different interpolation methods use different information and use it in different ways.

Lagrange Interpolation. Lagrange interpolation is a one-dimensional polynomial interpolation method which takes a collection of n points in R^2, $(x_i, y_i), i = 1, \cdots, n$, where the x_i are distinct, and finds a degree $n-1$ polynomial, $p(x)$, such that $y_i = p(x_i), i = 1, \cdots, n$. In this problem there are n distinct points imposing n constraints on the choice of n unknown polynomial coefficients of $p(x)$.

The Lagrange formula demonstrates that there is such interpolating polynomial. Define

$$\ell_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

Note that $\ell_i(x)$ is unity at $x = x_i$ and zero at $x = x_j$ for $i \neq j$. This property implies that the polynomial

$$p(x) = \sum_{i=1}^{n} y_i \ell_i(x)$$

interpolates the data, that is, $y_i = p(x_i), i = 1, \cdots, n$. This is also the unique such polynomial.

Schumaker's Shape-Preserving Splines. In this section we will construct the shape-preserving quadratic spline described in Schumaker (1983). First, we examine a Hermite interpolation problem. Then we discuss the Lagrange problem.

The basic Hermite problem is, given z_1, z_2, s_1, s_2 find a piecewise quadratic function $s \in C^1[t_1, t_2]$ such that

$$s(t_1) = z_i, \ s'(t_i) = s_i, \ i = 1, 2 \tag{8}$$

We first examine the nongeneric case where a quadratic works.

Lemma 1. *If* $\frac{s_1+s_2}{2} = \frac{z_2-z_1}{t_2-t_1}$, *then the quadratic polynomial*

$$s(t) = z_1 + s_1(t - t_1) + \frac{(s_2 - s_1)(t - t_1)^2}{2(t_2 - t_1)}$$

satisfies (8).

The shape-preserving properties of this construction are clear. First, if $s_1 \cdot s_2 \geq 0$, then $s'(t)$ has the same sign as s_1 and s_2 throughout $[t_1, t_2]$. Therefore, if the data indicate a monotone increasing (decreasing) function on $[t_1, t_2]$, then $s(t)$ is such a function. Second, if $s_1 < s_2$, then the data indicate a convex function, which $s(t)$ is since $s''(t) = (s_2 - s_1)/(t_2 - t_1)$. Similarly, if $s_1 > s_2$, then $s(t)$ and the data are concave.

In general, we need to add a knot to the interval (t_1, t_2) to solve (8).

Theorem 2. *There exists* ξ_1 *and* ξ_2 *such that* $t_1 < \xi_1 < \xi_2 < t_2$ *and for every* $\xi \in (\xi_1, \xi_2)$,

there is a unique quadratic spline solving (8) with a knot at ξ. It is

$$\begin{aligned}
s(t) &= \begin{array}{l} A_1 + B_1(t-t_1) + C_1(t-t_1)^2, \quad t \in [t_1, \xi], \\ A_2 + B_2(t-\xi) + C_2(t-\xi)^2, \ t \in [\xi, t_2], \end{array} \\
A_1 &= z_1, \ B_1 = s_1, C_1 = \tfrac{(\bar{s}-s_1)}{(2\alpha)}, \\
A_2 &= A_1 + \alpha B_1 + \alpha^2 C_1, B_2 = \bar{s}, \ C_2 = \tfrac{(s_2-\bar{s})}{2\beta}, \\
\bar{s} &= \tfrac{(2(z_2-z_1)-(\alpha s_1+\beta s_2))}{(t_2-t_1)}, \\
\alpha &= \xi - t_1, \ \beta = t_2 - \xi.
\end{aligned}$$

The shape-preserving properties of this quadratic spline are more complex. First, if $s_1 \cdot s_2 \geq 0$, then $s(t)$ is monotone if and only if $s_1 \bar{s} \geq 0$, which is equivalent to

$$\begin{aligned}
2(z_2 - z_1) &\geq (\xi - t_1)s_1 + (t_2 - \xi)s_2 \ if \ s_1, s_2 \geq 0, \\
2(z_2 - z_1) &\leq (\xi - t_1)s_1 + (t_2 - \xi)s_2 \ if \ s_1, s_2 \leq 0.
\end{aligned}$$

To deal with curvature, we compute $\delta = (z_2 - z_1)/(t_2 - t_1)$. We then note that if $(s_2 - \delta)(s_1 - \delta) \geq 0$, there must be an inflection point in $[t_1, t_2]$, and we can have neither a concave nor convex interpolant.

Otherwise, if $|s_2 - \delta| < |s_1 - \delta|$, then if ξ satisfies

$$t_1 < \xi \leq \bar{\xi} \equiv t_1 + \frac{2(t_2 - t_1)(s_2 - \delta)}{s_2 - s_1}$$

$s(t)$ is convex (concave) if $s_1 < s_2 (s_1 > s_2)$. Furthermore, if $s_1 s_2 > 0$, it is also monotone. If $|s_2 - \delta| > |s_1 - \delta|$, then, if

$$t_2 + \frac{2(t_2 - t_1)(s_1 - \delta)}{s_2 - s_1} \equiv \underline{\xi} \leq \xi < t_2$$

then s is convex (concave) if $s_1 < s_2 (s_1 > s_2)$, and if $s_1 s_2 > 0$, s is monotone.

These formulas were both for a single interval. We next consider a general interpolation problem. If we have Hermite data, i.e., we have $\{(z_i, s_i, t_i) \mid i = 1, \cdots, n\}$, we then apply the theorem to each interval. If we have Lagrange data, $\{(z_i, t_i) \mid i = 1, \cdots, n\}$, we first add estimates of the slopes. Schumaker suggests the formulas

$$\begin{aligned}
L_i &= \left[(t_{i+1} - t_i)^2 + (z_{i+1} - z_i)\right]^{1/2}, \ i = 1, \cdots, n-1 \\
\delta_i &= (z_{i+1} - z_i)/(t_{i+1} - t_i), \ i = 1, \cdots, n-1 \\
s_i &= (L_{i-1}\delta_{i-1} + L_i \delta_k)/(L_{i-1} + L_i), \ i = 2, \cdots, n-1 \\
s_1 &= (3\delta_1 - s_2)/2 \\
s_n &= (3\delta_{n-1} - s_{n-1})/2
\end{aligned}$$

These formulas are adjusted by adding the conditions

$$s_i = 0, \ \text{if } \delta_{i-1} \ \delta_i \leq 0,$$

that is, if the neighboring secants have opposite slope, we set the derivative to be zero at t. With the Hermite data, we can then proceed locally, computing the spline on each interval $[t_i, t_{i+1}] \equiv I_i$.

The only remaining problem is choosing the $\xi_i \in I_i$. The theorem above actually gives us a range of choices. On each I_i, we first determine the ξ_i which are consistent with

montonicity if $s_i \cdot s_{i+1} \geq 0$. If $(s_{i+1} - \delta_i)(s_i - \delta_i) < 0$, the data is either convex or concave and the choice of ξ_i is further restricted to preserve curvature. For specificity, choose the midpoint of the interval of ξ_i's which satisfy both restrictions. Other criterion could be to make the choice which results in minimum total curvature, or minimum maximal curvature.

The result of this scheme is a quadratic spline which is globally monotone if the data are monotone. Similarly for convexity and concavity. Furthermore, the spline is "co-monotone," that is, s is increasing (decreasing) on I_i iff $z_i < z_{i+1}$ ($z_i > z_{i+1}$). This property follows from the local monotonicity of the quadratic spline and our imposing $s_i = 0$ wherever the slope changes.

Preserving local curvature is also accomplished. If $\delta_i < \delta_{i+1} < \delta_{i+2} < \delta_{i+3}$ the data appears to be convex on I_{i+1}, and our construction is convex on I_{i+1} since $s_{i+1} < s_{i+2}$ by construction.

3.5. Error Bounds of Shape-Preserving Methods. We will next examine error bounds for dynamic programming algorithms. Let X be the interpolation nodes. Let

$$v_i = \max_u \; v(u, x_i) + \beta E\{V(x^+)|x_i, u\}$$

and let $\hat{V}(x; v)$ be the interpolation of the v_i. We will let $\hat{T}V$ denote the mapping from V to $\hat{TV}$. The typical numerical scheme attempts to find a $\hat{V}$ such that $\hat{V} = \hat{T}\hat{V}$. The key question is how the fixed points of $\hat{T}$ are related to the fixed points of T.

The first important facts are the following error bounds for T:

$$\| V - V^\infty \| \leq \| TV - V \| / (1 - \beta)$$

The contraction mapping implies

$$\| TV_1 - TV_2 \| \leq \beta \| V_1 - V_2 \|$$

This may not hold for the approximations, $\hat{V}$. The relevant approximation relation is:

$$\begin{aligned} \| \hat{T}V_1 - \hat{T}V_2 \| \leq & \| \hat{T}V_1 - TV_1 + TV_1 - TV_2 + TV_2 - \hat{T}V_2 \| \\ \leq & \| TV_1 - TV_2 \| + \| \hat{T}V_1 - TV_1 \| + \| \hat{T}V_2 - TV_2 \| \end{aligned}$$

This decomposes $\| TV_1 - TV_2 \|$ into three pieces, the first being $\| TV_1 - TV_2 \|$, which is bounded by $\beta \| V_1 - V_2 \|$ by the contraction theorem, and the other two being interpolation errors of the form $\| \hat{T}V - TV \|$

The formula

$$\begin{aligned} \| TV - V \| = & \| TV - \hat{T}V + \hat{T}V - V \| \\ \leq & \| TV - \hat{T}V \| + \| \hat{T}V - V \| \\ \equiv & \; \delta \end{aligned}$$

decomposes $\| TV - V \|$ into the approximation error and the maximum change. Furthermore, the McQueen-Porteus error formula implies

$$\| V - V^\infty \| \leq \delta / (1 - \beta).$$

If $\hat{T}V = V$, then

$$\| TV - V \| = \| TV - \hat{T}V \| \equiv \hat{e}$$

and

$$\| V - V^{\infty} \| \leq \hat{e}/(1-\beta)$$

Many dynamic programming problems have structure which we can exploit. For example, if the state variable is a vector of capital stocks then the value function is an increasing function of the state in many interesting cases. Shape-preservation is particularly valuable in concave problems, where T is a shape-preserving operator, as shown in a theorem of Lucas.

Theorem 3. *Suppose $v(u,x)$ is concave (increasing) in (u,x), the constraint set is convex for each x, and V is concave (increasing). Then $(TV)(x)$ is also concave (monotone), as is the fixed point, $TV^* = V^*$.*

This result is relevant even when we examine only a finite number of states. In particular, if $\hat{V}$ is concave in x, then the $(T\hat{V})(x_i)$ points, $x_i \in X$, will be consistent with concavity, and a shape-preserving scheme will cause the function approximation $(\hat{T}\hat{V})(x)$ to be a concave function of x.

Linear Interpolation. Shape-preserving approximations will not generate the internodal oscillations which may arise in the approximation step. In particular, these internodal oscillations can be avoided by the simplest of all interpolation schemes – linear interpolation, which is shape-preserving. Therefore, if the v_i points are increasing and concave, so will be the interpolating function. Linear interpolation satisfies particularly desirable properties for approximate dynamic programming.

Suppose that we know V is monotonically increasing. Consider the interval $I_i \equiv [x_i, x_{i+1}]$. Monotonicity implies, for $x \in [x_i, x_{i+1}]$

$$v_i \leq (TV)(x) \leq v_{i+1}$$

Therefore, the approximation error on I_i is at most $v_{i+1} - v_i \equiv \Delta_i$. Linear approximation method implies

$$\| \hat{T}V - TV \| \leq \max_i \ \Delta_i$$

Concavity-Preserving, C^1 Interpolation. We will next discuss error bounds for numerical approximations to solutions of concave dynamic programming problems when we use methods which preserve concavity and monotonicity of the value function.

Suppose that we know V is concave and monotone, but that we use linear approximation. Consider Figure 2, where we have displayed the values at x_{i-1}, x_i, x_{i+1}, and x_{i+2}. Define v_i^- to be the left derivative of $(\hat{T}V)(x_i)$ at x_i,

$$v_i^- \equiv \frac{v_i - v_{i-1}}{x_i - x_{i-1}}$$

and define v_{i+1}^+ to be the right derivative of $(\hat{T}V)(x_{i+1})$ at x_{i+1},

$$v_{i+1}^+ \equiv \frac{v_{i+2} - v_{i+1}}{x_{i+2} - x_{i+1}}$$

Then on $[x_i, x_{i+1}]$ $(TV)(x)$ is bounded below by the chord BC and above by BEC. The maximum error of the linear approximation can occur at

$$x_i^* \equiv \frac{v_{i+1} - v_i + v_i^- x_i - v_{i+1}^+ x_{i+1}}{v_i^- - v_{i+1}^+}$$

and the maximum vertical distance is

$$\Delta_i \equiv (v_i^- - \frac{v_{i+1} - v_i}{x_{i+1} - x_i})(x_i^* - x_i)$$

We can do better if we use slope information at the interpolation nodes. Concavity and slope information at the nodes implies

$$\begin{aligned}\hat{V}(x) \leq (TV)(x) &\leq \min\{v_i + v_i'(x - x_i), v_{i+1} + v_{i+1}'(x - x_{i+1}\} \\ &\leq \min\{v_i + v_i'(x_{i+1} - x_i), v_{i+1} + v_{i+1}'(x_i - x_{i+1}\}\end{aligned}$$

Again, on $[x_i, x_{i+1}]$ $(TV)(x)$ is bounded below by the chord BC, but we now have an upper bound of BFC. The maximum error of the linear approximation can occur at

$$x_i^* \equiv \frac{v_{i+1} - v_i + v_i' x_i - v_{i+1}' x_{i+1}}{v_i' - v_{i+1}'}$$

and the maximum vertical distance is

$$\Delta_i' \equiv (v_i' - \frac{v_{i+1} - v_i}{x_{i+1} - x_i})(x_i^* - x_i)$$

Hermite shape-preserving interpolation which uses this information implies

$$\| \hat{T}V - TV \| \leq \max_i \Delta_i'.$$

We can now incorporate these approximation error bounds into an error bound for the dynamic programming problem. The formula

$$\begin{aligned}\| TV - V \| &= \| TV - \hat{T}V + \hat{T}V - V \| \\ &\leq \| TV - \hat{T}V \| + \| \hat{T}V - V \| \\ &\equiv \delta\end{aligned}$$

decomposes $\| TV - V \|$ into the approximation error and the maximum change. Furthermore,

$$\| V - V^\infty \| \leq \delta/(1 - \beta).$$

If $\hat{T}V = V$, then

$$\| TV - V \| = \| TV - \hat{T}V \| \equiv \hat{e}$$

and

$$\| V - V^\infty \| \leq \hat{e}/(1 - \beta)$$

If we use piecewise linear interpolation, we can set $\hat{e} = \max_i \Delta_i$.

If we use a Hermite interpolation approach, we can set $\hat{e} = \max_i \Delta_i'$.

An even better estimate for $\hat{e}$ is given by Taylor's theorem. If $(TV)''(x)$ is bounded above by M, then

$$\| \hat{T}V - TV \| \leq \frac{1}{2} M (\max_i |x_{i+1} - x_i|)^2$$

If one can compute a finite M, this bound will generate an error term which is quadratic in the mesh size. With any of this information, one can derive formulas which will tell us when to stop value iteration.

Convergence of Linear Interpolation Method. For linear interpolation, we have another tight connection between the theoretical properites of T and its computational implementation. We will show that $\hat{T}$ is itself a contraction mapping on the space of piecewise linear functions.

Theorem 4. *If the interpolation scheme is a piecewise linear interpolation, then the approximation map, $\hat{T}$, is a contraction map.*

Proof. The proof is again straightforward. We know that T is a contraction map. Therefore, if V_1 and V_2 are piecewise linear functions with nodes at X, then

$$\begin{aligned}
\|\hat{T}V_1 - \hat{T}V_2\| &\equiv \max\, (\hat{T}V_1 - \hat{T}V_2)(x) \\
&= \max_{x \in X}\, (\hat{T}V_1 - \hat{T}V_2)(x) \\
&= \max_{x \in X}\, (TV_1 - TV_2)(x) \\
&\leq \max\, (TV_1 - TV_2)(x) \\
&= \|TV_1 - TV_2\| \\
&\leq \beta \|V_1 - V_2\|
\end{aligned}$$

where the critical second step follows from the fact that the extremal values of the difference of two piecewise linear functions with the same nodes are achieved at the nodes. ■

This fact also implies that we can use any of the standard acceleration methods with linear interpolation.

Monotone Operator Equations and Approximation. The key fact about dynamic programming which is exploited above is that the operator T a contraction map. It is also true that T is a monotone map, in the sense that $V_1 > V_2 \Rightarrow TV_1 > TV_2$. These are two distinct facts. We can use these facts to construct convergent schemes for solving (4) if we use approximation schemes with the appropriate properties.

The key theorem is that monotonicity in the data, that is, the approximate value function increases at all x as the interpolation data at X increases, implies that $\hat{T}$ is a monotone map.

Definition 5. *An interpolation scheme $V(x; v)$ is a monotone in the data iff $v_1 \geq v_2 \Rightarrow V(x; v_1) \geq V(x; v_2)$ for all x.*

Theorem 6. *If the interpolation scheme is monotonic in the data, then the approximation map, $\hat{T}$, is a monotone map.*

Proof. We know that T is a monotone map. Therefore,

$$\begin{aligned} V_1 \leq V_2 &\Rightarrow (TV_1)(x) \leq (TV_2)(x),\ x \in X \\ &\Rightarrow (\hat{T}V_1)(x) \leq (\hat{T}V_2)(x),\ x \in X \\ &\Rightarrow \hat{T}V_1 \leq \hat{T}V_2 \end{aligned}$$

where the last step follows from the monotonicity of the approximation scheme used to compute $\hat{T}V$ from the values of $\hat{T}V$ on $x \in X$. ■

Monotonicity is a powerful computational tool. First, convergence is guaranteed as long as there are upper and lower bounds on the solution. Second, the computed approximations inherit the monotonicity properties of the theoretical mathematical objects, providing a tight connection between the theory and the computational method.

Again, linear interpolation is the obvious example of an approximation scheme which is monotone in the data.

3.6. Convergence of Shape-Preserving Approximations. Consider monotone problems, that is, where the value function is monotone and T preserves monotonicity. We will next show that the shape-preserving approximaitions will converge to the true solution as we take a finer mesh. We do this by squeezing the shape-preserving approximations between two coarser approximations which we know will converge to the true value function

Define the upper and lower bound interpolation approximations for monotone T:

$$\begin{aligned} (T_U V)(x) &= (TV)(x_{i+1}),\ \text{for } x_i < x \leq x_{i+1} \\ (T_L V)(x) &= (TV)(x_i),\ \text{for } x_i \leq x < x_{i+1}. \end{aligned}$$

$T_U V$ produces an approximation to TV which is a step function and lies above TV; $T_L V$ is similarly a lower bound step function approximation.

Lemma 7. *T_U, T_L are contractions.*

$$\begin{aligned} \|T_U V_1 - T_U V_2\| &= \sup_x |(T_U V_1 - T_U V_2)(x)| \\ &= \sup_{x \in X} |(T_U V_1 - T_U V_2)(x)| \\ &= \sup_{x \in X} |(TV_1 - TV_2)(x)| \\ &\leq \sup_x |(TV_1 - TV_2)(x)| \\ &= \|TV_1 - TV_2\| \\ &\leq \beta \|V_1 - V_2\| \end{aligned}$$

The proof for T_L is similar.

We will next sandwich our shape-preserving approximations between T_U and T_L. Let SV be a shape preserving spline algorithm applied to the points $\{TV(x)\}_{x\in X}$. This can be one which uses only the the values of TV or one which also uses slope information.

Theorem 8. *If V is monotone increasing, T preserves monotonicity, and S is shape-preserving, then S^nV is monotone increasing.*

Proof. It suffices to show that SV is monotone. Since V is monotone increasing and T preserves monotonicity, $\{TV(x)\}_{x\in X}$ is a monotone increasing collection of points. Since S preserves shape, SV is monotone increasing. ■

Theorem 9. *If V is a concave function, T preserves concavity, and S is shape-preserving, then S^nV is concave.*

Proof. It suffices to show that SV is concave. Since V is monotone and T preserves concavity, $\{TV(x)\}_{x\in X}$ is a concave collection of points. Since S preserves shape, SV is concave. ■

Theorem 10. *If V is monotone increasing, T is monotone increasing and preserves monotonicity, and S is shape preserving then $T_L^nV \le S^nV \le T_U^nV$.*

Proof. by induction. ■

$n = 1$: Since SV is monotone increasing, $TV(x_i) \le SV(x) \le TV(x_{i+1})$ for $x_i \le x \le x_{i+1}$. Then by the definitions of T_L and T_U, $T_LV \le SV \le T_UV$.

$n = k$: Since T is monotone increasing, $T(T_L^{k-1}V)(x) \le T(S^{k-1}V)(x) \le T(T_U^{k-1}V)(x)$ for $x \in X$. Since S^kV is monotone increasing,

$$\begin{aligned} T(S^{k-1}V)(x_i) \le S^kV(x) \le T(S^{k-1}V)(x_{i+1}) &\text{ for } x \in [x_i, x_{i+1}] \\ \Rightarrow T_L^kV(x_i) \le S^kV(x) \le T_U^kV(x_{i+1}) &\text{ for } x \in [x_i, x_{i+1}] \\ \Rightarrow T_L^kV \le S^kV \le T_U^kV & \end{aligned}$$

Theorem 11. *Suppose that the value function, V, is a monotone function on X, has bounded second derivative on X, and that T is monotone increasing and preserves monotonicity. Then, for sufficiently fine mesh and sufficiently large n, S^nV is arbitrarily close to V.*

Proof. Follows from the fact that the fixed points of T_L and T_U converge to V as the mesh is refined, and from the inequality $T_L^kV \le S^kV \le T_U^kV$ proved in the last theorem. ■

Suppose $\hat{T}$ is our computational operator, and we iterate for a while until we reach some $V = \hat{T}^nV_0$ so that $\|\hat{T}V - V\| = \gamma$. Define

$$\epsilon = \|TV - \hat{T}V\| \le \max_i |TV(x_{i+1}) - TV(x_i)|.$$

Then

$$\begin{aligned} \|TV - V\| &\le \|TV - \hat{T}V\| + \|\hat{T}V - V\| \\ &= \epsilon + \gamma \end{aligned}$$

and

$$\begin{aligned}\|V - V^\infty\| &\leq \sum_{i=0}^{\infty} \|T^{(i+1)}V - T^{(i)}V\| \\ &\leq \|TV - V\| \sum_{i=0}^{\infty} \beta^i \\ &= \frac{\epsilon + \gamma}{1 - \beta}.\end{aligned}$$

This provides a bound on how close our current iterate V is from the true value function V^∞. Note that we do not require $\hat{T}$ to converge, only that successive iterates are within γ. This is not an ideal stopping criterion, since for a given mesh, the user may not be able to achieve a particular tolerance, even if the iterates converge.

The next thing to do is find conditions so that as we refine the mesh, we can make ϵ and γ as small as we wish.

If the iterates V^i are continuous and have a uniformly bounded first derivative (wherever it exists) then

$$\epsilon \leq \max_i |TV(x_{i+1}) - TV(x_i)| \leq M\delta$$

where δ is the size of the mesh and M is the bound on the derivative.

Since T_U and T_L are contractions, we know that they converge. If $V_U = \lim T_U^n V_0$ and $V_L = \lim T_L^n v_0$, then

$$\begin{aligned}\|V_U - V^\infty\| &\leq \frac{\epsilon}{1 - \beta} \\ \|V_L - V^\infty\| &\leq \frac{\epsilon}{1 - \beta}.\end{aligned}$$

This implies that, for large enough n, there is $V = S^n V_0$ so that

$$\|SV - V\| \leq \frac{2\epsilon}{1 - \beta}$$

where S is the shape preserving operator, since $T_L^n V \leq S^n V \leq T_U^n V$.

This proves that the shape-preserving approach will converge in an appropriate sense to the true value function. We now examine the relative efficiency of shape-preserving approximation.

4. Shape-Preserving Spline Approximations of Accumulation Examples

We next report on results concerning the application of shape-preserving splines to our simple example.

We demonstrate the power of shape-preserving approximation methods by applying them to the simple optimal growth problem

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

$$k_{t+1} = f(k_t) - c_t$$

where c_t is consumption in period t, $u(c)$ is the utility function at each date, k_t is the capital stock at the beginning of period t, and $f(k)$ is the aggregate production function in each period. We used the following specifications for preferences and technology:

$$u(c) = \frac{c^{1+\gamma}}{1+\gamma}$$

$$f(k) = k + Ak^{\alpha}$$

where $A = (1-\beta)/\alpha\beta$ is a normalization which produces a steady state capital stock at $k = 1$.

We solved this problem using three techniques: a discretization method, a piecewise linear approximation for the value function and a shape preserving quadratic approximation for the value function. We used the following parameter values: $\alpha = .25$, $\beta = .95, .99$, $\gamma = -10, -2, -.5$ over the capital interval [.4,1.6]. We ran the discretization method using mesh sizes $\Delta k = .01, .001, .0001, .00001$. We ran the two value function interpolation algorithms using mesh sizes $\Delta k = .3, .1, .03, .01$. We computed L_2 norm errors over the interior interval [.7, 1.3] using the value function and policy function from the discretization using the results from the cases with 120000 points as the "true" solutions.

Table 1 reports the true solutions for the consumption function. Table 3 reports the true value function solutions. Tables 2 and 4 report the absolute errors for various cases. Table 5 reports the relative errors in the consumption function. For the policy function, linear interpolation is roughly 3 times more accurate than the discrete method, and shape preserving interpolation is 70 times more accurate than the discrete method when all techniques use the same mesh size of .01. In fact, the accuracy of the policy function using shape preserving interpolation and a 120 point mesh is greater than using the discretization method and a mesh of 12000 points.

For the value function, linear interpolation is at least 3 times more accurate than the discrete method, and shape preserving interpolation is 1300 times more accurate than the discrete method when all techniques use the same mesh size of .01. Moreover, using shape preserving interpolation and a mesh of only 12 points provides better accuracy than discrete methods using 120 points.

Table 6 reports relative running times. The shape-preserving method is quite competitive with all other methods in terms of time. Table 7 reports convergence properties when we move the minimimum capital stock to $k = .01$. This case strains most methods because of the high curvature in the production function when k is small. Linear interpolation always converges, but accuracy is low. The cubic spline and polynomial methods had difficulty converging. The shape-preserving methods always converged and were efficient since they did not need to use an excessive number of nodes.

These error results indicate an important advantage to using shape preserving methods. Greater accuracy can be achieved while using fewer grid points.

5. CONCLUSIONS

In dynamic programming problems, shape properties can be exploited to compute efficient solutions.

Table 1: Consumption Function Solutions

k	(β,γ) :	(.95,-10.)	(.95,-2.)	(.95,-.5)	(.99,-10.)	(.99,-2.)	(.99,-.5)
0.4		0.16072542	0.14400542	0.11411542	0.03076215	0.02731215	0.02100215
0.5		0.17131082	0.15722082	0.13160082	0.03281561	0.02992561	0.02451561
0.6		0.18062668	0.16928668	0.14833668	0.03462007	0.03230007	0.02789007
0.7		0.18902657	0.18049657	0.16449657	0.03624722	0.03450722	0.03113722
0.8		0.19672350	0.19102350	0.18019350	0.03773178	0.03657178	0.03429178
0.9		0.20385342	0.20100342	0.19552342	0.03911369	0.03853369	0.03738369
1.0		0.21052632	0.21052632	0.21052632	0.04040404	0.04040404	0.04040404
1.1		0.21681288	0.21967288	0.22526288	0.04161833	0.04219833	0.04336833
1.2		0.22277424	0.22848424	0.23976424	0.04277829	0.04392829	0.04629829
1.3		0.22844789	0.23700789	0.25405789	0.04387303	0.04560303	0.04917303
1.4		0.23388154	0.24527154	0.26816154	0.04491979	0.04721979	0.05201979
1.5		0.23909567	0.25331567	0.28210567	0.04593442	0.04880442	0.05483442
1.6		0.24411529	0.26115529	0.29589529	0.04690172	0.05034172	0.05761172

Table 2: Errors

Discrete model: L_2 errors over the interval $k \in [.7, 1.3]$						
N	(.95,-10.)	(.95,-2.)	(.95,-.5)	(.99,-10.)	(.99,-2.)	(.99,-.5)
120	1.0e-02	1.5e-02	1.5e-02	1.0e-02	3.6e-02	1.3e-01
1200	1.4e-03	1.4e-03	1.4e-03	8.6e-03	8.0e-03	8.2e-03
12000	1.4e-04	1.4e-04	1.4e-04	7.2e-04	7.4e-04	7.5e-04
Linear Interpolation: L_2 errors over $k \in [.7, 1.3]$						
4	5.5e-02	9.3e-02	1.7e-01	5.7e-02	1.0e-01	1.9e-01
12	2.3e-02	3.8e-02	6.2e-02	2.5e-02	4.2e-02	7.6e-02
40	7.4e-03	1.0e-02	1.6e-02	8.1e-03	1.3e-02	2.2e-02
120	2.3e-03	2.9e-03	4.7e-03	2.7e-03	3.9e-03	6.1e-03
Shape Preserving Quadratic Interpolation: L_2 errors over $k \in [.7, 1.3]$:						
4	8.5e-03	7.5e-03	1.0e-02	9.3e-03	8.5e-03	1.3e-02
12	1.5e-03	9.8e-04	9.1e-04	1.6e-03	1.3e-03	1.8e-03
40	1.4e-04	7.2e-05	8.1e-05	2.5e-04	2.3e-04	2.6e-04
120	3.2e-05	3.1e-05	3.4e-05	1.5e-04	1.5e-04	1.6e-04

Table 3: Value function solutions

	(β,γ) :					
k	(.95,-10.)	(.95,-2.)	(.95,-.5)	(.99,-10.)	(.99,-2.)	(.99,-.5)
0.4	0.16962423	0.17279166	0.17523392	0.03257365	0.03319877	0.03367471
0.5	0.17839679	0.18032422	0.18184295	0.03424907	0.03462909	0.03492537
0.6	0.18608025	0.18718254	0.18806682	0.03571859	0.03593576	0.03610839
0.7	0.19297622	0.19353818	0.19399595	0.03703891	0.03714956	0.03723898
0.8	0.19927200	0.19950083	0.19968973	0.03824534	0.03829037	0.03832728
0.9	0.20509268	0.20514553	0.20518969	0.03936149	0.03937189	0.03938052
1.0	0.21052632	0.21052632	0.21052632	0.04040404	0.04040404	0.04040404
1.1	0.21563752	0.21568354	0.21572277	0.04138521	0.04139425	0.04140193
1.2	0.22047521	0.22064829	0.22079716	0.04231427	0.04234829	0.04237741
1.3	0.22507743	0.22544512	0.22576397	0.04319845	0.04327070	0.04333308
1.4	0.22947441	0.23009381	0.23063503	0.04404348	0.04416517	0.04427107
1.5	0.23369059	0.23461055	0.23542011	0.04485401	0.04503471	0.04519314
1.6	0.23774607	0.23900881	0.24012743	0.04563387	0.04588185	0.04610079

Table 4: Value function errors

	Errors in value function					
N	(.95,-10.)	(.95,-2.)	(.95,-.5)	(.99,-10.)	(.99,-2.)	(.99,-.5)
Discrete model:						
120	2.1e-03	4.1e-04	6.6e-05	2.2e-03	5.2e-03	2.1e-03
1200	2.6e-05	3.7e-06	7.7e-07	7.6e-04	1.0e-04	1.6e-05
12000	2.4e-07	3.2e-08	8.7e-09	6.6e-06	9.5e-07	1.8e-07
Linear Interpolation:						
4	1.3e-02	6.3e-03	3.7e-03	1.3e-02	6.4e-03	3.9e-03
12	3.2e-03	2.0e-03	1.1e-03	3.2e-03	2.1e-03	1.2e-03
40	9.0e-04	4.6e-04	1.5e-04	9.7e-04	6.1e-04	3.4e-04
120	2.3e-04	5.5e-05	1.8e-05	3.1e-04	1.8e-04	8.0e-05
Shape Preserving Quadratic Interpolation:						
4	1.0e-03	3.2e-04	1.2e-04	1.1e-03	3.5e-04	1.5e-04
12	1.2e-04	2.8e-05	7.7e-06	1.3e-04	3.6e-05	1.5e-05
40	8.8e-06	9.7e-07	1.5e-07	1.1e-05	2.9e-06	9.8e-07
120	3.8e-07	2.7e-08	5.1e-09	1.6e-06	2.1e-07	3.7e-08

Table 5: L_2 norm of relative errors in consumption over [0.7,1.3]

N	(β,γ) :					
	(.95,-10.)	(.95,-2.)	(.95,-.5)	(.99,-10.)	(.99,-2.)	(.99,-.5)
Discrete model						
12	7.6e-02	2.8e-03	5.3e-03	7.9e-01	1.8e-01	1.1e-02
120	1.2e-03	2.2e-04	5.3e-04	4.5e-02	6.6e-02	1.3e-03
1200	1.0e-04	2.1e-05	5.4e-05	2.9e-03	5.4e-03	1.3e-04
Linear Interpolation						
4	7.9e-03	4.1e-03	2.4e-03	8.0e-03	4.1e-03	2.4e-03
12	1.5e-03	9.8e-04	5.6e-04	1.5e-03	1.0e-03	6.3e-04
40	3.9e-04	2.3e-04	9.8e-05	4.2e-04	2.8e-04	1.6e-04
120	1.1e-04	3.7e-05	1.3e-05	1.4e-04	8.4e-05	4.2e-05
Cubic Spline						
4	6.6e-03	5.0e-04	1.3e-04	7.1e-03	5.7e-04	1.8e-04
12	8.7e-05	1.5e-06	1.8e-07	1.3e-04	4.9e-06	1.1e-06
40	7.2e-08	1.8e-08	5.5e-09	7.6e-07	8.8e-09	4.9e-09
120	5.3e-09	5.6e-10	1.3e-10	4.2e-07	4.1e-09	1.5e-09
Polynomial without slopes						
4	DNC	5.4e-04	1.6e-04	1.4e-02	5.6e-04	1.7e-04
12	3.0e-07	2.0e-09	4.3e-10	5.8e-07	4.5e-09	1.5e-09
Shape Preserving Quadratic Hermite Interpolation						
4	4.7e-04	1.5e-04	6.0e-05	5.0e-04	1.7e-04	7.3e-05
12	3.8e-05	1.1e-05	3.7e-06	5.9e-05	1.7e-05	6.3e-06
40	3.2e-06	5.7e-07	9.3e-08	1.4e-05	2.6e-06	5.1e-07
120	2.2e-07	1.7e-08	3.1e-09	4.0e-06	4.6e-07	5.9e-08
Shape Preserving Quadratic Interpolation (ignoring slopes)						
4	1.1e-02	3.8e-03	1.2e-03	2.2e-02	7.3e-03	2.2e-03
12	6.7e-04	1.1e-04	3.1e-05	1.2e-03	2.1e-04	5.7e-05
40	3.5e-05	5.8e-06	8.3e-07	4.5e-05	9.3e-06	3.0e-06
120	2.5e-06	1.5e-07	2.2e-08	4.3e-06	8.5e-07	1.9e-07

Table 6: Relative times of the algorithms

No. states:	(.95,-10.)	(.95,-2.)	(.95,-.5)	(.99,-10.)	(.99,-2.)	(.99,-.5)
Shape Preserving Quadratic Interpolation						
4	1.7	1.1	1.1	1.7	1.1	1.3
12	4.9	2.7	3.8	4.9	2.9	3.5
40	14.5	8.5	11.4	16.1	10.2	11.0
120	46.4	26.4	30.9	58.9	63.4	33.6
Linear Interpolation						
4	2.2	1.1	1.2	2.3	1.0	1.6
12	5.8	2.5	2.9	6.0	2.9	3.5
40	17.3	8.3	10.3	16.7	9.7	10.6
120	44.8	27.5	30.9	49.5	31.0	41.2
Polynomial without slopes						
4	DNC	2.2	2.2	3.3	2.8	2.6
12	17.9	15.6	13.4	20.1	18.3	16.0
Cubic Spline						
4	1.7	1.1	1.2	2.0	1.2	1.6
12	5.3	3.3	3.8	5.8	4.3	4.8
40	19.4	13.0	13.3	24.0	19.8	21.2
120	83.6	58.3	57.0	105.3	88.7	76.2

Table 7: Stability Comparison Convergence of Algorithm when Left-most point is 0.01

Method:	Success rate:
Linear Interpolation	24/24
Cubic Spline	8/24
Polynomial	8/12
Shape Preserving Quadratic Interpolation	24/24
Shape Preserving Quadratic Interpolation, ignoring slopes	24/24

References

[1] Bellman, Richard, Kalaba, Robert, and Kotkin, Bella. "Polynomial Approximation — A New Computational Technique in Dynamic Programming: Allocation Processes." *Mathematics of Computation* 17 (1963): 155–161.

[2] Bertsekas, D. *Dynamic Programming and Stochastic Control*, Academic Press, 1976.

[3] 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.

[4] Cheney, E.W. *Introduction to Approximation Theory*, New York, NY: McGraw-Hill, 1966.

[5] Costantini, P. "On Monotone and Convex Spline Interpolation," *Math Comp. 46 (1986) 203–214.*

[6] *Davis, Philip J. Interpolation and Approximation*, New York, NY: Dover Press, 1975.

[7] de Boor, C. and B. Swartz, "Piecewise Monotone Interpolation," *J. Approx. Theory* 21 (1977) 411–416.

[8] de Boor, C. *A Practical Guide to Splines*, New York, NY: Springer, 1978.

[9] Daniel, J.W. "Splines and Efficiency in Dynamic Programming." *Journal of Mathematical Analysis and Application*s 54 (1976): 402–407.

[10] Dantzig, G.B., Harvey, R.P., Lansdowne, Z.F., and McKnight, R.D. "DYGAM – A Computer System for the Solutions of Dynamic Programs." Control Analysis Corporation, Palo Alto, CA, August, 1974.

[11] Johnson, S., J. R. Stedinger, C. A. Shoemaker, Y. Li, and J. A. Tehada–Guibert. 1993. Numerical Solution of Continuous–State Dynamic Programs using Linear and Spline Interpolation. *Operations Research* 41:484–500.

[12] Phelan, C. J. , and R. M. Townsend. 1991. Formulating and Computing Solutions to the Infinite Period Principal-Agent Problem. *Review of Economic Studies* 58: 853–81.

[13] Putterman, Martin L., and Brumelle, Shelby L. "On the Convergence of Policy Iteration in Stationary Dynamic Programming." *Mathematics of Operations Researc*h 4 (No. 1, February, 1979): 60–69.

[14] Putterman, Martin L., and Shin, Moon Chirl. "Modified Policy Iteration Algorithms for Discounted Markov Decision Problems." *Management Science* 24 (July, 1978): 1127–1137.

[15] Schumaker, Larry L. "On Shape-Preserving Quadratic Spline Interpolation," *SIAM Journal of Numerical Analysis* 20 (No. 4, August 1983): 854–864.