CONTINUOUS STATE DYNAMIC PROGRAMMING VIA NONEXPANSIVE APPROXIMATION

JOHN STACHURSKI

ABSTRACT. This paper studies fitted value iteration for continuous state numerical dynamic programming using nonexpansive function approximators. A number of approximation schemes are discussed. The main contribution is to provide error bounds for approximate optimal policies generated by the value iteration algorithm. *Journal of Economic Literature Classifications: C61, C63*

1. INTRODUCTION

In dynamic programming, Bellman's principle of optimality permits computation of optimal policies from the relevant value function, denoted below by v^*. When no analytical representation of v^* is available an approximation can be obtained numerically through value iteration, which involves iterating the Bellman operator T on some initial function v.[1] Under mild assumptions T is supremum-norm contracting, and the resulting sequence $(T^n v)_{n=1}^{\infty}$ converges geometrically to v^*.

If the state space is infinite, one cannot in general implement the functions $Tv, T^2v, \ldots, T^nv$ on a computer. One feasible alternative is discretization, where the state space is replaced with a finite grid, and the original model with a "similar" model which evolves on

Date: August 3, 2007.

Key words and phrases. Numerical dynamic programming, nonexpansive approximation.

This paper has benefitted from the comments of an anonymous referee, financial support from Australian Research Council Grant DP0557625 and the Murata Science Foundation, and from helpful discussions with Takashi Kamihigashi, Kazuo Nishimura, Kevin Reffett and Rabee Tourkey.

[1]For background see the excellent survey of Rust (1996).

this grid. A second is fitted value iteration, a standard algorithm for which is

1 initialize v;
2 **repeat**
3 | evaluate Tv at finite set of grid points $\{x_i\}$;
4 | use the values to construct an approximation $w \in \mathscr{F}$ of Tv;
5 | set $v = w$;
6 **until** *a suitable stopping rule is satisfied* ;
7 compute an approximate optimal policy using v in place of v^*;

Here $\mathscr{F}$ is a class of functions with finite parametric representation. The map $v \mapsto w$ is in effect an approximate Bellman operator $\hat{T}$, and fitted value iteration is equivalent to iteration with $\hat{T}$ in place of T.[2]

Popular architectures for constructing the approximation w of Tv in line 4 include Chebychev polynomials, cubic splines and neural nets. These architectures are popular because they are often able to accurately represent Tv with relatively few grid points. However, it should be recalled that the ultimate objective is not to minimize the distance between w and Tv. Rather it is to minimize some measure of distance between the optimal policy and the approximate optimal policy computed from $\hat{T}^n v$. In particular, attention must be paid to whether or not the approximation scheme interacts well with the iteration scheme needed to compute the fixed point v^*. A scheme

[2]Aside from discretization and fitted value iteration, numerous alternative techniques have been proposed for numerical dynamic programming. They include a variety of perturbation and projection methods which act directly on the Euler equation. A comparison of alternative methods for the stochastic optimal growth model is available in Aruoba, Fernandez-Villaverde and Rubio-Ramirez (2006). Our study treats a general dynamic programming problem where Euler equations do not necessarily exist.

which represents the function Tv well at each iteration may in fact still lead to poor dynamic properties for the sequence $(\hat{T}^n v)$.[3]

At issue is the lack of compatibility between the sup-norm contraction property of T—which drives convergence of $(T^n v)_{n=1}^{\infty}$ to v^*—and the potentially expansive properties of the approximation operator. To clarify this point, let us decompose $\hat{T}$ into the action of two operators L and T: First T is applied to v—in practice Tv is evaluated only at finitely many points—and then an approximation operator L sends the result into $w = \hat{T}v \in \mathscr{F}$. Thus, $\hat{T} = L \circ T$. As T is a contraction, $\hat{T} = L \circ T$ is contracting whenever L is, but L is not generally contracting.

In the present paper we pursue a suggestion of Gordon (1995), restricting attention to approximation architectures such that L is nonexpansive with respect to the sup-norm; from which it follows that the composition $\hat{T} := L \circ T$ is a contraction mapping. We exploit the contractiveness of $\hat{T}$ to obtain a general set of error bounds for approximate optimal policies which apply to any nonexpansive approximation architecture. Our focus is on structures suitable for economic applications.[4]

We pay special attention to the case where L sends functions into piecewise constant functions—a kind of nonexpansive approximator. Iteration with $L \circ T$ provides an algorithm that can be thought of as combining aspects of discretization and fitted value iteration. Compared to the common practice of one-off discretization onto a finite grid, the algorithm is simple to program, preserves and exploits

[3] As approximation errors are compounded at each iteration, $\lim_{n\to\infty} \hat{T}^n v$ may deviate substantially from $\lim_{n\to\infty} T^n v = v^*$; in fact the sequence may fail to converge at all. See, for example, Tsitsiklis and Van Roy (1996, Section 4), which gives an example of divergence under least-squares approximation.

[4] Following Gordon (1995), Drummond (1996) investigated adding penalties to the derivatives of function approximators in order to prevent sup-norm expansiveness (overshooting). Guestrin et al. (2001) study nonexpansive approximations in factored Markov Decision Processes. See also Tsitsiklis and Van Roy (1996), who provided error bounds for optimal policies when the state and action spaces are finite.

information about the model primitives between grid points, admits the use of adaptive grids, and is always nonexpansive.

A third contribution of this paper is to investigate the expansiveness of shape-preserving function approximators. Previously, Judd and Solnick (1994) highlighted the computational advantages of such approximators, where the "shapes" of greatest interest are monotonicity and convexity (or concavity). We show that a certain class of shape-preserving quasi-interpolants popular in computer aided design are also nonexpansive.

Within the economic literature, a number of studies have been made of approximation architectures which turn out to be nonexpansive. Judd and Solnick (1994) noted that a class of spline interpolants preserve the contraction property of $\hat{T}$, and exploited this fact in their discussion of errors. Santos and Vigo-Aguiar (1998) considered a finite element method using piecewise affine functions. They also observed that their approximation scheme preserves the contraction property of $\hat{T}$, and these ideas were subsequently extended by Grüen and Semmler (2004). Rust (1997) studies a random discretized Bellman operator which is a probability one contraction. The present paper unifies and generalize some of these ideas.

Section 2 formulates the dynamic programming problem. Section 3 discusses nonexpansive approximation schemes. Section 4 states results, Section 5 looks at applications, and Section 6 gives proofs.

2. Formulation of the Problem

If (U, d) is a metric space, then $\mathscr{B}(U)$ denotes the Borel subsets of U and $b\mathscr{B}(U)$ is the bounded Borel measurable functions from U to $\mathbb{R}$. For $f \in b\mathscr{B}(U)$, let $\|f\| := \sup_{x \in U} |f(x)|$ denote the supremum norm. A map $M \colon b\mathscr{B}(U) \to b\mathscr{B}(U)$ is called *nonexpansive* if

$$\|Mw - Mw'\| \leq \|w - w'\| \qquad \forall\, w, w' \in b\mathscr{B}(U) \tag{1}$$

and a *contraction of modulus* ρ if there exists a $\rho \in [0, 1)$ with

$$\|Mw - Mw'\| \leq \rho \|w - w'\| \qquad \forall\, w, w' \in b\mathscr{B}(U) \tag{2}$$

Let M_1 and M_2 map $b\mathscr{B}(U)$ to itself. Note that if M_1 is a contraction of modulus ρ and M_2 is nonexpansive, then $M_2 \circ M_1$ is a contraction of modulus ρ. (In much of what follows we write the composition $M_2 \circ M_1$ more simply as M_2M_1, and similarly for other compositions.)

Consider an infinite horizon stochastic dynamic programming problem with state space S, action space A (both Borel subsets of finite-dimensional Euclidean space) and a correspondence Γ mapping S into $\mathscr{B}(A)$, with $\Gamma(x)$ interpreted as the set of feasible actions at state x. Given S, A and Γ, define

$$\operatorname{gr}\Gamma := \{(x,u) \in S \times A \,:\, u \in \Gamma(x)\}$$

This collection of points (the graph of Γ) is called the set of all feasible state/action pairs. A "reward function" r sends $\operatorname{gr}\Gamma$ into $\mathbb{R}$, $\rho \in (0,1)$ is a discount factor, and $\mathbf{M}(x,u;dy)$ is a distribution over S for each feasible state/action pair $(x,u) \in \operatorname{gr}\Gamma$. Here $\mathbf{M}(x,u;dy)$ should be be interpreted as the conditional distribution of X_{t+1} when the current state $X_t = x$ and the current action $U_t = u$.[5] For example, suppose the future state is determined according to

$$X_{t+1} = F(X_t, U_t, W_{t+1}) \qquad W_t \overset{\text{IID}}{\sim} G(dz) \tag{3}$$

Then $\mathbf{M}(x,u;dy)$ is the distribution of $F(x,u,W)$ when $W \sim G$, and, for arbitrary integrable $w\colon S \to \mathbb{R}$,

$$\int w(y)\mathbf{M}(x,u;dy) = \int w[F(x,u,z)]G(dz) \tag{4}$$

The system evolves as follows. At the start of time, the agent observes $X_0 = x_0 \in S$, where x_0 is some fixed initial condition, and then chooses action $U_0 \in \Gamma(X_0) \subset A$. After choosing U_0, the agent receives reward $r(X_0, U_0)$. The next state X_1 is now drawn according to distribution $\mathbf{M}(X_0, U_0; dy)$ and the process repeats.

Let Π denote the set of all measurable functions $\pi\colon S \to A$ with $\pi(x) \in \Gamma(x)$ for all $x \in S$. We refer to Π as the set of *feasible policies*.

[5] By a distribution on S is meant a probability measure on $(S, \mathscr{B}(S))$. In addition, $(x,u) \mapsto \mathbf{M}(x,u;B)$ is required to be measurable, $\forall B \in \mathscr{B}(S)$.

Each $\pi \in \Pi$ and initial condition $x_0 \in S$ defines a Markov process $(X_t)_{t \geq 0}$, where X_0 is set equal to x_0, and then $X_{t+1} \sim \mathbf{M}(X_t, \pi(X_t); dy)$.

Let $\mathbf{P}_\pi^{x_0}$ be the joint distribution on sequence space $(S^\infty, \otimes_{n=1}^\infty \mathscr{B}(S))$ induced by $(X_t)_{t \geq 0}$, and let $\mathbf{E}_\pi^{x_0}$ be the expectation operator corresponding to $\mathbf{P}_\pi^{x_0}$. Define a map $\Pi \times S \ni (\pi, x_0) \mapsto v_\pi(x_0) \in \mathbb{R}$ by

$$v_\pi(x_0) := \mathbf{E}_\pi^{x_0} \left[\sum_{t=0}^{\infty} \rho^t r(X_t, \pi(X_t)) \right] \tag{5}$$

Thus $v_\pi(x_0)$ is the value of following the policy π when starting at initial condition x_0. The *value function* $v^* \colon S \to \mathbb{R}$ is

$$v^*(x_0) := \sup_{\pi \in \Pi} v_\pi(x_0) \qquad (x_0 \in S) \tag{6}$$

Existence of v^* as a well defined function follows from the existence of suprema for bounded subsets of $\mathbb{R}$ (see Assumption 2.1 below). A policy $\pi^* \in \Pi$ is called *optimal* if it attains the supremum in (6) for every $x_0 \in S$ (equivalently, $v_{\pi^*} = v^*$ on S).

Assumption 2.1. The map r is continuous and bounded on $\operatorname{gr} \Gamma$, while Γ is continuous, nonempty and compact valued. Further,

$$(x, u) \mapsto \int w(y) \mathbf{M}(x, u; dy) \tag{7}$$

is continuous as a map from $\operatorname{gr} \Gamma$ to $\mathbb{R}$ whenever w is bounded and continuous on S.

The continuity assumption in (7) is a version of the so-called Feller property.[6] In the case of the transition rule in (3) and (4), continuity of (7) holds whenever $(x, u) \mapsto F(x, u, z)$ is continuous for all z. Under Assumption 2.1 the following optimality result obtains.[7]

Theorem 2.1. *The value function v^* is continuous, and is the unique function in $b\mathscr{B}(S)$ which satisfies*

$$v^*(x) = \sup_{u \in \Gamma(x)} \left\{ r(x, u) + \rho \int v^*(y) \mathbf{M}(x, u; dy) \right\} \qquad (x \in S) \tag{8}$$

[6] See, for example, Stokey, Lucas and Prescott (1989, Chapter 8).

[7] See, for example, Hernández-Lerma and Lasserre (1999, Section 8.5).

If π^* is an element of Π and

$$v^*(x) = r(x, \pi^*(x)) + \rho \int v^*(y)\mathbf{M}(x, \pi^*(x); dy) \qquad (x \in S) \tag{9}$$

then π^* is optimal. At least one such optimal policy $\pi^* \in \Pi$ exists. Conversely, if π^* is an optimal policy then it satisfies (9)*.*

Two kinds of contraction mappings are used to study the optimality results. First, let $T_\pi \colon b\mathscr{B}(S) \to b\mathscr{B}(S)$ be defined for all $\pi \in \Pi$ by

$$T_\pi w(x) = r(x, \pi(x)) + \rho \int w(y)\mathbf{M}(x, \pi(x); dy) \qquad (x \in S)$$

Further, let $T \colon b\mathscr{B}(S) \to b\mathscr{B}(S)$ be defined by

$$Tw(x) = \sup_{u \in \Gamma(x)} \left\{ r(x, u) + \rho \int w(y)\mathbf{M}(x, u; dy) \right\} \qquad (x \in S)$$

The second operator T is usually called the *Bellman operator*. In view of Theorem 2.1, v^* is the unique fixed point of T in $b\mathscr{B}(S)$.

It is well-known that for every $\pi \in \Pi$, the operator T_π is a contraction on $(b\mathscr{B}(S), \|\cdot\|)$ of modulus ρ. The unique fixed point of T_π in $b\mathscr{B}(S)$ is v_π, where the definition of v_π is given in (5). In addition, T_π is monotone on $b\mathscr{B}(S)$, in the sense that if $w, w' \in b\mathscr{B}(S)$ and $w \leq w'$, then $T_\pi w \leq T_\pi w'$.[8] Similarly, the Bellman operator is also a contraction of modulus ρ; and monotone on $b\mathscr{B}(S)$.[9]

3. The Approximation Operator

To carry out fitted value iteration we use an approximation operator L which maps $b\mathscr{B}(S)$ into a collection of functions $\mathscr{F} \subset b\mathscr{B}(S)$. In general, L constructs an approximation $Lv \in \mathscr{F}$ to $v \in b\mathscr{B}(S)$ according to a sample $\{v(x_i)\}_{i=1}^k$ of evaluations of v on grid points $\{x_i\}_{i=1}^k \subset S$. As discussed in the introduction, we focus on architectures with the property that L is nonexpansive with respect to the

[8] Inequalities such as $w \leq w'$ are pointwise inequalities on S.

[9] These results are standard. See, for example, Puterman (1994), Stokey, Lucas and Prescott (1989) or Hernández-Lerma and Lasserre (1999).

sup norm:

$$\|Lv - Lw\| \leq \|v - w\| \qquad \forall\, v, w \in b\mathscr{B}(S) \tag{10}$$

In this section we discuss examples of approximation operators with the nonexpansive property. The discussion is largely expository, although the observation that Schoenberg's variation diminishing operator is nonexpansive appears to be new.

Example 3.1. (**Piecewise constant approximation**) An elementary approximation architecture is provided by piecewise constant approximation. Suppose that $\{x_i\}_{i=1}^k$ is a sequence of grid points in S, and that $\{J_i\}_{i=1}^k$ is a partition of S with $x_i \in J_i$ for each i, $J_m \cap J_n = \emptyset$ when $m \neq n$, and $S = \cup_{i=1}^k J_i$. For any function $v\colon S \to \mathbb{R}$, set

$$Lv(x) = v(x_i) \qquad \forall\, x \in J_i$$

Thus Lv takes only finitely many values. Moreover, L is nonexpansive. To see this, pick any $w, v \in b\mathscr{B}(S)$ and any $x \in S$. Without loss of generality, suppose that $x \in J_m$. Then

$$|Lw(x) - Lv(x)| = |w(x_m) - v(x_m)| \leq \sup_{1 \leq i \leq k} |w(x_i) - v(x_i)|$$

$$\therefore \quad \|Lw - Lv\| \leq \sup_{1 \leq i \leq k} |w(x_i) - v(x_i)| \leq \|w - v\| \tag{11}$$

Iteration with $\hat{T} = LT$ provides an implementation of discretization for dynamic programs that has several theoretical and practical advantages over traditional discretization. The ideas are discussed in detail in Section 5, and applied to Samuelson's (1971) commodity pricing model.

Example 3.2. (**Kernel averagers**) Kernel-based approximators have attracted much attention in recent years, partly because they possess good properties in high-dimensional state spaces. One of these methods is the kernel averager, which can be represented by an expression of the form

$$Lv(x) = \frac{\sum_{i=1}^k K_h(x_i - x) v(x_i)}{\sum_{i=1}^k K_h(x_i - x)} \tag{12}$$

Here the kernel K_h is a nonnegative mapping from $S \to \mathbb{R}$ such as the radial basis function $e^{-\|\cdot\|/h}$. The value of the kernel decays to zero as x diverges from x_i. Thus, $Lv(x)$ is a convex combination of the observations $v(x_1), \ldots, v(x_k)$ with larger weight being given to those observations $v(x_i)$ for which x_i is close to x.[10]

The operator L in (12) is easily seen to be nonexpansive on $b\mathscr{B}(S)$: Pick any $x \in S$, and let $\lambda(x,i) := K_h(x_i - x) / \sum_{j=1}^k K_h(x_j - x)$. Using $\sum_{i=1}^k \lambda(x,i) = 1$, we have

$$\begin{aligned} |Lw(x) - Lv(x)| &= \left| \sum_{i=1}^k \lambda(x,i)(w(x_i) - v(x_i)) \right| \\ &\leq \sum_{i=1}^k \lambda(x,i)|w(x_i) - v(x_i)| \leq \sup_{1 \leq i \leq k} |w(x_i) - v(x_i)| \end{aligned}$$

Since x is arbitrary the claim in the lemma holds.

Example 3.3. (**Continuous piecewise linear interpolation**) A common form of approximation in dynamic programming is piecewise linear (piecewise affine) spline interpolation.[11] To describe a general set up, let $\{x_i\}_{i=1}^k$ be a finite subset of $S \subset \mathbb{R}^d$ with the property that the convex hull of $\{x_i\}_{i=1}^k$ equals S, and let $\mathscr{T}$ be a triangularization of S relative to the nodes $\{x_i\}_{i=1}^k$. In other words, $\mathscr{T}$ is a partition of S into a finite collection of non-overlapping, non-degenerate simplexes, where, for each $\Delta \in \mathscr{T}$, the set of vertices $\{\zeta_i\}_{i=1}^{d+1} \subset \{x_i\}_{i=1}^k$.[12]

Each $x \in \Delta$ can be represented uniquely by its barycentric coordinates relative to Δ:

$$x = \sum_{i=1}^{d+1} \lambda(x,i)\zeta_i, \quad \text{where} \quad \lambda(x,i) \geq 0 \text{ and } \sum_{i=1}^{d+1} \lambda(x,i) = 1$$

[10] The smoothing parameter h controls the weight assigned to more distant observations.

[11] See, for example, Santos and Vigo-Aguiar (1998) and Munos and Moore (1999).

[12] A simplex is called non-degenerate if it has positive measure in $\mathbb{R}^d$.

For $v \in b\mathscr{B}(S)$ we define the interpolation operator L by

$$Lv(x) = \sum_{i=1}^{d+1} \lambda(x,i) v(\zeta_i)$$

Direct calculations show that if $v, w \in b\mathscr{B}(S)$, then at x we have

$$|Lw(x) - Lv(x)| \leq \sup_{1 \leq i \leq d+1} |w(\zeta_i) - v(\zeta_i)| \leq \|w - v\|$$

Since x is arbitrary, L is clearly nonexpansive.

Example 3.4. (**Schoenberg's variation diminishing operator**) In a well-known study, Judd and Solnick (1994) emphasize the advantages of fitted value iteration with shape-preserving approximators; the shapes of greatest interest are monotonicity and convexity, and approximators which preserve them not only incorporate known structure from the target function in approximation, they also allow monotonicity and convexity to be exploited in the optimization step of the value iteration algorithm.[13]

Judd and Solnick discuss several univariate shape-preserving architectures, including (nonsmooth) univariate piecewise linear interpolants and (smooth) Schumaker splines. Here we describe a further class of smooth, shape-preserving univariate approximators known as Schoenberg variation diminishing splines.

To construct the operator we set $S = [a, b] \subset \mathbb{R}$, and in place of a standard grid we use for each $d \in \mathbb{N}$ a $d+1$-regular knot sequence $(t_i)_{i=1}^{k+d+1}$, which satisfies

$$a = t_1 = \cdots = t_{d+1} < t_{d+2} < \cdots < t_{k+1} = \cdots = t_{k+d+1} = b$$

Here d is the order of the spline, so that, for example, $d = 3$ corresponds to a cubic spline. The Schoenberg splines are built using k

[13] Monotonicity is exploited as follows: In monotone programs the optimal action is often increasing in the state, in which case one need not search for optimal actions in that subset of the action space which is dominated by the optimal action at a lower state. The importance of convexity in optimization needs no illustration here.

basis functions which are known as B-splines. The latter are defined recursively by

$$B_{i,0} := \mathbb{1}_{[t_i,t_{i+1})}, \quad i = 1,\dots,k$$

and then, $i = 1,\dots,k$,

$$B_{i,d}(x) := \frac{x - t_i}{t_{i+d} - t_i} B_{i,d-1}(x) + \frac{t_{i+d+1} - x}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(x)$$

where in the definition we are using the convention that $0/0 = 0$. For fixed d the basis functions $B_{1,d},\dots,B_{k,d}$ are linearly independent and satisfy $\sum_{i=1}^{k} B_{i,d} = \mathbb{1}_S$. Their span is often denoted by $\mathbb{S}_d$:

$$\mathbb{S}_d := \left\{ \sum_{i=1}^{k} \alpha_i B_{i,d} \,:\, (\alpha_1,\dots,\alpha_k) \in \mathbb{R}^k \right\}$$

Clearly $\mathbb{S}_d \subset b\mathscr{B}(S)$. Setting $t_i^* := (t_{i+1} + \cdots + t_{i+d})/d$, Schoenberg's variation diminishing operator is given by

$$L\colon b\mathscr{B}(S) \ni v \mapsto \sum_{i=1}^{k} v(t_i^*) B_{i,d} \in \mathbb{S}_d$$

It is well-known that L preserves monotonicity and convexity (concavity) in v.[14] It is easy to see that L is also nonexpansive on $b\mathscr{B}(S)$. To check this, pick any $v, w \in b\mathscr{B}(S)$, and let $x \in S$.

$$\begin{aligned} |Lw(x) - Lv(x)| &\le \sum_{i=1}^{k} B_{i,d}(x) |w(t_i^*) - v(t_i^*)| \\ &\le \sum_{i=1}^{k} B_{i,d}(x) \sup_{1\le j\le k} |w(t_j^*) - v(t_j^*)| \end{aligned}$$

Using $\sum_{i=1}^{k} B_{i,d}(x) = 1$, we have

$$|Lw(x) - Lv(x)| \le \sup_{1\le j\le k} |w(t_j^*) - v(t_j^*)| \le \|v - w\|$$

Since x is arbitrary the proof is done.

[14] See, for example, Lyche and Mørken (2002, Chapter 5).

4. RESULTS

In this section we develop error bounds for fitted value iteration with nonexpansive approximation. The error associated with policy π is

$$E(\pi) := \sup_{x \in S} \{v_{\pi^*}(x) - v_\pi(x)\} = \sup_{x \in S} \{v^*(x) - v_\pi(x)\} \tag{13}$$

where π^* is the optimal policy. The value of $E(\pi)$ gives the cost of using π rather than the optimal policy π^* in terms of total reward. Policies with small E-error are consistent with the incentives of agents in the model.[15]

Algorithm 1: FVI algorithm

```
initialize v_0 ∈ 𝓕 and set n = 0 ;
repeat
    set n = n + 1 ;
    evaluate T v_{n-1} at a finite set of grid points {x_i}_{i=1}^k;
    compute the approximation L T v_{n-1} from these values;
    set v_n = L T v_{n-1} ;
    set e_n = ‖v_n − v_{n-1}‖ ;
until e_n falls below some tolerance ;
solve for a v_n-greedy policy π ;
return π and e_n
```

Consider Algorithm 1. In the algorithm, L is any nonexpansive approximation operator sending $b\mathscr{B}(S)$ into $\mathscr{F} \subset b\mathscr{B}(S)$. For $v \in b\mathscr{B}(S)$ the notation v-greedy means that π satisfies

$$\pi(x) \in \operatorname{argmax}_{u \in \Gamma(x)} \left\{ r(x,u) + \rho \int v(y) \mathbf{M}(x,u;dy) \right\} \tag{14}$$

for all $x \in S$. Observe that the algorithm terminates in finite time for any strictly positive tolerance, as the nonexpansiveness of L implies

[15] An alternative measure is Euler residuals, as discussed in Judd (1992) and Santos (2000). Since not all dynamic programs satisfy Euler equations the measure (13) is more general. (For alternative treatments of policy errors using geometric measures see Santos and Vigo-Aguiar, 1998, or Maldonado and Svaiter, 2001.)

that $\hat{T} := LT$ is a contraction mapping of modulus ρ, and hence $e_n \leq \rho^n \|v_1 - v_0\| \to 0$.

Below we adopt the convention that N always denotes the number of iterations after which the FVI algorithm terminates. It follows that the policy π returned by the algorithm is v_N-greedy.[16]

Combining ideas found in Puterman (1994), Judd and Solnick (1994), Gordon (1995), Rust (1996) and Santos and Vigo-Aguiar (1998), one can establish the following result:

Theorem 4.1. *If the FVI algorithm terminates after N iterations, then for every $x \in S$ we have*

$$v^*(x) - v_\pi(x) \leq \frac{2}{1-\rho} \times (\rho e_N + \|LTv_N - Tv_N\|)$$

The two sources of error in this bound are e_N, the deviation of v_N from v_{N-1} under the supremum norm, and $\|LTv_N - Tv_N\|$, which measures the ability of L to approximate the function Tv_N. The latter is not directly observable, and requires further analysis to bound. Below we give some indications of how this can be done.

The following remarks highlight some key points of the theorem.

Remark 4.1. The bound in Theorem 4.1 should be compared to the bound $v^*(x) - v_\pi(x) \leq 2\rho e_N/(1-\rho)$ given by Puterman (1994, Theorem 6.3.1) for the finite state case, where no approximation is used and value iteration can be carried out exactly. In the present case, if there is no approximation error (i.e., if $LTv_N = Tv_N$), then the bound in Theorem 4.1 reduces to Puterman's bound. This suggests that our bound is relatively tight.

Remark 4.2. It may seem that the error e_n in the FVI algorithm will be difficult to evaluate accurately. However, both v_n and v_{n-1} lie in

[16] In general lines 9 and 10 of the algorithm cannot be implemented exactly. Hence it may be preferable to return the approximate value function v_N and evaluate optimal actions by solving the maximization in (14) as required. In practice, however, computing a good approximation to π by evaluating the right hand side of (14) at many points in S takes very little computational effort relative to the FVI algorithm itself, as only one function (i.e., π itself) need be approximated.

the simple parametric class $\mathscr{F}$. As a result, evaluation of the error is typically straightforward.

Now we turn to the second theorem of the paper. Suppose for some reason that the term $\|LTv_N - Tv_N\|$ in Theorem 4.1 is difficult to evaluate or bound efficiently. In that case it may be easier to assess the approximation error $\|Lv^* - v^*\|$. The next result gives a bound using $\|Lv^* - v^*\|$ instead of $\|LTv_N - Tv_N\|$, albeit at the cost of a larger constant term:

Theorem 4.2. *If the FVI algorithm terminates after N iterations, then for every $x \in S$ we have*

$$v^*(x) - v_\pi(x) \leq \frac{2}{(1-\rho)^2} \times (\rho e_N + \|Lv^* - v^*\|)$$

The proofs of Theorems 4.1 and 4.2 are given in Section 6.

5. Repeated Partial Discretization

Let us consider the FVI algorithm in more detail for the case where L uses piecewise constant approximation (see Section 3). For reasons that become clear below, we refer to this case as repeated partial discretization (RPD). The following RPD algorithm is closely related to standard discretization, where a continuous state model is replaced by a similar model evolving on a finite grid. At the same time, RPD possesses several important advantages. One is that, since the approximation operator is nonexpansive, the error bounds developed above all apply. We show how, in many cases, bounds are easily derived from a term automatically generated by the FVI algorithm.

A second advantage is that the reward function r and the law of motion are never themselves discretized—and nor need they be, as these are primitives which presumably can be implemented without discretization. Thus, RPD does not discard the information contained in the values of these functions between the grid points.

Finally, in RPD it is possible to adjust the location and size of the grid at each iteration. A number of algorithms use variable grids for discretized dynamic programming, which allows one to place relatively many grid points near the areas of greatest curvature (i.e.,

the areas where approximation is most difficult) at each iteration. We do not discuss variable grid methods further in this paper.

In what follows we discuss RPD in some detail and give direct error bounds. The method is then applied to a model of stochastic speculative prices due to Samuelson (1971).

To begin, let S be a subset of $\mathbb{R}^d$ with the usual partial order.[17] Suppose that S can be written as the union of finitely many disjoint rectangles $\{J_i\}_{i=1}^k$, where

$$J_i := [x_i, y_i) := [x_i^1, y_i^1) \times \cdots \times [x_i^d, y_i^d)$$

Let $\mathscr{C}$ be the set of functions from S to $\mathbb{R}$ which are constant on each J_i. For $w \in b\mathscr{B}(S)$, define the operator $L\colon b\mathscr{B}(S) \to \mathscr{C}$ by $Lw(x) = w(x_i)$ when $x \in J_i$. Below we use the shorthand notation $\text{step}[a_1, \ldots, a_k]$ to mean the piecewise constant function on S given by the vector $(a_1, \ldots, a_k)$, in the sense that $\text{step}[a_1, \ldots, a_k] = a_i$ on J_i. In this notation, $Lw = \text{step}[w(x_1), \ldots, w(x_k)]$.

5.1. **The RPD Algorithm.** Consider Algorithm 2, which is a specialized version of the FVI algorithm corresponding to piecewise constant approximation.[18] In many situations Algorithm 2 (henceforth, the RPD algorithm) lends itself to simple implementation and yields an error bound which is completely specified by observables. The details are in Proposition 5.1.

Proposition 5.1. *Let π, R and e_N be as returned by the RPD algorithm, which is assumed to terminate after N iterations. If T preserves monotonicity, then*

$$v^*(x) - v_\pi(x) \leq \frac{2}{1-\rho} \times (\rho e_N + R) \qquad (x \in S)$$

whenever the initial condition v_0 is monotone increasing.[19]

[17] Let $x = (x^j)_{j=1}^d$ and $y = (y^j)_{j=1}^d$. Say that $x \leq y$ if $x^j \leq y^j$ for all j.

[18] One apparent difference is that the FVI algorithm defines e_n as $\|v_n - v_{n-1}\|$, while Algorithm 2 defines $e_n = \max_{1 \leq i \leq k} |v_n(x_i) - v_{n-1}(x_i)|$. But since $v_j \in \mathscr{C}$ for all j it should be clear that these two definitions are equivalent.

[19] Preservation of monotonicity means that Tw is increasing whenever $w \in b\mathscr{B}(S)$ is increasing.

Algorithm 2: RPD algorithm

initialize $v_0 \in \mathscr{C}$ and set $n = 0$;
repeat
  set $n = n + 1$;
  for *i in 1 to k* **do** evaluate $Tv_{n-1}(x_i)$;
  set $LTv_{n-1} = \text{step}[Tv_{n-1}(x_1), \ldots, Tv_{n-1}(x_k)]$;
  set $v_n = LTv_{n-1}$;
  set $e_n = \max_{1 \leq i \leq k} |v_n(x_i) - v_{n-1}(x_i)|$;
until *e_n falls below some tolerance* ;
solve for a v_n-greedy policy π ;
set $R = \max_{1 \leq i \leq k} |Tv_n(y_i) - Tv_n(x_i)|$;
return *π, R and e_n*

Remark 5.1. The condition that T preserves monotonicity holds in many applications. For example, suppose as in (3) that the law of motion is given by $X_{t+1} = F(X_t, U_t, W_{t+1})$, where $W_t \overset{\text{IID}}{\sim} G(dz)$. If $x \mapsto F(x, u, z)$ is increasing for all fixed u and z, and, moreover, $x \mapsto r(x, u)$ is increasing for all u, then T preserves monotonicity. These kinds of results are well-known and further details are omitted.

Remark 5.2. In the RPD algorithm, note that evaluation of R needs no further optimization, as π and v_N are available, π is v_N-greedy, and

$$Tv_N(x) = r(x, \pi(x)) + \rho \int v_N(y) \mathbf{M}(x, \pi(x); dy) \qquad (x \in S)$$

5.2. **Application.** To further illustrate the ideas, we now apply the RPD algorithm to Samuelson's (1971) theory of price equilibrium in a commodity market with speculative investment.[20] The model has recently been the basis of active empirical study of commodity prices. We follow Chambers and Bailey (1996) and Deaton and Laroque (1996) in considering a commodity price model with correlated supply shocks.

[20]Code for the following is available on request from the author.

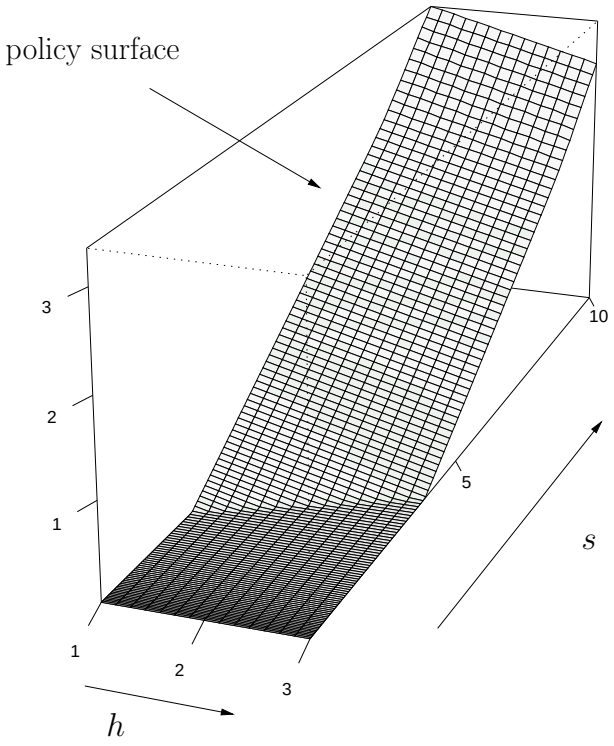

FIGURE 1. Optimal investment policy.

Briefly, the model describes intertemporal equilibrium in a single commodity market with consumption demand c_t determined by inverse demand function P ($p_t = P(c_t)$), and speculative demand q_t. Supply s_t consists of the "harvest" h_t plus λq_{t-1}, where $\lambda < 1$ is a "shrinkage" parameter and q_{t-1} is carryover from the last period. The harvest process $(h_t)_{t=0}^{\infty}$ is correlated. We assume in particular that

$$h_{t+1} = \theta h_t + (1-\theta)W_{t+1}$$

where $(W_t)_{t=1}^{\infty}$ is an independent shock process with identical cumulative distribution function G, and $\theta \in (0,1)$ is a parameter.

Equilibrium prices are determined by arbitrage conditions. Samuelson (1971) famously demonstrated that these restrictions correspond to the first order conditions of a dynamic programming problem with discount factor $\rho := (1+r)^{-1}$ and period utility function $U(c) = \int_0^c P(x)dx$. The program in question is $\max \mathbf{E}\left[\sum_{t=0}^{\infty} \rho^t U(c_t)\right]$ subject

to restrictions

$$c_t + q_t = s_t, \quad s_{t+1} = \lambda q_t + h_{t+1}, \quad s_0, h_0 \text{ given}$$

In the framework of Section 2, the control is q and two-dimensional state is $x = (s, h)$. The law of motion is

$$(s, h, q, z) \mapsto \begin{pmatrix} s' \\ h' \end{pmatrix} = \begin{pmatrix} \lambda q + \theta h + (1-\theta) z \\ \theta h + (1-\theta) z \end{pmatrix}$$

where a prime denotes next period's value. If $(W_t)_{t=0}^{\infty}$ takes values in $[\check{z}, \hat{z}]$, then the same is true of $(h_t)_{t=0}^{\infty}$. The state space can be set as

$$S := [\check{z}, \hat{z}/(1-\lambda)] \times [\check{z}, \hat{z}]$$

and the feasible correspondence as $\Gamma(s, h) = [0, s]$. In particular, $(s, h) \in S$ and $q \in \Gamma(s, h)$ implies $(s', h') \in S$ with probability one. The reward function is $U(s - q)$.[21]

We set $U(c) = c^{\alpha}$, and let $W = a + bV$, where V is beta$(5, 5)$. The parameters are set to $\rho = 0.9$, $\lambda = 0.7$, $\theta = 0.3$, $\alpha = 0.2$, $a = 1$ and $b = 2$. As a result, $\check{z} = 1$ and $\hat{z} = 3$. The RPD algorithm was initialized with $v(s, h) = U(s_k)$ on J_{kj}.

To study the RPD algorithm, the policy function and the error bound were computed for different grid sizes after $N = 40$ iterations. Figure 1 gives the approximate optimal policy when the grid size is 960. Table 1 shows the error bound

$$\frac{2}{1-\rho} \times (\rho e_N + R)$$

from Proposition 5.1 for different grid sizes (column 2). To put the errors in context, we used this upper bound on the absolute error to compute a lower bound on the fraction of total value $v^*(x)$ obtained by the approximate optimal policy. (The method is outlined in the appendix.) The results are shown in column 3. The bounds guarantee that the approximate optimal policy obtains over 95% of available value for a grid size of 1800 and 40 iterations.

[21] The graph gr Γ is all (s, h, q) with $(s, h) \in S$ and $0 \leq q \leq s$.

grid size	error bound	value obtained
500	0.765	$\geq 93.1\%$
960	0.623	$\geq 94.4\%$
1800	0.479	$\geq 95.6\%$
5000	0.451	$\geq 95.9\%$

TABLE 1. Error bounds by grid size, 40 iterations

6. PROOFS

Let us now address the proof of Theorem 4.1. Since the initial condition x will vary according to the problem, we construct a bound on the deviation $v^*(x) - v_\pi(x)$ which is uniform over $x \in S$. In practice, this is done by bounding the sup-norm error $\|v^* - v_\pi\|$. Using the triangle inequality, the sup-norm error is broken down as

$$\|v^* - v_\pi\| \leq \|v^* - v_N\| + \|v_N - v_\pi\| \tag{15}$$

where $v_N \in b\mathscr{B}(S)$ and N are as in Theorem 4.1. First we bound the first term on the right hand side of (15):

Claim 6.1. *We have* $(1-\rho)\|v^* - v_N\| \leq \rho e_N + \|LTv_N - Tv_N\|$.

Proof. By the triangle inequality and the contraction property of T,

$$\begin{aligned} \|v^* - v_N\| &\leq \|v^* - Tv_N\| + \|Tv_N - v_N\| \\ &\leq \rho\|v^* - v_N\| + \|Tv_N - v_N\| \end{aligned}$$

$$\therefore \quad (1-\rho)\|v^* - v_N\| \leq \|Tv_N - v_N\| \tag{16}$$

Moreover,

$$\begin{aligned} \|Tv_N - v_N\| &\leq \|Tv_N - v_{N+1}\| + \|v_{N+1} - v_N\| \\ &\leq \|v_{N+1} - Tv_N\| + \rho\|v_N - v_{N-1}\| \\ &= \|LTv_N - Tv_N\| + \rho e_N \end{aligned}$$

Putting this together with (16) gives the bound in Claim 6.1. □

Next consider the second term on the right hand side of (15).

Claim 6.2. *We have* $(1-\rho)\|v_N - v_\pi\| \leq \rho e_N + \|LTv_N - Tv_N\|$.

Proof. By the triangle inequality,

$$\|v_N - v_\pi\| \leq \|v_N - Tv_N\| + \|Tv_N - v_\pi\| \tag{17}$$

Consider the second term in the right hand side of (17). From the definition of the Bellman operator T we have

$$Tv_N(x) = \max_{u \in \Gamma(x)} \left\{ r(x,u) + \rho \int v_N(y) \mathbf{M}(x,u;dy) \right\}$$

One the other hand, by the definition of the operator T_π,

$$T_\pi v_N(x) = r(x,\pi(x)) + \rho \int v_N(y) \mathbf{M}(x,\pi(x);dy)$$

Since π is v_N-greedy, Tv_N and $T_\pi v_N$ are equal. Moreover, we know that T_π is a contraction of modulus ρ, and v_π is the unique fixed point. Hence

$$\|Tv_N - v_\pi\| = \|T_\pi v_N - T_\pi v_\pi\| \leq \rho \|v_N - v_\pi\|$$

Substituting this into (17) we get

$$\|v_N - v_\pi\| \leq \|v_N - Tv_N\| + \rho \|v_N - v_\pi\|$$

$$\therefore \quad (1-\rho)\|v_N - v_\pi\| \leq \|v_N - Tv_N\| \tag{18}$$

Finally, we have already shown in the proof of Claim 6.1 that

$$\|Tv_N - v_N\| \leq \|LTv_N - Tv_N\| + \rho e_N$$

Substituting this into (18) gives the bound that we are seeking. □

Proof of Theorem 4.1. From (15) and Claims 6.1 and 6.2 we get

$$(1-\rho)\|v^* - v_\pi\| \leq 2(\rho e_N + \|LTv_N - Tv_N\|)$$

The bound in Theorem 4.1 follows immediately. □

Next we turn to the proof of Theorem 4.2. The proof is based on the following two estimates:

Claim 6.3. *If π is v_N-greedy, then we have*

$$(1-\rho)\|v^* - v_\pi\| \leq 2\|v_N - v^*\| \tag{19}$$

Proof. We have

$$\|v^* - v_\pi\| \leq \|v^* - v_N\| + \|v_N - v_\pi\| \tag{20}$$

On the other hand,

$$\|v_N - v_\pi\| \leq \|v_N - Tv_N\| + \|Tv_N - v_\pi\| \tag{21}$$

Consider the first term on the right hand side of (21). Observe that for any $w \in b\mathscr{B}(S)$ we have

$$\begin{aligned}\|w - Tw\| &\leq \|w - v^*\| + \|v^* - Tw\| \\ &\leq \|w - v^*\| + \rho\|v^* - w\| = (1+\rho)\|w - v^*\|\end{aligned}$$

Substituting in v_N for w, we obtain

$$\|v_N - Tv_N\| \leq (1+\rho)\|v_N - v^*\| \tag{22}$$

Now consider the second term on the right hand side of (21). It has already been observed that for this particular policy π we have $Tv_N = T_\pi v_N$, so

$$\|Tv_N - v_\pi\| = \|T_\pi v_N - v_\pi\| = \|T_\pi v_N - T_\pi v_\pi\| \leq \rho\|v_N - v_\pi\|$$

Substituting this bound and (22) into (21), we obtain

$$\|v_N - v_\pi\| \leq (1+\rho)\|v_N - v^*\| + \rho\|v_N - v_\pi\|$$

$$\therefore \quad \|v_N - v_\pi\| \leq \frac{1+\rho}{1-\rho}\|v_N - v^*\|$$

This inequality and (20) together give

$$\|v^* - v_\pi\| \leq \|v^* - v_N\| + \frac{1+\rho}{1-\rho}\|v_N - v^*\|$$

Simple algebra now gives (19). □

Claim 6.4. *For every $n \in \mathbb{N}$ we have*

$$(1-\rho)\|v^* - v_n\| \leq \|v^* - Lv^*\| + \rho\|v_n - v_{n-1}\|$$

Proof. Let $\hat{v}$ be the fixed point of $\hat{T}$. By the triangle inequality,

$$\|v^* - v_n\| \leq \|v^* - \hat{v}\| + \|\hat{v} - v_n\| \tag{23}$$

Regarding the first term on the right hand side of (23), we have

$$\begin{aligned}\|v^* - \hat{v}\| &\leq \|v^* - \hat{T}v^*\| + \|\hat{T}v^* - \hat{v}\| \\ &= \|v^* - Lv^*\| + \|\hat{T}v^* - \hat{T}\hat{v}\| \leq \|v^* - Lv^*\| + \rho\|v^* - \hat{v}\|\end{aligned}$$

$$\therefore \quad (1-\rho)\|v^* - \hat{v}\| \leq \|v^* - Lv^*\| \tag{24}$$

Regarding the second term in the sum (23), we have

$$\begin{aligned}\|\hat{v} - v_n\| &\leq \|\hat{v} - \hat{T}^{n+1}v_0\| + \|\hat{T}^{n+1}v_0 - \hat{T}^n v_0\| \\ &\leq \rho\|\hat{v} - v_n\| + \rho\|v_n - v_{n-1}\|\end{aligned}$$

$$\therefore \quad (1-\rho)\|\hat{v} - v_n\| \leq \rho\|v_n - v_{n-1}\| \tag{25}$$

Combining (23), (24) and (25) gives the bound we are seeking. □

Proof of Theorem 4.2. Pick any $x \in S$, and suppose that the value iteration algorithm terminates after N steps. By Claim 6.3 we have

$$v^*(x) - v_\pi(x) \leq \frac{2}{1-\rho}\|v_N - v^*\|$$

Applying Claim 6.4, this becomes

$$v^*(x) - v_\pi(x) \leq \frac{2}{(1-\rho)^2}(\rho\|v_N - v_{N-1}\| + \|v^* - Lv^*\|)$$

The claim in Theorem 4.2 now follows from the definition of e_N. □

Proof of Proposition 5.1. The proof is almost trivial. Since the RPD algorithm is a special case of the FVI algorithm, Theorem 4.1 gives

$$v^*(x) - v_\pi(x) \leq \frac{2}{1-\rho} \times (\rho e_N + \|LTv_N - Tv_N\|) \qquad (x \in S)$$

where N is the number of iterations after which the RPD algorithm terminates. Thus we need only show that $\|Lw - w\| \leq R$, where $w := Tv_N$. In doing so, notice that w is monotone increasing, as the initial condition v_0 has this property, and T preserves monotonicity.[22]

[22]Clearly L automatically preserves monotonicity, and hence so does $\hat{T}$.

So suppose to the contrary that there is an $x \in S$ with $|Lw(x) - w(x)| > R$. Without loss of generality, assume that $x \in J_m$, so that

$$R < |Lw(x) - w(x)| = |w(x_m) - w(x)| = w(x) - w(x_m)$$

where the last step is by monotonicity. On the other hand, we have

$$w(y_m) - w(x_m) \leq \max_{1 \leq i \leq k} |w(y_i) - w(x_i)| =: R$$

Putting these two inequalities together gives $w(y_m) < w(x_m)$, which contradicts monotonicity of w. □

Finally we discuss the technique used to obtain the lower bound on the fraction of total value $v^*(x)$ obtained by the approximate optimal policy π, as shown in column 3 of Table 1. The fraction in question is $v_\pi(x)/v^*(x)$. Regarding this fraction, observe that

$$\begin{aligned} \frac{v^*(x) - v_\pi(x)}{v^*(x)} &\leq \frac{v^*(x) - v_\pi(x)}{v^*(\check{z}, \check{z})} \\ &\leq \frac{v^*(x) - v_\pi(x)}{v_N(\check{z}, \check{z})} \leq \frac{2}{1-\rho} \times \frac{\rho e_N + R}{v_N(\check{z}, \check{z})} := \beta \end{aligned}$$

where the first equation is by monotonicity of v^* and the second follows from our choice of initial condition. The value β can be computed from the second column of Table 1 and the term $v_N(\check{z}, \check{z})$. Now note that

$$\frac{v_\pi(x)}{v^*(x)} = 1 - \frac{v^*(x) - v_\pi(x)}{v^*(x)} \geq 1 - \beta$$

REFERENCES

[1] Aruoba, S. B., J. Fernandez-Villaverde and J. F. Rubio-Ramirez (2006): "Comparing Solution Methods for Dynamic Equilibrium Economies, *Journal of Economic Dynamics and Control,* 30 (12), 2477–2508.

[2] M.J. Chambers and R.J. Bailey (1996): "A Theory of Commodity Price Fluctuations," *Journal of Political Economy,* 104(5), 924–957.

[3] Deaton, A. and G. Laroque (1996): "Comptetitive Storage and Commodity Price Dynamics," *Journal of Political Economy,* 104(5), 896–923.

[4] Drummond, C (1996): "Preventing Overshoot of Splines with Application to reinforcement Learning," Computer Science Dept. Ottawa TR-96-05.

[5] Gordon, G.J. (1995): "Stable Function Approximation in Dynamic Programming," Proceedings of the 12th International Conference on Machine Learning.

[6] Guestrin, C., D. Koller and R. Parr (2001): "Max-Norm Projections for Factored MDPs," International Joint Conference on Artificial Intelligence, Vol. 1, 673–680.

[7] Hernández-Lerma, O., and J.B. Lasserre (1999): *Further Topics in Discrete-Time Markov Control Processes*, Springer-Verlag, New York.

[8] Judd, K.L. (1992): "Projection Methods for Solving Aggregate Growth Models," *Journal of Economic Theory*, 58 (2), 410–452.

[9] Judd, K.L. and A. Solnick (1994): "Numerical Dynamic Programming with Shape-Preserving Splines," unpublished manuscript.

[10] Lyche, T and K. Mørken (2002): *Spline Methods*, mimeo, University of Oslo.

[11] Maldonado, W. L. and B. F. Svaiter (2001): "On the Accuracy of the Estimated Policy Function using the Bellman Contraction Method," *Economics Bulletin*, (3) 15, 1–8.

[12] Munos, R. and A. Moore (1999): "Variable Resolution Discretization in Optimal Control," *Machine Learning*, 1, 1–24.

[13] Puterman, M. (1994): *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, John Wiley & Sons, New York.

[14] Rust, J. (1996): "Numerical Dynamic Programming in Economics," in H. Amman, D. Kendrick and J. Rust (eds.) *Handbook of Computational Economics*, Elsevier, North Holland.

[15] Rust, J. (1997): "Using Randomization to Break the Curse of Dimensionality," *Econometrica*, 65 (3), 487–516.

[16] Samuelson, P.A. (1971): "Stochastic Speculative Price," *Proceedings of the National Academy of Science*, 68 (2), 335–337.

[17] Santos, M.S. and J. Vigo-Aguiar (1998): "Analysis of a Numerical Dynamic Programming Algorithm Applied to Economic Models," *Econometrica*, 66(2), 409–426.

[18] Santos, M.S. (2000): "Accuracy of Numerical Solutions Using the Euler Equation Residuals," *Econometrica*, 68 (6), 1377–1402.

[19] Stokey, N. L., R. E. Lucas and E. C. Prescott (1989): *Recursive Methods in Economic Dynamics*, Harvard University Press, Massachusetts.

[20] Tsitsiklis, J.N. and B. Van Roy (1996): "Feature-Based Methods for Large Scale Dynamic Programming," *Machine Learning*, 22, 59–94.

INSTITUTE OF ECONOMIC RESEARCH, KYOTO UNIVERSITY, YOSHIDA-HONMACHI, SAKYO-KU, KYOTO 606-8501, JAPAN

E-mail address: john@kier.kyoto-u.ac.jp