Numerical Dynamic Programming in Economics

John Rust
Yale University

Contents [1]

---

[1] Revised November 1994 draft for *Handbook of Computational Economics*, H. Amman, D. Kendrick and J. Rust, (eds.). I am grateful for helpful comments by Hans Amman, Ken Judd, David Kendrick, Martin Puterman, and two not very anonymous referees, Charles Tapiero and John Tsitsiklis.

7. References

1. Introduction

This chapter surveys numerical methods for solving dynamic programming (DP) problems. The DP framework has been extensively used in economic modeling because it is sufficiently rich to model almost any problem involving sequential decision making over time and under uncertainty.[2] By a simple re-definition of variables virtually any DP problem can be formulated as a Markov decision process (MDP) in which a decision maker who is in state s_t at time $t = 1, \ldots, T$ takes an action a_t that determines current utility $u(s_t, a_t)$ and affects the distribution of next period's state s_{t+1} via a Markov transition probability $p(s_{t+1}|s_t, a_t)$. The problem is to determine an *optimal decision rule* α that solves $V(s) \equiv max_\alpha E_\alpha \left\{\sum_{t=0}^{T} \beta^t u(s_t, a_t)|s_0 = s\right\}$ where E_α denotes expectation with respect to the controlled stochastic process $\{s_t, a_t\}$ induced by the decision rule $\alpha \equiv \{\alpha_1, \ldots, \alpha_T\}$, and $\beta \in (0, 1)$ denotes the discount factor. What makes these problems especially difficult is that instead of optimizing over an *ex ante* fixed sequence of actions $\{a_0, \ldots, a_T\}$ one needs to optimize over sequences of *functions* $\{\alpha_0, \ldots, \alpha_T\}$ that allow the *ex post* decision a_t to vary as a best response to the current state of the process, i.e. $a_t = \alpha_t(s_t)$. The method of dynamic programming provides a constructive, recursive procedure for computing α using the *value function* V as a "shadow price" to decentralize a complicated stochastic/multiperiod optimization problem into a sequence of simpler deterministic/static optimization problems. [3] In infinite horizon problems V is the unique solution to a *Bellman's equation,* $V = \Gamma(V)$, where the *Bellman operator* Γ is defined by:

$$\Gamma(V)(s) = \max_{a \in A(s)} [u(s, a) + \beta \int V(s')p(ds'|s, a)], \tag{1.1}$$

and the optimal decision rule can be recovered from V by finding a value $\alpha(s) \in A(s)$ that attains the maximum in (1.1) for each $s \in S$. We review the main theoretical results about MDP's in section 2, providing an intuitive derivation Bellman's equation in the infinite horizon case and showing that the value function V_α corresponding to a policy α is simply a multidimensional generalization of a geometric series. This implies that V_α can be computed as the solution to a system of linear equations, the key step in the Bellman 1955, 1957 and Howard 1960 *policy iteration* algorithm. The Bellman operator has a particularly nice mathematical property: Γ is a *contraction mapping.*

---

[2] See Stokey and Lucas 1987 for examples of DP models in economic theory. See Rust 1994a, 1994b for examples of of DP models in econometrics.

[3] In finite horizon problems V actually denotes an entire sequence of value functions, $V \equiv \{V_0^T, \ldots, V_T^T\}$, just as α denotes a sequence of decision rules. In the infinite-horizon case, the solution (V, α) reduces to a pair of functions of the current state s.

A large number of numerical solution methods exploit the contraction property to yield convergent, numerically stable methods for computing approximate solutions to Bellman's equation, including the classic method of *successive approximations.* Since V can be recovered from α and vice versa, the rest of this chapter focuses on methods for computing the value function V, with separate discussions of numerical problems involved in approximating α where appropriate. [4]

From the standpoint of computation, there is an important distinction between *discrete MDP's* whose state and control variables can assume only a finite number of possible values, and *continuous MDP's* whose state and control variables can assume a continuum of possible values. The value functions for discrete MDP's live in a subset B of the finite-dimensional Euclidean space $R^{|S|}$ (where $|S|$ is the number of elements in S), whereas the value functions of continuous MDP's live in a subset B of the infinite-dimensional Banach space $\mathcal{B}(S)$ of bounded, measurable real-valued functions on S. Thus, discrete MDP problems can be solved exactly (modulo rounding error in arithmetic operations), whereas the the solutions to continuous MDP's can generally only be approximated. Approximate solution methods may also be attractive for solving discrete MDP's with a large number of possible states $|S|$ or actions $|A|$.

There are two basic strategies for approximating the solution to a continuous MDP: 1) discrete approximation and 2) smooth approximation. Discrete approximation methods solve a finite state MDP problem that approximates the original continuous MDP problem over a finite grid of points in the state space S and action space A. Since the methods for solving discrete MDP's have been well developed and exposited in the operations research literature (e.g. Bertsekas, 1987, Porteus, 1980 and Puterman, 1990, 1994), this chapter will only briefly review the standard approaches, focusing instead on recent research on discretization methods for solving continuous MDP's. Although smooth approximation methods also have a long history (dating back to Bellman, 1957 Bellman and Dreyfus, 1962 and Bellman, Kalaba and Kotkin 1963), there has been a recent resurgence of interest in these methods as a potentially more efficient alternative to discretization methods (Christiano and Fisher, 1994, Johnson *et. al.* 1993, Judd and Solnick 1994, Marcet and Marshall, 1994, and Smith 1991). Smooth approximation methods treat the value function V and/or the decision rule α as smooth, flexible functions of s and a finite-dimensional parameter vector θ: examples include linear interpolation and a variety of more sophisticated approximation methods

[4] Special care must be taken in problems with a continuum of actions, since discontinuities or kinks in numerical estimates of the conditional expectation of V can create problems for numerical optimization algorithms used to recover α. Also, certain solution methods focus on computing α directly rather than indirectly by first computing V. We will provide special discussions of these methods in section 5.

such as polynomial series expansions, neural networks, and cubic splines. The objective of these methods is to choose a parameter $\hat{\theta}$ such that the implied approximations $V_{\hat{\theta}}$ or $\alpha_{\hat{\theta}}$ "best fit" the true solution V and α according to some metric. [5] In order to insure the convergence of a smooth approximation method, we need a parametrization that is sufficiently flexible to allow us to approximate arbitrary value functions V in the set B. One way to do this is via an expanding sequence of parameterizations that are ultimately dense in B in the sense that for each $V \in B$[6]

$$\lim_{k\to\infty} inf_{\theta\in R^k} \|V_\theta - V\| = 0, \tag{1.2}$$

where $\|V\| = \sup_{s\in S} |V(s)|$ denotes the usual "sup-norm". For example, consider an infinite horizon MDP problem with state space $S = [-1, 1]$. A natural choice for an expanding family of smooth approximations to V is

$$V_\theta(s) = \sum_{i=1}^{k} \theta_i p_i(s) \tag{1.3}$$

where $p_i(s) = \cos\left(i \cos^{-1}(s)\right)$ is the i^{th} ***Chebyshev polynomial.***[7] The Weierstrauss approximation theorem implies that the Chebyshev polynomials are ultimately dense in the space $B = C[-1, 1]$ of continuous functions on $[-1, 1]$.[8] Under the least squares criterion of goodness of fit, the problem is to choose a parameter $\theta \in \Theta \subset R^k$ to minimize the error function σ_N defined by:

$$\sigma_N(\theta) \equiv \sqrt{\sum_{i=1}^{N} |V_\theta(s_i) - \hat{\Gamma}(V_\theta)(s_i)|^2}, \tag{1.4}$$

where $\hat{\Gamma}$ is some computable approximation to the true Bellman operator Γ. Methods of this form are known as *minimum residual* (MR) methods since the parameter θ is chosen to set the *residual function* $R(V_\theta)(s) = V_\theta(s) - \hat{\Gamma}(V_\theta)(s)$ as close to the zero function as possible. The philosophy of these methods is that if the true value function V can be well approximated by a flexible parametric function V_θ for a small number of parameters k, we will be able to find a better approximation to

[5] Other variants of smooth approximation methods use nonlinear equation solvers to find a value θ^* that satisfies a set of "orthogonality conditions". In finite horizon MDP's interpolation methods are used to store the value function. Other methods such as Johnson *et. al.* 1993 use local polynomial interpolation of V over a grid of points in S.

[6] If the MDP problem has further structure that allows us to restrict the true value function to some subset of B such as the space of continuous functions, $C(S)$, then the limit in equation (1.2) condition need only for each V in this subset.

[7] See Judd chapter 6 for a definition of Chebyshev and other families of orthogonal polynomials.

[8] In fact we have the following error bound for equation (1.2): $\inf_{\theta\in R^k} \|V_\theta - V\| \leq b \log(k)/k$ for some constant $b > 0$. See Judd chapter 6 for a formal statement of this result and Rivlin 1969 for a proof.

V in significantly less cpu time by minimizing an error function such as $\sigma_N(\theta)$ in (1.4) than by "brute force" discretization methods.

At a sufficient level of abstraction the distinction between discrete and smooth approximation methods starts to evaporate since discrete approximation based on N grid points can always be regarded as a particular way of parametrizing the solution to the MDP problem (e.g. as a simple linear spline function that interpolates V_N at the N grid points). Also, as equation (1.4) illustrates, practical implementation of smooth approximation methods require discretization methods of various sorts to compute various integrals, to provide grid points for interpolation, knots for splines, and so forth. However there is a distinction between the two approaches that we believe is key to understanding their relative strengths and weaknesses: *Discrete approximation methods focus on solving an approximate finite MDP problem whose associated Bellman operator $\Gamma_N : R^N \to R^N$ fully preserves the contraction mapping property of the true Bellman operator $\Gamma : B \to B$. In addition, an approximate fixed point V_N to Γ_N typically preserves the monotonicity and concavity properties of the true value function V. Smooth approximation methods such as the minimum residual method outlined in equations (1.3) to (1.4) transform the MDP problem into the problem of minimizing a smooth function $\sigma_N(\theta)$ that generally does not have any easily exploitable mathematical structure (beyond smoothness in θ). Furthermore the parametrized value function V_θ may not retain any of the concavity or monotonicity properties that are known to hold for the true solution V.* Of course, there are exceptions to this distinction as well. Section 5 describes a discrete approximation method of Coleman 1990, 1993 that computes the optimal decision rule α directly as the solution to a functional equation known as the *Euler equation.* The Euler equation has an associated *Euler operator* that maps policy functions into policy functions just as the Bellman equation has an associated Bellman operator that maps value functions into value functions. Unfortunately, the Euler operator is not a contraction mapping, which makes it more difficult to derive error bounds and prove the convergence of iterative methods such as successive approximations. Another exception is the smooth approximation method of Judd and Solnick 1994 that uses "shape preserving splines" to preserve concavity and monotonicity properties of V.

Approximate solution methods force us to confront a tradeoff between the maximum allowable error ϵ in the numerical solution and the amount of computer time (and storage space) needed to compute it. Solution time will also be an increasing function of any relevant measure of the size or *dimension* d of the MDP problem. It goes without saying that economists are interested in using the fastest possible algorithm to solve their MDP problem given any specified values

of (ϵ, d). Economists are also concerned about using algorithms that are both accurate and numerically stable since the MDP solution algorithm is often embedded or "nested" as a subroutine inside a larger optimization or equilibrium problem. Examples include computing competitive equilibria of stochastic general equilibrium models with incomplete markets (Hansen and Sargent, 1993, Imrohoroğlu and Imrohoroğlu, 1993, McGuire and Pakes, 1994), maximum likelihood estimation of unknown parameters of u and p using data on observed states and decisions of actual decision-makers (Eckstein and Wolpin, 1987, Rust, 1994 and Sargent, 1979), and computation and econometric estimation of Bayesian Nash equilibria of dynamic games (McKelvey and Palfrey, 1992). All of these problems are solved by "polyalgorithms" that contain MDP solution subroutines. [9] Since the MDP problem must be repeatedly re-solved for various trial sequences of the "primitives" (β, u, p), speed, accuracy, and numerical stability are critical.

A large number of alternative solution methods have been proposed in the last 40 years. Bellman was one of the original contributors to this literature, with pioneering work on both discrete and smooth approximation methods (e.g. the previously cited work on policy iteration and polynomial approximations of the value function). Recently there has been considerable controversy in the economics literature about the relative merits of discrete versus smooth approximation methods for solving continuous MDP problems. Part of the controversy arose from a "horse race" in the 1990 *Journal of Business and Economic Statistics* (Taylor and Uhlig, 1990) in which a number of alternative solution methods competed in their ability to solve the classical Brock-Mirman stochastic growth model which we will describe in section 2.6. More recently Judd 1993 has claimed that

> Approximating continuous-state problems with finite state Markov chains limits the range of problems which can be analyzed. Fortunately, state-space discretization is unnecessary. For the past thirty years, the standard procedure in Operations Research literature (see Bellman, 1963, Dantzig 1974, Daniel, 1976) has been to approximate value functions and policy rules over continuous state spaces with orthogonal polynomials, splines, or other suitable families of functions. This results in far faster algorithms and avoids the errors associated with making the problem unrealistically "lumpy". (p. 3)

This chapter offers some new perspectives on this debate by providing a conceptual framework for analyzing the accuracy and efficiency of various discrete and smooth approximation methods for continuous MDP problems. The framework is the theory of *computational complexity* (Garey and Johnson 1983, Traub and Woźniakowski 1980, and Traub, Wasilikowski and Woźniakowski

[9] We will see that the MDP subroutine is itself a fairly complicated polyalgorithm consisting of individual subroutines for numerical integration, optimization, approximation, and solution of systems of linear and nonlinear equations.

(TWW), 1988). We provide a brief review of the main results of complexity theory in section 3. Complexity theory is of particular interest because it has succeeded in characterizing the form of *optimal algorithms* for various mathematical problems. Chow and Tsitsiklis 1989 used this theory to establish a lower bound on the amount of computer time required to solve of a general class of continuous MDP's. A subsequent paper, Chow and Tsitsiklis 1991, presented a particular discrete approximation method — a *multigrid algorithm* — that is approximately optimal in a sense to be made precise in section 5.3. Thus, we will be appealing to complexity theory as a way of *finding optimal strategies for finding optimal strategies.*

There are two main branches of complexity theory, corresponding to discrete and continuous problems. Discrete (or algebraic) computational complexity applies to finite problems that can be solved exactly such as matrix multiplication, the traveling salesman problem, linear programming, and discrete MDP's. [10] The *size* of a discrete problem is indexed by an integer d and the (worst case) complexity, $comp(d)$, denotes the minimal number of computer operations necessary to solve the hardest possible problem of size d, (or ∞ if there is no algorithm capable of solving the problem). Continuous computational complexity theory applies to continuous problems such as multivariate integration, function approximation, nonlinear programming, and continuous MDP problems. None of these problems can be solved exactly, but in each case the true solution can be approximated to within an arbitrarily small error tolerance ϵ. Problem size is indexed by an integer d denoting the dimension of the space that the continuous variable lives in (typically a subset of R^d), and the complexity, $comp(\epsilon, d)$, is defined as the minimal computational cost (cpu time) of solving the hardest possible d-dimensional problem to within a tolerance of ϵ.

Complexity theory enables us to formalize an important practical limitation to our ability to solve increasingly detailed and realistic MDP problems: namely Bellman's 1955 *curse of dimensionality.* This is the well-known exponential rise in the amount of time and space required to compute the solution to a continuous MDP problem as the number of dimensions d_s of the state variable or the number of dimensions d_a of the control variable increases. Although one typically thinks of the curse of dimensionality as arising from the discretization of continuous MDP's, it also occurs in finite MDP's that have many state and control variables. For example, a finite MDP with d_s state variables each of which can take on $|S| > 1$ possible values has a total of $|S|^{d_s}$ possible states. The amount of computer time and storage required to solve such a problem increases exponentially fast as d_s increases. Smooth approximation methods cannot escape the curse of

[10] In the latter two problems we abstract from rounding error computer arithmetic.

dimensionality: for example using the Chebyshev approximation in equation (1.3), the dimension k of the parameter vector θ must increase at a sufficiently rapid rate as d_s increases in order to guarantee that $\|V_{\hat{\theta}} - V\| \leq \epsilon$. Indeed, the literature on approximation theory (Pinkus, 1985, Novak 1988) shows that k must increase at rate $(1/\epsilon)^{(d/r)}$ in order to obtain uniform ϵ-approximations of smooth multidimensional functions which are r-times continuously differentiable. Although there are certain nonlinear approximation methods such as neural networks that can be shown to require a polynomially rather than exponentially increasing number of parameters k to obtain ϵ-approximations for certain subclasses of functions (i.e. only $k = O(1/\epsilon^2)$ parameters are required to approximate the subclass of functions considered in Barron 1993, and Hornik, Stinchombe and White 1992) are still subject to curse of dimensionality due to curse of dimensionality involved in fitting the parameters θ of the neural net by some minimization such as nonlinear least squares: the amount of computer time required to find an ϵ-approximation to a global minimizer of an arbitrary smooth function increases exponentially fast as the number of parameters k increases (Nemirovsky and Yudin, 1983), at least on a worst case basis. Variations of smooth approximation methods that involve solutions of systems k nonlinear equations or which require multivariate interpolation or approximation are also subject to the curse of dimensionality (Sikorski, 1985, TWW, 1988).

Definition: *A class of discrete MDP problems with d_s state variables and d_a control variables is subject to the curse of dimensionality if $comp(d_s, d_a) = \Omega(2^{d_s+d_a})$. A class of continuous MDP problems is subject to the curse of dimensionality if $comp(\epsilon, d_a, d_a) = \Omega(1/\epsilon^{(d_s+d_a)})$.*[11]

In the computer science literature problems that are subject to the curse of dimensionality are called *intractable.*[12] If the complexity of the problem has an upper bound that only grows polynomially in d we say that the problem is in the class P of polynomial-time problems. Computer scientists refer to polynomial-time problems as *tractable.*[13]

---

[11] The notation Ω denotes a "lower bound" on complexity, i.e. $comp(d) = \Omega\left(g(d)\right)$ if $\overline{\lim}_{d\to\infty}|g(d)/\, comp(d)| < \infty$.

[12] We prefer the terminology "curse of dimensionality" since the common use of the term "intractable" connotes a problem that can't be solved. Computer scientists have a specific terminology for problems that can't be solved in any finite amount time: these problems have infinite complexity, and are classified as *non-computable.* However even though intractable problems are computable problems in the computer science terminology, as the problem grows large the lower bound on the solution time grows so quickly that these problems are not computable in any practical sense.

[13] Here again it is important to note the difference between the common meaning of the term "tractable" and the computer science definition. Even so-called "tractable" polynomial-time problems can quickly become computationally infeasible if complexity satisfies $comp(d) \geq O(d^k)$ for some large exponent k. However it seems to be a fortunate act of nature that the maximum exponent k for most common polynomial time problems is fairly small; typically $k \in [2, 4]$.

A large number of mathematical problems have been proven to be intractable on a worst case basis. These problems include multivariate integration, optimization, and function approximation (TWW, 1988), and solution of multidimensional partial differential equations (PDE's), and Fredhom integral equations (Werschulz, 1991). Lest the reader get too depressed about the potential usefulness of numerical methods at the outset, we note that there are some mathematical problems that are not subject to the curse of dimensionality. Problems such as linear programming and solution of ordinary differential equations have been proven to be in the class P of polynomial time problems (Nemirovsky and Yudin, 1983 and Werschulz, 1991). Unfortunately Chow and Tsitsiklis 1989 proved that the general continuous MDP problem is also intractable. They showed that the complexity function for continuous MDP's with d_s state variables and d_a control variables is given by:

$$comp(\epsilon, d_s, d_a, \beta) = \Theta\left(\frac{1}{\left((1-\beta)^2\epsilon\right)^{2d_s+d_a}}\right), \tag{1.5}$$

where the symbol Θ denotes both an upper and lower bound on complexity. [14] In subsequent work, Chow and Tsitsiklis 1991 developed that a "one way multigrid" algorithm that comes within a factor of $1/|\log(\beta)|$ of achieving this lower bound on complexity, and in this sense is an approximately optimal algorithm for the MDP problem. As we will see, the multigrid algorithm is a particularly simple example of a discrete approximation method that is based on simple equi-spaced grids of S and A. [15]

The fact that the Chow-Tsitsiklis lower bound on complexity increases exponentially fast in d_s and d_a tells us that the curse of dimensionality is an inherent aspect of continuous MDP problems that can't be circumvented by any solution algorithm, no matter how brilliantly designed. There are, however, three potential ways to legitimately circumvent the curse of dimensionality: 1) we can restrict attention to a limited class of MDP's such as the class of linear-quadratic MDP's, 2) we can use algorithms that work well on an average rather than worst case basis, or 3) we can use random rather than deterministic algorithms.

Chapter 5 by Sargent, McGrattan and Hansen demonstrates the payoffs to the first approach: they present highly efficient polynomial-time algorithms for solving the subclass of linear-quadratic

[14] Formally, $comp(d) = \Theta(g(d))$ if there exist constants $0 \leq c_1 \leq c_2$ such that $c_1 g(d) \leq comp(d) \leq c_2 g(d)$.

[15] Discrete approximation methods have been found to be approximately optimal algorithms for other mathematical problems as well. For example Werschulz 1991 showed that the standard finite element method (FEM) is a nearly optimal complexity algorithm for solving linear elliptic PDE's.

MDP's (LQ-MDP's) and associated "matrix Ricatti equation". These algorithms allow routine solution of very high dimensional LQ problems on desktop workstations.

The idea behind the second approach to circumventing the curse of dimensionality is that even though an algorithm may perform poorly in solving a "worst case" problem, it may in fact perform very satisfactorily for most problems that are typically encountered in economic applications. A classic example is the simplex algorithm for linear programming. Klee and Minty 1967 concocted an artificial sequence of LP problems that force the simplex algorithm to visit an exponentially increasing number of vertices before it finds an optimal solution. Nevertheless the simplex algorithm performs quite satisfactorily for most problems encountered in practical applications. There are alternative algorithms that are guaranteed to solve all possible LP problems in polynomial-time, (e.g. Karmarkar's 1985 interior point algorithm), but numerical comparisons reveal that for "typical" problems the simplex method is as fast if not faster. One can resolve this paradoxical finding by appealing to the concept of *average case complexity.* This requires us to specify a prior probability measure μ to represent the likelihood of encountering various problems and defines complexity by minimizing the *expected time* required to solve problems with an *expected error* of ϵ or less. Section 3 provides a formal definition of average case complexity. This concept is of interest since several problems such as multivariate integration, elliptic PDE's, and Fredholm integral equations of the second kind that are intractable on a worst case basis have been shown to be tractable on an average case basis (Woźniakowski 1991 and Werschulz 1991). Since multivariate integration constitutes a major part of the work involved in solving for V, this result provides hope that the MDP problem might be tractable on an average case basis. In section 3.3 we conjecture that recent results of Woźniakowski (1991,1992) can be used to show that certain classes of MDP's become tractable on an average case basis, i.e. the average case complexity function is bounded above by a polynomial function of the dimension d, and inverse error $1/\epsilon$. Section 5.4 describes several algorithms that we believe will perform well on an average case basis. One of these methods uses the optimal integration algorithm of Woźniakowski 1991 that evaluates integrands at the set of shifted *Hammersley points,* and another evaluates integrands at the set of *Sobol' points* (Paskov, 1994).

A final strategy for circumventing the curse of dimensionality is to use random rather than deterministic algorithms. Randomization can sometimes succeed in breaking the curse of dimensionality because it relaxes the requirement that the error in the numerical approximation is less than ϵ with probability one. However similar to the notion of average complexity, it can only offer the weaker assurance that the *expected error* in an approximate solution is less than ϵ. A classic example

where randomization succeeds in breaking the curse of dimensionality is multivariate integration. The monte carlo estimate of the integral $\int f(s)ds$ of a function f on the d-dimensional hypercube $S = [0,1]^d$ is formed by drawing a random sample of points $\{\tilde{s}_1, \ldots, \tilde{s}_N\}$ uniformly from S and forming the sample average $\sum_{i=1}^N f(s_i)/N$. Hölder's inequality implies that the expected error in a monte carlo integral converges to zero at rate $1/\sqrt{N}$, independent of the dimension d. However randomization does not always succeed in breaking the curse of dimensionality: problems such as multivariate optimization (Nemirovsky and Yudin, 1983), function approximation (TWW, 1988), and solution of linear elliptic PDE's and Fredhom integral equations of the second kind (Werschulz, 1991) are intractable regardless of whether or not randomization is used.

Section 5 discusses a recent result due to Rust (1994c) that proves that randomization does break the curse of dimensionality for a subclass of MDP's known as *discrete decision processes* (DDP's), i.e. MDP's with a finite number of possible actions. Rust's upper bound on the worst case randomized complexity of an infinite horizon DDP problem with $|A|$ possible choices and a d-dimensional continuous state vector s_t is given by:

$$comp^{\textit{wor-ran}}(\epsilon, d) = O\left(\frac{|A|^3 d^4}{|log(\beta)|(1-\beta)^8\epsilon^4}\right), \tag{1.6}$$

which implies that the DDP problem can be solved in polynomial time once randomization is allowed. Rust's proof is constructive since he presents a "random multigrid algorithm" that attains the upper bound on complexity in (1.6). The multigrid algorithm is based on the *random Bellman operator* $\tilde{\Gamma}_N : B \to B$ defined by:

$$\tilde{\Gamma}_N(V)(s) = \max_{a \in A(s)} [u(s,a) + \frac{\beta}{N}\sum_{i=1}^N V(\tilde{s}_i)p(\tilde{s}_i|s,a)], \tag{1.7}$$

where $\{\tilde{s}_1, \ldots, \tilde{s}_N\}$ is an *IID* random sample of points uniformly distributed over the d-dimensional state space $S = [0,1]^d$. A particularly nice feature of the random Bellman operator is that it is *self-approximating*, i.e. $\tilde{\Gamma}_N(V)(s)$ is a well-defined smooth function of $s \in S$ whenever u and p are smooth functions of s. Thus, one doesn't need to resort to auxilliary interpolation or approximation procedures to be able to evaluate $\tilde{\Gamma}(V)(s)$ at any point $s \in S$.

Unfortunately randomization cannot break the curse of dimensionality for the class of *continuous decision processes* (CDP's), i.e. MDP's with continuous choice sets $A(s)$. The reason is quite simple: since the general nonlinear programming problem is a special case of the MDP problem when $\beta = 0$, the general MDP problem must be at least as hard as the general (static) nonlinear programming problem. However as we noted above, the general nonlinear programming problem

is intractable regardless of whether deterministic or random algorithms are employed. In section 5.4 we consider whether there are certain subclasses of CDP's for which randomization might succeed in breaking the curse of dimensionality. One such class is the set of "convex CDP's" with convex choice sets $A(s)$ for each $s \in S$ and $u(s,a)$ and $p(s'|s,a)$ are concave functions of a for each pair $(s',s) \in S \times S$. An algorithm that might succeed in breaking the curse of dimensionality for this class of problems is a random multigrid algorithm using the random Bellman operator defined in (1.7), modified to use numerical optimization to find an approximate maximizer over the continuum of actions in $A(s)$. Due to the convex structure of the problem, there are polynomial-time numerical optimization methods that are guaranteed to find an ϵ-approximation to the maximizing element $a \in A(s)$.

The results presented so far show that discrete approximation methods have certain optimality properties (e.g. the Chow-Tsitsiklis 1991 multigrid algorithm is an almost optimal algorithm for continuous MDP problems, and Rust's 1994 random multigrid algorithm succeeds in breaking the curse of dimensionality for continuous MDP's with finite action sets). Do smooth approximation methods have any similar optimality properties? In order to analyze the convergence and complexity properties of smooth approximation methods we need to have a good understanding of the general *approximation problem:* what is the most efficient way to approximate a multivariate function V in some set B of functions using only a finite number N of (potentially noisy) observations of V at specific points in it's domain? Section 3.3 reviews some of the key results on error and complexity bounds for multivariate function approximation. Unfortunately these results show that all of the standard function approximation algorithms are subject to a curse of dimensionality, at least on worst case basis and for reasonably general classes of functions (such as the class B of Lipschitz continuous functions). Furthermore randomization does not succeed in breaking the curse of dimensionality of multivariate function approximation. Thus, the intractability of the embedded multivariate optimization and approximation problems turns out to be the Achilles heel of smooth approximation methods. If we want to design smooth approximation methods that are capable of breaking the curse of dimensionality we need to do a better job of exploiting the special structure of certain subclasses of MDP problems, or we must analyze the cost of these methods on an average instead of a worst case basis.

While complexity theory provides a useful general guide to the analysis and design of efficient numerical methods for MDP's, there are limits to its ability to make precise efficiency comparison of specific algorithms. Probably the most useful way to evaluate the performance of different methods is to compare their performance over a suite of test problems that are commonly

encountered in economic applications. While we provide comparisons of some of the standard solution methods for discrete MDP's in section 5.2.2 and discuss what we know about the actual performance of various methods wherever possible, we do not have sufficient space in this chapter to provide a rigorous comparison of the wide variety of different solution methods for a reasonably comprehensive set of multidimensional test problems. This is a task we leave to future work.

Sections 5.1 and 5.2 examine the question of whether massive parallel processing (MPP) technology can be effective in solving large scale discrete MDP's. Although many aspects of the dynamic programming procedure are inherently sequential, we have already noted that a major part of the work involved in solving finite horizon problems is the computation of the conditional expectation of next period's value function for each possible value of the current state s. This can be carried out in parallel, say, with $|S|$ separate processors computing the conditional expectations for each state $s = 1, \ldots, |S|$. However, Papadimitriou and Tsitsiklis 1987 argued that infinite horizon discrete MDP's cannot be *efficiently* parallelized by showing that the problem is *P-complete.* This means, roughly speaking, that the MDP problem is at least as difficult as any other polynomial-time problem such as linear programming. Polynomial-time problems that can be effectively parallelized are said to be in the class *NC:* formally, *NC* consists of all problems that can be solved in poly-log time (i.e. a polynomial of $\log(|S|)$, where $|S|$ indexes the size of the problem) using an expandable massive parallel processor whose processors equal some polynomial in $|S|$. The P-completeness of the MDP problem "means, intuitively, that such problems are as hard as any problem in P; thus, they can be massively parallelized only if *NC* $= P$, that is if *all* problems in P can, and thus there are no inherently sequential problems. Since this event is considered unlikely, *P*-completeness is taken as evidence that a problem is not *NC,* and thus cannot be satisfactorily paralellized." (Papadimitriou and Tsitsiklis, 1987, p. 442). We argue that this result is an artifact of the use of the Turing model of computation and the algebraic notion of computational complexity. We show that if we adopt the real model of computation (which assumes that computers can carry out on infinite precision arithmetic) and the continuous notion of computational complexity, then the MDP problem can be effectively parallelized using the Bellman/Howard policy iteration algorithm and the parallel Newton algorithm of Pan and Reif 1985 to carry out each policy evaluation step. The Pan-Reif algorithm approximately solves a system of $|S|$ linear equations in $|S|$ unknowns in $O(\log(|S|)^2)$ time using $|S|^\omega$ processors where $\omega \leq 2.376$ is the best current exponent on the complexity of matrix multiplication. [16] Although this result is currently only of theoretical interest since we do not yet have access to the truly massive parallelism that such an algorithm would require, we do

[16] The exponent ω provides an upper bound on the complexity of matrix multiplication: $comp(|S|) = O\left(|S|^{2.376}\right)$.

discuss an alternative algorithm known as the *Generalized Minimum Residual Algorithm* that is capable of producing fast and accurate solutions to extremely large scale systems using the more limited scale parallelism (including vector processing) that is currently available.

Our review of MDP's and numerical methods is necessarily selective given the space constraints of this chapter. To provide some perspective of how our review fits into the larger literature on numerical dynamic programming and stochastic control, it is useful to briefly review the main variations of control problems encountered in the literature:

- deterministic vs. stochastic
- Markovian vs. non-Markovian
- discrete vs. continuous time
- finite vs. infinite horizon
- linear/quadratic vs. general nonlinear problems
- discrete vs. continuous state
- discrete vs. continuous control
- discounted vs. long-run average rewards
- perfect vs. imperfect state information

This chapter focuses on stochastic control since deterministic control problems are a special case and can generally be effectively solved using the same algorithms developed for stochastic problems. [17] We also focus on Markovian decision processes with additively separable utility functionals. Although the method of dynamic programming can be extended in a straightforward manner to solve finite-horizon problems with non-Markovian uncertainty and general utility functionals (see Hinderer 1970 or Gihman and Skorohod, 1979), these methods require keeping track of complete histories which make them computationally intractable for solving all but the smallest problems. We also focus on MDP's formulated in discrete rather than continuous time. While there is an elegant theory of continuous time dynamic programming in which the value function V can be shown to be the solution to a system of partial differential equations known as the *Hamilton-Bellman-Jacobi* (HBJ) equation (see, e.g. Doshi, 1976, and Fleming and Soner, 1993), under very general conditions one can show that a continuous time MDP can be approximated arbitrary closely by a discrete time MDP when the time interval is sufficiently small (van Dijk, 1984). Indeed, the predominant solution method for solving approximate discrete time formulations of the MDP problem rather than attempting to numerically solve the HBJ equation, which frequently reduces to a

[17] This is not completely true. There is a sense in which deterministic MDP problems are inherently easier than stochastic MDP problems. In particular, in deterministic MDP's we don't have the problem of multivariate integration which is a large part of the work involved in solving stochastic MDP's. Papadimitriou and Tsitsiklis 1987 have shown that certain deterministic MDP's are in the class *NC* whereas the general stochastic MDP problem is P-complete, which is further theoretical evidence that deterministic problems are inherently easier.

nonlinear system of PDE's of the second order (see Kushner, 1990 and Kushner and Dupuis, 1992). [18] As we already noted, even systems of linear elliptic PDE's are intractable on a worst case basis, which may explain why the best approach to solving these problems is to solve an approximate discrete time MDP. In order to obtain good approximations, we need discrete time MDP's with very short time intervals Δt whose discount factors $\beta = e^{\Delta t}$ are very close to 1. However we can see from the Chow Tsitsiklis complexity bound (1.5) that the complexity of this problem tends to infinity as $\beta \to 1$. This provides an indication of the inherent difficulty of the continuous time MDP problem. A similar problem is encountered in solving MDP's under the long-run average rather than discounted criterion. Although there is a special theory for the solution of MDP problems under the long-run average reward criterion (see chapter 7 of Bertsekas, 1987), we focus on solving problems with discounted returns since a generalization of a classical theorem due to Abel (see Bhattacharya and Majumdar, 1989, or Dutta, 1989) shows that under weak regularity conditions $(1-\beta)V$ converges to the optimal long-run average reward as β tends to 1. This implies that we can approximate the stationary optimal decision rule under long-run average rewards by solving a discounted MDP with β close to 1.

We also focus on problems of perfect state information, since one can always reduce a problem with imperfect state information to a problem with perfect state information by treating the conditional distribution of the unobserved components of the state variable given the observed components as an additional state variable (see, e.g. Bertsekas, 1987). However while this Bayesian approach is conceptually simple, the reformulation falls prey to the curse of dimensionality since the new state variable in the reformulated problem is a conditional probability distribution, an element of an infinite-dimensional space. These problems are inherently much more difficult. For example Papadimitriou and Tsitsiklis 1987 have shown that discrete state MDP's with partially observed states are significantly harder problems than discrete MDP's with fully observed states. Unlike the fully observed MDP which is in the class P, Papadimitriou and Tsitsiklis showed that the partially observed MDP problem is *NP-complete.* [19] It is an open question whether these

[18] Chapter 9 of Fleming and Soner reviews closely related numerical methods based on solving finite-difference approximations to the HJB equations, which Kushner and Dupuis 1992 show are closely related to methods based on solving approximate discrete time MDP's. Semmler 1994 provides an interesting application of the time discretization approach that demonstrates the ability of discrete time models to approximate chaotic trajectories in the limiting continuous time model as the time interval tends to zero. See Tapiero and Sulem 1994 for a recent survey of numerical methods for continuous time stochastic control problems and Ortega and Voigt 1988 for a review of the literature on numerical methods for PDE's.

[19] The class *NP* consists of problems that can be solved in polynomial time using a "non-deterministic" computer. See section 3.1 for a more detailed description of this class of problems.

non-Bayesian learning algorithms can avoid the curse of dimensionality inherent in the Bayesian approach. The remaining subject divisions: finite vs. infinite horizon, discrete vs. continuous states and controls are all covered in separate sections of this chapter. Readers who are interested in efficient algorithms for solving LQ-MDP problems should consult chapter 5.

A final caveat is that the MDP framework involves the implicit assumption that decision-makers have time-separable preferences and are expected utility maximizers. Experimental tests of human decision-making show that neither of these assumptions may be valid (e.g. the famous "Allais paradox", see Machina, 1987). Recently, a new theory of sequential decision-making has emerged that allows for time and state non-separable preferences (see Epstein and Zin, 1989, and Ozaki and Streufert, 1992), and it's possible that this generalized dynamic choice theory could result in more realistic economic models. Apart from some special cases that have been solved numerically (e.g. Hansen and Sargent, 1993) there is virtually no theory or well developed numerical procedure for solving general versions of these problems. However these problems have a recursive structure, so it is quite likely that many of the methods outlined in this chapter can ultimately be adapted to compute approximate solutions to these more general sequential decision processes (SDP's). We content ourselves with the observation, formalized in Rust (1994a), that the class of MDP's is already sufficiently broad to enable one to generate virtually any type of decision rule (behavior) via an appropriate specification of preferences u and law of motion p.

2. MDP's and the Theory of Dynamic Programming: A Brief Review

This section reviews the main theoretical results on dynamic programming in finite and infinite horizon problems. Readers who are already familiar with this theory may simply wish to skim over this section to determine the notation we are using, and skip directly to the presentation of numerical methods in sections 3, 4 and 5. We focus on time stationary problems for notational simplicity: it will become clear that in finite horizon problems the utility and transition probability functions can be general functions of time t without affecting any of our results.

2.1 Definitions of MDP's, DDP's and CDP's

Definition 2.1: *A (discrete-time)* ***Markovian Decision Process (MDP)*** *consists of the following objects:*

- A *time index* $t \in \{0, 1, 2, \ldots, T\},\ T \leq \infty$
- A *state space* S
- An *action space* A
- A family of *constraint sets* $\{A(s) \subseteq A\}$
- A *transition probability* $p(ds'|s, a) = Pr\{s_{t+1} \in ds'|s_t = s, a_t = a\}$.
- A *discount factor* $\beta \in (0, 1)$.
- A *single period utility function* $u(s, a)$ such that the utility functional U_T has the additively separable decomposition: [20]

$$U_T(\mathbf{s}, \mathbf{d}) = \sum_{t=0}^{T} \beta^t u(s_t, a_t). \tag{2.1}$$

The agent's optimization problem is to choose an optimal decision rule $\alpha = (\alpha_0, \ldots, \alpha_T)$ to solve the following problem:

$$\max_{\alpha=(\alpha_0,\ldots,\alpha_T)} E_\alpha\{U_T(\mathbf{s}, \mathbf{d})\} \equiv \int_{s_0} \cdots \int_{s_T} \left[\sum_{t=0}^{T} \beta^t u(s_t, \alpha_t(s_t))\right] \prod_{t=1}^{T} p(ds_t|s_{t-1}, \alpha_{t-1}(s_{t-1}))p_0(ds_0), \tag{2.2}$$

[20] The boldface notation denotes sequences: $\mathbf{s} = (s_0, \ldots, s_T)$. We will subsequently impose explicit topological structure on S and A and smoothness conditions on u and p later in this section and in sections 4 and 5.

where p_0 is a probability distribution over the initial state s_0. Stated in this form, the optimization problem is extremely daunting. We must search over an infi nite-dimensional space of sequences of potentially history-dependent functions $(\alpha_0, \ldots, \alpha_T)$, and each evaluation of the objective function in (2.2) requires $(T+1)$-fold multivariate integration. We now show how dynamic programming can be used to vastly simplify this potentially intractable optimization problem. Before doing this, we distinguish two important subclasses of MDP's known as DDP's and CDP's:

Definition 2.2: *A* ***Discrete Decision Process (DDP)*** *is an MDP with the following property:*

- There is a fi nite set A such that $A(s) \subset A$ for each $s \in S$.

For simplicity, section 5 will make the further assumption that $A(s) = A$ for all $s \in S$. This apparent restriction actually does not involve any loss of generality, since we can mimic the outcome of a problem with state-dependent choice sets $A(s)$ by a problem with a state-independent choice set A by choosing the utility function $u(s,a)$ so that the utility of any "infeasible" action $a \in A \cap A(s)^c$ is so low that it will in fact never be chosen.

Definition 2.3: *A* ***Continuous Decision Process (CDP)*** *is an MDP with the following property:*

- For each $s \in S$, the action set $A(s)$ is a compact subset of R^{d_a} with non-empty interior.

2.2 Bellman's Equation, Contraction Mappings, and Blackwell's Theorem

In the fi nite-horizon case $(T < \infty)$, dynamic programming amounts to simple backward induction. In the terminal period V_T and α_T are defi ned by:

$$\alpha_T(s_T) = \underset{a_T \in A(s_T)}{argmax}\,[u(s_T, a_T)], \tag{2.3}$$

$$V_T(s_T) = \underset{a_T \in A(s_T)}{max}\,[u(s_T, a_T)]. \tag{2.4}$$

In periods $t = 0, \ldots, T-1$, V_t and α_t are defi ned by the recursions:

$$\alpha_t(s_t) = \underset{a_t \in A(s_t)}{argmax}[u(s_t, a_t) + \beta \int V_{t+1}(s_{t+1}) p(ds_{t+1}|s_t, a_t)], \tag{2.5}$$

$$V_t(s_t) = \underset{a_t \in A(s_t)}{max}[u(s_t, a_t) + \beta \int V_{t+1}(s_{t+1}) p(ds_{t+1}|s_t, a_t)]. \tag{2.6}$$

It's straightforward to verify that at time $t = 0$ the value function $V_0(s_0)$ represents the maximized expected discounted value of utility in all future periods. Since dynamic programming has recursively generated the optimal decision rule $\alpha = (\alpha_0, \ldots, \alpha_T)$, it follows that

$$V_0(s) = \max_{\alpha} E_\alpha\{U_T(\tilde{\mathbf{s}}, \tilde{\mathbf{d}})|s_0 = s\}. \tag{2.7}$$

In the infinite horizon case $T = \infty$ there is no "last" period from which to start the backward induction to carry out the dynamic programming algorithm. However if the per period utility function u is uniformly bounded and the discount factor β is in the $(0, 1)$ interval, then we can approximate the infinite horizon utility functional $U_\infty(\mathbf{s}, \mathbf{d})$ arbitrarily closely by a finite horizon utility functional $U_T(\mathbf{s}, \mathbf{d})$ for T sufficiently large. This is a basic idea underlying many numerical methods for solving infinite horizon MDP's such as successive approximations.

Almost all infinite-horizon MDP's formulated in economic applications have the further characteristic of *stationarity:* i.e. the transition probabilities and utility functions are the same for all t. In the finite horizon case the time homogeneity of u and p does not lead to any significant simplifications since there still is a fundamental nonstationarity induced by the fact that the remaining utility $\sum_{j=t}^{T} \beta^j u(s_j, a_j)$ depends on t. However in the infinite-horizon case, the stationary Markovian structure of the problem implies that the future looks the same whether the agent is in state s_t at time t or in state s_{t+k} at time $t + k$ provided that $s_t = s_{t+k}$. In other words, the only variable that affects the agent's view about the future is the value of his current state s. This suggests that the optimal decision rule and corresponding value function should be time invariant. Removing time subscripts from the recursion (2.5), we obtain the following equations characterizing the optimal stationary decision rule α and value function V:

$$\alpha(s) = \underset{a \in A(s)}{argmax}[u(s, a) + \beta \int V(s')p(ds'|s, a)], \tag{2.8}$$

where V is the solution to:

$$V(s) = \max_{a \in A(s)} [u(s, a) + \beta \int V(s')p(ds'|s, a)]. \tag{2.9}$$

Equation (2.9) is known as *Bellman's Equation.*[21] In mathematical terms Bellman's equation is a *functional equation* and the value function V is a *fixed point* to this functional equation.

[21] Bellman was not the first to discover this equation (for example versions of it appear in prior work by Arrow *et. al.* 1951 on optimal inventory policy), however the equation bears his name due to Bellman's systematic application of the approach to solving a wide variety of problems.

To establish the existence and uniqueness of a solution V to Bellman's equation, assume for the moment the following regularity conditions hold: 1) S and A are complete metric spaces, 2) $u(s,a)$ is jointly continuous and bounded in (s,a), 3) $s \to A(s)$ is a continuous correspondence. Let $\mathcal{B}(S)$ denote the Banach space of all measurable, bounded functions $f\colon S \to R$ under the (essential) supremum norm, $\|f\| = \sup_{s\in S} |f(s)|$. Define the *Bellman operator* $\Gamma\colon \mathcal{B}(S) \to \mathcal{B}(S)$ by:

$$\Gamma(W)(s) = \max_{a\in A(s)} [u(s,a) + \beta \int W(s')p(ds'|s,a)]. \tag{2.10}$$

Bellman's equation can then be re-written in operator notation as:

$$V = \Gamma(V), \tag{2.11}$$

i.e. V is a fixed point of the mapping Γ. Denardo 1967 noted that the Bellman operator has a particularly nice property: it is a *contraction mapping.* This means that for any $V, W \in B$ we have:

$$\|\Gamma(V) - \Gamma(W)\| \leq \beta \|V - W\| \tag{2.12}$$

The theory of contraction mappings allows us to establish the existence and uniqueness of the solution V to Bellman's equation. In addition, since the theory of contraction mappings provides the basis for many of the solution methods and error bounds developed in sections 4 and 5, we repeat the statement and proof of the well known contraction mapping theorem below.

Contraction Mapping Theorem: *If Γ is a contraction mapping on a Banach Space B, then Γ has a unique fixed point $V \in B$.*

The uniqueness of the fixed point is a direct consequence of the contraction property (2.12): if W and V are fixed points to Γ then (2.12) implies that

$$\|V - W\| = \|\Gamma(V) - \Gamma(W)\| \leq \beta\|V - W\|. \tag{2.13}$$

Since $\beta \in (0,1)$ then only possible solution to (2.13) is $\|V - W\| = 0$. The existence of a fixed point is a result of the completeness of the Banach space B. Starting from any initial element of B (such as 0), the contraction property (2.12) implies that the following sequence of successive approximations forms a Cauchy sequence in B:

$$\{0, \Gamma(0), \Gamma^2(0), \Gamma^3(0), \dots, \Gamma^n(0), \dots\}. \tag{2.14}$$

Since the Banach space B is complete, the Cauchy sequence converges to a point $V \in B$, so existence follows by showing that V is a fixed point of Γ. To see this, note that a contraction Γ is uniformly continuous, so:

$$V = \lim_{n\to\infty} \Gamma^n(0) = \lim_{n\to\infty} \Gamma(\Gamma^{n-1}(0)) = \Gamma(V), \tag{2.15}$$

i.e., V is indeed the required fixed point of Γ. The next theorem shows that the stationary decision rule defined from V in equation (2.8) is an optimal decision rule.

Blackwell's Theorem: *The stationary, Markovian, infinite-horizon policy given by $\alpha = (\alpha, \alpha, \dots)$ where α is defined in (2.8) and (2.9) constitutes an optimal decision rule for the infinite-horizon MDP problem (2.2).*

This result follows by showing that the unique solution V to Bellman's equation coincides with the optimal value function V_∞ defined in (2.7), which we restate below for convenience:

$$V_\infty(s) \equiv \max_{\alpha} \; E_\alpha\Big\{\sum_{t=0}^{\infty} \beta^t u(s_t, a_t)\Big| s_0 = s\Big\}. \tag{2.16}$$

Consider approximating the infinite-horizon problem by the solution to a finite-horizon problem with value function:

$$V_T(s) = \max_{\alpha} \; E_\alpha\Big\{\sum_{t=0}^{T} \beta^t u(s_t, a_t)\Big| s_0 = s\Big\}. \tag{2.17}$$

Since u is bounded and continuous, $\sum_{t=0}^{T} \beta^t u(s_t, a_t)$ converges to $\sum_{t=0}^{\infty} \beta^t u(s_t, a_t)$ for any sequences $\mathbf{s} = (s_0, s_1, \dots)$ and $\mathbf{a} = (a_0, a_1, \dots)$. Using the Lebesgue dominated convergence theorem and properties of suprema, one can show that for each $s \in S$, $V_T(s)$ converges to the infinite-horizon value function $V_\infty(s)$:

$$\lim_{T\to\infty} V_T(s) = V_\infty(s) \qquad \forall s \in S. \tag{2.18}$$

But the contraction mapping theorem also implies that this same sequence converges uniformly to V (since $V_T = \Gamma^T(0)$), so it follows that $V = V_\infty$. Since V is the expected present discounted value of utility under the policy α (a result we demonstrate in the next subsection), the fact that $V = V_\infty$ immediately implies the optimality of the stationary infinite horizon decision rule α.

The optimality of the infinite horizon policy α can be proved under weaker conditions that allow $u(s, a)$ to be an unbounded, upper semicontinuous function of s and a, see Bhattarcharya and Majumdar, 1989. Although problems with unbounded state spaces S, decision spaces A, and

utility functions u arise frequently in economic applications, in practice most of the computational methods presented in section 5 require bounded domains and utility functions. We can approximate the solution to a problem with unbounded state space, decision space, and utility function via a sequence of bounded problems where the bounds tend to infinity. Furthermore, the solution to problems with upper-semicontinuous utility functions can be approximated as the limit of a sequence of solutions to problems with continuous utility functions, see Gihman and Skorohod 1979, Lemma 1.8.

2.3 Error Bounds for Approximate Fixed Points to Approximate Bellman Operators

In problems where S contains an infinite number of possible states the value function V is an element of the infinite dimensional Banach space $\mathcal{B}(S)$, so it can generally only be approximated. A general strategy for approximating the fixed point $V = \Gamma(V)$ in $\mathcal{B}(S)$ is to compute the fixed point $V_N = \Gamma_N(V_N)$ of an approximate Bellman operator $\Gamma_N : B_N \to B_N$, where B_N is a finite dimensional subspace of $\mathcal{B}(S)$. The following lemmas provide error bounds that are the key to proving the convergence of these approximation methods.

Lemma 2.1:. *Suppose $\{\Gamma_N\}$ are a family of contraction mappings on a Banach space B indexed by N that that converge pointwise: i.e. $\forall W \in B$ we have*

$$\lim_{N\to\infty} \Gamma_N(W) = \Gamma(W). \tag{2.19}$$

Then $\lim_{N\to\infty} \|V_N - V\| = 0$ since the approximate fixed point $V_N = \Gamma_N(V_N)$ satisfies the error bound:

$$\|V_N - V\| \leq \frac{\|\Gamma_N(V) - \Gamma(V)\|}{(1-\beta)}. \tag{2.20}$$

The proof of this lemma is a simple application of the triangle inequality:

$$\begin{aligned} \|V_N - V\| &= \|\Gamma_N(V_N) - \Gamma(V)\| \\ &\leq \|\Gamma_N(V_N) - \Gamma_N(V)\| + \|\Gamma_N(V) - \Gamma(V)\| \\ &\leq \beta\|V_N - V\| + \|\Gamma_N(V) - \Gamma(V)\|. \end{aligned} \tag{2.21}$$

The next problem is that even if there was no need to approximate the Bellman operator Γ, we generally never compute its exact fixed point $V = \Gamma(V)$, but rather only an approximation to this fixed point. The standard method for approximating fixed points to contraction mappings is the method of successive approximations starting from an arbitrary element $W \in B$. After t successive approximations steps our estimate of the fixed point is given by $\Gamma^t(W)$. The following Lemma provides key error bounds between $\Gamma^t(W)$ and the true fixed point V:

Lemma 2.2:. *Let Γ be a contraction mapping on a Banach space B with fixed point $V = \Gamma(V)$. If W is an arbitrary element of B the following inequalities hold:*

$$\|W - \Gamma(W)\| \leq (1+\beta)\|V - W\| \tag{2.22}$$

$$\|\Gamma^{t+1}(W) - V\| \leq \beta\|\Gamma^t(W) - V\|. \tag{2.23}$$

$$\|\Gamma^t(W) - V\| \leq \beta^t\|\Gamma(W) - W\|/(1-\beta) \tag{2.24}$$

Error bound (2.22) shows that if W is close to the fixed point V then W must also be close to $\Gamma(W)$. Error bound (2.24), known as an *a priori* error bound, shows the converse result: the maximum error in a sequence of successive approximations $\{\Gamma^t(W)\}$ starting from W is a geometrically declining function of the initial error $\|V - \Gamma(W)\|$. These two inequalities will be used to establish the convergence of the various parametric approximation methods presented in section 4. Inequality (2.23) is referred to as an *a posteriori* error bound: it enables us to bound the maximum distance of $\Gamma^{t+1}(W)$ to the fixed point V in terms of the (observable) change in the value function from iteration t to $t+1$. We state this bound for completeness, since special case where Γ is the Bellman operator one can derive a sharper *a posteriori* error bound known as the *McQueen-Porteus error bound* in section 5.2. The final lemma is a useful result that implies an *a priori* bound on the magnitude of the value function V.

Lemma 2.3: *Let Γ be a Bellman operator. Then we have:*

$$\Gamma : B\left(0, \frac{K}{(1-\beta)}\right) \to B\left(0, \frac{K}{(1-\beta)}\right), \tag{2.25}$$

where $B(0,r) = \{V \in B \Big| \ \|V\| \leq r\}$ and the constant K is given by:

$$K \equiv \sup_{s \in S} \sup_{a \in A(s)} |u(s,a)|. \tag{2.26}$$

2.4 A Geometric Series Representation for Infinite Horizon MDP's

Infinite-horizon MDP's have a simple algebraic structure that can be interpreted as a multidimensional generalization of an ordinary geometric series. [22] Since several of the numerical methods presented in section 5 directly exploit this algebraic structure, particularly the method of *policy iteration*, we provide a brief review here. To avoid technicalities, we first present the basic results for the case of finite MDP's where, without loss of generality, both the state space S and the decision space A can be identified as finite sets of integers, $S = \{1, \ldots, |S|\}$ and $A(s) = \{1, \ldots, |A(s)|\}$. In this framework a feasible stationary decision rule α is an $|S|$-dimensional vector satisfying $\alpha(s) \in \{1, \ldots, |A(s)|\}, s = 1, \ldots, |S|$, and the value function V is an $|S|$-dimensional vector in the Euclidean space $R^{|S|}$. Given α we can define a vector $u_\alpha \in R^{|S|}$ whose i^{th} component is $u(i, \alpha(i))$, and an $|S| \times |S|$ *transition probability matrix* M_α whose (i, j) element is $p(j|i, \alpha(i)) = Pr\{s_{t+1} = j | s_t = i, a_t = \alpha(i)\}$. Given a stationary, Markovian decision rule $\alpha = (\alpha, \alpha, \ldots)$, define the associated value function $V_\alpha \in R^{|S|}$ as the vector of expected discounted utilities under policy α. It is straightforward to show that V_α is the solution to the following system of $|S|$ linear equations in $|S|$ unknowns:

$$V_\alpha = G_\alpha(V_\alpha) \equiv u_\alpha + \beta M_\alpha V_\alpha. \tag{2.27}$$

It turns out that this equation can always be solved by simple matrix inversion:

$$\begin{aligned} V_\alpha &= [I - \beta M_\alpha]^{-1} u_\alpha \\ &= u_\alpha + \beta M_\alpha u_\alpha + \beta^2 M_\alpha^2 u_\alpha + \beta^3 M_\alpha^3 u_\alpha + \cdots \end{aligned} \tag{2.28}$$

The last equation in (2.28) is simply a geometric series expansion for V_α in powers of β and M_α. As is well known $M_\alpha^k = (M_\alpha)^k$ is simply the *k-stage transition probability matrix*, whose (i, j) element equals $Pr\{s_{t+k} = j | s_t = i, \alpha\}$, where the presence of α as a conditioning argument denotes the fact that all intervening decisions satisfy $a_{t+j} = \alpha(s_{t+j}), j = 0, \ldots, k$. Since $\beta^k M_\alpha^k u_\alpha$ is the expected discounted utility received in period k under policy α, formula (2.28) can be thought of as a vector generalization of a geometric series, showing explicitly how V_α equals the sum of expected discounted utilities under α in all future periods. Note that since M_α^k is a transition probability matrix (i.e. all of its elements are between 0 and 1, and its rows sum to unity), it follows that $\lim_{k \to \infty} \beta^k M_\alpha^k = 0$, which guarantees that the geometric series representation in (2.28) is convergent and therefore that the $|S| \times |S|$ matrix $[I - \beta M_\alpha]$ is invertible for any Markovian

[22] This observation may have prompted Lucas's 1978 observation that "a little knowledge of geometric series goes a long way." (p. 1443).

decision rule α and all $\beta \in [0,1)$. Economists will recognize that this same algebraic structure appears in input-output theory (Leontief, 1966) and mathematicians will recognize this as a special case of the more general theory of *M-matrices*.[23]

In fact, the basic geometric series representation for V_α in (2.28) also holds in problems where the state and decision spaces are infinite sets, in which case mathematicians refer to the series expansion (2.28) as a *Neumann series* instead of a geometric series. When the state space S has infinitely many elements M_α is no longer represented by a transition probability matrix, but by a special kind of linear operator known as a *Markov operator* given by:[24]

$$M_\alpha(V)(s) = \int V(s')p(ds'|s,\alpha(s)) \tag{2.29}$$

Using the standard definition of a norm $\|L\|$ of a linear operator $L : B \to B$,

$$\|L\| = \sup_{V \neq 0} \frac{\|L(V)\|}{\|V\|}, \tag{2.30}$$

it is straightforward to verify that the linear operator M_α has unit norm, $\|M_\alpha\| = 1$. It is also easy to verify that the operator $M_\alpha^2 \equiv M_\alpha(M_\alpha)$ is also a Markov operator. By induction, it follows that $M_\alpha^k = M_\alpha(M_\alpha^{k-1})$ is also Markov operator. Furthermore using the definition of the operator norm it is easy to see that

$$\|M_\alpha^k\| \leq \|M_\alpha\|^k, \tag{2.31}$$

which implies that the Neumann series expansion in equation (2.28) is convergent. Banach's Theorem (Theorem 4.4.3 of Kantorovich and Aikilov, 1982) then implies that the inverse operator $[I - \beta M_\alpha]^{-1}$ exists and is a bounded linear operator on B. Thus, the value function V_α is well-defined even when S is an infinite set, a result that also follows directly from the fact that the operator G_α defined in (2.27) is a contraction mapping.

In section 5 we will establish error bounds and prove the convergence of discretization methods as the number of states S used in the finite approximations tend to infinity. The basic idea is to approximate the Markov operator M_α on an infinite-dimensional function space B by an $|S| \times |S|$ transition probability matrix $\hat{M}_\alpha$ on the ordinary Euclidean space $R^{|S|}$, and show that this implies that the true solution $V_\alpha \in B$ from equation (2.28) can be approximated by appropriately interpolating the approximate finite-dimensional solution $\hat{V}_\alpha \in R^{|S|}$.

---

[23] See Berman and Plemmons, 1993 for a general characterization of necessary and sufficient conditions for matrices of the form $I - A$ to be invertible. These theorems subsume the Hawkins-Simon 1949 condition that guarantees that $\lim_{n\to\infty} A^n = 0$ provided that certain conditions on the principal minors of A are satisfied.

[24] If the state space consists of a countably infinite number of points, then M_α can be represented as an infinite matrix.

2.5 Examples of Analytic Solutions to Bellman's Equation for Specific "Test Problems"

We now provide several concrete examples of MDP's that arise in typical economic problems, showing how the theory in the previous sections can be used to solve them. Although the underlying thesis of this chapter is that analytical solutions to DP problems are rare and non-robust (in the sense that small perturbations of the problem formulation leads to a problem with no analytic solution), we present analytical solutions in order to provide a "test bed" of problems to compare the accuracy and speed of various numerical solution methods presented in section 5. Examples 2 and 3 provide examples of a CDP and a DDP, respectively. Since CDP's require optimization over a continuum of possible decisions $a \in A(s)$, they are typically intrinsically more difficult to solve than DDP's. In order to differentiate these two cases we will use the notation $a \in A(s)$ to denote the case of discrete choice (where $A(s)$ contains a finite or countably infinite number of possible values), and $c \in A(s)$ to denote the case of continuous choice (where $A(s)$ contains a continuum of possible values, such as a convex subset of Euclidean space).

Example 1: A trivial problem. Consider a problem where $u(s,a) = 1$ for all $a \in A(s)$ and all $s \in S$. Given that the utility function is a constant, it is reasonable to conjecture that V is a constant also. Substituting this conjecture into Bellman's equation we obtain:

$$V = \max_{a \in A(s)} [1 + \beta \int V p(ds'|s,a)], \tag{2.32}$$

the unique solution to which is easily seen to be $V = 1/(1-\beta)$. This is the well known formula for a geometric series, $V = [1 + \beta + \beta^2 + \cdots]$ which is clearly equal to expected utility in this case since u is identically equal to 1. This provides a simple and basic test of any solution method for infinite horizon MDP's: the method should return a value function identically equal to $1/(1-\beta)$ whenever we solve the problem with a utility function that is identically 1.

Example 2: A problem with continuous state and control variables. Consider the problem of optimal consumption and savings analyzed by Phelps (1962). In this case the state variable s denotes a consumer's current wealth, and the decision a is how much to consume in the current period. Since consumption is a continuous decision, we will use c_t rather than a_t to denote the value of the control variable, and let w_t denote wealth at time t. The consumer is allowed to save, but is not allowed to borrow against future income. Thus, the constraint set is $A(w) = \{c \,|\, 0 \leq c \leq w\}$. The consumer can invest his savings in a single risky asset with random rate of return $\{\tilde{R}_t\}$ is an *IID* process (i.e. independently and identically distributed over time) with marginal distribution F. Thus there is a two-dimensional state vector for this problem, $s_t = (w_t, R_t)$, although since R_t

is *IID* it is easy to see that w_t is the only relevant state variable entering the value function. Let the consumer's utility function be given by $u(w,c) = \log(c)$. Then Bellman's equation for this problem is given by:

$$V(w) = \max_{0 \le c \le w} [\log(c) + \beta \int_0^\infty V(R(w-c))F(dR)] \tag{2.33}$$

Working backward from an initial conjecture $V = 0$ we see that at each stage $V_t = \Gamma^t(0)$ has the form, $V_t(w) = f_t \log(w) + g_t$ for constants f_t and g_t. Thus, it is reasonable to conjecture that this form holds in the limit as well. Inserting the conjectured functional form $V(w) = f_\infty \log(w) + g_\infty$ into (2.33) and solving for the unknown coeffi cients f_∞ and g_∞ we fi nd:

$$\begin{aligned} f_\infty &= 1/(1-\beta) \\ g_\infty &= \log(1-\beta)/(1-\beta) + \beta \log(\beta)/(1-\beta)^2 + \beta E\{\log(\tilde{R})\}/(1-\beta)^2, \end{aligned} \tag{2.34}$$

and the optimal decision rule or *consumption function* is given by:

$$\alpha(w) = (1-\beta)w. \tag{2.35}$$

Thus, the logarithmic specifi cation implies that a strong form of the *permanent income hypothesis* holds in which optimal consumption is a constant fraction of current wealth independent of the distribution F of investment returns.

Example 3: A problem with discrete control and continuous state variable. Consider the problem of optimal replacement of durable assets analyzed in Rust (1985,1986). In this case the state space $S = R_+$, where s_t is interpreted as a measure of the accumulated utilization of the durable (such as the odometer reading on a car). Thus $s_t = 0$ denotes a brand new durable good. At each time t there are two possible decisions {keep,replace} corresponding to the binary constraint set $A(s) = \{0,1\}$ where $a_t = 1$ corresponds to selling the existing durable for scrap price $\underline{P}$ and replacing it with a new durable at cost $\overline{P}$. Suppose the level of utilization of the asset each period has an exogenous exponential distribution. This corresponds to a transition probability p is given by:

$$p(ds_{t+1}|s_t, a_t) = \begin{cases} 1 - \exp\{-\lambda(ds_{t+1} - s_t)\} & \text{if } a_t = 0 \text{ and } s_{t+1} \ge s_t \\ 1 - \exp\{-\lambda(ds_{t+1} - 0)\} & \text{if } a_t = 1 \text{ and } s_{t+1} \ge 0 \\ 0 & \textit{otherwise.} \end{cases} \tag{2.36}$$

Assume the per-period cost of operating the asset in state s is given by a function $c(s)$ and that the objective is to fi nd an optimal replacement policy to minimize the expected discounted costs

of owning the durable over an infinite horizon. Since minimizing a function is equivalent to maximizing its negative, we can define the utility function by:

$$u(s_t, a_t) = \begin{cases} -c(s_t) & \text{if } a_t = 0 \\ -[\overline{P} - \underline{P}] - c(0) & \text{if } a_t = 1. \end{cases} \tag{2.37}$$

Bellman's equation takes the form:

$$\begin{aligned} V(s) = max\Big[-c(s) + \beta \int_s^\infty V(s')\lambda \exp\{-\lambda(s'-s)\}ds', \\ -[\overline{P} - \underline{P}] - c(0) + \beta \int_0^\infty V(s')\lambda \exp\{-\lambda(s')\}ds'\Big]. \end{aligned} \tag{2.38}$$

Observe that V is a non-increasing, continuous function of s and that the second term on the right hand side of (2.38), the value of replacing the durable, is a constant independent of s. Note also that $\overline{P} > \underline{P}$ implies that it is never optimal to replace a brand-new durable $s = 0$. Let γ be the smallest value of s such that the agent is indifferent between keeping and replacing. Differentiating Bellman's equation (2.38), it follows that on the *continuation region*, $[0, \gamma)$, $\alpha(s) = 0$ (i.e. keep the current durable) and V satisfies the differential equation:

$$V'(s) = -c'(s) + \lambda c(s) + \lambda(1-\beta)V(s). \tag{2.39}$$

This is known as a *free boundary value problem* since the boundary condition:

$$V(\gamma) = [\overline{P} - \underline{P}] + V(0) = -c(\gamma) + \beta V(\gamma) = \frac{-c(\gamma)}{1-\beta}, \tag{2.40}$$

is determined endogenously. Equation (2.39) is a linear first order differential equation that can be integrated to yield the following closed-form solution for V:

$$V(s) = max\Big[\frac{-c(\gamma)}{1-\beta}, \frac{-c(\gamma)}{1-\beta} + \int_s^\gamma \frac{c'(y)}{1-\beta}[1 - \beta e^{-\lambda(1-\beta)(y-s)}]dy\Big], \tag{2.41}$$

where γ is the unique solution to:

$$[\overline{P} - \underline{P}] = \int_0^\gamma \frac{c'(y)}{1-\beta}[1 - \beta e^{-\lambda(1-\beta)y}]dy. \tag{2.42}$$

2.6 Euler Equations and Euler Operators

There is a subclass of CDP's for which we can effectively bypass solving Bellman's equation and derive a first order necessary condition characterizing the optimal decision rule α that doesn't depend on the value function V. Using the envelope theorem, we can transform this first order condition into a functional equation for α known as the *Euler equation.* The following definition characterizes a class of CDP's for which Euler equations can be derived.

Definition 2.4: *A class of CDP's is an* ***Euler class*** *if the following conditions hold:*[25]

[25] This is a slight generalization of a class of problems analyzed in chapters 9 and 10 of Stokey and Lucas, 1989.

- The action space A is a subset of R^M.
- The state space S is the product space $S = Y \times Z$, where Y is a closed subset of R^J and Z is a closed subset of R^K.
- $A(s) = A(y, z)$ is an upper hemicontinuous correspondence which is increasing in its first argument: $y \leq y' \Rightarrow A(y, z) \subset A(y', z)$.
- $A(\cdot, \cdot)$ is convex in its first argument: i.e. for all $z \in Z$ and all $y, y' \in A$ if $c \in A(y, z)$ and $c' \in A(y', z)$ then $\theta c + (1 - \theta)c' \in A(\theta y + (1 - \theta)y', z)$ for all $\theta \in [0, 1]$.
- The transition probability $p(ds_{t+1}|s_t, c_t)$ factors as:

$$p(ds_{t+1}|s_t, c_t) = p(dy_{t+1}, dz_{t+1}|y_t, z_t, c_t) = I\{dy_{t+1} = r(y_t, c_t, z_{t+1}, z_t)\}q(dz_{t+1}|z_t). \tag{2.43}$$

  where $q(\cdot|z)$ is a weakly continuous function of z, and r is continuously differentiable.
- For each (y, c, z', z) there exists an $M \times J$ matrix $h(y, c, z)$ satisfying:

$$\frac{\partial r(y, c, z', z)}{\partial y} = \frac{\partial r(y, c, z', z)}{\partial c} h(y, c, z). \tag{2.44}$$

- The utility function $u(s, c) = u(y, z, c)$ is a concave function of its first and third arguments for each $z \in Z$.

The interpretation of this class of problems is that the state variable $s_t = (y_t, z_t)$ consists of two components: an "endogenous" state variable y_t and an "exogenous" state variable z_t. The exogenous state variable is so named because its law of motion $q(dz_{t+1}|z_t)$ is unaffected by the agent's action c_t. The exogenous state variable y_t is affected by the agent's action but in a particular way: with probability 1 the value of y_{t+1} is given by $y_{t+1} = r(y_t, c_t, z_{t+1}, z_t)$. Bellman's equation for this class of problems is given by: [26]

$$V(s) = V(y, z) = \max_{c \in A(y,z)} [u(y, z, c) + \beta \int V(r(y, c, z', z), z')q(dz'|z)]. \tag{2.45}$$

[26] This generalizes the MDP framework in Stokey and Lucas (1989) which assumes that $y_{t+1} = c_t$ with probability 1.

Lemma 2.4:. *Let α be the optimal decision rule and V be the value function for the Euler class of CDP's given in Definition 2.4 and equation (2.45). If $y \in int\, Y$ and $\alpha(y,z) \in int\, A(y,z)$, and if $V(y,z)$ is a continuously differentiable function of y at the point (y,z), then its derivative is given by:*

$$\frac{\partial V(y,z)}{\partial y} = \frac{\partial u(y,z,\alpha(y,z))}{\partial y} - \frac{\partial u(y,z,\alpha(y,z))}{\partial c}\, h(y,\alpha(y,z),z). \tag{2.46}$$

The proof of the Lemma 2.4 is a straightforward extension of the proof of Theorem 9.10 of Stokey and Lucas (1989). If equation (2.46) holds for *all* points $(y,z) \in Y \times Z$, we obtain the following general form of the *stochastic Euler equation:*

$$0 = \frac{\partial u(y,z,c)}{\partial c} + \beta \int \left[\frac{\partial u(y',z',c')}{\partial y} - \frac{\partial u(y',z',c')}{\partial c}\, h(y',c',z') \right] \frac{\partial r(y',c,z',z)}{\partial c}\, q(dz'|z), \tag{2.47}$$

where $c = \alpha(y,z)$, $y' = r(y,\alpha(y,z),z',z)$, and $c' = \alpha(y',z')$. The Euler equation is simply a first order necessary condition characterizing the optimal decision rule $\alpha(y,z)$. It says that the change in expected discounted expected utility from a small change in c will be zero when $c = \alpha(y,z)$. The remarkable feature of the first order condition for this class of problems is that the impact on expected discounted utility can be evaluated only in terms of the value of marginal utility in period t and the expected discounted value of marginal utility in period $t+1$: since V drops out of the first order condition it is not necessary to consider the impact on marginal utility on all future periods $t+2, t+3, \ldots$

Notice that (2.47) is a recursive formula characterizing the optimal decision rule α, similar to the recursive characterization of V in Bellman's equation (2.45). Just as Bellman's equation defines V as the fixed point to the Bellman operator Γ, the Euler equation defines the optimal decision rule α as the fixed point to an operator we call the *Euler operator* Ψ. Following Coleman 1990, 1991, we define $\Psi(\alpha)(y,z)$ pointwise as the solution to:

$$0 = u_c(y,z,\Psi(\alpha)(y,z)) + \beta \int F\left(z', z, r(y,\Psi(\alpha)(y,z),z',z), y, \alpha(r(y,\Psi(\alpha)(y,z),z',z),z'), \alpha(y,z)\right) q(dz'|z), \tag{2.48}$$

where the integrand $F(z',z,y',y,c',c)$ is defined by:

$$F(z',z,y',y,c',c) = \left[\frac{\partial u}{\partial y}(y',z',c') - \frac{\partial u}{\partial c}(y',z',c') h(y',c',z') \right] \frac{\partial r}{\partial c}(y',c,z',z). \tag{2.49}$$

It is not immediately apparent, however, that this particular functional equation is any simpler to solve than the functional equation (2.45) characterizing V. Indeed, even though under certain

conditions one can prove that the Euler equation has a unique solution (Coleman, 1991), the Euler operator Ψ is generally is generally not a contraction mapping. Another restriction that limits the applicability of Euler equation methods is the assumption that $\alpha(c,z)$ is an interior point of the constraint set. In cases when the maximizing value of c in (2.45) occurs on the boundary of the constraint set, the first order conditions to (2.45) take the form of *inequalities* rather than equalities, so that the standard form of the Euler equation (2.47) is no longer valid. However there are methods that can handle these *boundary solutions* by generalizing the Euler equation to include a Lagrange multiplier function $\lambda(y,z)$ which is positive when the constraints are binding. An especially simple way of handling these contraints is by the method of *parameterized expectations* of Marcet (1990) which we present in section 5.4.

Example 1. Consider Phelp's model of optimal consumption and saving in example 2 of the previous section. This is an example of a continuous decision process with exogenous shocks since the *IID* investment return process $\{R_t\}$ is a special case of the Markovian exogenous state variable $\{z_t\}$ given in (3.38), where $q(dz'|z) = F(dz')$. The endogenous state variable is current wealth w. Thus the state variable in this problem is $s = (w,z)$, and the control variable c is consumption, and the law of motion for the endogenous state variable is $w_{t+1} = r(w_t, c_t, z_{t+1}, z_t) = z_{t+1}(w_t - c_t)$. The h function in this case is given by $h(w,c) = -1$, and the utility function is given by $u(c)$, independent of the state s. One can also verify that Phelp's problem also satisfies all of the other regularity conditions listed in our definition of a CDP. The stochastic Euler equation (2.47) for this problem takes the form:

$$u'(\alpha(w)) = \beta \int z' u'(\alpha(z'(w - \alpha(w)))F(dz'). \tag{2.50}$$

Note that in this case, the assumption that the exogenous state variable is *IID* implies that the optimal decision rule α is a function only of w_t. It is straightforward to verify that the stochastic Euler equation does indeed hold in the special case $u(c) = \log(c)$. Substituting $\log(c)$ and the optimal decision rule $\alpha(w_t) = (1-\beta)w_t$ into (2.50) we get:

$$\frac{1}{(1-\beta)w} = \beta \int \frac{z'}{(1-\beta)w'} F(dz') = \beta \int \frac{z'}{(1-\beta)(z'(w-(1-\beta)w)} F(dz') = \frac{1}{(1-\beta)w}. \tag{2.51}$$

Example 2. Consider the Brock-Mirman (1972) stochastic growth model. This problem was used as a canonical test problem for the effectiveness of various numerical methods in the 1990 symposium in the *Journal of Economic and Business Statistics.* The state variable for this problem

is $s_t = (k_t, z_t)$ where the endogenous state variable k_t is the current capital stock and the exogenous state variable z_t is interpreted as a "technology shock" with Markov transition probability $q(dz_{t+1}|z_t)$. Output is produced by a production function $f(k, z)$ which is increasing and strictly concave in k and increasing in z. Capital and output are assumed to be the same commodity, so the existing capital stock k and current period output $f(k, z)$ can be either reinvested or used for current consumption c. Thus the constraint set is given by $A(k, z) = \{c| \, 0 \leq c \leq f(k, z) + k\}$, and the law of motion r for the next period's capital stock is $k_{t+1} = r(k_t, c_t, z_{t+1}, z_t) = f(k_t, z_t) + k_t - c_t$. The value function V for this problem can be written as:

$$V(s) = V(k, z) = \max_{0 \leq c \leq f(k,z)+k} [u(c) + \beta \int V(f(k, z) + k - c, z')q(dz'|z)]. \tag{2.52}$$

Since this specification of the stochastic growth problem meets all the regularity conditions of definition 2.4 and Lemma 2.4, the stochastic Euler equation (2.47) for this problem takes the form:

$$0 = u'(\alpha(k, z)) - \beta \int u'(\alpha(f(k, z) + k - \alpha(k, z), z')) \left[1 + \frac{\partial f}{\partial k}(f(k, z) + k - \alpha(k, z), z')\right] q(dz'|z). \tag{2.53}$$

By imposing further restrictions on functional form, we can use the Euler equation to derive closed-form solutions for α. For example suppose that $u(c) = c$, $f(k, z) = zk^\gamma$, $\gamma \in (0, 1)$, and assume that the technology shock $\{z_t\}$ follows the log-normal process $\log(z_{t+1}) = \rho \log(z_t) + \epsilon_t$, where ϵ_t is distributed $N(0, \sigma^2)$. Then as Tauchen (1990) noted, we can solve the Euler equation for α to obtain:

$$\alpha(k, z) = zk^\gamma + k - \left(\frac{z^\rho \sigma^2 \beta \gamma}{2(1 - \beta)}\right)^{\frac{1}{1-\gamma}}. \tag{2.54}$$

Alternatively, if $u(c) = \log(c)$ and $f(k, z) = zk^\gamma - k$ (i.e. the case of 100% depreciation of the capital stock each period), then it is easy to verify that the following decision rule satisfies the Euler equation:

$$\alpha(k, z) = (1 - \gamma\beta)zk^\gamma. \tag{2.55}$$

The advantage of using Euler equations in this case is that it has enabled us to obtain a closed-form solution for the optimal decision rule in a problem with two continuous state variables and one continuous control variable, whereas it does not seem possible to derive these solutions from direct analysis of Bellman's equation.

3. Computational Complexity and Optimal Algorithms

To keep this chapter self contained, this section provides a brief review of the main concepts of the theory of computational complexity. There are two main branches of complexity theory: *discrete* (or algebraic) complexity theory and *continuous* (or information based) complexity. The two branches differ in their assumptions about the underlying model of computation (Turing vs. real), the types of problems being solved (discrete vs. continuous), and the nature of the solution (exact vs. approximate). We appeal to computational complexity theory in section 5 as a tool for helping us think about the relative effi ciency of various algorithms for solving MDP's. In particular section 5.4.1 reviews recent work by Chow and Tsitsiklis 1989 who derived lower bounds on the computational complexity of solving a general class of continuous MDP's. Their lower bounds show that the MDP problem is subject to an inherent curse of dimensionality. In a subsequent paper Chow and Tsitsiklis 1991 characterized a nearly optimal algorithm for the MDP problem, i.e. an algorithm that minimizes the (worst case) cpu time required to compute an ϵ-approximation of the value function V for an arbitrary MDP problem in a specifi ed class. Since function approximation is one of the key numerical subproblems involved in approximating solutions to continuous MDP's, section 3.3 reviews basic results on the computational complexity of the *approximation problem* i.e. the problem of approximating a given smooth multidimensional function $f : S \to R$ that is costly to evaluate at any $s \in S$ by smooth functions such as linear combinations of polynomials, splines, or neural networks that can be rapidly evaluated at any $s \in S$.

3.1 Discrete Computational Complexity

Discrete computational complexity theory deals with mathematical problems such as matrix multiplication or the travelling salesman problem that can be fully specifi ed by a fi nite number of parameters and whose exact solution can be computed in a fi nite number of steps.[27] This theory is based on the *Turing model of computation*, which is universal model of computation that assumes that a computer can perform a fi nite number of possible operations on a fi nite number of possible symbols, although with an unbounded memory and disk storage capacity. Thus, a Turing machine is a generic model of a computer that can compute functions defi ned over domains of fi nitely representable objects such as the integers or rational numbers, Z. A function f is said to be *computable* if there exists a computer program or *algorithm* that can compute $f(x)$ for any input x

[27] Good references to this theory include Garey and Johnson, 1979 and Kronsjö, 1985.

in its domain in a finite amount of time using a finite amount of memory and storage.[28] However knowing that an algorithm can compute the value of a function in finite amount of time is not very useful if the amount of time or storage required to compute it is unreasonably large. The theory of computational complexity classifies various problems in terms of their *complexity* defined as the minimal amount computer time or space required by any algorithm to compute $f(x)$ for any particular value of x. In order to formally define complexity, we need some notion of the "size" or inherent difficulty of computing a particular input x. Although the notation x suggests the input is a number, inputs can be more abstract objects such as a finite list of symbols. For example in the traveling salesman problem, the input x consists of a finite list $\{c_1, \ldots, c_n\}$ of cities, and a corresponding list $\{d(c_i, c_j)\}$ of the distances between each of the cities c_i and c_j. The function f returns a solution to the travelling salesman problem, i.e. an ordering $(c_{\pi(1)}, \ldots, c_{\pi(n)})$ that minimizes the length of a tour of all n cities beginning in city $c_{\pi(1)}$, visiting each city in sequence, and then returning to city $c_{\pi(1)}$ from city $c_{\pi(n)}$. Here it is natural to index the "size" or difficulty of the input x by the number of cities n. The *complexity function* $comp(n)$ denotes the minimal time required by any algorithm to compute a solution to a problem with input size n. Computer scientists have classified various mathematical problems in terms of their inherent difficulty, as measured by $comp(n)$. A *polynomial-time problem* has $comp(n) = O(P(n))$ for some polynomial function $P(n)$. If the problem complexity function cannot be uniformly bounded above by any polynomial function $P(n)$ it is refered to as an *exponential-time problem.* In the computer science literature, polynomial-time problems are referred to as *tractable* and exponential-time problems are referred to as *intractable.* Examples of tractable problems include multiplication of two $n \times n$ matrices (for which $comp(n) = O(n^{2.376})$) and linear programming. In section 5 we show that an n-dimensional discrete dynamic programming problem is an example of an intractable problem, since its complexity is given by $comp(n) = \Omega(|S|^n)$, where $|S| > 1$ is the number of possible states in each dimension.

There is an intermediate class of problems, known as *NP,* that can be solved in polynomial time using a "non-deterministic computer".[29] The class *NP* includes the class P of polynomial-time problems. However it is an open question whether all *NP* problems can actually be solved in polynomial time on an ordinary deterministic computer such as a Turing machine. The hardest

[28] Conversely a problem is said to be *non-computable* (or *undecidable*) if there is no algorithm that can be guaranteed to compute $f(x)$ in a finite amount of time. Examples of non-computable problems include the *halting problem* and Hilbert's tenth problem, i.e. solvability of polynomial equations over the integers.

[29] This is, roughly speaking, a computer that can verify in polynomial time whether any conjectured solution to a problem is in fact an actual solution to the problem. See Kronsjö 1985 p. 79 for a more precise definition.

problems in *NP*, known as *NP-complete*, have been shown to be an equivalence class in the sense that there exists polynomial time algorithms that can transform any problem in this class into any other problem in this class. Thus, if any particular *NP*-complete problem can be solved in polynomial time on a deterministic computer, then all *NP*-complete problems can be solved in polynomial time, i.e. $P = NP$. Despite a number of unsuccessful proofs to the contrary, most mathematicians and computer scientists believe that $P \neq NP$, although the so-called $P = NP$ *problem* remains a famous unsolved mathematical puzzle. The traveling salesman problem is one example of an NP-complete problem.

Within the class P of polynomial-time problems, computer scientists have identified a subclass of problems known as *NC* that can be efficiently solved on massively parallel processors.[30] The formal definition of an *NC* problem is one that can be solved in poly-log time using a polynomial number of processors, i.e. the complexity of the problem is given by $comp(n) = O(P_1(\log(n))$ when solved on a massive parallel processor with $P_2(n)$ processors, where P_1 and P_2 are polynomials. An example of a problem in *NC* is matrix multiplication, which has parallel complexity $comp(n) = O(\log(n))$ using $O(n^{2.376})$ processors (see, e.g. Pan and Reif, 1985). A subset of the hardest problems in P, known as P*-complete,* constitute an equivalence class in the sense that there exist poly-log time algorithms that translate any P-complete problem into any other P-complete problem. Thus, if one P-complete problem can be shown to be in the class *NC*, then all P-complete problems are in the class *NC*, i.e. $P = NC$. Similar to the $P = NP$ problem, the "$P = NC$ *problem*" is an unsolved problem in computer science. Most computer scientists believe that $P \neq NC$, i.e. not all polynomial time problems can be efficiently solved on massive parallel processors. An example of a P-complete problem is linear programming. In section 5.1 we review a recent result of Papadimitriou and Tsitsiklis 1986 that shows that the MDP problem is P-complete, at least under the Turing model of computation and the discrete notion of computational complexity. They interpret this result as providing strong evidence that the MDP problem cannot be effectively parallelized. In section 5.2 we show that the MDP problem is in the class *NC* if we adopt the real model of computation and the continuous notion of computational complexity. The next section provides a formal definition of these concepts.

---

[30] The acronym *NC* stands for "Nick's class" in honor of Nicholas Pippenger who first introduced this concept.

3.2 Continuous Computational Complexity

Continuous computational complexity deals with continuous mathematical problems defined over infinite-dimensional spaces such as multivariate integration, solution of PDE's, and solution of continuous-state MDP's.[31] None of these problems can be solved exactly on ordinary computers, although one can in principle compute arbitrarily accurate approximations to these solutions, to within an arbitrary error tolerance $\epsilon > 0$. Unlike discrete complexity theory which is based on the Turing model of computation, continuous complexity theory is based on the *real number model* of computation, i.e. a computer that can conduct infinite precision arithmetic and store exact values of arbitrary real numbers such as π. Thus, continuous complexity theory abstracts from certain practical issues such as numerical stability and round-off error. The other key distinguishing feature of a continuous mathematical problem is that it generally cannot be fully specified by a finite list of parameters. Since computers can only store finite sets of real numbers, it follows that algorithms for solving continuous mathematical problems must make due with partial information on the problem to be solved. For example in the integration problem algorithms must ordinarily rely on evaluations of an integrand f at only a finite number of points in the domain. Recognizing the fact that information on the problem inputs is generally only partial, continuous complexity theory is also referred to as *information-based complexity* (IBC). A number of important developments in IBC theory over the last decade has lead to a fairly complete categorization of the difficulty of computing approximate solutions to a wide variety of continuous mathematical problems, including the continuous MDP problem. It has also lead to fairly precise characterizations of the form of optimal and near-optimal algorithms for various problems. The remainder of this section reviews the main elements of this theory.

A continuous mathematical problem can be defined abstractly as a mapping $\Lambda : F \to B$ from an infinite-dimensional space F into an infinite-dimensional space B. The mapping Λ is known as the *solution operator*. For example, in the case of multivariate integration the solution operator Λ is given by:

$$\Lambda(f) = \int_{[0,1]^d} f(s)\lambda(ds), \tag{3.1}$$

[31] This section closely follows the excellent treatment in Traub, Wasilikowski and Woźniakowski 1988 with only minor changes in notation. Tsitsiklis 1993 provides a review of applications of complexity theory to MDP's. Traub 1993 presents a review of recent developments in complexity theory for solving more general continuous problems such as multivariate integration and function approximation.

where λ denotes Lebesgue measure. The range of Λ is the set $B = R$ and the domain of Λ, the set of admissible integrands f, is given by:

$$F = \left\{ f : [0,1]^d \to R \;\middle|\; D^r f \text{ is continuous and } \|D^r f\| \leq 1 \right\}, \tag{3.2}$$

where $\|D^r f\|$ denotes the largest mixed partial derivatives of order r of f, i.e.

$$\|D^r f\| = \max_{k_1,\ldots,k_d} \sup_{s_1,\ldots,s_d} \left| \frac{\partial^r f(s_1,\ldots,s_d)}{\partial^{k_1} s_1, \ldots, \partial^{k_d} s_d} \right|, \quad \text{subject to: } r = k_1 + \cdots k_d. \tag{3.3}$$

In the case of the MDP problem, the set F of problem elements consist of pairs $f = (u, p)$ satisfying certain regularity conditions. For example, in sections 4 and 5 we will require that u and p are elements of a certain class of Lipschitz continuous functions. The solution operator $\Lambda : F \to B$ can be written as $V = \Lambda(u, p)$, where V denotes the $T + 1$ value functions $(V_0, \ldots, V_T)$ given by the recursions $(2.2), \ldots, (2.5)$ in the finite horizon case, or the unique solution to Bellman's equation (2.10) in the infinite horizon case. Under appropriate regularity conditions on the set of problem elements F, the range of Λ is the set $B = C(S)$, where $C(S)$ is the Banach space of bounded continuous functions.

Since the problem elements $f \in F$ live in an infinite-dimensional space and computers are finite-state devices, we will only be able to approximate the true solution $\Lambda(f)$ using a "computable" mapping $U : F \to B$ that uses only a finite amount of information about f and can be calculated using only a finite number of elementary operations (e.g. addition, multiplication, comparisons, etc.). Given a norm on the space B we can define the error involved in using the approximate operator U instead of the true solution operator Λ by $\|\Lambda(f) - U(f)\|$. We say that $U(f)$ is an *ϵ-approximation of* $f \in F$ if $\|\Lambda(f) - U(f)\| \leq \epsilon$. Since we restrict $U(f)$ to depend only on a finite amount of information about f, we can represent it as the composition of two distinct operators:

$$U(f) = \phi_N \left(I_N(f) \right), \tag{3.4}$$

where $I_N(f) : F \to R^N$ represents an *information operator* providing information about the problem element $f \in F$, and $\phi_N : R^N \to B$ is an *algorithm* that maps the information about f into an approximate solution. [32] In general, the information about f will be given by a finite number N of functionals of f, $I_N(f) = \Big(L_1(f), \ldots, L_N(f) \Big)$. In the problems we are considering here, $F : S \to R$ is a function space and we will be focusing on the evaluation functionals,

[32] The information operator I_N is sometimes referred to an an *oracle.*

$L_i(f) = f(s_i)$ for some point $s_i \in S$. The resulting information operator $I_N(f)$ is referred to as the *standard information:*

$$I_N(f) = \Big(f(s_1), \ldots, f(s_N)\Big), \tag{3.5}$$

where $\{s_1, \ldots, s_N\}$ can be thought of as defining "grid points" in the domain of f. The information operator I_N, the algorithm ϕ_N, and the number of grid points N are all treated as variables to be chosen in order to obtain approximations of a specified degree of accuracy, ϵ. Thus, we can think of an arbitrary algorithm as consisting of the pair $U = (I_N, \phi_N)$.[33]

The total computational cost of computing an approximate solution $U(f)$, denoted by $c(U, f)$, is assumed to consist of two components:

$$c(U, f) = c_1(I_N, f) + c_2(\phi_N, I_N(f)), \tag{3.6}$$

where $c_1(I_N, f)$ is the cost of computing the N information points $I_N(f)$ (i.e. of evaluating the function f at the N grid points $\{s_1, \ldots, s_N\}$), and $c_2(\phi_N, I_N(f))$ denotes the combinatorial cost of using the information $I_N(f) \in R^N$ to compute the approximate solution $U(f) = \phi_N(I_N(f))$. In many problems such as the multivariate integration problem one can prove that $c(I_N, f)$ and $c(\phi_N, I_N(f))$ are both proportional to N. For example, this is true of the multivariate integration problem when $I_N(f)$ is the standard information (3.5) and ϕ_N is a linear algorithm such as the simple sample average algorithm:

$$\phi_N\left(I_N(f)\right) = \frac{1}{N}\sum_{i=1}^{N} f(s_i). \tag{3.7}$$

We are now ready to define the *ϵ-complexity,* which roughly speaking is the minimal cost of computing an ϵ-approximation to a mathematical problem $\Lambda(f)$. We begin by defining the concept of *worst-case complexity* which measures the cost of computing the hardest possible problem element $f \in F$.

Definition 3.1: *The* ***worst case deterministic complexity*** *of a mathematical problem Λ is given by:*

$$comp^{\textit{wor-det}}(\epsilon) = \inf_{U}\left\{c(U) \,\middle|\, e(U) \le \epsilon\right\}, \tag{3.8}$$

---

[33] We have oversimplified the treatment of information here in the interest of brevity. TWW 1988 distinguish between *adaptive information* versus *nonadaptive information.* In the latter case, we allow each of the N functionals of f to be conditioned on the outcome of the previous functionals, i.e. $I_N(f) = \{L_1(f), L_2(f, y_1), \ldots, L_N(f, y_1, \ldots, y_{N-1})\}$ where the y_i are defined recursively by $y_1 = L_1(f)$ and $y_i = L_i(f, y_1, \ldots, y_{i-1})$, $i = 2, \ldots, N$. We have presented the special case of nonadaptive information since for many linear problems, one can prove that there is no gain to using adaptive information.

where the functions $e(U)$ *and* $c(U)$ *are defined by:*

$$\begin{aligned} e(U) &= \sup_{f \in F} \|\Lambda(f) - U(f)\|, \\ c(U) &= \sup_{f \in F} c(U, f). \end{aligned} \tag{3.9}$$

Thus, worst case complexity is a minimax bound on the cost of computing an ϵ-approximation to the problem Λ. Tight upper and lower bounds on the complexity have been established for various mathematical problems. For example, the worst case complexity of integration of functions that are differentiable to order r over the set $S = [0,1]^d$ is known to have a worst case complexity given by[34]

$$comp^{\textit{wor-det}}(\epsilon, d, r) = \Theta\left(\frac{1}{\epsilon^{d/r}}\right). \tag{3.10}$$

Thus, multivariate integration is subject to the curse of dimensionality. In section 4 we review recent results of Chow and Tsitsiklis 1989, 1991 that proves that the general MDP problem is also subject to the curse of dimensionality — at least using deterministic algorithms and measuring complexity on a worst case basis.

One way to break the curse of dimensionality is to use random rather than deterministic algorithms. Similar to the case of deterministic algorithms, a random algorithm $\tilde{U}$ can be decomposed as a composition of a randomized information operator $\tilde{I}_N$ and a randomized combinatory algorithm $\tilde{\phi}_N$:

$$\tilde{U}(f) = \tilde{\phi}_N\left(\tilde{I}_N(f)\right). \tag{3.11}$$

Monte carlo integration is a simple example of a randomized algorithm $\tilde{U}$. It is based on the randomized information

$$\tilde{I}_N(f) = \{f(\tilde{s}_1), \ldots, f(\tilde{s}_N)\}, \tag{3.12}$$

where $\{\tilde{s}_1, \ldots, \tilde{s}_N\}$ are *IID* uniform random draws from $[0,1]^d$, and the deterministic linear algorithm $\phi_N : R^N \to R$ given in equation (3.7). In order to ensure that a random algorithm $\tilde{U}$ is well defi ned we need to defi ne an underlying probability space $(\Omega, \Im, \mu)$ and measurable mappings $\tilde{I}_N : \Omega \to R^N$ and $\tilde{\phi}_N : R^N \times \Omega \to B$ such that $\tilde{U}(f)$ is a well defi ned random element of B for each $f \in F$. We can think of a random algorithm as a kind of *mixed strategy* $\tilde{U} = (\tilde{\phi}_N, \tilde{I}_N, \Omega, \Im, \mu)$. Clearly just as pure strategies are a special case of mixed strategies, deterministic algorithms are special cases of random algorithms when μ is restricted to be a unit mass. It follows immediately

[34] We include the parameters d and r as arguments to the complexity function to emphasize its dependence on the dimension and smoothness of the set of problem elements F. Obviously we do not treat d as a variable in F since otherwise Defi nition 3.1 implies that complexity will generally be infi nite (i.e. take the supremum as $d \to \infty$).

that randomized complexity can never be greater than deterministic complexity, although for certain mathematical problems (such as multivariate integration) randomized complexity is strictly less than deterministic complexity.

Definition 3.2: *The* ***worst case randomized complexity*** *of a mathematical problem* Λ *is given by:*

$$comp^{\textbf{wor-ran}}(\epsilon) \equiv \inf_{\tilde{U}} \left\{ c(\tilde{U}) \,\middle|\, e(\tilde{U}) \leq \epsilon \right\}, \tag{3.13}$$

where $e(\tilde{U})$ *and* $c(\tilde{U})$ *are defined by:*

$$\begin{aligned} e(\tilde{U}) &= \sup_{f \in F} \int_{\Omega} \left\| \Lambda(f) - \tilde{U}(\omega, f) \right\| \mu(d\omega), \\ c(\tilde{U}) &= \sup_{f \in F} \int_{\Omega} c\left(\tilde{U}(\omega, f), f \right) \mu(d\omega). \end{aligned} \tag{3.14}$$

To illustrate definition 2.2, consider the randomized complexity of the multivariate integration problem Λ defined in equations (3.1) and (3.2). The simple monte carlo integration algorithm puts an upper bound on the complexity of the problem given by:

$$comp^{\textbf{wor-ran}}(\epsilon, d) = O\left(\frac{1}{\epsilon^2} \right). \tag{3.15}$$

To see this, note that F is a uniformly bounded class of uniformly equicontinuous functions, so the classical Ascoli Theorem implies that F is a compact subset of $C[0,1]^d$. Define the *covariance operator* $\Sigma : C[0,1]^d \times C[0,1]^d \to R$ by:

$$f \Sigma g = \int f(s) g(s) \lambda(ds) - \int f(s) \lambda(ds) \int g(s) \lambda(ds). \tag{3.16}$$

It is elementary to show that Σ is a bounded, bilinear, positive definite operator. Indeed, the "standard deviation" operator $\sigma : C[0,1]^d \to R$ satisfies:

$$\sigma(f) \equiv \sqrt{f \Sigma f} \leq \|f\|, \tag{3.17}$$

which not only implies that Σ is bounded, but that the functional σ is Lipschitz continuous. Since the set F is compact, it follows that $\sup_{f \in F} \sigma(f) < \infty$. Using Hölder's inequality we see that

$$e(\tilde{U}) = \sup_{f \in F} E \left\{ \left| \frac{1}{N} \sum_{i=1}^{N} f(\tilde{s}_i) - \int f(s) \lambda(ds) \right| \right\} \leq \frac{\sup_{f \in F} \sigma(f)}{\sqrt{N}}. \tag{3.18}$$

Assuming that $c(\tilde{U}) = \gamma(d)N$ for some polynomial function $\gamma(d)$ (which will typically be linear in d), it follows that by setting $N = (K/\epsilon)^2$, $K = \sup_{f \in F} \sigma(f)$, we are assured that $e(\tilde{U}) \leq \epsilon$. It follows that the worst case randomized complexity of multivariate integration is given by:

$$comp^{\textsf{wor-ran}}(\epsilon, d) = O\left(\frac{\gamma(d)}{\epsilon^2}\right). \tag{3.19}$$

Comparing this bound to the deterministic complexity bound in (3.10), it follows that randomization does succeed in breaking the curse of dimensionality of multivariate integration. Bakhvalov 1959 and Novak 1987 have established tight upper and lower bounds on the randomized worst case complexity of multivariate integration. For completeness we state their bound below:

Theorem 3.1 *The worst case randomized complexity of the multivariate integration problem (3.1) and (3.2) is given by:*

$$comp^{\textsf{wor-ran}}(\epsilon, d, r) = \Theta\left(\frac{\gamma(d)}{\epsilon^m}\right), \tag{3.20}$$

where m is given by:

$$m = \frac{2}{\left(\frac{2r}{d}\right) + 1}. \tag{3.21}$$

It follows that in high dimensional problems with fixed r simple monte carlo integration is an approximately *optimal algorithm.*

An implicit assumption underlying the use of random algorithms is that computers are capable of generating true *IID* random draws from arbitrary distributions such as the uniform distribution on $[0, 1]$. Of course, actual computers only generate *pseudorandom* numbers using deterministic algorithms. To the extent that these algorithms are deterministic it would appear that the results on deterministic computational complexity ought to apply, leading to the conclusion that monte carlo integration using actual random number generators cannot be capable of breaking the curse of dimensionality of the multivariate integration problem. There has been a large, almost philisophical debate over this issue. However we agree with the view expressed in Traub, Wasilikowski and Woźniakowski 1988 that "pseudo-random computation may be viewed as a close approximation of random computation, and that randomness is a very powerful tool for computation even if implemented on deterministic computers" (p. 414). Indeed, Traub and Woźniakowski 1992 have shown that monte carlo algorithms based on a linear congruential generator of period M with a uniformly distributed initial seed "behaves as for the uniform distribution and its expected error is roughly $N^{-1/2}$ as long as the number N of function values is less than M^2." (p. 323).

As we noted in the introduction, randomization (even if implemented using "truly random" generators) does not always succeed in breaking the curse of dimensionality: for example randomization doesn't help for problems such as nonlinear optimization, solution of PDE's or Fredholm integral equations, as well as the general approximation problem. A final way to break the curse of dimensionality is to evaluate algorithms on an average rather than worst case basis. To do this we need to specify a prior probability measure μ over the space F of problem inputs.

Definition 3.3: *The* ***average case deterministic complexity*** *of a mathematical problem* Λ *is given by:*

$$comp^{\textbf{avg-det}}(\epsilon) \equiv \inf_{U} \left\{ c(U) \;\middle|\; e(U) \leq \epsilon \right\}, \tag{3.22}$$

where $e(U)$ *and* $c(U)$ *are defined by:*

$$\begin{aligned} e(U) &= \int_F \|\Lambda(f) - U(f)\| \, \mu(df), \\ c(U) &= \int_F c\left(U(f), f\right) \mu(df). \end{aligned} \tag{3.23}$$

Thus, the average case complexity measure is based on minimizing the expected cost of solving a problem with an expected error no greater than ϵ, where the expectation is taken with respect to the prior probability measure μ. One can also define the concept of average randomized complexity, $comp^{\textbf{avg-ran}}(\epsilon)$. However TWW 1988 showed that randomization does not help in the average case setting in the sense that $comp^{\textbf{avg-ran}}(\epsilon) = comp^{\textbf{avg-det}}(\epsilon)$. Thus, there is no loss in generality in restricting attention to deterministic information and algorithms in the average case setting. By choosing the prior μ appropriately, the average case complexity measure can reflect the types of problems one typically encounters in practical applications. However in practice it is difficult to specify priors over infinite dimensional spaces and most results assume fairly specific prior probability measures μ such as "Wiener sheet measure" over the space $F = C[0,1]^d$. Relative to this prior, Woźniakowski 1991 showed that multivariate integration is a tractable problem:

Theorem 3.2 *The average case deterministic complexity of the multivariate integration problem (3.1) and (3.2) is given by:*

$$comp^{\textbf{avg-det}}(\epsilon, d) = \Theta\left(\frac{\gamma(d)}{\epsilon} \left(\log \frac{1}{\epsilon} \right)^{(d-1)/2} \right). \tag{3.24}$$

Woźniakowski provided a full characterization of a nearly optimal integration algorithm: it is the sample average algorithm (3.7), where the integrand f is evaluated at the *Hammersley* points

in $[0,1]^d$.[35] In section 5 we will argue that low discrepancy points are a good choice for grid points for solving the integration subproblem of the MDP problem.

A number of other problems that are intractable in the worst case and randomized settings have been shown to be tractable on in the average case setting. These problems include multivariate function approximation (Woźniakowski, 1993), solution of linear elliptic PDE's, and Fredholm integral equations (Werschulz, 1991). However very little is known about the average case complexity of nonlinear problems; for example it is still an open question whether multivariate optimization is tractable on an average case basis.

3.3 Computational Complexity of the Approximation Problem

The *approximation problem* can be formally stated as the problem of finding a computable approximation $U(f)$ to the mapping $\Lambda : F \to F$ where $\Lambda = I$ is the identity operator on a collection of functions F, where each $f \in F$ is a real valued function of d arguments defined on a domain $S \subset R^d$. The computational complexity of the approximation problem is defined by:

$$\begin{aligned} comp^{wor,det}(\epsilon, d) &= \inf_U \left\{ c(U) \middle| e(U) \leq \epsilon \right\}, \\ e(U) &= \sup_{f \in F} \|f - U(f)\|, \\ c(U) &= \sup_{f \in F} c(U, f), \end{aligned} \tag{3.25}$$

where $c(U, f)$ is the cost of computing the approximation $U(f)$ using the algorithm U.

We can describe the approximation problem in more intuitive terms as follows. Suppose we are interested in approximating a given function $f \in F$ that is "difficult" to compute by an approximate function $U(f)$ that is "simple" to compute. More precisely, approximation is useful in a situation where it takes a large amount of cpu time in order to compute $f(s)$ at any point $s \in S$ whereas the amount of cpu time required to compute $U(f)(s)$ at any $s \in S$ is relatively small. This is certainly the case in continuous MDP problems where we are interested in approximating *value functions*, i.e. functions f of the form:

$$f(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \int V(s') p(ds'|s,a) \right]. \tag{3.26}$$

[35] See Niederreiter 1992 for a definition of the Hammersley points. Woźniakowski actually characterized the optimal integration algorithm: the optimal integration algorithm is a sample average of *shifted* Hammersley points. This is not fully constructive since the amount that the Hammersley points need to be shifted is a quantity that appears to be very difficult to compute. Paskov 1993 characterized the optimal algorithm for integration of smooth functions under the folded Wiener sheet measure prior. He derived a nearly optimal algorithm that breaks the curse of dimensionality: a simple average of the integrand f at points derived from the *hyperbolic cross points.*

Obviously evaluating f at any point $s \in S$ requires solution of multidimensional maximization and integration subproblems which can be quite cpu intensive. So it is clearly vital to find efficient methods to approximate the difficult to evaluate function f by a simple to evaluate function $U(f)$.

We consider computable approximations to f given by mappings U of the form $U(f) = \phi_N(I_N(f))$ where $I_N : \mathcal{B}(S) \rightarrow R^N$ denotes the standard information operator $I_N(f) = (f(s_1), \ldots, f(s_N))$ and ϕ_N denotes a combinatory algorithm that projects the N-dimensional information vector $I_N(f)$ into an approximate function $U(f) \in \mathcal{B}(S)$. Since each evaluation of f and its derivatives is quite costly we are interested in finding an ϵ-approximation at minimal possible cost. The standard treatment of the approximation problem assumes that we can evaluate f exactly, although it is clear from equation (3.26) that in practice we will only be able to obtain approximate values of f. While the theory can be generalized to allow for "noisy information" on f (see, e.g. chapter 12 of TWW, 1988), we will avoid these complications here and assume that f can be exactly evaluated. Also, the standard treatment of the approximation problem does not actually require that it should be faster to evaluate the approximate function $U(f)$ than the original function f at each $s \in S$. However as we will see below it turns out that optimal approximations $U^*(f)$ typically do have this property.

Linear Approximations and Kolmogorov N-Widths. Perhaps the simplest way to approximate a smooth function f is via a simple linear combination of a set of N *basis functions* $p_1, \ldots, p_N$:

$$U(f)(s) = \phi_N(I_N(f))(s) = \sum_{i=1}^{N} f(s_i)p_i(s), \tag{3.27}$$

where $(s_1, \ldots, s_N)$ are pre-specified points in the domain S. A classic example of a linear approximation of this form is *Chebyshev approximation* where the $\{p_i\}$ are linear combinations of the first N Chebyshev polynomials (see section 5.3.2 for the exact formula for Chebyshev approximation). As long as the basis functions $\{p_i\}$ are polynomial functions, it is clear that it will be very easy to compute the approximating function $U(f)$ at any point $s \in S$ and computational effort will be proportional to the number of basis functions N. The question then turns to characterizing the form of *optimal subspaces* F_N. This question was first posed by Kolmogorov in 1936 who defined a measure of approximation error that has since become known as a *Kolmogorov N-width.*

Definition 3.4 *The **Kolmogorov N-width** $d_N(F)$ of a set of functions $F \subset \mathcal{B}(S)$, is defined by:*

$$d_N(F) = \inf_{F_N} \sup_{f \in F} \inf_{v \in F_N} \|f - v\| \tag{3.28}$$

where F_N denotes an N-dimensional subspace of $\mathcal{B}(S)$.

The Kolmogorov N-width provides the worst case error in approximating an arbitrary function $f \in F$ by linear combinations of a set of N basis functions. The following theorem characterizes the classes of functions that can be consistently approximated, i.e. for which $\lim_{N\to\infty} d_N(F) = 0$.

Theorem 3.5 $\lim_{N\to\infty} d_N(F) = 0$ *iff* $F \subset K + V$ *where* K *is a compact set and* V *is a finite dimensional subspace of* $\mathcal{B}(S)$.

Theorem 3.5 shows that the sets of functions F that can be consistently approximated by linear combinations of basis functions are "essentially" finite-dimensional subsets of $\mathcal{B}(S)$. This follows from a result of Araujo and Giné 1980 that compact subsets K of a Banach space B are *flat*, i.e. for each $\epsilon > 0$ there is a finite-dimensional subspace $L \subset B$ such that $K \subset L_\epsilon = \{f \in B | \, \|f - L\| \leq \epsilon\}$. In particular, compact subsets of infinite-dimensional linear spaces necessarily have empty interiors.

If F_N^* attains the infimum in (3.28) we say that F_N^* is an *optimal subspace* for F. Optimal subspaces have been characterized for many types of approximation problems: for example Kolmogorov proved that the sine and cosine functions $\{\sin(x), \ldots, \sin(Nx), \cos(x), \ldots, \cos(Nx)\}$ are an optimal subspace for approximating periodic functions. In addition to finding the form of optimal subspaces, we are interested in characterizing the rate at which the approximation error $d_N(F)$ tends to zero. The following theorem of Novak 1988 provides the exact N-widths for the compact set F of Lipschitz continuous functions $f \in \mathcal{B}(S)$ with Lipschitz bound 1:

Theorem 3.6 *Let* $S = [0,1]^d$ *and* $F = \{f \in \mathcal{B}(S) | |f(s) - f(s')| \leq max_i |s_i - s_i'|\}$. *Then* $d_N(F)$ *is given by:*

$$d_N(F) = \frac{1}{4}\frac{1}{N^{\frac{1}{d}}}. \tag{3.29}$$

Theorem 3.6 implies that linear approximation of Lipschitz continuous functions is subject to the curse of dimensionality since $d_N(F)$ tends to 0 at an increasingly slow rate as d increases. In particular it is easy to see that the value of N required to obtain an ϵ-approximation of an arbitrary Lipschitz continuous function f is given by

$$N = \left(\frac{1}{4\epsilon}\right)^d. \tag{3.30}$$

so linear approximation algorithms are clearly subject to the curse of dimensionality. It turns out that imposing additional smoothness conditions on F, say restricting F to the Sobolev space of functions with r derivatives and $(r+1)^{st}$ derivative uniformly bounded by 1 does not break the

curse of dimensionality: the exponent in (3.30) decreases from d to $\frac{d}{r}$ (see Pinkus 1985, Chapter VII). These N-width bounds results imply that all linear approximation methods are subject to a curse of dimensionality, at least if we are interested in approximating a reasonably broad class of functions (such as the Lipschitz or Sobolev classes) on a worst case basis.

Error Bounds for Nonlinear and Adaptive Approximation Methods. Novak 1988 derived bounds on the approximation error for the much broader class of adaptive, nonlinear algorithms U^{ad}. This class allows for nonlinear combinatory algorithms ϕ_N and adaptive information operators I_N^{ad} of the form:

$$I_N^{ad}(f) = \{f(s_1), f(s_2(f(s_1)), \ldots, f(s_N(f(s_1), \ldots, f(s_{N-1}))\}\,. \tag{3.31}$$

Defi ne the worst case error bound $e_N^{ad}(F)$ by:

$$e_N^{ad}(F) = \inf_{\phi_N} \inf_{I_N^{ad}} \sup_{f \in F} \left\| f - \phi_N(I_N^{ad}(f)) \right\|. \tag{3.32}$$

Let $e_N(F)$ denote the corresponding error bound when we restrict attention to non-adaptive information operators I_N. Since $d_N(F)$, $e_N^{ad}(F)$ and $e_N(F)$ can all be regarded as measures of the "size" of the class of functions F, it should not be surprising that these error bounds are related.

Theorem 3.7 (Novak). *The following inequalities hold:*

$$e_N^{ad}(F) \leq e_N(F) \leq (1+N) d_N(F). \tag{3.33}$$

Note that it is generally not the case that $e_N(F) \leq d_N(F)$ since the defi nition of the N-width $d_N(F)$ does not place any computability restrictions such forcing the approximation to depend only on a fi nite amount of information on the function f to be approximated. Thus $d_N(F)$ can in principle be smaller than $e_N(F)$ even though $d_N(F)$ is defi ned in terms of linear approximations to $f \in F$ whereas $e_N(F)$ encompasses arbitrary nonlinear forms of approximation. Surprisingly, Novak 1988 proved that there are certain classes of functions F for which there is no gain to using nonlinear algorithms ϕ_N or adaptive information operators I_N^{ad} to approximate the functions $f \in F$. For these classes the N-width $d_N(F)$ provides a lower on the approximation error $e_N(F)$.

Theorem 3.8 *Let F be a convex, symmetric set (i.e. if $f \in F$ then $-f \in F$). Then we have:*

$$(1+N) d_N(F) \geq e_N^{ad}(F) = e_N(F) = \inf_{s_1, \ldots, s_N} \inf_{p_i \in \mathcal{B}(S)} \sup_{f \in F} \left\| f - \sum_{i=1}^{N} f(s_i) p_i \right\| \geq d_N(F). \tag{3.34}$$

Since the class F of Lipschitz continuous functions is convex and symmetric Theorem 3.8 implies that linear, nonadaptive approximation algorithms of the form (3.27) are in fact optimal for this class, and the results of Theorem 3.6 imply that there is an inherent curse of dimensionality in approximating functions in this class regardless of whether linear, nonlinear, adaptive or nonadaptive algorithms are used.

Approximation by Neural Networks. Recent results of Barron (1993) and Hornik, Stinchombe, White and Auer (1992) and others have shown that it is possible approximate certain restricted subclasses of functions F using neural networks with a relatively small number N of unknown parameters. In particular, their work shows that the number of coeffi cients N required to obtain an ϵ-approximation of functions f in certain subclasses F grows only polynomially fast in the dimension d:

$$N = O\left(\frac{p(d)}{\epsilon^2}\right), \tag{3.35}$$

where $p(d)$ denotes a polynomial function of the problem dimension d. Barron and Hornik *et. al.* focus on the class of *single layer feedforward networks* defi ned by:

$$f_\theta(s) = \lambda_0 + \sum_{i=1}^{k} \lambda_i \phi(\delta_i' s + \rho_i), \tag{3.36}$$

where $\theta = (\lambda_0, \ldots, \lambda_k, \delta_1, \ldots, \delta_k, \rho_1, \ldots, \rho_k)$ has a total of $N = (d+2)k + 1$ components (since the δ_i are $d \times 1$ vectors) and $\phi : R \to R$ is a sigmoidal function satisfying $\lim_{x\to-\infty} \phi(x) = 0$ and $\lim_{x\to\infty} \phi(x) = 1$. In the lingo of neural nets, ϕ is known as a *squashing function,* s is the *input vector,* $f_\theta(s)$ is the *network output,* k is the number of *hidden units,* the $\{\delta_i, \rho_i\}$ coeffi cients are called hidden unit *weights* and *biases,* and the $\{\lambda_i\}$ coeffi cients are referred to as *output weights.* Typical choices for the squashing function ϕ include the *cosine squasher* $\phi(x) = \cos(x)$ and the *logistic squasher* $\phi(x) = 1/(1 + e^x)$.[36]

Barron and HSW have shown that the error in approximating a function f using neural nets of the form (3.36) with k hidden units decreases at rate $1/\sqrt{k}$ independent of the dimension d of the domain S of f. Barron established this result for a class of functions F which have Fourier representations of the form

$$f(s) = \int_{R^d} e^{i\omega' s} \tilde{f}(\omega) d\omega, \tag{3.37}$$

[36] Since the cosine function is periodic, we can modify ϕ to have sigmoidal properties by defi ning $\phi(x) = \cos(x)$ for $x \in (-\frac{\pi}{2}, 0)$ and $\phi(x) = 1$ for $x \geq 0$ and $\phi(x) = 0$ for $x \leq -\frac{\pi}{2}$.

where $\tilde{f}(\omega)$ is a complex-valued function satisfying

$$C_f \equiv \int_{R^d} \sqrt{\omega'\omega}|\tilde{f}(\omega)|d\omega \leq C < \infty. \tag{3.38}$$

In particular, conditions (3.37) and (3.38) imply that functions in the class F are continuously differentiable. Barron provides a number of different sufficient conditions for functions to be a member of the class F. For example, if f has derivatives of order $r = \lfloor d/2 \rfloor + 2$ which are square-integrable on R^d, then $f \in F$. Barron's main approximation result is stated below:

Theorem 3.9 (Barron). *For every function $f \in F$ with C_f finite, and for every $k \geq 1$, there exists a neural network f_θ with k hidden units satisfying*

$$\sqrt{\int_{B_r} [f(s) - f_\theta(s)]^2 \, \mu(ds)} \leq \frac{2rC_f}{\sqrt{k}}, \tag{3.39}$$

where $B_r = \{s \,|\, \|s\| \leq r\}$, and μ is an arbitrary probability measure on B_r.

Barron's result can be regarded as a bound on the L_2-norm on the neural net approximation error: HSWA extended Barron's result to other norms such as the L_p-norms, the sup-norm, and Sobolev norms. In each case the error bounds involve the factor $1/\sqrt{k}$ where k is the number of hidden units. This result is quite remarkable since if F is the class of functions whose constants C_f are uniformly bounded by some constant C, Barron's inequality (3.39) implies that we can obtain an ϵ-approximation to any $f \in F$ using only $k = O(1/\epsilon^2)$ hidden units. Given that the neural net has a total of $N = (d+2)k + 1$ coefficients, if the bounding constant C_f can be shown to increase polynomially in the problem dimension d, then unlike series and spline approximators, the total number of coefficients in a neural network approximation increases polynomially rather than exponentially in d.

Does Barron's result imply that neural networks succeed in breaking the curse of dimensionality of the approximation problem? Unfortunately the answer is no. Note that Barron's class F is convex and symmetric, so that Novak's result (Theorem 3.8) applies, and the N-width $d_N(F)$ provides a lower bound on the approximation error $e_N^{ad}(F)$ using *any* linear, nonlinear, adaptive or nonadaptive approximation algorithm including neural networks. However Barron showed that a lower bound on the N-width for his class F is proportional to $1/N^{\frac{1}{d}}$, i.e. that $d_N(F) = \Omega(1/N^{\frac{1}{d}})$. Thus, neural networks do not succeed in overcoming the curse of dimensionality associated with approximating even Barron's relatively narrow class of functions F, at least on a worst case basis.

[37] An intuitive way to reconcile the apparently conflicting results of Novak and Barron is to note that Barron's theorem only states that *there exists* neural nets with relatively small numbers of parameters that succeed in providing an ϵ-approximation to functions in the class F on a worst case basis. However his theorem does not provide a constructive computational procedure for finding these parameters. Although HSWA show that the θ parameters satisfying Barron's bound can be computed by nonlinear least squares, doing this requires approximating the solution to a *global* minimization problem which is itself subject to a curse of dimensionality.

Worst Case Complexity Bounds for the Approximation Problem. We have seen that nonlinear and adaptive approximation algorithms do not succeed in breaking the curse of dimensionality of the approximation problem. Another possible way to break the curse of dimensionality is to employ randomized algorithms. We have seen in the previous section that randomization succeeds in breaking the curse of dimensionality of multivariate integration. Unfortunately, the following result of TWW 1988 shows that randomization does not succeed in breaking the curse of dimensionality of the approximation problem, even if we consider algorithms that use information of the derivatives of f.

Theorem 3.10 *Let F be the Sobolev class, $F = \{f : [0,1]^d \rightarrow R \,\big|\, \|f^{(r)}\| \leq 1\}$. Randomization does not succeed in breaking the curse of dimensionality associated with approximating functions in F:*

$$comp^{wor,ran}(\epsilon, d, r) = \Theta\left(comp^{wor,det}(\epsilon, d, r)\right) = \Theta\left(\frac{1}{\epsilon^{\frac{d}{r}}}\right). \tag{3.40}$$

Although randomization doesn't help to break the curse of dimensionality for functions in the class F, we will show in section 5.3.1 and 5.4.1 that randomization can be used to break the curse of dimensionality associated with approximating value functions for the subclass of discrete decision processes. The intuition behind this result is that the value functions for such problems take the form of the maximum of a finite number of linear integral operators. It turns out that randomization can be used to break the curse of dimensionality of approximating functions of this form.

Average Case Complexity Bounds for the Approximation Problem. Theorem 3.10 shows that there is essentially no way around the curse of dimensionality of approximating a reasonably

[37] A related problem is that even though the bounding constant C_f in (3.38) is finite, it may increase exponentially fast in d and in such a case the neural network would still require an exponentially increasing number of coefficients N in order to obtain a specified approximation error ϵ. As Barron notes: "The constant C_f involves a d-dimensional integral, and it is not surprising that often it can be exponentially large in d." (p. 932) Although Barron has characterized certain classes of functions F for which C_f increases only polynomially fast in d, it appears that most of these classes are too restrictive to include value functions encountered in typical economic applications.

general class of functions on a worst case basis. However recent results of Woźniakowski 1992 that function approximation is tractable on an average case base under a "folded Wiener sheet measure" prior distribution over the space of functions F. We do not have space here to describe Woźniakowski's algorithm in detail beyond the observation that the approximation is of the form

$$U(f) = \phi_N(I_N(f))(s) = \sum_{s_j \in H_N} f(h(s_j))p_j(s). \tag{3.41}$$

where H_N is the set of N *hyperbolic cross points* and the $\{p_j\}$ are a set of basis functions derived from functions known as the *de la Vallée–Poussin kernels.* The idea behind the hyperbolic cross points and de la Vallée–Pousin kernel is due to Temlyakov 1987 who used them to construct an algorithm that breaks the curse of dimensionality of approximating a certain class of multivariate periodic functions on a worst case basis. Woźniakowski extended Temlyakov's algorithm to a larger class of non-periodic functions and proved the following theorem:

Theorem 3.11 (Woźniakowski) *Let F be a class of functions with r continuous derivatives on the domain $S = [0,1]^d$ and suppose all partial derivatives of $f(s)$ vanish when one of the components of s equals 0. Then if μ is the Wiener Sheet measure on F, the average case computational complexity of multivariate function approximation is given by:*

$$comp^{avg}(\epsilon, d) = \Theta\left(\left(\frac{1}{\epsilon}\right)^{\frac{1}{(r+1/2)}} \left(\log\frac{1}{\epsilon}\right)^{\frac{(d-1)(r+1)}{(r+1/2)}}\right). \tag{3.42}$$

This result shows that we can indeed break the curse of dimensionality of multivariate function approximation on an average case basis. It suggests the possibility that there may be smooth approximation algorithms that succeed in breaking the curse of dimensionality associated with approximating the value functions to certain classes of continuous MDP problems. However there are difficulties in applying Woźniakowski's average case results to the MDP problem since his results are based on a prior given by folded Weiner sheet measure over the space F. Such a prior is inappropriate for the transition probability density $p(s'|s,a)$ in the MDP problem since it doesn't guarantee that p is nonnegative and integrates to 1.

4. Numerical Methods for Contraction Fixed Points

In section 2 we saw that in stationary infinite horizon MDP's the Bellman operator Γ is a contraction mapping and the value function V is the fixed point to this contraction mapping. This section provides a brief review of the two main algorithms for computing fixed points to general contraction mappings, the method of *successive approximations* and the *Newton-Kantorovich method.* We will see in section 5.2 that analogs of these methods are the predominant solution methods for discrete infinite horizon MDP's, although the Bellman operator has further structure that can allow us to design even more specialized and efficient iterative methods than are available for general contraction mappings. However we have included this section in this survey since there are many problems in economics that require solutions to general contraction mappings that don't take the form of Bellman operators. An example of a more general contraction operator is the *smoothed Bellman operator* $\Gamma_\sigma : \mathcal{B}(S) \to \mathcal{B}(S)$ defined by:

$$\Gamma_\sigma(V)(s) = \sigma \log \left[\sum_{a \in A(s)} \exp \left\{ \frac{1}{\sigma} \left[u(s,a) + \beta \int V(s') p(ds'|s,a) \right] \right\} \right]. \tag{4.1}$$

We encounter fixed points to operators of this form in maximum likelihood estimation of DDP's with unobserved state variables: see, for example, Rust 1994a. Dini's Theorem implies that for each $V \in \mathcal{B}(S)$ we have:

$$\lim_{\sigma \to 0} \Gamma_\sigma(V) = \Gamma(V), \tag{4.2}$$

i.e. the smoothed Bellman operator converges to the Bellman operator as the smoothing parameter $\sigma \to 0$. This result is useful for simplifying the analysis of various algorithms for computing approximate value functions since $\Gamma_\sigma(V)(s)$ is a nice smooth function of V for all $s \in S$ whereas the max operator inside the Bellman operator can introduce kinks or nondifferentiabilities in $\Gamma(V)(s)$ at certain points $s \in S$. These kinks create complications for certain smooth approximation algorithms such as the minimum residual algorithm to be described in section 5.4.2. The errors involved in replacing Γ by Γ_σ can be made as small as desired since a straightforward application of Lemma 2.1 shows that the fixed point V_σ to the smoothed Bellman operator Γ_σ converges to the fixed point V of the Bellman operator Γ as $\sigma \to 0$.

The Contraction Mapping Theorem implies that a globally convergent algorithm for computing the fixed point to any contraction mapping Γ is the method of successive approximations:

$$V_k = \Gamma(V_{k-1}) = \Gamma^k(V_0). \tag{4.3}$$

However an alternative method is to approximate the fixed point V using *Newton-Kantorovich* iterations:

$$V_{k+1} = V_k - [I - \Gamma'(V_k)]^{-1}(I - \Gamma)(V_k), \tag{4.4}$$

where I denotes the identity operator on $\mathcal{B}(S)$, and $\Gamma'(V)$ denotes the *Gateaux derivative* of Γ evaluated at the point $V \in B$, i.e. the linear operator $\Gamma'(V) : \mathcal{B}(S) \to \mathcal{B}(S)$ defined by

$$\Gamma'(V)(W) \equiv \lim_{t \to 0} \frac{[\Gamma(V + tW) - \Gamma(V)]}{t}. \tag{4.5}$$

It follows that $\Gamma'(V) \in L(\mathcal{B}(S), \mathcal{B}(S))$, the space of all continuous linear operators from $\mathcal{B}(S)$ into itself.[38] In the case of the smoothed Bellman operator it is straightforward to show that Γ'_σ is given by:

$$\Gamma'_\sigma(V)(W)(s) = \beta \sum_{a \in A(s)} \int W(s') p(ds'|s,a) P_\sigma(a|s), \tag{4.6}$$

where $P_\sigma(a|s)$ is a *conditional choice probability* given by:

$$P_\sigma(a|s) = \frac{\exp\left\{\frac{1}{\sigma}\left[u(s,a) + \beta \int V(s') p(ds'|s,a)\right]\right\}}{\sum_{a' \in A(s)} \exp\left\{\frac{1}{\sigma}\left[u(s,a') + \beta \int V(s') p(ds'|s,a')\right]\right\}}, \tag{4.7}$$

$P_\sigma(a|s)$ is known in physics as the *Boltzman distribution* with temperature parameter $1/\sigma$.

An argument exactly analogous to the series expansion argument used to proved the existence of $[I - \beta M_\alpha]^{-1}$ in section 2.4 can be used to establish that the linear operator $[I - \Gamma'(V)]^{-1}$ exists, is continuous, and has a convergent Neumann series expansion provided that $\|\Gamma'(V)\| \in (0, 1)$, a result known as *Banach's Theorem* (see Kantorovich and Akilov, 1982, p. 154). It is easy to see that the fact that Γ is a contraction mapping implies that $\|\Gamma'(V)\| \leq \beta$, where the norm of a linear operator was defined in equation (2.30) of section 2.4. Thus, the iterations defined in (4.4) are always well-defined. Kantorovich's Theorem guarantees that given a starting point V_0 in a *domain of attraction* of the fixed point V of Γ, the Newton-Kantorovich iterations will converge to V at a *quadratic rate:*

Theorem 4.1 (Kantorovich): *Suppose that Γ has continuous first and second derivatives satisfying $\|\Gamma''(V)\| \leq K$, then for any initial point V_0 satisfying:*

$$\|I - \Gamma(V_0)\| = \eta \leq (1 - \beta)^2/(2K), \tag{4.8}$$

[38] If the above limit is uniform for all V such that $\|V\| = 1$, $\Gamma'(V)$ is known as the Fréchet derivative of Γ.

then all subsequent Newton-Kantorovich iterations (4.4) satisfy the inequality:

$$\|V_k - V\| \leq \frac{1}{2^k}\left(\frac{2K\eta}{(1-\beta)^2}\right)^{2^k}(1-\beta)^2/K. \tag{4.9}$$

Equation (4.9) shows that provided we start the Newton-Kantorovich iterations from a point V_0 suffi ciently close to V, convergence will be extremely rapid, since (4.9) implies that:

$$\|V_{k+1} - V\| \leq K'\|V_k - V\|^2, \tag{4.10}$$

for a constant K'. The rate of convergence can be further accelerated by using a quasi-Newton method which involves evaluating Γ' in (4.4) at the intermediate point $W_k = \lambda V_k + (1-\lambda)\Gamma(V_k)$ for $\lambda \in (0,1)$. Werner (1984) showed that evaluation at the point $\lambda = 1/2$ typically improves the speed of convergence by reducing the constant K' in (4.10) in comparison to the constant implied by the standard Newton-Kantorovich iteration.

Thus, Newton-Kantorovich iterations yield rapid quadratic rates of convergence but are only guaranteed to converge for initial estimates V_0 in a domain of attraction of V whereas successive approximations yields much slower linear rates of convergence rates but are always guaranteed to converge to V starting from any initial point $V_0 \in B$. This suggests the following a hybrid method or *polyalgorithm:* start with successive approximations, and when the error bounds in Lemma 2.2 indicate that one is suffi ciently close to V, switch over to Newton-Kantorovich iterations to rapidly converge to the solution. The next section will show that in the case where Γ is a Bellman operator, the Newton-Kantorovich method actually coincides with the method of policy iteration, a result fi rst published in the important paper by Puterman and Brumelle, 1979. However, in the case of policy-iteration for discrete MDP's, the domain of attraction widens to the entire space $\mathcal{B}(S) = R^{|S|}$: no initial successive approximation iterations are necessary to guarantee convergence.

We now show that the number of Newton-Kantorovich iterations required to compute an ϵ-approximation to V are substantially fewer than the number of successive approximation steps. If we adopt the relative error criterion and stop successive approximations when the relative error

$$\frac{\|V_k - V\|}{\|V\|} \leq \epsilon, \tag{4.11}$$

and if we start successive approximations from the initial estimate $V_0 = 0$, then then an easy calculation shows that a total of $T_s(\epsilon, \beta)$ successive approximation steps are required to obtain an ϵ-approximation of V, where $T_s(\epsilon, \beta)$ is given by:

$$T_s(\epsilon, \beta) = \lceil \log \frac{1}{\epsilon} / \log \frac{1}{\beta} \rceil.$$

However if we start Newton-Kantorovich iterations from an initial estimate V_0 satisfying inequality (4.8) (which require at most $\log(2K/(1-\beta^2))/\log(1/\beta)$ successive approximation iterations starting from $V_0 = 0$), then only $T_n(\epsilon,\beta)$ Newton-Kantorovich iterations are required to obtain an ϵ-approximation, where $T_n(\epsilon,\beta)$ is given by:

$$T_n(\epsilon,\beta) = O\left(\log\left[\log(1/\epsilon) + \log\left(1/(1-\beta)\right)\right]\right). \tag{4.12}$$

This implies that for small values of ϵ vastly fewer Newton-Kantorovich iterations are required to find an ϵ-approximation of V, and the number of such iterations is asymptotically independent of the discount factor β as $\epsilon \to 0$. Although Newton-Kantorovich iterations converge much faster than successive approximations, the work per iteration is much larger. So we face the issue of which method can compute an ϵ-approximation in less cpu time. It is difficult to say much about this issue generality without imposing more restrictions on the class of contraction mappings we are considering. We will be able to say more about the relative efficiency of the two methods in section 5.2 for the special case where Γ is a Bellman operator.

We conclude this section by reviewing a result of Sikorski and Woźniakowski 1987 on the worst case complexity of the contraction fixed point problem. They consider the class of algorithms that use the information I_N of evaluations of Γ at N adaptively chosen points. Let $T^*(\epsilon,\beta)$ denote the minimum number of evaluations of Γ required in order to find an ϵ-approximation to V for all contraction mappings $\Gamma : R^{|S|} \to R^{|S|}$.

Theorem 4.2: *Suppose that $V \in R^{|S|}$ for some $|S| < \infty$ and $|S| \geq \lceil \log(1/\epsilon)/\log(1/\beta) \rceil$. Then:*

$$T^*(\epsilon,\beta) \geq \frac{1}{2} T_s(\epsilon,\beta) - 1. \tag{4.13}$$

This result says that in high-dimensional problems, the method of successive approximations is nearly an optimal algorithm, at least relative to the information I_N consisting of evaluations of Γ. However if $|S| < \lceil \log(1/\epsilon)/\log(1/\beta) \rceil$ then better algorithms than successive approximations can be constructed. For example in the case $|S| = 1$ Woźniakowski and Sikorski showed that the ***fixed point envelope*** (FPE) algorithm attains the lower bound $T^*(\epsilon,\beta)$ given by:

$$T^*(\epsilon,\beta) = \frac{\log(1/2\epsilon)}{\left[\log\left(\frac{3+\beta}{1+3\beta} + .9\right) + \log\log\left(\frac{1}{1-\beta}\right)\right]}. \tag{4.14}$$

When $\epsilon = 10^{-4}$ and $\beta = 1 - 10^{-4}$ successive approximations requires 5000 times more function evaluations than the FPE algorithm to find an ϵ-approximation. This suggests that the optimal algorithm for computing contraction fixed points depends on the problem dimension $|S|$, but in

sufficiently high dimensional problems simple successive approximations is nearly the best that we can do. We do not know yet what kinds of algorithms are optimal in intermediate dimension problems. Also, the Woźniakowski and Sikorski result does not consider more general forms of information. In sections 5.2 and 5.4 we will be able to say more about the relative efficiency of various fixed point algorithms by exploiting the additional structure of of the Bellman operator. In particular, we explicitly consider the issue of how to numerically evaluate $\Gamma(V)$ and how the cost of evaluating it depends on the number of states $|S|$. This issue was ignored in the Woźniakowski and Sikorski analysis, who assumed that the cost of evaluating Γ is c, independent of $|S|$.

5. Numerical Solution Methods for General MDP's

This section surveys the solution methods for MDP's. We cover both standard approaches that have been used for many years as well as several promising recent methods for which we have little or no practical experience. We devote separate subsections to the main categories of MDP's: finite and infinite horizon problems and discrete and continuous problems. This organization is dictated by the substantial differences in solution methods for these various categories of problems. We also provide separate treatment of solution methods for CDP's and DDP's since the nature of the control variable also has a big impact on the type of algorithm and the computational complexity of these problems.

5.1 Discrete Finite Horizon MDP's

The main numerical method for solving finite horizon MDP's is simple backward recursion (see equations (2.3) to (2.6) in section 2). The integration operator in the finite state case reduces to simple summation:

$$V_t(s) = \Gamma(V_t)(s) = \max_{a \in A(s)} \left[u(s,a) + \beta \sum_{s=1}^{|S|} V_{t+1}(s') p(s'|s,a) \right]. \tag{5.1}$$

Assume for simplicity that the choice sets $A(s)$ contain a common finite number of possible actions A. Then it is easy to see that computation of (5.1) for each state s requires a total of $2(|A||S| + |A|)$ additions, multiplications, and comparison operations, or a total of $2(|A||S|^2 + |A||S|)$ operations to compute the entire time t value function V_t. Thus the total operation count in carrying out the dynamic programming procedure is is $2T(|A||S|^2 + |A||S|)$, which is dominated by the squared term for problems where $|S|$ is large. Note that the storage requirements are also $O(|A||S|^2)$, representing the space required to store the transition probabilities $\{p(s'|s,d)\}$. It follows that the complexity of solving discrete finite horizon MDP's is $O(T|A||S|^2)$, which implies that it is a member of the class P of polynomial time problems, *provided that we measure the size of the problem by the number of discrete states* $|S|$. However if we measure the size of the discrete MDP problem by the *dimension* of the state vector s_t rather than by the total number of states $|S|$, then one can show that the MDP problem is in the class of exponential-time problems.

Theorem 5.1: *Under the Turing model of computation the finite horizon discrete MDP problem is P-complete if problem size is measured by the pair* $(|A|, |S|)$ *representing the total number of possible decisions and states in the MDP. If problem size is measured by the pair* (d_a, d_s)*, resulting in an MDP problem with*

$(|A|^{d_a}, |S|^{d_s})$ *decisions and states for some* $|A| > 0$ *and* $|S| > 1$*, then the MDP problem is in the class of exponential-time problems. In the latter case, the MDP problem is not even NP-complete since the amount of time required to verify a candidate solution to the MDP increases exponentially in* (d_a, d_s).

Notice that the main bulk of the work required to solve a discrete finite horizon MDP problem is the computation of the conditional expectations of the value function for each possible combination of the state s, action a, and time period t: the remaining summation and maximization operations are of order $O(T|A||S|)$ which are negligible compared to the $O(T|A||S|^2)$ operations needed to compute the conditional expectations. There are four main ways to speed up the latter calculations: 1) exploiting special "sparsity structure" of the transition probability $p(s'|s,d)$, 2) using massively parallel processors, 3) using fast matrix multiplication algorithms, and 4) using smooth approximation methods to reduce the cost of computing and storing value functions for discrete MDP's with huge numbers of states and controls. We defer discussion of the last approach to our discussion of smooth approximation methods in sections 5.3.2 and 5.4.2.

Exploiting special structure of the MDP problem. Many economic problems such as the discretized version of the optimal replacement problem and the optimal consumption and saving problems presented in section 2 have transition probabilities that are sparse and they often have a highly regular, recursive structure. For example, by restricting the distribution of investment returns to a finite interval one obtains a discrete representation of the consumption/savings problem with a banded transition probability matrix p. In order to obtain the largest speedups and storage reductions we need to fully exploit our *a priori* knowledge of the particular economic problem. However determining the best way to do this gets more complicated in problems with multidimensional state variables. To apply formula (5.1) we typically recode the multidimensional state vector s_t to have a linearly ordered discrete state representation $s = 1, \ldots, |S|$. By doing this coding in the right way we can obtain a representation for the finite state transition probability matrix that has a desirable *sparsity pattern* that substantially reduces the burden of computing the conditional expectation of the value functions in (5.1). Rust (1991) provides an example of this approach in the case of an MDP model of optimal retirement behavior where the state variable s_t has seven components, i.e. $d_s = 7$. It turns out that for this problem any coding procedure yields a matrix representation for $p(s'|s,d)$ that is a direct product of a circulant matrix C, a banded matrix B, and a dense matrix D. Depending on how we order these component matrices to form the overall transition probability matrix p we obtain various sparsity patterns which are more or less amenable to rapid computation on parallel and vector computers. It turns out that the optimal ordering for a vector processor like the Cray-2 is $p = C \otimes B \otimes D$, which yields an upper block triangular representation for p (for details

see Rust, 1991). This strategy is even more effective for solving infinite horizon problems by the policy iteration approaches presented in the next section since policy iteration involves solution of systems of $|S|$ equations in $|S|$ unknowns, which takes $\Theta(|S|^3)$ time using conventional algorithms for solving systems of linear equations (e.g. LU factorization and recursive back-substitution). Relative to naive policy iteration methods that treat p as a dense matrix, Rust showed that using the optimal ordering $p = C \otimes B \otimes D$ fully exploits the sparsity pattern of p and results in speedups of $\Theta(|C|^3)$ where $|C|$ is the order of the circulant matrix C. For the retirement problem considered in Rust's 1991 paper, this resulted in a speed-up of 27,000 times relative to standard policy iteration algorithms that don't exploit the sparsity structure of p.

Massive Parallel Processing Technology. Massive parallel processing is just beginning to be applied in computational economics. Note that the backward recursion approach to solving DP problems is inherently sequential and cannot be parallelized. However, the majority of the work in each step of backward induction are the $O(|A||S|^2)$ operations necessary to compute the conditional expectations of the value function. This task can clearly be performed in parallel. For example, if we have access to an expandable massive parallel processor with $O(|S|)$-processors, then we can assign each separate processor to compute the summation in (5.1) for each state $s = 1, \ldots, |S|$. It follows that the MDP problem can now be solved in $O(T|A||S|)$ time using $|S|$ processors as opposed to the $O(T|A||S|^2)$ time required by a single processor. However there is an unsolved issue about the how much additional speedup can be attained by employing parallel processors with more than $|S|$ processors. In the next section we argue that the infinite horizon MDP problem can be effectively parallelized, in the sense that it can be solved in $O(\log(|S|)^2)$ time on $O(|S|^{2.376})$ processors. Although it seems clear that massively parallel computers will enable us to significantly speed up the calculation of solutions to finite horizon MDP's as well, most current applications of this technology (e.g. Coleman 1993) have focused on infinite horizon problems so we will defer further discussion of this approach to sections 5.2 and 5.4.

Fast Matrix Multiplication Algorithms. Although fast matrix multiplication methods also offer the potential for large speedups on either serial or parallel machines, there have been few practical implementations of these algorithms. However this may change in the near future since the potential gains to using these algorithms are getting increasingly large — especially in large scale problems where the "overhead" of these algorithms can be "amortized". For example, the number of operations required to multiply two $n \times n$ matrices using standard matrix multiplication algorithms is $2n^3 - n^2$. Strassen's 1972 algorithm computes the product in $4.7n^{2.807}$ operations, Pan's 1980 algorithm requires $O(n^{2.795})$ operations, and Coppersmith and Winograd's 1987 algorithm

requires only $O(n^{2.376})$ operations. [39] However these fast matrix multiplication algorithms have larger space requirements and higher fixed costs relative to the conventional matrix multiplication algorithm. For example, Strassen's algorithm requires $11/3n^2$ memory locations which exceeds the $3n^2$ locations needed for conventional matrix multiplication, and Pan's algorithm requires 24 operations to multiply two 2×2 matrices compared to only 8 operations required by the conventional algorithm. However the break-even point for overcoming the overhead costs associated with fast matrix multiplication may not be very large. For example when $n = 128$ the traditional algorithm requires 2,097,152 operations versus 823,723 for Strassen's algorithm and 797,184 for Pan's algorithm. [40] This suggests that implementation of fast matrix multiplication algorithms may be a path worth investigating for solving discrete MDP's with large, ill-structured, and non-sparse transition probability matrices.

5.2 Discrete Infinite Horizon MDP's

We begin by reviewing the "standard methods" for MDP's. We do not attempt to offer a comprehensive survey of the huge number of different methods that have been proposed, but rather focus on the methods which we view as most effective for problems arising in economic applications. As mentioned in section 2, the solution to infinite horizon MDP problems is mathematically equivalent to computing a fixed point of the Bellman operator $V = \Gamma(V)$. The fixed point problem can also be posed as the problem of finding a zero to the nonlinear functional $F(V) = 0$, where $F = [I - \Gamma]$. Section 4 reviewed the two most prominent methods for computing fixed points to contraction mappings, successive approximations and the Newton-Kantorovich method. Most of the other methods that have been proposed for the solution of infinite horizon MDP's are applications of more general methods for solving systems of nonlinear equations. This has spawned an almost overwhelmingly large number of iterative techniques for solving infinite horizon MDP's including variants of Newton's method, Gauss-Seidel, successive overrelaxation, and so forth. For a review of these methods in the case of general nonlinear equations, we refer the reader to the classic text by Ortega and Rheinboldt 1970. Readers who are interested in seeing how these methods can be applied to the MDP problems should consult Kushner and Kleinman, 1971, Porteus, 1980, and Puterman, 1990, 1994.

---

[39] Surprisingly, the computational complexity of matrix multiplication is not known. All that is known is that its complexity, $comp(n)$, lies within the following lower and upper bounds: $comp(n) = \Omega(n^2)$ and $comp(n) = O(n^{2.376})$.

[40] These figures are taken from table 2.4.2 of Kronsjö, 1985.

5.2.1 Successive Approximation, Policy Iteration, and Related Methods

This subsection provides brief descriptions of the methods that are most commonly used for solving discrete infi nite horizon MDP's that arise in economic and operations research applications. The main issue is the relative effi ciency of successive approximations versus policy iteration in solving large scale problems. Although the standard implementation of policy iteration becomes computationally infeasible in problems with $|S| \geq 10,000$, there are modifi ed versions of policy iteration that appear to dominate successive approximation and its variants. Section 5.2.2 verifi es this conclusion in a numerical comparison of the accuracy and effi ciency of the various methods in solving a discretized version of the auto replacement problem in example 3 of section 2.5.

Successive Approximations: [41] Starting with an arbitrary initial guess V_0 of the solution of Bellman's equation, one simply iterates the Bellman operator

$$V_{k+1} = \Gamma(V_k) = \Gamma^{k+1}(V_0). \tag{5.2}$$

Note that in the case where we set $V_0 = 0$, the method of successive approximations is equivalent to solving an approximate fi nite-horizon problem by backward induction. The Contraction Mapping theorem (see Theorem 2.1 in section 2) guarantees the consistency of this algorithm: in particular the contraction property implies that $\|V - V_k\|$ converges to zero at a geometric rate, sometimes referred to as *linear* convergence since inequality (2.24) in Lemma 2.2 implies that the ratios of approximation errors satisfy $\|V_k - V\|/\|V_{k-1} - V\| \leq \beta$. Furthermore this same error bound can be used to determine an upper bound on the number of iterations required in order to be within some pre-specifi ed tolerance ϵ of the true solution V. Indeed, a straightforward calculation shows that a maximum of $T(\epsilon, \beta)$ successive approximation steps are required to obtain an ϵ-approximation, where $T(\epsilon, \beta)$ is given by:

$$T(\epsilon, \beta) = \frac{1}{|\log(\beta)|} \log\left(\frac{1}{(1-\beta)\epsilon}\right). \tag{5.3}$$

This upper bound is tight: one can construct MDP's that take exactly $T(\epsilon, \beta) - 1$ successive approximation steps in order to compute an ϵ-approximation to V (for example consider the case where $u(s, a) = 1$ so the MDP reduces to a simple geometric series). In problems where the discount factor β is very close to 1 such as problems where the time intervals are relatively short

[41] The method also goes by name *value iteration, contraction iteration, backward induction,* or simply *dynamic programming.*

(such as daily or weekly), so we can see from (5.3) that an unacceptably large number of successive approximation steps will be required to obtain any desired level of numerical accuracy ϵ.

Accelerated Successive Approximations. In certain circumstances the method of successive approximations can be signifi cantly accelerated by employing the *McQueen-Porteus Error Bounds*. If V is the true solution to Bellman's equation, $V = \Gamma(V)$, and V_0 is any initial starting estimate for successive approximations, then after k successive approximations steps V must lie within upper and lower bounds given by:

$$\Gamma^k(V_0) + \underline{b}_k e \leq V \leq \Gamma^k(V_0) + \overline{b}_k e, \tag{5.4}$$

where e denotes an $|S| \times 1$ vector of 1's, and:

$$\begin{aligned} \underline{b}_k &= \beta/(1-\beta)\, min[\Gamma^k(V_0) - \Gamma^{k-1}(V_0)] \\ \overline{b}_k &= \beta/(1-\beta)\, max[\Gamma^k(V_0) - \Gamma^{k-1}(V_0)]. \end{aligned} \tag{5.5}$$

The contraction property guarantees that $\underline{b}_k$ and $\overline{b}_k$ approach each other geomerically at rate β. That fact that the fi xed point V is bracketed within these bounds suggests that we can obtain an improved estimate of V by terminating the iterations (5.2) when $|\overline{b}_k - \underline{b}_k| < \epsilon$, setting the fi nal estimate of V to be the median bracketed value:

$$\hat{V}_k = \Gamma^k(V_0) + \left(\frac{\overline{b}_k + \underline{b}_k}{2}\right) e. \tag{5.6}$$

Bertsekas (1987) p. 195 shows that the rate of convergence of $\{\hat{V}_k\}$ to V is geometric at rate $\beta|\lambda_2|$, where λ_2 is the subdominant eigenvalue of M_α. In cases where $|\lambda_2| < 1$ the use of the error bounds can lead to signifi cant speed-ups in the convergence of successive approximations at essentially no extra computational cost. However in problems where M_α has multiple ergodic sets $|\lambda_2| = 1$ and the error bounds will not lead to an appreciable speed improvement as illustrated in computational results in table 5.2 of Bertsekas (1987).

Policy Iteration Methods. In relatively small scale problems ($|S| < 500$) with discount factors suffi ciently close to 1 (e.g. $\beta > .95$) the method of *Policy Iteration* is generally regarded as one of the fastest methods for computing V and the associated optimal decision rule α. The method starts by choosing an arbitrary initial policy, α_0.[42] Next a *policy evaluation* step is carried out to compute the value function V_{α_0} implied by the stationary decision rule α_0. This requires solving the linear

[42] One possible choice is $\alpha_0(s) = argmax_{a \in A(s)}[u(s,a)]$.

system (2.27), $V_{\alpha_0} = u_{\alpha_0} + \beta M_{\alpha_0} V_{\alpha_0}$. Once the solution V_{α_0} is obtained, a *policy improvement* step is used to generate an updated policy α_1:

$$\alpha_1(s) = \underset{a \in A(s)}{argmax}[u(s,a) + \beta \sum_{s'=1}^{|S|} V_{\alpha_0}(s') p(s'|s,a)]. \tag{5.7}$$

Given α_1 one continues the cycle of policy valuation and policy improvement steps until the first iteration k such that $\alpha_k = \alpha_{k-1}$ (or alternatively $V_{\alpha_k} = V_{\alpha_{k-1}}$). It is easy to see from (2.27) and (5.7) that such a V_{α_k} satisfies Bellman's equation, so that by Theorem 2.3 the stationary Markovian decision rule $\alpha = \alpha_k$ is optimal. Policy iteration always generates an improved policy, i.e. $V_{\alpha_k} \geq V_{\alpha_{k-1}}$. To see this, note that by definition of the policy improvement step we have:

$$\begin{aligned} G_{\alpha_k}(V_{\alpha_{k-1}})(s) &= \underset{a \in A(s)}{max}[u(s,a) + \beta \sum_{s'=1}^{|S|} V_{\alpha_{k-1}}(s') p(s'|s,a)] \\ &\geq [u(s, \alpha_{k-1}(s)) + \beta \sum_{s'=1}^{|S|} V_{\alpha_{k-1}}(s') p(s'|s, \alpha_{k-1}(s))] \\ &= V_{\alpha_{k-1}}(s). \end{aligned} \tag{5.8}$$

Since the operator G_α is monotonic, we have:

$$V_{\alpha_{k-1}} \leq G_{\alpha_k}(V_{\alpha_{k-1}}) \leq \cdots \leq G^N_{\alpha_k}(V_{\alpha_{k-1}}), \tag{5.9}$$

and since G_α is a contraction mapping it follows that:

$$\lim_{N \to \infty} G^N_{\alpha_k}(V_{\alpha_{k-1}}) = V_{\alpha_k} \geq V_{\alpha_{k-1}}. \tag{5.10}$$

In fact, policy iteration always generates a strict improvement in V_{α_k}, since if $V_{\alpha_k} = V_{\alpha_{k-1}}$ the method has already converged. Since there are only a finite number $|A(1)| \times \cdots \times |A(|S|)|$ of feasible stationary Markov policies, it follows that policy iteration always converges to the optimal decision rule α in a finite number of iterations.

Policy iteration is able to discover the optimal decision rule after testing an amazingly small number of trial policies α_k: in our experience the method typically converges in under 20 iterations. The reason for the fast convergence of this method is closely connected to the very rapid quadratic convergence rates of Newton's method for nonlinear equations: indeed Puterman and Brumelle 1979 showed that policy iteration is a form of the Newton-Kantorovich method for finding a zero to the nonlinear mapping $F : R^{|S|} \to R^{|S|}$ defined by $F(V) = [I - \Gamma](V)$. Since the error bounds

for Newton-Kantorovich iterations are valid for general Banach spaces, it follows that the number of policy iteration steps required to discover the optimal policy is independent of the number of states, $|S|$. However the amount of work per iteration of policy iteration does depend on $|S|$ and is signifi cantly larger than for successive approximations. Since the number of algebraic operations needed to solve the linear system (2.28) for V_{α_k} is $O(|S|^3)$, the standard policy iteration algorithm becomes impractical for $|S|$ much larger than 1,000. [43] To solve very large scale MDP problems, it seems that the best strategy is to use policy iteration, but to only attempt to approximately solve for V_α in each policy evaluation step. It is easy to see that it is unnecessary to obtain an exact solution for V_α in order to uncover the optimal decision rule α. It is only necessary to compute an approximate solution $\hat{V}_\alpha$ to the linear system $V_\alpha = G_\alpha(V_\alpha)$ that is uniformly within a certain distance of the true solution V_α, which we refer to as the *radius* $r(\alpha)$:

$$r(\alpha) = \inf_{s=1,\ldots,|S|} \left| \max_{a \in A(s)} \left[u(s,a) + \beta \sum_{s'=1}^{|S|} V_\alpha(s') p(s'|s,a) \right] - \max_{a \in A(s)/\alpha(s)} \left[u(s,a) + \sum_{s'=1}^{|S|} V_\alpha(s') p(s'|s,a) \right] \right|. \tag{5.11}$$

Intuitively, $r(\alpha)$ is difference in expected discounted utility between the utility maximizing choice $\alpha(s)$ and the next best choice in the set of remaining alternatives, $a \in A(s)/\alpha(s)$, minimized over all $s \in S$. It follows that any approximate solution $\hat{V}_\alpha$ satisfying $\|\hat{V}_\alpha - V_\alpha\| < r(\alpha)$ will generate the same policy α' in the policy evaluation step (5.7). Although the defi nition of $r(\alpha)$ involves the true solution V_α which is what we want to avoid computing exactly, in practice we don't need to know it: as long as the iterative procedure for computing an approximate solution $\hat{V}_\alpha$ yields monotonically decreasing errors (as is true using successive approximations in the modifi ed policy iteration method described below, at least after a suffi ciently large number of iterations) once we fi nd that further increases in the accuracy of $\hat{V}_\alpha$ do not change the policy α' we deduce that we are within $r(\alpha)$ of V_α and no further iterations are necessary. One may even want to consider using even coarser estimates $\hat{V}_\alpha$ that are outside the radius $r(\alpha)$ of V_α in order to further reduce the computational burden of policy iteration. This amounts to fi nding the right trade-off between a larger number of faster approximate policy iteration steps using relatively inaccurate solutions $\hat{V}_\alpha$ to the policy valuation step versus a smaller number of more expensive policy iteration steps with more accurate solutions. An example of this approach is modifi ed policy iteration which we describe below.

[43] Supercomputers using combinations of vector processing and multitasking can now solve dense linear systems exceeding 1,000 equations and unknowns in under 1 CPU second. See for example, Dongarra (1986).

Modified Policy Iteration. This approach uses successive approximations to approximate the solution to the linear system $V_\alpha = G_\alpha(V_\alpha)$ rather than computing the exact solution $V_\alpha = [I - \beta M_\alpha]^{-1} u_\alpha$ at each policy valuation step. The successive approximations iterations for the operator G_α can be further accelerated by using the McQeen-Porteus error bounds described above. The following Theorem of Puterman and Shin (1978) shows that asymptotically, each modified policy iteration is equivalent to performing $N+1$ successive approximation steps.

Theorem 5.2:. *Let α be an optimal policy for a stationary discounted MDP problem, and let α_k be the decision rule generated at step k of the modified policy iteration algorithm that uses N successive approximation steps as an approximate solution for V_{α_k} in each policy evaluation step. If:*

$$\lim_{k\to\infty} |M_{\alpha_k} - M_\alpha| = 0,$$

then

$$\overline{\lim_{N\to\infty}} \frac{|V_{N+1} - V|}{|V_N - V|} = \beta^{N+1}. \tag{5.12}$$

Thus, modified policy iteration can also be thought of as an accelerated form of successive approximations. It can be effective in problems where β is relatively low, although in problems where β is close to 1 it tends to suffer from the same slow convergence properties as successive approximations. However our numerical results in section 3.5 demonstrate that when we use the McQueen-Porteus error bounds to accelerate the use of successive approximations to solve the fixed point problem $V_\alpha = G_\alpha(V_\alpha)$ modified policy iteration comes close to achieving the rapid convergence properties of standard policy iteration, but requires significantly less work to compute each policy valuation step.

Policy Iteration with State Aggregation. The strategy of this class of methods is to use policy iteration and standard linear equation solvers to compute the exact solution $\overline{v}_\alpha$ to a lower dimensional version of (2.28) that groups the $|S|$ elemental states of the original problem into a smaller number K of aggregate states. In an *aggregation step* we choose a partition of the state space $S_1, \ldots, S_K$ (methods for choosing the partition will be described shortly). If aggregate state i has N_i elements, we can define a transition probability matrix $\overline{M}_\alpha$ on the aggregated state space by:

$$\overline{M}_\alpha(i,j) = \frac{1}{N_i} \sum_{s\in S_i} \sum_{s'\in S_j} p(s'|s, \alpha(s)). \tag{5.13}$$

Let $\overline{u}_\alpha$ be an $K \times 1$ vector of the average utilities in each of the K aggregate states. Then if K is sufficiently small, one can use standard Gaussian elimination algorithms to rapidly compute the

exact solution $\overline{v}_\alpha = [I - \beta\overline{M}_\alpha]^{-1}\overline{u}_\alpha$ to the K-state problem. In a *disaggregation step* we use $\overline{v}_\alpha$ of to construct an approximation of the $|S|$-state value function V_α. The partition $S_1, \ldots, S_K$ can be represented by an $|S| \times K$ *partition matrix* W defi ned by:

$$W_{ij} = \begin{cases} 1 & \text{if } i \in S_j \\ 0 & \text{otherwise.} \end{cases} \tag{5.14}$$

Using W we can then compute an approximate solution $\hat{V}_\alpha = W\overline{v}_\alpha$ to the original problem (2.28). Notice this approximate solution will be a step function equal to $\overline{v}_\alpha(i)$ for each elemental state $s \in S_i$.

Bertsekas and Castañon (1989) have shown that a better approach is to use $W\overline{v}_\alpha$ as an additive correction in an iterative procedure that takes an initial estimate V and generates a more accurate solution $\hat{V}$ that avoids the jaggedness of the step function approximation $W\hat{v}_\alpha$. In this case the appropriate formula for $\overline{u}_\alpha$ becomes:

$$\overline{u}_\alpha(i) = \frac{1}{N_i} \sum_{s \in S_i} [G_\alpha(V)(s) - V(s)]. \tag{5.15}$$

To see why (5.15) is apprporiate, suppose there exists an $K \times 1$ vector y that solves the equation

$$V_\alpha = V + Wy, \tag{5.16}$$

where V_α is the solution to (2.28). Using the equations $G_\alpha(V) = u_\alpha + \beta M_\alpha V$ and $V_\alpha = G_\alpha(V_\alpha) = u_\alpha + \beta M_\alpha V_\alpha$ we have:

$$[I - \beta M_\alpha](V_\alpha - V) = G_\alpha(V) - V. \tag{5.17}$$

Multiplying both sides of (5.17) by $(W'W)^{-1}W'$ and substituting $Wy = (V_\alpha - V)$ in the left hand side we obtain:

$$(W'W)^{-1}W'[I - \beta M_\alpha]Wy = (W'W)^{-1}W'(G_\alpha(V) - V). \tag{5.18}$$

Notice that since W is a partition matrix, $(W'W)^{-1}$ is an $K \times K$ diagonal matrix whose i^{th} diagonal element is $1/N_i$. It follows that the solution to equation (5.18) can be written as:

$$\overline{v}_\alpha = [I - \beta\overline{M}_\alpha]^{-1}\overline{u}_\alpha, \tag{5.19}$$

where I is the $K \times K$ identity matrix and $\overline{u}_\alpha$ is given by (5.15). Thus, (5.19) is the appropriate form of the aggregation step when the aggregation term $W\overline{v}_\alpha$ is treated as an additive correction

to an initial $|S| \times 1$ estimate V as in (5.16). An alternative formula for the disaggregation step can be found by applying G_α to both sides of the equation $V_\alpha = V + W\overline{v}_\alpha$ yielding:

$$V_\alpha \simeq G_\alpha(V_\alpha) + \beta M_\alpha W \overline{v}_\alpha. \tag{5.20}$$

Bertsekas and Castañon show that by interspersing a number of successive approximation steps $V_{t+1} = G_\alpha(V_t)$ between each pair of aggregation/disaggregation steps (5.19) and (5.20), one can guarantee that the method will converge to the fixed point V_α. Thus, in the initial stages the additive correction term $W\overline{v}_\alpha$ succeeds in shifting an initial estimate V to a neighborhood of V_α, and this estimate is refined by a small number successive approximation steps. This cycle is then repeated, and as the resulting sequence $\{V_t\}$ converges to V it is easy to see from (5.15) that the additive corrections $W\overline{v}_\alpha$ in (5.16) converge to 0.

The aggregate states can be chosen on basis of a fixed partition of the state space, or can be chosen adaptively after each aggregation step k in order to minimize the residual variation in $G_\alpha(V_k) - V_k$. One way to do this is to divide the total variation $\Delta = max[G_\alpha(V_k) - V_k] - min[G_\alpha(V_k) - V_k]$ into K equal intervals, assigning states with residuals $G_\alpha(V_k)(s) - V_k(s)$ in the highest interval to aggregate state 1, states with residuals in the next highest interval to aggregate state 2, and so on, i.e.

$$s \in S_k \quad \text{if} \quad G_\alpha(V)(s) - V(s) - \underline{b} - (k-1)\Delta \in (0, \Delta], \tag{5.21}$$

where $\underline{b} = min[G_\alpha(V_k) - V_k]$.

5.2.2 Comparison of Methods in the Auto Replacement Problem

As an illustration, we report the results of a comparison of the performance and accuracy of each of the methods presented in the previous section in the finite-state version of the automobile replacement problem in Example 3 of section 2.5, a problem that was originally posed and solved by policy iteration in Howard's 1960 monograph. The parameters of the finite-state problem were chosen to match the analytical solution given in (2.35) in the case where $\lambda = .5$, $\beta = .95$, $c(s) = 200s$, and $\overline{P} - \underline{P} = 100,000$. Calculating the optimal stopping boundary γ in (2.36), we see that it is optimal to replace the automobile when $s_t > \gamma = 52.87$. The corresponding value function is plotted as the solid line in figure 5.1.

An approximate discrete-state version of the problem was solved using $|S| = 100$ states and the same discount factor, cost function, and replacement costs. The exponential specification (2.30)

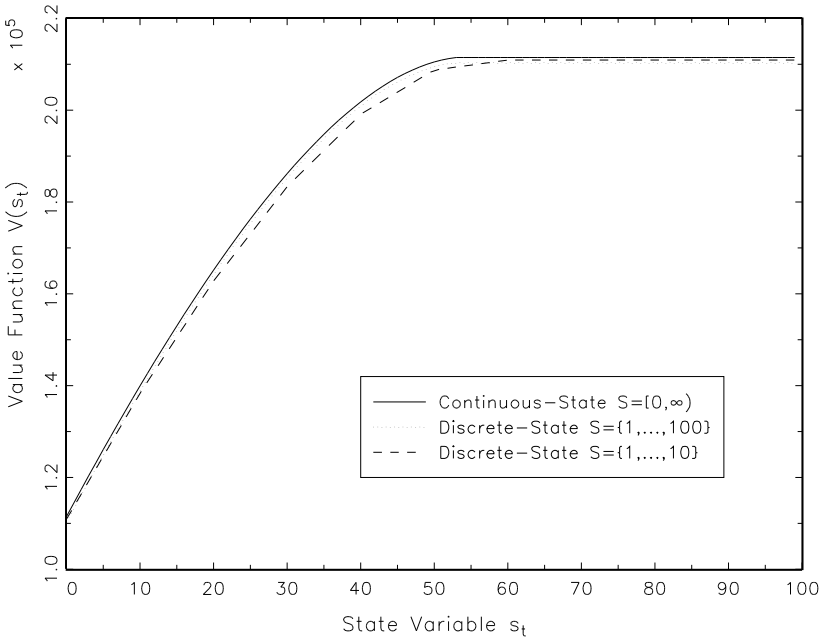

Figure 5.1: Comparison of Discrete vs. Continuous Value Functions in Auto Replacement Problem

for $p(\cdot|s_t, d_t)$ in the continuous case was approximated with a 12-point probability distribution in the discrete case, using a simple continuity correction. Thus, each of the 12 mass points were computed as the probability an exponential random variable falls within plus or minus $.5$ of the integer values j that s_t assumes in the discrete case:

$$p(s_{t+1} = s_t + j|s_t, d_t) = \begin{cases} \int_0^{.5} \lambda \exp^{-\lambda y}\, dy & j = 0 \\ \int_{j-.5}^{j+.5} \lambda \exp^{-\lambda y}\, dy & j = 1, \ldots, 10 \\ \int_{10.5}^{\infty} \lambda \exp^{-\lambda y}\, dy & j = 11. \end{cases} \tag{5.22}$$

The discrete-state value function (computed by policy iteration) is plotted as the dotted line in figure 5.1, with one dot for each point in the state space. One can see from figure 5.1 that the discrete-state value function approximates the continuous-state version quite closely: the maximum absolute deviation between the two functions was 1164, and the maximum percentage deviation was just over 1%. Recall from our previous discussion, that in discrete MDP's it is not necessary to solve the policy valuation step $V_\alpha = G_\alpha(V_\alpha)$ in order to recover the optimal policy α: any approximate solution $\hat{V}_\alpha$ satisfying $\|\hat{V}_\alpha - V_\alpha\| < r(\alpha)$ also generates the same optimal policy α. The radius of the optimal policy α when $|S| = 100$ is $r(\alpha) = 98.3$, which can be regarded as an

alternative benchmark for the level of accuracy necessary in an approximate solution $\hat{V}_\alpha$. Figure 5.1 also plots an interpolated value function solved with only $|S| = 10$ states. The maximum error in this case is 3553, representing a 1.8% deviation. One can see that even very coarse discretizations are able to do a good job of approximating a continuous underlying value function. This provides some insight into the potential effectiveness of solution methods based on state aggregation.

Tables 5.1 and 5.2 present a comparison of six alternative solution algorithms used to solve the 100 state replacement problem. [44] The tables present the number of iterations and CPU times required by each algorithm to compute an estimate of V to within a tolerance of 1164, the maximum deviation between the continuous and discrete-state formulas for V. [45] The modified policy iteration algorithm used $N = 20$ successive approximation steps to compute an approximate fixed point of G_α. The McQueen-Porteus error bounds were employed to test for convergence and produce an improved final estimate of V_α. [46] The state aggregation methods used $M = 10$ aggregate states. The fixed state method simply partitioned the states into 10 equal groups, $S_i = \{10*(i-1) + 1, \ldots, 10*i\}$. The adaptive state aggregation method partitioned the states endogenously, according to the magnitude of the residuals $G_\alpha(V) - V$ as given in (5.21). In both of the aggregation methods, we set a relatively loose inner convergence tolerance: iterate V_k was deemed to be a sufficiently good approximation of the fixed point $V_\alpha = G_\alpha(V_\alpha)$ if the McQueen-Porteus error bounds $\underline{b}_k$, $\overline{b}_k$ for the operator G_α satisfy $\overline{b}_k - \underline{b}_k < 30000(1-\beta)/\beta$. As soon as the policy converged $(\alpha_k = \alpha_{k-1})$, additional policy evaluation steps were carried out with a tighter convergence tolerance of $500(1-\beta)/\beta$ until the resulting estimate for V_{α_k} was within 1164 of V (computed via the McQueen-Porteus error bounds for Γ). Table 5.2 presents a comparison of the six methods in the case where $\beta = .9999$.

---

[44] Each of these methods were programmed in Gauss and run on an IBM 386/SX computer. The code, also written in Matlab and C, is available from the author upon request.

[45] Successive approximations were terminated when the crude estimate of the maximum deviation between V_t and V was less than 1164. Successive approximations with error bounds were stopped when the more refined McQeen-Porteus error bounds indicated that the maximum deviation of V_t and V was less than 1164. The various policy iteration algorithms were terminated at the first iteration k such that $\alpha_k = \alpha_{k-1}$, although in the case of the approximate policy iteration algorithms, additional steps were continued until the McQueen-Porteus error bounds for the operator Γ indicated that the estimated value function V_{α_k} was within 1164 of V.

[46] More precisely, one successive approximations step $V_{21} = \Gamma(V_{20})$ was performed using the operator Γ after 20 successive approximations steps $V_{t+1} = G_{\alpha_k}(V_t)$ were performed with the operator G_{α_k}. The final estimate of V_{α_k} in modified policy iteration step k is then given by: $\hat{V}_k = V_{21} + (\overline{b} + \underline{b})/2$ where $\overline{b}$ and $\underline{b}$ are the McQueen-Porteus error bounds for the operator Γ.

Method	Iterations	CPU Seconds	$\mid V - V^* \mid$	$\overline{b} - \underline{b}$	$\mid V - \Gamma(V) \mid$
1. Successive Approximations	114	48.9	571.7	26.5	30.6
2. Error Bounds	65	29.7	503.4	1141.0	48.5
3. Policy Iteration	6	46.1	$1.1E^{-9}$	$1.1E^{-9}$	$5.8E^{-11}$
4. Modified Policy Iteration	5	21.8	52.3	301.4	8.6
5. Fixed State Aggregation	6	31.4	20.9	336.0	8.8
6. Adaptive State Aggregation	8	171.8	40.4	291.2	7.7

Table 5.1: Comparison of Solution Methods $\beta = .95$

Method	Iterations	CPU Seconds	$\mid V - V^* \mid$	$\overline{b} - \underline{b}$	$\mid V - \Gamma(V) \mid$
1. Successive Approximations	$> 10,000$	$> 4,600$	$> 30,000$	$1.1E^{-8}$	3.0
2. Error Bounds	166	75.7	$4.4E^{-1}$	114.5	$5.7E^{-3}$
3. Policy Iteration	8	71.0	$2.9E^{-7}$	$2.9E^{-7}$	$1.5E^{-11}$
4. Modified Policy Iteration	11	50.0	174.9	219.4	$2.9E^{-2}$
5. Fixed State Aggregation	10	51.9	3.4	93.8	$4.7E^{-3}$
6. Adaptive State Aggregation	15	1296	$2.4E^{-1}$	58.4	$2.9E^{-3}$

Table 5.2: Comparison of Solution Methods $\beta = .9999$

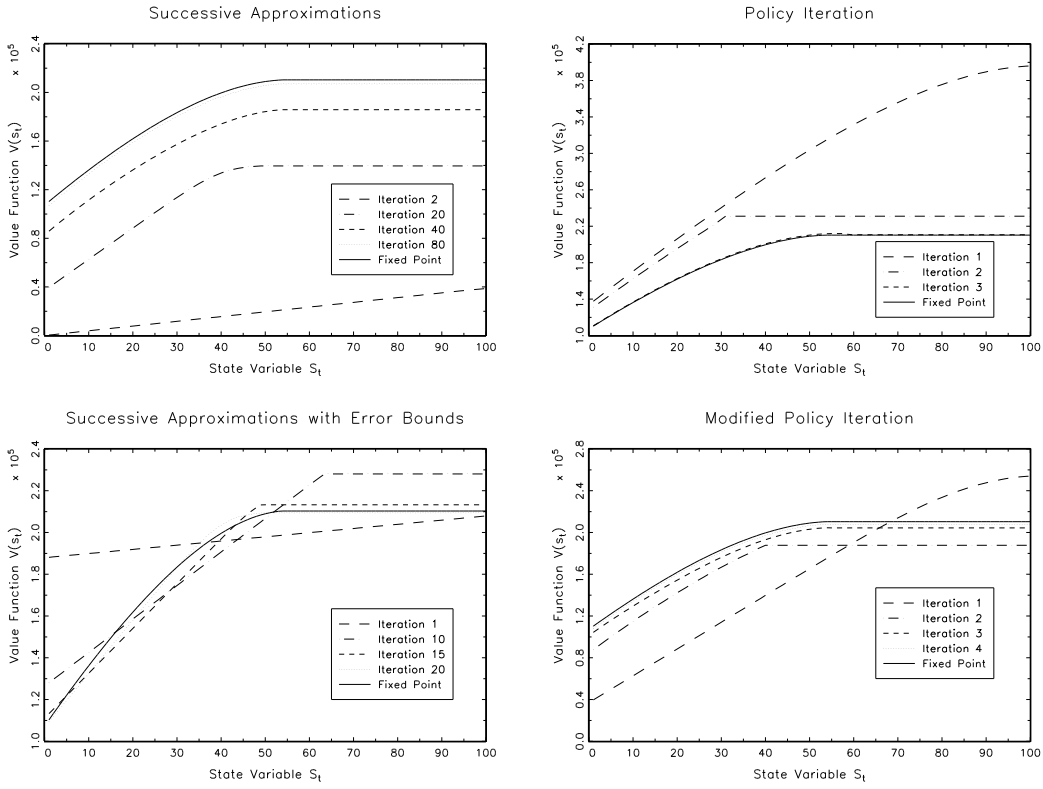

Figure 5.2: Convergence of V_t to V when $\beta = .95$

Figure 5.2 illustrates the convergence trajectories of 4 of the algorithms. Successive approximations converges to V from below, reflecting the fact that it is solving a finite-horizon approximation to the infinite-horizon problem. Policy iteration converges to V from above, reflecting the fact that each successive policy α_t represents an improvement over the previous policy as established in (5.10). Convergence to V under the error bounds and modified policy iteration procedures is not necessarily monotonic. In the former case one can see that the addition of the correction term $(\overline{b}+\underline{b})/2$ quickly translates the iterates V_t into a region much closer to V (compare the first iterate V_1 using the error bounds correction to the the second iterate V_2 under successive approximations). Once in a general neighborhood of V, successive approximations succeed in "bending" V_t into the precise shape of V. Modified policy iteration is not monotonic because approximating the fixed point $V_\alpha = G_\alpha(V_\alpha)$ by N successive approximation steps implies that each iterate $V_k = G^N_{\alpha_k}(V_{k-1})$ will be an underestimate of the true solution V_{α_k}. Thus, the sequence $\{V_k\}$ generated by modified policy iteration is essentially the same as under policy iteration, but translated downward. In this case the sequence $\{V_k\}$ ended up converging to V from below. Note also the inexactness of the approximate solutions under modified policy iteration imply that more iterations are required to get close to V in comparison to policy iteration.

Overall the results indicate that in this problem policy iteration using an approximate rather than exact solution to the inner fixed point problem $V_\alpha = G_\alpha(V_\alpha)$ was the fastest of the methods considered. In the early stages of policy iteration, it is not necessary to solve (2.28) exactly to insure rapid progress towards V and the optimal decision rule α. Since the decision rules α_k are the results of finite maximizations, V_t does not have to be an extremely precise estimate of V to insure that the corresponding policy α_t coincides with α. However once $\alpha_t = \alpha$ (which is indicated by the fact that α_t does not change in successive steps of the policy iteration) one can set finer tolerances for the approximate solution of $V_\alpha = G_\alpha(V_\alpha)$ in (2.28) to generate a more accurate estimate of V and insure that the solution α_k really does correspond to the optimal decision rule α. We found that the Bertsekas-Castañon adaptive state aggregation procedure was not effective in this particular problem. We noticed that the adaptively chosen states typically varied quite substantially over successive aggregation steps. The variations in the membership of the aggregate states frequently resulted in an approximate disaggregate solution $G_\alpha(V) + \beta P_\alpha W \overline{v}_\alpha$ that would tend to move away from the fixed point V_α, requiring a large number of intervening successive approximations steps to move the solution back towards V_α. Use of a fixed set of aggregate states performed much more effectively in this problem, reflecting our ability to closely approximate a

continuous value function using very coarse discretizations as demonstrated in figure 5.1.[47] The McQueen-Porteus error bounds are also a very cost-effective method for accelerating convergence, particularly when used to accelerate the method of successive approximations for the fixed point problem $V_\alpha = G_\alpha(V_\alpha)$ encountered at each of the policy valuation steps.

5.2.3 New Approaches to Solving Large Scale Problems

The rest of this subsection presents several of the most promising recent ideas for speeding up the solution to large scale discrete infinite horizon MDP's. The first example describes the use of linear programming methods as a potentially attractive alternative to policy iteration. The second example describes an idea of Pan and Reif 1985 to use massive parallel processors and Newton's method to solve this linear system in $O(\log(|S|)^2)$ time using $O(|S|^\omega)$ processors, where $\omega \leq 2.376$ is the best lower bound on fast matrix multiplication. The third example discusses the use of a parametric approximation method known as the "minimum residual algorithm" (MR) to approximately solve the system of linear equations involved in carrying out the policy iteration algorithm. For certain MDP's with sparse transition probabilities the MR algorithm can reduce the time required to solve the policy valuation step from $O(|S|^3)$ to as little as $O(|S|)$ time depending on the sparsity of the transition probability matrix and the level of precision ϵ required in the solution.

Linear Programming. It is well known (see, e.g. Bertsekas, 1987) that the value function for a discrete infinite horizon MDP problem is the solution to the following linear programming problem:

$$\min_{V \in R^{|S|}} \sum_{s=1}^{|S|} V(s) \tag{5.23}$$

subject to:

$$V(s) \geq u(s,a) + \beta \sum_{s'=1}^{|S|} V(s')p(s'|s,a), \quad a = 1, \ldots, |A(s)|, \;\; s = 1, \ldots, |S|. \tag{5.24}$$

Many people have presumed that solving the MDP problem as an LP problem is a relatively inefficient way to solve large-scale MDP's. However a recent paper by Trick and Zin 1993

[47] However in less structured problems it may not be clear how to choose the aggregate states, and in problems with multiple ergodic classes use of a fixed set of aggregate states is typically ineffective. In these problems, Bertsekas and Castañon (1989) have found that the adaptive aggregation method can outperform the modified policy iteration algorithm.

observes that generally only $|S|$ of the $|A||S|$ constraints (5.24) will be binding since there will generally be a unique optimal action. Therefore they propose solving the LP problem using a technique known as *constraint generation* which solves an expanding sequence of LP's starting from a very small LP problem that only imposes a small subset of the constraints (5.24) and then sequentially re-solves the LP problem adding discarded constraints that were violated by the previous trial solution until the optimal solution V is found. They claim that constraint generation has been "spectacularly successful" in solving large scale LP's in other applications including "a formulation for the traveling salesman problem that is estimated to have 2^{60} constraints is solved in a matter of hours on a workstation." (p. 8). Trick and Zin used constraint generation to solve a discretized version of the continuous MDP test problem used in the *JBES* 1990 "horse race" (see example 2 in section 2.5). Their discretization used a two-state Markov chain approximation for the exogenous "technology shock" process $\{z_t\}$ and varying size discretizations for the endogenous capital stock variable k_t. They found that even without employing constraint generation, the standard commercial linear programming code "CPLEX", was able to solve the MDP problem between 33 to 13 times faster than policy iteration as the number of capital grid points ranged from 33 to 513. By switching to constraint generation they were able to speed up the solution of the 513 state problem by another 2.5 times, or a total speedup of nearly 33 times relative to policy iteration. Trick and Zin also describe multigrid versions of their LP/MDP constraint generation algorithm that are similar in many respects to the Bertsekas- Castañon adaptive aggregation approach to policy iteration. Their first version, which they refer to as "grid generation" begins by solving the LP problem using only a subset of the states $s \in S$ and then sequentially adds additional states to the linear programming problem using the previous solution as a starting point to the problem with a larger set of states. A second version, which they refer to as "adaptive grid generation" is described as follows:

> "We begin by optimizing over a coarse evenly spaced grid using constraint generation. Each state s on this coarse grid has an optimal action $\alpha(s)$ and a shadow price $\lambda(s)$ generated by the LP solution. This shadow price measures the impact that on the sum of the value function ordinates (the objective function (5.23)), of a small change in the constraint, specifically $u(s, \alpha(s))$. We next calculate the *slack* of each constraint adjacent to to the constraint for $\alpha(s)$. This amounts to finding the change in u that would result in the constraint holding with equality. We then multiply these slacks by the shadow price to obtain a measure of the impact on the objective function of placing a new grid point adjacent to $\alpha(s)$. We do this for all s. We then add the actions and the states corresponding to the highest values of this product of shadow price and slack. The new points are always the midpoint between two existing points." (paraphrase of Trick and Zin 1993 p. 17 in the notation of this chapter).

Trick and Zin found that their adaptive grid generation procedure does a better job of approximating the underlying continuous state value function than methods that use a fixed grid, especially in regions where the value function has lots of curvature since the method tends to place more grid points in these regions.

Massively Parallel Policy Iteration Although the results of Trick and Zin are very encouraging, it is an open question whether LP solution methods can dominate approximate policy iteration methods on massively parallel architectures. As we discussed in section 3.1, both the LP problem and the MDP problem are members of the class of P*-complete* problems under the Turing model of computation. Specifically, Papadimitriou and Tsitsiklis 1987 proved that the MDP problem is P-complete by showing that a known P-complete problem, the *Circuit value problem*, reduces to and thus can be solved as a special case of the general MDP problem. This implies that the MDP problem is a member of the equivalence class of the "hardest" polynomial time problems including linear programming. Given that most computer scientists believe that $P \neq NC$, there is a strong presumption that neither the LP or MDP problems are in the class *NC* of problems that can be effectively parallelized. As we discussed in section 3.1, these results are based on the algebraic notion of computational complexity that assumes exact solutions are required. We now show that if we are willing to allow approximate solutions with an arbitrarily small error $\epsilon > 0$, then the MDP problem can be effectively parallelized using policy iteration and the massively parallel linear equation solver of Pan and Reif 1985 to approximately solve each policy valuation step. The Pan-Reif algorithm uses Newton's method to compute an approximate inverse of the $|S| \times |S|$ matrix $M = [I - \beta M_\alpha]$ of the linear system that must be solved at each policy iteration step in $O(\log(|S|)^2)$ time using $O(|S|^\omega)$ processors, where ω is the best lower bound on fast matrix multiplication, currently $\omega = 2.376$. The Newton algorithm for matrix inversion works as follows. Given an initial estimate L_0 of the inverse M^{-1} which satisfies $\|I - L_0 M\| = q < 1$, one can use Newton's method to solve the matrix equation $F(X) = [I - XM] = 0$ yielding iterations of the form:

$$L_k = (2I - L_{k-1}M)L_{k-1}. \tag{5.25}$$

An alternative iterative procedure that yields similar results is

$$L_k = L_0 \prod_{i=0}^{k-1} [I + (I - L_0 M)]^{2^i}. \tag{5.26}$$

As is characteristic of Newton's method, the iterates L_k converge quadratically to the zero M^{-1} of $F(X)$:

$$\|M^{-1} - L_k\| \leq \frac{q^{2^k}\|L_0\|}{(1-q)}. \tag{5.27}$$

Using this error bound, Pan and Reif established the following lemma:

Lemma 5.1 *Suppose that the initial $|S| \times |S|$ starting matrix L_0 for Newton's iteration (5.25) is chosen so that $q \equiv \|I - L_0 M\|$ satisfies:*

$$q = 1 - \frac{1}{|S|^{O(1)}} \quad \textit{as} \quad |S| \to \infty. \tag{5.28}$$

Let c be a positive constant. Then $O(\log(|S|)$ iterations of the Newton algorithm suffice in order to compute an approximate inverse $\hat{M}^{-1}$ that satisfies:

$$\|M^{-1} - \hat{M}^{-1}\| \leq \frac{1}{2^{n^c}}\|L_0\|. \tag{5.29}$$

Now observe that Newton's method involves repeated multiplication of $|S| \times |S|$ matrices. This can be done in $O(\log(|S|))$ time using $O(|S|^{\omega})$ processors where $\omega \leq 2.376$. By Lemma 5.1 only $O(\log(|S|))$ Newton iterations are required to obtain an approximate inverse L_k satisfying (5.29). Therefore we have the following theorem:

Theorem 5.3: *Suppose that the initial estimate L_0 of M^{-1} satisfies inequality (5.28), and let c be a positive constant. Then a massive parallel processor with $O(|S|^{\omega})$ processors can find an approximate solution L_k satisfying the convergence criterion (5.29) in $O(\log(|S|)^2)$ time.*

Since policy iteration is also a version of Newton's method, we can use the Pan-Reif Newton algorithm to generate a suffi ciently precise approximate solution to the value function $\hat{V}_\alpha$ at each policy iteration step, guaranteeing that policy iteration using the approximate solution $\hat{V}_\alpha$ is still quadratically convergent. Since the number of policy iteration steps required to converge to the optimal policy is independent of the number of states $|S|$ we have the following result:

Theorem 5.4: *Let $\epsilon > 0$ be an arbitrary solution tolerance between an approximate solution $\hat{V}$ and the true solution $V = \Gamma(V)$ to an $|S|$-state infinite horizon MDP problem. Then the massively parallel policy iteration algorithm using the Pan-Reif algorithm to approximately solve each policy valuation step generates an approximate solution $\hat{V}$ satisfying $\|\hat{V} - V\| < \epsilon$ in at most in $O(\log(\log(1/\epsilon))\log(|S|)^2)$ time steps using a massively parallel processor with $O(|S|^{\omega})$ processors.*

The proof of this result is given in Rust 1994d. The significance is that it proves that the MDP problem can be efficiently parallelized. Rust's result does not contradict the results of Papadimitriou and Tsitsiklis 1987 since it is based on the real rather than Turing model of computation and uses the continuous rather than algebraic notion of computational complexity. [48] The algorithm is numerically stable since the Pan-Reif algorithm is numerically stable and self-correcting. Furthermore, a starting estimate L_0 satisfying inequality (5.28) can always be obtained from the Neumann expansion:

$$L_0 = I + \beta M_\alpha + \beta^2 M_\alpha^2 + \cdots + \beta^J M_\alpha^J, \tag{5.30}$$

for a sufficiently large integer J.[49]

Policy Iteration Using Minimum Residual Algorithms. The result of the previous section is currently primarily of theoretical interest due to the fact that most economists do not have access to the truly massive parallel processing technology required by the Pan-Reif method as well as the fact that fast matrix multiplication algorithms required by the Pan-Reif algorithm described above are not yet widely available in standard numerical libraries. A further drawback to the use of Newton's method to invert the $|S| \times |S|$ matrix $I - \beta M_\alpha$ is that it involves unnecessary work if our objective is to simply solve a single system of linear equations $V_\alpha = u_\alpha + \beta M_\alpha V_\alpha$ at each policy valuation step. We now describe a class of "minimum residual algorithms" (MR) for solving this system that is highly efficient, especially in cases where the transition probability matrix M_α corresponding to $p(s'|s,a)$ is sparse. MR algorithms have been proven extremely effective in solving very large scale systems arising in chemistry and physics applications. The MR algorithm is similar to modified policy iteration in that it is an iterative method that only attempts to approximately solve the linear system $V_\alpha = G_\alpha(V_\alpha)$ at each policy valuation step. In order to motivate the method, recall that modified policy iteration produces an approximate solution $\hat{V}_\alpha$ given by the following linear combination:

$$\hat{V}_\alpha = \sum_{i=1}^{k} \beta^i M_\alpha^i u_\alpha, \tag{5.31}$$

---

[48] Traub 1994 cites other examples of problems that appear more difficult under the Turing model of computation and algebraic notion of complexity than the real model of computation and continuous notion of complexity. It is our belief that the latter framework is more relevant for understanding the behavior of actual numerical methods even though actual computers are only finite precision devices.

[49] Pan and Reif outline faster methods for obtaining a starting estimate L_0 that can be implemented in $O(\log(|S|))$ time using $|S|^2$ processors.

where M_α is the $|S| \times |S|$ matrix representation of $p(s'|s, \alpha(s))$ and u_α is the $|S| \times 1$ vector with elements $u(s, \alpha(s))$. If M_α is a dense matrix, each policy evaluation step of modifi ed policy iteration requires $O(k|S|^2)$ operations, since each matrix-vector multiplication requires $O(|S|^2)$ operations. Modifi ed policy iteration will dominate ordinary policy iteration if k can be substantially less than $|S|$ without substantially increasing the total number of iterations necessary to achieve convergence. In addition if M_α is a sparse matrix with $O(|S|)$ non-zero elements, then only $O(k|S|)$ operations are required to carry out each policy evaluation step.

We now show that there are better methods for approximately solving the policy valuation step that require a signifi cantly smaller number k of matrix-vector multiplications than required by modifi ed policy iteration to obtain approximate solutions $\hat{V}_\alpha$ of specifi ed accuracy. We can write the linear system (2.28) in generic form as:

$$u = LV, \tag{5.32}$$

where the matrix $L = [I - \beta M]$ has condition number $\|L\|\|L^{-1}\|$ that is bounded above by 1. The MR method computes an approximate solution to the linear system (5.32) as the linear combination V_k given by:

$$\hat{V}_k = \sum_{i=1}^{k} c_i^* L^{i-1} u, \tag{5.33}$$

where the coeffi cients $(c_1^*, \ldots, c_k^*)$ are chosen by the method of ordinary least squares minimize the residual difference $\|u - M\hat{V}_k\|_2$:

$$(c_1^*, \ldots, c_k^*) = \underset{(c_1,\ldots,c_k)\in R^k}{argmin} \|u - \sum_{i=1}^{k} c_i L^{i-1} u\|_2. \tag{5.34}$$

The MR algorithm is just one example of many algorithms that use the *Krylov information:* [50]

$$I_k(L, u) = (u, Lu, L^2u, \ldots, L^k u). \tag{5.35}$$

Let F denote the class of *orthogonally invariant matrices.* [51] Traub and Woźniakowski 1984 proved that the MR algorithm is nearly optimal for solving the linear system (5.32) provided the matrix L is in the class F of orthogonally invariant matrices. More precisely, they proved that the

[50] Other methods that use this information include the generalized conjugate residual method (GCR), the Chebyshev algorithm, and simple successive approximations.

[51] A class of matrices F is orthogonally invariant if for each $L \in F$ we have $QLQ' \in F$ for any orthogonal matrix Q. It is easy to verify that the class of all matrices with uniformly bounded condition numbers is orthogonally invariant.

value of k required by the MR algorithm to produce an ϵ-approximation of the true solution V to (5.32) is at most one step larger than the lower bound $k(\epsilon, F)$ on the value of k required to produce an ϵ-approximation for each $L \in F$. [52] Further Chou 1987 proved that the Krylov information is almost optimal in the sense that the value of $k(\epsilon, F)$ under the Krylov information is at most twice the lower bound on this index using the "optimal information" for this problem. Traub and Woźniakowski derived an explicit expression for the value of k required by the MR algorithm to obtain an ϵ-approximation for the class of matrices F defined by $F = \{L|L = I - \beta M,\ \|M\| \leq 1\}$. Notice that this class includes the matrices arising in MDP problems. For this class F their formula for the minimal value of k, denoted by $k_{MR}(\epsilon, F)$, is given by:

$$k_{MR}(\epsilon, F) = min\left(|S|, \left\lfloor \frac{\log(\epsilon)}{\log(\beta)}(1-\delta) \right\rfloor + 1\right), \tag{5.36}$$

where $\delta \in [0, \log(1 - \beta^2 + \beta^2\epsilon^2)/\log(\epsilon^2)]$. For problems where extremely high precision is not required we have $k_{MR}(\epsilon, F) \ll |S|$, so if the matrix L is sparse with $O(|S|)$ nonzero elements, then the total work involved in carrying out each policy evaluation step using the MR algorithm is only $O(|S|)$ compared to $O(|S|^3)$ using standard linear equation solvers.

Saad and Schultz 1986 present numerical comparisons of a generalized iterative version of the MR algorithm known as GMRES. This algorithm has been successfully used to solve extremely high-dimensional systems in chemistry and physics. [53] The GMRES algorithm uses Gram-Schmidt orthogonalization of the Krylov information $(u, Lu, \ldots, L^k u)$ to allow sequential solution of the least squares problem (5.34). This allows periodic restarting of the GMRES algorithm and adaptive monitoring of the approximation error $\|u - L\hat{V}_k\|$, allowing one to stop the algorithm when desired solution tolerance ϵ is obtained rather than precommitting *ex ante* to a fixed value of $k_{MR}(\epsilon, F)$. Elman 1982 proved that the sequence of iterates from the GMRES algorithm with m restarts satisfy the following error bound:

$$\|u - L\hat{V}_k\| \leq \left[1 - \frac{1}{\gamma}\right]^{m/2} \|u - L\hat{V}_0\|, \tag{5.37}$$

where $\gamma = \lambda_{max}(L'L)$, and $\lambda_{max}(L'L)$ denotes the largest eigenvalue of the matrix $L'L$, (which is equal to $(1-\beta)^2$ in this case). This formula shows that increasing the number of restarts

[52] The function $k(\epsilon, F)$ is called the *optimal class index* and can be regarded as a type of complexity bound, see Traub and Woźniakowski 1984 for a definition.

[53] Some applications report using GMRES to approximately solve systems with over 100,000 equations and unknowns.

improves the rate of convergence of the GMRES algorithm. In their numerical experiments Saad and Schultz 1986 found that $m \in [5, 20]$ seemed to work well, yielding significant speed-ups over other approximate solution methods. The fact that GMRES is considered to be an "industrial strength" equation solver in numerical chemistry and physics suggests that this algorithm could have substantial promise in economic problems for use as a subroutine to approximately solve the policy valuation steps in discrete MDP's with very large numbers of states.

5.3 Continuous Finite Horizon MDP's

Recall from section 2.1 that continuous MDP's have state variables that can assume a continuum of possible values. For concreteness, we will assume here that the state space S is an uncountable subset of the d_s-dimensional Euclidean space R^{d_s}. As we noted in section 2.1, there two main subclasses of continuous MDP's: 1) discrete decision processes (DDP's) for which the action set A is finite, and 2) continuous decision processes (CDP's) for which the constraint set $A(s)$ is a compact, convex subset of the d_a-dimensional Euclidean space R^{d_a}. We will see that there are important differences in the solution algorithms and associated computational complexity bounds for CDP's and DDP's. In particular, we show that there are randomized algorithms that succeed in breaking the curse of dimensionality of approximating the solution to DDP's whereas no deterministic or random algorithm succeeds in breaking the curse of dimensionality for CDP's, at least on a worst case basis.

Since the solution to a continuous T-period MDP problem is the sequence of value functions $V = (V_0, \ldots, V_T)$ and each V_t is an element of the infinite-dimensional Banach space $\mathcal{B}(S)$ it is clear that it is generally impossible to compute the solution exactly, although under certain conditions it will be possible to approximate the true solution to within an arbitrarily small tolerance ϵ. We organize this section around the two general approaches to approximation: section 5.3.1 reviews discrete approximation methods and section 5.3.2 reviews smooth approximation methods. The analysis of approximation methods is complicated by the fact that many MDP's are naturally formulated on unbounded state and action spaces. Examples of MDP's with unbounded state spaces include the auto replacement problem or the Brock-Mirman stochastic growth model in section 2. The analysis of this section assumes that the state and action spaces are compact subsets of Euclidean space. This assumption is without significant loss of generality since under very general conditions we can truncate a problem with unbounded state and action spaces and analyze an approximate truncated problem with compact sets S and A with the property that the value function for the truncated problem is arbitrarily close to the value function for the original

untruncated problem over the truncated state space S. Since it is usually reasonably clear how to form an approximate truncated problem, we do not provide a general analysis of the problem here although it is useful to briefly illustrate some of the issues involved by considering the automobile replacement example introduced in section 2.5.

Recall that the value function $V(s)$ in the auto problem is constant for $s \geq \gamma$. Thus, it is natural to consider the truncated state space $S = [0, \gamma]$. However the value γ is not known *a priori* and imposing a value that is too small could cause the truncated problem to poorly approximate the true solution. So instead we need to consider a truncated version of this problem defined on a compact interval $[0, \overline{s}]$ where $\overline{s}$ is an upper bound on the stopping threshold γ which can be thought of as a maximum state of degradation of the automobile. In the auto problem it is easy to compute an upper bound to γ using the equation (2.42) determining the optimal stopping threshold γ. The solution $\overline{s}$ to the equation $c(\overline{s}) - c(0) = \overline{P} - \underline{P}$ can be easily shown to be an upper bound on γ, and the value function for the truncated problem on the compact interval $[0, \overline{s}]$ will coincide with the value function to the unbounded problem over this same interval. We used the value $\overline{s}$ in the numerical solution of the discretized version of the automobile replacement problem in the previous section. In most problems encountered in economic applications, it seems reasonable to assume that the truncation error will be negligible in comparison to the approximation errors resulting from discretization of the state space. Since nearly all approximation methods assume compact state and action spaces we will henceforth assume that an appropriate truncation of the original unbounded MDP problem has already been chosen in order to focus on the more important issue of characterizing efficient approximation methods for continuous MDP's.

5.3.1 Discrete Approximation Methods

Discrete approximation methods compute the value functions V_t and decision rules α_t, $t = 1, \ldots, T$ at a finite set of points in the state space S and constraint sets $A(s)$. We will refer to a finite subset of points of S as a *grid*, and denote it by $\{s_1, \ldots, s_N\}$. Note that this definition does presume that the points in the grid need to be evenly spaced. There are many variants of discrete approximation, but the general structure can be described as follows. One first chooses grids for the state and action spaces, and then performs backward induction for a finite MDP problem defined on these grids. Thus, discretization amounts to replacing the continuous-state version of the backward induction step

$$V_t(s) = \Gamma(V_{t+1})(s) \equiv \max_{a \in A(s)} [u(s,a) + \beta \int V_{t+1}(s') p(ds'|s,a)], \quad s \in S \tag{5.38}$$

by a discretized equivalent

$$\hat{V}_t(s) = \hat{\Gamma}_N(\hat{V}_{t+1})(s) \equiv \max_{a \in \hat{A}(s)} [u(s,a) + \beta \sum_{k=1}^{N} \hat{V}_{t+1}(s_k) p_N(s_k|s,a)], \quad s \in \{s_1, \ldots, s_N\}. \tag{5.39}$$

We can represent the recursion (5.39) more compactly as successive applications of $\hat{\Gamma}_N$ starting from the 0 function:

$$\hat{V}_t = \hat{\Gamma}_N^{T-t}(0), \quad t = 0, \ldots, T. \tag{5.40}$$

Note there is no particular reason to assume that the number of points N making up the grid of S is the same as the number of grid points making up the grids of the constraint sets $A(s)$. Also, there is no reason why one can't choose different grids for each constraint set $A(s_i)$, $i = 1, \ldots, N$. Indeed, it is not even necessary to discretize the constraint sets if continuous optimization methods such as gradient hill climbing algorithms, the Nelder-Mead 1965 polytope algorithm, or stochastic search algorithms such as simulated annealing are used to solve the maximization problem in (5.39). [54] The key restriction is that the state space is discretized, which implies that discrete approximation methods amount to a replacement of the true Bellman operator $\Gamma : \mathcal{B}(S) \to \mathcal{B}(S)$ by an approximate "discretized" Bellman operator $\hat{\Gamma}_N : R^N \to R^N$ but otherwise the backward induction process is identical. However even though the discrete solution $\hat{V} = (\hat{V}_0, \ldots, \hat{V}_T)$ consists of $T+1$ vectors in R^N, we can use these vectors as "data" for a variety of interpolation or "smoothing" methods to produce continuous solutions defi ned over the entire state space S. For example, in our plots of the numerical solutions to the automobile replacement problem in fi gure 5.1 we used simple linear interpolations of successive values of the $|S| \times 1$ vector $\hat{V}$. In multidimensional problems other more elaborate interpolation schemes can be employed such as cubic splines (which guarantee that the interpolated function is twice continuously differentiable at the "knot points" $\{s_1, \ldots, s_N\}$), or various versions of local polynomial fi ts such multivariate Hermite polynomials and tensor product splines. The details of these various interpolation and approximation methods will be presented in our survey of smooth approximation methods in the next subsection.

All of the discrete approximation methods that we will be presenting in this subsection have discretized transition probabilities $p_N(s_k|s,a)$ that are nice functions (i.e. continuous or

[54] To keep the length of this chapter within bounds, we do not survey the wide range of algorithms available for solving the embedded constrained optimization problems required to evaluate (5.39). We refer the reader to Gill, Murray and Wright 1985 for an excellent survey of constrained and unconstrained optimization methods. Hwang 1980 and Solis and Wets 1981 are good starting points for the theory of and practical experience with randomized maximization methods. Geman and Hwang 1986, Goffe, Ferrier and Rogers 1992, and Bertsekas and Tsitsiklis 1993 are good starting points for the theory of and practical experience with simulated annealing methods.

continuously differentiable) of s. This is significant since it implies that the the approximate Bellman operator $\hat{\Gamma}_N$ has a very important property: it is *self-approximating,* i.e. we can evaluate $\hat{\Gamma}_N(\hat{V}_t)$ at any point s in the continuous state space S and not just at the finite set of grid points $\{s_1, \ldots, s_N\}$. Furthermore $\hat{V}_t(s) = \hat{\Gamma}_N(\hat{V}_{t+1})(s)$ is a maximum of continuous functions of s and is therefore a well-defined continuous function of s. The self-approximating property leads to a dual interpretation of $\hat{\Gamma}_N$: we can regard it as a contraction mapping on R^N when we consider its restriction to the finite grid $\{s_1, \ldots, s_N\}$ or we can regard it as a contraction mapping directly on the infinite-dimensional space $C(S)$. This simplifies computation since we don't need to employ any auxiliary interpolation, approximation or smoothing methods to evaluate $\hat{\Gamma}_N(W)$ at any point $s \in S$, and it also makes it easier to analyze the convergence of these methods as $N \to \infty$. However the most important reason for emphasizing the self-approximating property is that under certain circumstances it allows us to break the curse of dimensionality of approximating the value functions to certain MDP's. More specifically, we will show that backward induction using a *random Bellman operator* $\tilde{\Gamma}_N$ breaks the curse of dimensionality for the class of DDP's, i.e. MDP's with finite action sets A.

There is a "grey area" of discrete approximation methods for which $\hat{\Gamma}_N$ is not self-approximating. These methods produce approximate solutions $\hat{V}_t \in R^N$ that are only defined at a finite number of points in S. Since these methods require auxiliary interpolation, approximation, or smoothing algorithms to extend $\hat{V}_t$ for a function defined over all of S, we classify these approaches as smooth approximation methods and cover them in section 5.3.2. An example of this approach is the method proposed by Keane and Wolpin 1994 to approximate value functions to DDP's. The significance of this seemingly predantic distinction between discrete and smooth approximation methods is due to a basic conclusion about the complexity of the approximation problem discussed in section 3.3: all of the standard multivariate function approximation algorithms are subject to an inherent curse of dimensionality regardless of whether deterministic or randomized algorithms are used. Thus, even though the Keaner-Wolpin algorithm uses monte carlo integration which breaks the curse of dimensionality of the integration subproblem, its reliance on regression smoothing methods to construct an estimate of $\hat{V}_t$ that is defined over all of S means that the method falls prey to the curse of dimensionality of the approximation problem.

We now outline a simple approach for proving the convergence of discrete approximation methods based on self-approximating Bellman operators $\hat{\Gamma}_N$: we show that if the approximate Bellman operator $\hat{\Gamma}_N$ converges uniformly to the true Bellman operator Γ as $N \to \infty$, then the sequence of approximate value functions $\hat{V} = \{\hat{V}_0, \ldots, \hat{V}_T\}$ will converge to the true sequence

of value functions $V = \{V_0, \ldots, V_T\}$. However we must exercise sufficient control over the approximation errors generated at each step of the backward induction process otherwise the errors could compound at each stage, i.e. it is conceivable that $\|\hat{V}_t - V_t\| \geq \|\hat{V}_{t+1} - V_{t+1}\|$. The following Lemma provides a simple sufficient condition that guarantees that the approximation errors are uniformly bounded by ϵ at each step of the backward induction process.

Lemma 5.2: *Suppose there exists an integer $\overline{N}(\epsilon, \beta)$ such that for all $N \geq \overline{N}(\epsilon, \beta)$ we have:*

$$\left\|\hat{\Gamma}_N(W) - \Gamma(W)\right\| \leq (1-\beta)\epsilon, \tag{5.41}$$

uniformly for all $W \in B$ satisfying:

$$\|V\| \leq K \equiv \sup_{s \in S} \sup_{a \in A(s)} \frac{|u(s,d)|}{(1-\beta)}. \tag{5.42}$$

If we begin the backward induction using any estimate of the terminal value function $\hat{V}_T$ satisfying:

$$\begin{aligned} \|\hat{V}_T - V_T\| &\leq \epsilon, \\ \|\hat{V}_T\| &\leq K/(1-\beta), \end{aligned} \tag{5.43}$$

then $\hat{V}_t$ will be uniformly within ϵ of V_t for all t, i.e.

$$\max_{t \in \{1,\ldots,T\}} \|\hat{V}_t - V_t\| \leq \epsilon. \tag{5.44}$$

Proof: We prove the result by induction on t. Starting at $t = T-1$ suppose we choose N and $\hat{V}_T$ satisfying (5.41), (5.42), and (5.43). Then we have:

$$\begin{aligned} \left\|\hat{V}_{T-1} - V_{T-1}\right\| &= \left\|\hat{\Gamma}_N(\hat{V}_T) - \Gamma(V_T)\right\| \\ &= \left\|\hat{\Gamma}_N(\hat{V}_T) - \hat{\Gamma}_N(V_T) + \hat{\Gamma}_N(V_T) - \Gamma(V_T)\right\| \\ &\leq \left\|\hat{\Gamma}_N(\hat{V}_T) - \hat{\Gamma}_N(V_T)\right\| + \left\|\hat{\Gamma}_N(V_T) - \Gamma(V_T)\right\| \\ &\leq \beta\epsilon + (1-\beta)\epsilon \\ &= \epsilon. \end{aligned} \tag{5.45}$$

This argument can be repeated for each $t = T-1, T-2, \ldots, 1$ provided we can show that for each t we have $|\hat{V}_t| \leq K/(1-\beta)$. However this follows follows from Lemma 2.3.

Lemma 5.2 shows that the problem of proving the consistency of various discrete approximation methods reduces to the problem of proving that the approximate Bellman operator $\hat{\Gamma}_N$ converges to Γ uniformly for $V \in B(0, K)$:

$$\lim_{N \to \infty} \sup_{\|V\| \leq K} \left\|\hat{\Gamma}_N(V) - \Gamma(V)\right\| = 0. \tag{5.46}$$

Furthermore, in order to compare the computational complexity of various discrete approximation methods we will also need to determine the *rate of convergence* of $\hat{\Gamma}_N$ to Γ, i.e. the rate at which the function $\overline{N}(\epsilon, \beta)$ increases as $\epsilon \to 0$ and $\beta \to 1$.

We now provide a brief overview of four main approaches to discretization. There are two key problems involved in developing an effective discretization procedure: 1) how to choose the N grid points $\{s_1, \ldots, s_N\}$ at which to evaluate V_{t+1} in the summation in (5.39), and 2) how to construct a discretized estimate p_N of the continuous transition probability p. It turns out that the first problem is the most important, since that methods for estimating p typically follow naturally once we have chosen a grid. We now present four different strategies for selecting the grid points $\{s_1, \ldots, s_N\}$: 1) uniform grids, 2) quadrature grids, 3) random grids, and 4) "low discrepancy" grids. The first two methods are subject to the curse of dimensionality of multivariate integration discussed in section 3: i.e. the minimal number of grid points $\overline{N}(\epsilon, \beta)$ required to approximate the Bellman operator Γ defined in (5.38) to within a maximum error of $(1-\beta)\epsilon$ uniformly for all V satisfying $|V| \leq K/(1-\beta)$ is $\overline{N} = \Theta(1/(\epsilon(1-\beta)^2)^{d_s})$ where d_s is the dimension of the state vector s. Furthermore the worst case complexity bounds established by Chow and Tsitsiklis 1989 show that general CDP's are subject to an inherent curse of dimensionality, at least on a worst case basis using deterministic methods. However Rust 1994c proved that the third approach, randomly chosen grid points, succeeds in breaking the curse of dimensionality for DDP's. The final approach uses "low discrepancy" grid points such as the Hammersley points and Sobol' points. As we discussed at the end of section 3.2, Woźniakowski 1991 proved that a quasi-monte carlo integration algorithm using the Hammersley points succeeds in breaking the curse of dimensionality associated with the multivariate integration problem on an average case basis. We conjecture that these grid points may also break the curse of dimensionality for the subclass of DDP's on an average case basis.

In order to establish the consistency and asymptotic properties of the various discretization procedures presented below we need to impose some additional regularity conditions on the state space S and the functions $\{u, p\}$. The following assumptions are stronger than necessary to prove many of the results stated below, but we will impose them here in order to simplify the proofs and unify the exposition. We believe that the assumption that the state and control spaces are subsets of the d_s and d_a-dimensional unit cubes is without significant loss of generality. The key restriction is that S and A are compact sets. If we have compactness then we can always do a change of coordinates to map these spaces into compact subsets of the hypercube.

Definition 5.1. *Let $\Lambda(u,p) = V$ denote the (finite or infinite horizon) MDP problem given by the dynamic programming recursion equations in section 2.2. The class of admissible problem elements $F = \{(u,p)\}$ for the MDP problem is given by:*

(A1) $S = [0,1]^{d_s}$; $A(s) = [0,1]^{d_a}$ $\forall s \in S$.

(A2) $p(ds'|s,a)$ *has a continuous and uniformly bounded density with respect to Lebesgue measure on* S *for each* $a \in A(s)$ *and* $s \in S$.

(A3) $u(s,a)$ *is jointly Lipschitz continuous in* (s,a) *with Lipschitz bound* K_u.[55]

(A4) $p(s'|s,a)$ *is a jointly Lipschitz continuous function of (s',s,a) with Lipschitz bound* K_p.

(A5) *The mapping* $s \rightarrow A(s)$ *is a Lipschitz continuous correspondence.*[56]

Uniform grid points. The obvious way to discretize a continuous state space is to use a uniform grid, consisting of equi-spaced points in S and A. Specifically, we partition the d_s-dimensional cube $[0,1]^{d_s}$ into equal subcubes of length h on each side. Assuming that $1/h$ is an integer, this results in a total of $N = (1/h)^{d_s}$ subcubes. In to obtain arbitrarily accurate approximations, it should be clear that we will need to have $h \rightarrow 0$. It follows that the simple uniform discretization is subject to the curse of dimensionality since the number of grid points, and hence the amount of work required to solve the approximate discrete MDP problem, increases exponentially in the dimensions d_s and d_a. For this reason, the uniform discretization has been traditionally been regarded as a "naive" approach to numerical approximation of MDP problems. Surprisingly, Chow and Tsitsiklis 1991 showed that the simple uniform uniform discretization is about the best we can do in the sense that a multigrid algorithm using a sequence of uniform grids for S and $A(s)$ nearly attains the lower bound on the worst case complexity of the MDP problem. Since their result was proven for the case of infinite horizon MDP's we will defer further discussion of this paradox until section 5.4. Here we will simply illustrate how they used a uniform grid to construct an approximate Bellman operator Γ_h that converges to Γ as $h \rightarrow 0$.

Let S_h denote the partition of S induced by the uniform partition of the unit cube, $[0,1]^{d_s}$. Similarly let A_h denote the partition induced by the discretization of the action space $A = [0,1]^{d_a}$. Thus, S_h consists of $N_s = (1/h)^{d_s}$ equal subcubes of length h on each side, and A_h consists of

[55] Lipschitz continuity means that $|u(s,d) - u(s',d')| \leq K_u|(s,d) - (s',d')|$, for all feasible pairs (s,d) and (s',d').

[56] This means that for any s, $s' \in S$ and any $a' \in A(s')$ there exists some $a \in A(s)$ such that $|a - a'| \leq K|s - s'|$.

$N_a = (1/h)^{d_a}$ equal subcubes of length h on each side. Let s_k denote an arbitrary element (grid point) in the k^{th} partition element of S_h and let $k(s)$ denote the index of the partition element of S_h that contains a given point $s \in S$. Similarly, let a_k denote an arbitrary element (grid point) in the k^{th} partition element of A_h. The discretized utility and transition probability are defined by:

$$\begin{aligned} u_h(s,a) &= u(s_{k(s)}, a) \\ p_h(s'|s,a) &= \frac{p(s_{k(s')}|s_{k(s)}, a)}{\int p(s_{k(s')}|s_{k(s)}, a)ds'} \end{aligned} \tag{5.47}$$

where the normalization of the second equation insures that p_h is a well defined transition probability density on S. Note that in many cases we will need to do numerical integration to compute the normalizing factors in the denominator of the discretized transition probability p_h as can be seen from formula (5.47). In practice the required numerical integration can be carried out by any numerical integration method such as the quadrature method to be described next. Note also that (5.47) defines u_h and p_h as discrete step function approximations to u and p. This is primarily for notational convenience: in practice one would probably want to use spline interpolations so that u_h and p_h are also Lipschitz continuous functions of (s', s, a). Given these objects, we can define the discretized Bellman operator by

$$\begin{aligned} \hat{\Gamma}_h(V)(s) &= \max_{a_k, k=1,\ldots,a_{Na}} [u_h(s,a_k) + \beta \int V(s')p_h(s'|s,a_k)ds'] \\ &= \max_{a_k, k=1,\ldots,a_{Na}} [u_h(s,a_k) + \frac{\beta}{N}\sum_{k=1}^{N} V(s_k)p_h(s_k|s_{k(s)},a_k)], \end{aligned} \tag{5.48}$$

where the second equality holds whenever V is a step function taking value $V(s_k)$ in partition element k. The following Theorem is the basis consistency result for the uniform discretization procedure, adapted from Theorem 3.1 of Chow and Tsitsiklis 1991:

Theorem 5.5: *There exist constants K_1 and K_2 such that for all h sufficiently small and all $V \in B$ we have:*

$$\|\Gamma(V) - \hat{\Gamma}_h(V)\| \leq (K_1 + \beta K_2 \|V\|)h. \tag{5.49}$$

Using Lemma 5.2, we need to choose h small enough so that the right hand side of inequality (5.49) is less than $(1-\beta)\epsilon$ uniformly for all V satisfying $\|V\| \leq K/(1-\beta)$, where K is the constant defined in equation (5.43). It is easy to see that choosing h proportional to $(1-\beta)^2\epsilon$ is sufficient to guarantee that this is satisfied. This implies that the discretized MDP requires a total of $|S| = O(1/((1-\beta)^2\epsilon)^{d_s})$ grid points. A similar analysis shows that $|A| = O(1/((1-\beta)^2)\epsilon)^{d_a}$ grid points are required in order to guarantee that the discrete maximizations in the approximate

Bellman operator $\hat{\Gamma}_h(V)$ are within $(1-\beta)\epsilon$ of $\Gamma(V)$ uniformly for all V satisfying $\|V\| \leq K/(1-\beta)$. Recalling from section 5.1 that $O(T|A||S|^2)$ operations are required to solve a discrete, finite horizon MDP, it follows that the uniform discretization requires $O(T/((1-\beta)^2\epsilon)^{(2d_s+d_a)})$ operations to compute an approximate solution $\hat{V} = (\hat{V}_0, \ldots, \hat{V}_T)$ that is uniformly within ϵ of the true sequence of value functions. It follows that discrete approximation methods using uniform grids are subject to the curse of dimensionality.

Quadrature grids. Tauchen and Hussey 1991 suggested the use of quadrature abscissa as a good set of grid points for discrete approximation of integral equations arising in linear rational expectations models. Their approach can be easily translated to provide a discrete approximation method for continuous MDP's. The motivation for this approach is the use of quadrature methods for approximating the integral of a function $f : S \to R$ with respect to a weighting function $p : S \to R$. In the unidimensional case, $S = [0, 1]$, quadrature involves the choice of N *quadrature abscissa* $\{s_1, \ldots, s_N\}$ and corresponding weights $\{w_1, \ldots, w_N\}$ such that

$$\int f(s)p(s)ds \simeq \sum_{k=1}^{N} f(s_k)w_k. \tag{5.50}$$

The weighting function p will typically be a probability density function, although in some cases it can be interpreted as an "importance sampling" weighting reflecting the analyst's *a priori* knowledge about the parts of the state space that are most important for calculating the integral. In the special case of Gaussian quadrature, $\{s_1, \ldots, s_N\}$ and $\{w_1, \ldots, w_N\}$ are determined by requiring that equation (5.50) to be exact for all polynomials of degree less than or equal to $2N-1$. The resulting grid points and weights depend only on the weighting function p and not the integrand f. Notice that the weights $\{w_1, \ldots, w_N\}$ generally do not satisfy $w_k = p(s_k)$, although they are always guaranteed to be positive. Applying Gaussian quadrature to approximate the conditional expectation of the value function V with respect to a weighting function p we obtain

$$\int V(s')p(s'|s,a)ds' = \int V(s')\frac{p(s'|s,a)}{p(s')}p(s')ds' \simeq \sum_{k=1}^{N} V(s_k)\frac{p(s_k|s,a)}{p(s_k)}w_k. \tag{5.51}$$

The quadrature weights and abscissa allow us to define an N-state Markov chain with state space consisting of the quadrature abscissa $\{s_1, \ldots, s_N\}$ with transition probability p_N defined probability by a simple normalization:

$$p_N(s_k|s_j,a) = \frac{p(s_k|s_j,a)w_k/p(s_k)}{\sum_{i=1}^{N} p(s_i|s_j,a)w_i/p(s_i)}, \quad j,k = 1,\ldots,N. \tag{5.52}$$

Using p_N we can define a discretized Bellman operator $\hat{\Gamma}_N : C(S) \rightarrow C(s)$ by:

$$\hat{\Gamma}_N(V)(s) = \max_{a \in A_N(s)} [u(s,a) + \beta \sum_{k=1}^{N} V(s_k) p_N(s_k|s,a)], \tag{5.53}$$

where $A_N(s)$ denotes a finite choice set with N points, a discretized version of $A(s)$ using the same procedure defined for the uniform discretization procedure above.[57] Notice that since $\hat{\Gamma}_N(V)(s)$ is the maximum of a sum of continuous functions of s, it has the self-approximating property that we discussed earlier. Adapting Tauchen and Hussey's Theorem 4.2 yields the following error bound on the deviation between $\Gamma(V)$ and $\hat{\Gamma}_N(V)$:

Theorem 5.6:. *Suppose $S = [0,1]$. There exist constants K_1 and K_2 such that for all $V \in C(S)$ and all N sufficiently large we have:*

$$\|\Gamma(V) - \hat{\Gamma}_N(V)\| \leq \frac{(K_1 + \beta K_2)}{N}. \tag{5.54}$$

Note that this is the same basic bound as obtained in the case of uniform discretization in Theorem 5.2 with inverse of the number of quadrature abscissa $1/N$ playing the same role as the grid size parameter h. Thus, just as in the case of uniform discretization, the quadrature approach produces a consistent estimate of the true value function V as $N \rightarrow \infty$.

Note that although Gaussian quadrature possesses several optimum properties (Davis and Rabinowitz, 1975) the ϵ-complexity of the quadrature algorithm is the same as uniform discretization since quadrature is a deterministic integration algorithm and we saw from section 3.2 that all deterministic integration algorithms are subject to the curse of dimensionality. We can see this more explicitly by considering the product rule formula for numerical quadrature of multidimensional functions $f : [0,1]^d \rightarrow R$:

$$\int f(s)ds \simeq \sum_{k_1=1}^{N} \cdots \sum_{k_d=1}^{N} f(s_{k_1}, \ldots, s_{k_d}) \prod_{i=1}^{d} w_{k_i}, \tag{5.55}$$

where the $\{s_{k_i}\}$ and $\{w_{k_i}\}$ are the same quadrature abscissa and weights as in the one-dimensional case. Clearly the amount of cpu time required to evaluate a d-dimensional product quadrature rule is proportional to N^d, even though the approximation error for $f \in C[0,1]^d$ still decreases at rate

[57] Of course there is no reason beyond notational simplicity to require that the number of points N used for the quadrature abscissa equals the number of points for the choice set discretization as long as both tend to infinity and the maximum grid size of the discretization of $A(s)$ tends to zero as $N \rightarrow \infty$.

inversely proportional to the number of quadrature abscissa used in each dimension, i.e. $1/N$. It follows that in the multivariate case the amount of computer time required by quadrature methods to approximate an integral with a maximum error of ϵ on a worst case basis is of order $\Theta(1/\epsilon^{d_s})$, the same as for the simple uniform discretization method described above. Thus, while the use of quadrature abscissa may be relatively effi cient for low-dimensional MDP problems, quadrature cannot break the curse of dimensionality of the continuous MDP problem, at least on a worst case basis. If we calculate the number of operations required to guarantee that backward induction using quadrature grids produces approximate value functions that are uniformly within ϵ of the true solution V, we fi nd that the amount of work required is of order $\Theta(T/((1-\beta)^2\epsilon)^{(2d_s+d_a)})$, which is same order of effort required by backward induction using a simple uniform discretization. The Chow and Tsitsiklis 1989 complexity bound presented in section 5.4 shows that this curse of dimensionality is an inherent feature of the MDP problem that can't be circumvented by *any* choice of deterministic solution algorithm. In order to circumvent the curse of dimensionality we must either focus on a subclass of MDP problems with further structure, or we must adopt a different notion of computational complexity such as randomized complexity or average case complexity.

Random grid points. In section 3.2 we discussed the use of randomization as a way to break the curse of dimensionality associated with approximate solution of various mathematical problems. Certain problems such as multivariate integration that are intractable when deterministic algorithms are used become tractable when random algorithms are allowed. Of course the use of random algorithms does not come without cost: we can't guarantee that the approximate solution is within ϵ of the true solution with probability 1 but only with probability arbitrarily close to 1. Also randomization does not always succeed in breaking the curse of dimensionality: Nemirovsky and Yudin 1978 showed that the general nonconvex multivariate optimization problem is intractable whether or not random or deterministic algorithms are used. This implies that randomization cannot succeed in breaking the curse of dimensionality of the general MDP problem with continuous action space A since it includes the general nonconvex multivariate optimization problem as a special case when $\beta = 0$ and $|S| = 1$. We record this result as

Theorem 5.7 *Randomization cannot succeed in breaking the curse of dimensionality of the class of all continuous MDP problems, i.e. a lower bound on the computational complexity is given by:*

$$comp^{\textsf{wor-ran}}(\epsilon, \beta, d_a, d_s) = \Omega\left(\frac{1}{[(1-\beta)^2\epsilon]^{d_a}}\right). \tag{5.56}$$

However Rust 1994c showed that randomization does succeed in breaking the curse of dimensionality of the subclass of DDP problems given in definition 2.2. The intuition underlying the result is fairly simple: a DDP has a finite number $|A|$ of possible actions but a continuous multidimensional state space $S = [0,1]^d$. Therefore essentially all the work involved in approximating the Bellman operator $\Gamma(V)$ is the multivariate integration problem to compute $\int V(s')p(s'|s,a)ds'$ at each pair (s,a). However we know from section 3.2 that randomization succeeds in breaking the dimensionality of the multivariate integration, so it stands to reason that it can also break the curse of dimensionality of the DDP problem. Proving this result is not quite as simple as this heuristic argument would indicate since the Bellman operator effectively involves the evaluation of infinitely many multidimensional integrals for each possible conditioning pair (s,a). The problem can be solved by introducing a *random Bellman operator* $\tilde{\Gamma}_N(V)$ defined by

$$\tilde{\Gamma}_N(V)(s) = \max_{a \in A(s)} [u(s,a) + \frac{\beta}{N} \sum_{i=1}^{N} V(\tilde{s}_i) p_N(\tilde{s}_i|s,a)], \tag{5.57}$$

where $(\tilde{s}_1, \ldots, \tilde{s}_N)$ are N *IID* random uniform draws from the d-dimensional hypercube $[0,1]^d$, and the transition probability p_N over this grid is defined by:

$$p_N(\tilde{s}_i|\tilde{s}_j,a) = \frac{p(\tilde{s}_i|\tilde{s}_j,a)}{\sum_{k=1}^{N} p(\tilde{s}_i|\tilde{s}_j,a)}, \quad i,j = 1,\ldots,N. \tag{5.58}$$

The resulting discrete approximation algorithm, $\tilde{V}_t = \tilde{\Gamma}_N^{T-t}(0)$, $t = 0,\ldots,T$ can be interpreted as simple backward induction over a randomly chosen set of grid points in $[0,1]^d$. Notice that the random Bellman operator is a self-approximating operator, i.e. we can use formula (5.57) to evaluate $\tilde{\Gamma}_N(V)(s)$ at any point $s \in S$ rather than just at the set of random grid points $(\tilde{s}_1, \ldots, \tilde{s}_N)$. This self-approximating property is the key to the proof of the convergence of the random Bellman operator. Rust 1994c proved that if the transition probability $p(s'|s,a)$ is a sufficiently smooth function of (s',s) (i.e. Lipschitz continuous), then the sample average $1/N \sum_{i=1}^{N} V(\tilde{s}_i) p_N(\tilde{s}_i|s,a)$ will not only be close to its expectation $\int V(s')p(s'|s,a)ds'$ at any specific pair $(s,a) \in S \times A$, it will be close uniformly for all $(s,a) \in S \times A$. This can be formalized in the following error bound established in Rust 1994c:

Theorem 5.8 *Suppose $S = [0,1]^d$ and (u,p) satisfy the Lipschitz conditions (A3) and (A4). Then for each $N \geq 1$ the expected error in the random Bellman operator satisfies the following uniform bound:*

$$\sup_p \sup_{\|V\| \leq K/(1-\beta)} E\left\{ \|\tilde{\Gamma}_N(V) - \Gamma(V)\| \right\} \leq \frac{\gamma(d)|A|K_p K}{(1-\beta)\sqrt{N}}, \tag{5.59}$$

where K_p is the Lipschitz bound on p and $\gamma(d)$ is a bounding constant given by:

$$\gamma(d) = \sqrt{\frac{\pi}{2}}\beta[1 + d\sqrt{\pi}C] \tag{5.60}$$

and C is an absolute constant independent of u, p, V, β or d.

Inequality (5.59) implies that randomization breaks the curse of dimensionality for the subclass of DDP problems because the expected error in the random Bellman operator decreases at rate $1/\sqrt{N}$ independent of the dimension d of the state space and the constant of proportionality $\gamma(d)$ grows linearly rather than exponentially in d. Using Lemma 5.1 we can insure that the expected error in the random sequence of value functions $\tilde{V} = (\tilde{V}_0, \ldots, \tilde{V}_T)$ generated by backward induction using $\tilde{\Gamma}_N$ will be within ϵ of the true sequence $V = (V_0, \ldots, V_T)$ provided we choose $N \geq \overline{N}(\epsilon, \beta)$ where $\overline{N}(\epsilon, \beta)$ is given by

$$\overline{N}(\epsilon, \beta) = \left(\frac{\gamma(d)|A|K_p K}{(1-\beta)^2 \epsilon} \right)^2. \tag{5.61}$$

Since the complexity of solving the DDP with N states and $|A|$ actions is $O(T|A|N^2)$, we obtain the following complexity bound in Rust 1994c:

Theorem 5.9: *Randomization breaks the curse of dimensionality of solving DDP's: i.e. the worst case complexity of the class of randomized algorithms for solving the DDP problems satisfying $(A1), \ldots, (A4)$ is given by:*

$$comp^{\textsf{wor-ran}}(\epsilon, \beta, d) = O\left(\frac{T d^4 |A|^3 K_p^4 K^4}{(1-\beta)^8 \epsilon^4} \right). \tag{5.62}$$

Rust's algorithm has not yet been programmed and evaluated in actual applications, so we do not know how well it really works in practice. The fact that simple backward recursion using the random Bellman operator has the nice theoretical property of breaking the curse of dimensionality does not necessarily imply that the method will outperform deterministic methods on low-dimensional problems in much the same way that deterministic integration algorithms such as quadrature systematically outperform crude monte carlo integration in low dimensional problems. However monte carlo integration methods have been successfully applied in a recent study by Keane and Wolpin 1994, although their method differs in many respects from the random Bellman operator approach described above. In particular, their method does not have the self-approximating feature of the random Bellman operator, so auxiliary interpolation routines must be employed to generate estimates of the continuous value function for points s that do not lie on a

preassigned grid of the state space S. Since this approach requires the use of smooth approximation methods, we defer its presentation until section 5.3.2.

"Low discrepancy" grid points. We conclude our survey of discrete approximation methods by suggesting that certain deterministically generated sets of grid points known as "low discrepancy sets" may be very effective for reducing the computational burden of the multivariate numerical integration subproblem of the general MDP problem. Recent research has shown that various low discrepancy sets such as the Hammersley, Halton, and Sobol' points possess certain optimality properties for numerical integration problems (for definitions of these sets, see Neiderreiter, 1992). For example, section 3.2 summarized Woźniakowski's 1991 result that the sample average $1/N \sum_{i=1}^{N} f(s_i)$ where $(s_1, \ldots, s_N)$ are the first N Hammersley points is a nearly optimal algorithm for computing an integral $\int_{[0,1]^d} f(s)ds$ in the sense that the average cost of this algorithm is close to the lower bound on the average complexity of numerical integration using a Gaussian prior over the space F of possible integrands. We can get some insight into why these particular deterministic sequences seem to perform so well in numerical integration from an inequality Neiderreiter 1992 refers to as the *Koksma-Hlwaka inequality:*

$$\left| \frac{1}{N} \sum_{i=1}^{N} f(s_i) - \int f(s)\lambda(ds) \right| \leq V(f) D_N^*(s_1, \ldots, s_N), \tag{5.63}$$

where λ is Lebesgue measure on $[0,1]^d$, $V(f)$ is the total variation in f in the sense of Hardy and Krause (see page 19 of Neiderreiter for a definition), and D_N^* is the *discrepancy:*

$$D_N^*(s_1, \ldots, s_N) \equiv \sup_{B \in \mathcal{B}} |\lambda_N(B) - \lambda(B)|, \tag{5.64}$$

where $\mathcal{B}$ is the class of (open) suborthants of $[0,1]^d$, ($\mathcal{B} = \{[0,s)^d \subset [0,1]^d | s \in [0,1]^d\}$) and λ_N is the empirical CDF corresponding to the points $(s_1, \ldots, s_N)$. The Koksma-Hlwaka inequality suggests that by choosing grid points $(s_1, \ldots, s_N)$ that make the discrepancy $D_N^*(s_1, \ldots, s_N)$ small we will be able to obtain very accurate estimates of the integral of functions f whose total variation $V(f)$ is not too large. An interesting literature on *discrepancy bounds* (surveyed in Neiderreiter 1992) provides upper and lower bounds on the rate of decrease of certain "low discrepancy" point sets such as the Hammersley, Halton, and Sobol' points. For example Roth's 1954 lower bound on the discrepancy of any set of points $(s_1, \ldots, s_N)$ in $[0,1]^d$ is given by:

$$D_N^*(s_1, \ldots, s_N) \geq K(d) \frac{(\log N)^{(d-1)/2}}{N}$$

where $K(d)$ is a universal constant that only depends on the dimension of the hypercube, d. An upper bound for the N-element Hammersley point set $P = \{s_1^h, \ldots, s_N^h\}$ is given by:

$$D_N^*(P) \leq \frac{\gamma(d)(\log N)^{d-1}}{N} + O(N^{-1}(\log N)^{d-2}).$$

Obviously for this class, the Hammersley points do much better than randomly selected points since the rate of decrease in the expected error in the latter is at the slower rate $O(1/\sqrt{N})$. A recent study by Paskov 1994 compared the accuracy of deterministic integration using deterministic low discrepancy grids such as the Halton and Sobol' points and standard monte carlo in high dimensional ($d = 360$) integration problems arising in finance applications. The these problems the deterministic algorithms provided much more accurate estimates of the integral in less cpu time. This suggests that these points might be better choices than randomly chosen grid points $(\tilde{s}_1, \ldots, \tilde{s}_N)$ in the definition of the random Bellman operator in equation (5.57).

That fact tht these deterministic algorithms outperform monte carlo integration for certain classes of functions F does not contradict the complexity bounds presented in section 3.2. The deterministic low-discrepancy integration algorithms are indeed subject to a curse of dimensionality, but on a *worst case basis.* However the theoretical work of Woźnaikowski and Paskov shows that low diescrepancy methods break the curse of dimensionality on an average case basis. Paskov's numerical results are consistent with the average case complexity bounds based on prior distributions which are concentrated on a subclass of problem elements with stronger regularity conditions than considered under the worst case complexity measure. Although monte carlo integration does a good job in approximating any measurable integrand with finite variance, certain deterministic integration algorithms may be more effective for certain restricted subclasses of functions that have more smoothness. For example, the Koksma-Hlwaka inequality suggests that low discrepancy point sets such as the Halton or Sobol' points will be effective in integrating the set of functions that have small Hardy-Krause variations $V(f)$, although they generally won't do a good job of integrating arbitrary continuous functions in high dimensional spaces since $V(f)$ can be shown to increase exponentially fast as the dimension of the space increases, at least for functions in certain subsets such as the subset $C[0,1]^d$ of continuous functions on the d-dimensional hypercube.

An important unanswered question is whether an analog of Paskov's and Woźniakowski's results extend to certain classes of MDP's, i.e. are certain classes of MDP problems tractable on an average case basis? We conjecture that the subclass of DDP problems is tractable on an average case basis. As we noted at the end of section 3 the difficulty in applying an average case analysis to the DDP problem is the necessity of specifying a reasonable prior over the space of admissible

transition probabilities. The typical prior used in multivariate integration problems, folded Wiener sheet measure, does not ensure that the transition probabilities are nonnegative and integrate to 1.

5.3.2 Smooth Approximation Methods

Smooth approximation methods differ from discrete approximation methods by approximating the value function $V = (V_0, \ldots, V_T)$ or decision rule $\alpha = (\alpha_0, \ldots, \alpha_T)$ by smooth functions of the state variable s rather than over a finite grid of points. For example many methods parameterize the value function or policy function in terms of a vector $\theta \in R^k$ of unknown parameters, choosing $\hat{\theta}$ so that the resulting estimates of $V_{\hat{\theta}}$ or $\alpha_{\hat{\theta}}$ are as close as possible to the true solutions V and α. As noted in the introduction many smooth approximation methods are closely connected to discrete approximation since they can be interpreted as using estimates of V and α at a finite grid of points $\{s_1, \ldots, s_N\}$ in S as "data" for any one of a variety of smoothing or approximation algorithms to obtain smoothed functions that are defined for all $s \in S$. In this sense all discrete approximation methods are actually a subclass of smooth approximation methods. From our point of view the key difference between the methods is whether or not a discrete approximation method is self-approximating in the sense defined in section 5.3.1, or whether we need to employ auxilliary approximation methods to produce estimates of V and α that are defined for all $s \in S$. Smooth approximation methods can be viewed from an abstract perspective as selecting estimates of $\hat{V}$ or $\hat{\alpha}$ from a finite-dimensional submanifold of the infinite dimensional space of value or policy functions. We shall see that the differences in various approximation methods are a result of different ways of parameterizing these submanifolds and of different metrics for locating particular elements $\hat{V}$ and $\hat{\alpha}$ that are "closest" to the true solution V and α.

In order to understand the general structure of smooth approximation methods and appreciate the role that approximation and interpolation methods play in carrying out the backward induction process, consider how one would compute the basic recursion relation (2.6) determining the value function V_t at time t. Assume that we have already computed a smoothed version of the time $t+1$ value function so that we can quickly evaluate $\hat{V}_{t+1}(s)$ at any point $s \in S$. Using $\hat{V}_{t+1}$ we can compute estimates of the time t value function $\hat{V}_t$ at any *finite* collection of points $\{s_1, \ldots, s_N\}$ using the standard backward induction formula of dynamic programming:

$$\hat{V}_t(s_i) = \hat{\Gamma}\left(\hat{V}_{t+1}\right)(s_i) \simeq \max_{a \in A(s_i)} [u(s_i, a) + \beta \int \hat{V}_{t+1}(s') p(ds'|s_i, a)], \quad i = 1, \ldots, N. \quad (5.65)$$

We use the notation $\hat{\Gamma}(\hat{V}_{t+1})(s_i)$ to denote an approximation to the value of the true Bellman operator $\Gamma(\hat{V}_{t+1})(s_i)$ at s_i in view of the fact that we will typically not be able to find analytic

solutions to the integration and maximization problems in (5.65) so we must rely on approximate numerical solutions to these problems. Since it is quite costly to compute approximate solutions to the right hand side of (5.65) at each of the N points $\{s_1, \ldots, s_N\}$, we would like to choose N as small as possible. However in order to increase the accuracy of solutions to the numerical optimization and integration problems the are involved in calculating the time $t-1$ value function $\hat{V}_{t-1} = \hat{\Gamma}(\hat{V}_t)$ we need to evaluate $\hat{V}_t$ at as large a number of points $s \in S$ as possible. Rather than evaluating $\hat{V}_t(s)$ at a large number of points $s \in S$ we will consider approximation methods that only require us to evaluate $\hat{V}_t$ at a relatively small number of points $\{\hat{V}_t(s_1), \ldots, \hat{V}_t(s_N)\}$ an use these points as "data" in order to generate accurate "predicted" values $\hat{V}_t(s)$ at other points $s \in S$ at much lower cost than evaluating equation (5.65) directly at each $s \in S$. A general approach is to approximate the value function by nonlinear least squares using a parametric family $V_\theta = (V_{0,\theta}, \ldots, V_{T,\theta})$ that are smooth functions of $s \in S$ and a $(T+1)k \times 1$ vector of unknown parameters $\theta = (\theta_0, \ldots, \theta_T)$. Then our estimate of the time t value function is $\hat{V}_t = V_{t,\hat{\theta}}$ where $\hat{\theta}_t$ is defi ned recursively by:

$$\hat{\theta}_t = \underset{\theta_t \in R^k}{argmin}\, \sigma_N(\theta) \equiv \sqrt{\sum_{i=1}^{N} \left| V_{\theta_t}(s_i) - \hat{\Gamma}(V_{\hat{\theta}_{t+1}})(s_i) \right|^2}, \tag{5.66}$$

where $V_{\hat{\theta}_{t+1}}$ is the least squares estimate of the time $t+1$ value function V_{t+1}. The degree to which the smooth approximation approach succeeds in providing accurate approximations for a small number N depends on the degree of smoothness in the true solution: much larger values of N will be required if we are unable to rule out the possibility that V_t is highly oscillatory as opposed to the case where we know *a priori* that V_t has certain monotonicity and concavity properties.

Notice that smooth approximation is similar to discrete approximation in that both methods require specifi cation of a "grid" $\{s_1, \ldots, s_N\}$. It follows that the same issues of optimal choice of grid points that arose in our review of discrete approximation methods will also arise in our discussion of continuous approximation methods. In addition to the choice of grid points (which is similar in many respects to the problem of selecting "optimal design points" in a regression/prediction problem) there are a number of other decisions that have to be made in order to implement a smooth approximation method:

1. Which object should we parametrize: V, α, or something else?

2. How should this object be parameterized: via a linear-in-parameters specifi cation such as a polynomial series approximation, or a nonlinear-in-parameters approximation such as a

neural network, or by piecewise polynomial approximations such as splines or Hermite polynomials?

3. How should $\hat{\theta}$ be determined: via nonlinear least squares fit, or via a projection method such as Chebyshev interpolation or Galerkin's method, or some other (possibly randomized) procedure?

4. Which numerical optimization and integration methods should be used to compute the right hand side of (5.65) at each of the grid points $\{s_1, \ldots, s_N\}$?

One can see that we obtain a wide variety of algorithms depending on how we answer each of these questions. We do not have the space here to provide an exhaustive analysis of all the possible variations. Instead we will survey some of the leading smooth approximation methods that have been used in practical applications, and then conclude with some general observations on the relative efficiency and accuracy of these methods. We will concentrate on providing intuitive presentations of methods rather than on formal proofs of their convergence. However it is straightforward to outline a general approach to proving the convergence of methods that approximate V. Starting in the terminal period, we use approximation error bounds for one of the particular approximation methods described below to find an approximation $\hat{V}_T$ to the terminal value function V_T that satisfies $\|\hat{V}_T - V_T\| \leq (1-\beta)\epsilon$. Then at each stage t of the backward induction process find an approximate value function $\hat{V}_t$ that satisfies $\|\hat{V}_t - \Gamma(\hat{V}_{t+1})\| \leq (1-\beta)\epsilon$, $t = 0, \ldots, T-1$. Note that it is necessary to find $\hat{V}_t$ that approximates the function $\Gamma(\hat{V}_{t+1})$ where Γ is the true Bellman operator. However in practice we can only approximate Γ. Using the triangle inequality we can break the approximation problem into two steps:

$$\|\hat{V}_t - \Gamma(\hat{V}_{t+1})\| \leq \|\hat{V}_t - \hat{\Gamma}_N(\hat{V}_{t+1})\| + \|\hat{\Gamma}_N(\hat{V}_{t+1}) - \Gamma(\hat{V}_{t+1})\|. \tag{5.67}$$

If we choose an approximation method that guarantees that the sum of both terms on the right hand side of (5.67) is less than $(1-\beta)\epsilon$, then by a simple modification of the proof to Lemma 5.2 it is easy to show that

$$\|\hat{V}_t - V_t\| \leq \epsilon, \quad t = 1, \ldots, T, \tag{5.68}$$

i.e. the approximate value functions $\hat{V} = (\hat{V}_0, \ldots, \hat{V}_T)$ will be uniformly within ϵ of the true sequence $V = (V_0, \ldots, V_T)$. Thus, the problem of establishing the consistency of any particular smooth approximation method reduces to the problem of proving that the particular approximation used is able to approximate smooth functions in some class F that contains the set B of possible value functions. While this strategy can be used to provide basic consistency results for most of

the methods presented below, the lower bounds on the complexity of the general approximation problem presented in section 3.3 implies that all of these methods are subject to the curse of dimensionality, at least on a worst case basis for value functions in a sufficiently broad class (e.g. if the set of possible value function B coincides with the set F of bounded Lipschitz continuous functions, or functions with r continuous partial derivatives).

Piecewise Linear Interpolation. [58] This is perhaps the simplest smooth approximation method, also referred to as *multilinear interpolation.* Versions of this approach have been implemented in the computer package DYGAM by Dantzig *et. al.* 1974. The basic idea is to use simple piecewise linear interpolation of V_t in each coordinate of the state vector s. Multilinear interpolation requires an underlying grid on the state space defined by a cartesian product of unidimensional grids over each coordinate of the vector $s = (s_1, \ldots, s_d)$. Let the grid over the k^{th} component of s be given by N points $s_{k,1} < s_{k,2} < \cdots < s_{k,N}$, so that the overall grid of the state space S contains N^d points. In order to estimate $V_t(s)$ at an arbitrary point $s \in S$, the multilinear interpolation procedure locates the grid hypercube that contains s and successively carries out linear interpolation over each coordinate of s, yielding an estimate of $V_t(s)$ that is a linear combination of the values of V_t at the vertices of the grid hypercube containing s. For example, consider a two-dimensional MDP problem $d = 2$ and suppose we want to evaluate V_t at the point $s = (s_1, s_2)$. First we locate the grid hypercube containing s: $s_1 \in (s_{1,i}, s_{1,i+1})$ and $s_2 \in (s_{2,j}, s_{2,j+1})$ for some indices i and j between 1 and N. Define the weights w_1 and w_2 by:

$$\begin{aligned} w_1 &= (s_1 - s_{1,i})/(s_{1,i+1} - s_{1,i}) \\ w_2 &= (s_2 - s_{2,j})/(s_{2,j+1} - s_{2,j}). \end{aligned} \tag{5.69}$$

then the multilinear approximation $\hat{V}_t(s)$ is defined by:

$$\hat{V}_t(s) = w_2 v_2(s_{2,j+1}) + (1 - w_2) v_2(s_{2,j}), \tag{5.70}$$

where the points $v_2(s_{2,j})$ and $v_2(s_{2,j+1})$ are linear interpolations over the first coordinate s_1 of s:

$$\begin{aligned} v_2(s_{2,j}) &= w_1 V_t(s_{1,i+1}, s_{2,j}) + (1 - w_1) V_t(s_{i,1}, s_{2,j}) \\ v_2(s_{2,j+1}) &= w_1 V_t(s_{1,i+1}, s_{2,j+1}) + (1 - w_1) V_t(s_{i,1}, s_{2,j+1}). \end{aligned} \tag{5.71}$$

In a d-dimensional problem, one applies the above steps recursively, generating a total of $2^d - 1$ linear interpolations and function evaluations to produce the interpolated value $\hat{V}_t(s)$. Since the

[58] This section borrows heavily from the presentation in Johnson *et. al.* 1993.

amount of work required to perform all of these interpolations increases exponentially in d_s it is clear that simple multilinear interpolation is subject to the curse of dimensionality. The method also produces an interpolated value function with kinks on the edges of each of the grid hypercubes. In certain MDP problems such as CDP's in the Euler class of Defi nition 2.4 the kinks in $\hat{V}_t$ are not smoothed out by the integration operator $\int \hat{V}_t(s')p(ds'|s,a)$ which creates diffi culties for nonlinear optimization algorithms that attempt to compute the maximizing value of the action a at each grid point s. The main advantage of linear interpolation, noted in Judd and Solnick 1994, is that in one dimensional problems linear interpolation preserves monotonicity and concavity properties of V. Unfortunately Judd and Solnick showed that these properties are not necessarily preserved by multilinear interpolation in higher dimensional problems.

The fact that multilinear interpolation is subject to the curse of dimensionality is not just of academic interest: practical experience with the method demonstrates that it is feasible only in relatively low-dimensional problems. In their numerical application of the multilinear DP algorithm to a multidimensional stochastic water reservoir problem, Johnson *et. al.* 1993 found that while numerical calculation of a reasonable approximation of V for a problem with $T = 3$ and $d = 2$ took only 10 seconds on an IBM 3090 supercomputer, solving a problem with $d = 5$ was estimated to require *70 cpu hours.* Thus, they concluded that "The effort required to solve practical problems having nonlinear objectives with a dimension d of more than 3 to 5 with reasonable accuracy is generally prohibitive with the linear interpolation method commonly employed." (p. 486).

Piecewise Cubic Interpolation. [59] Piecewise linear interpolation is an example of a *spline approximation,* which in one dimension is simply a piecewise polynomial approximation to a function $f(s)$ ($s \in R$) with boundary conditions at *knot points* $(s_1, \ldots, s_N)$ that ensure that the local polynomial approximations on each subinterval (s_i, s_{i+1}) join up to have a specifi ed degree of smoothness. A *spline of order* r is an approximation to f that is $(r-2)$ times continuously differentiable and therefore a polynomial of degree not greater than $(r-1)$ in each subinterval (s_i, s_{i+1}). Thus, a piecewise linear approximation is also known as a 2-spline. If the value function is known to be twice continuously differentiable, then much better approximations can be obtained using a 4-spline, i.e. piecewise cubic polynomials, since they produce an approximation that is everywhere twice continuously differentiable. The extra smoothness of cubic splines often result in much better approximations, allowing one to get by with a smaller number of knot points than are required by linear splines to achieve a specifi ed degree of accuracy. In general if f has a bounded

[59] This section borrows from the exposition of splines in Daniel 1976 and Johnson *et. al.* 1993.

r^{th} derivative, then it can be approximated by an r-spline over a finite interval with N equally space knot points with a maximum error of order $O(1/N^r)$. This result suggests that higher order splines ought to be able to obtain the same error of approximation using a smaller number of knot points N.

Multidimensional problems can be solved by forming *tensor product splines.* These are simply products of one-dimensional splines in each of the coordinate variables. For example, a d-dimensional cubic spline approximation to $f(s)$ takes the form:

$$\hat{f}(s) = \sum_{i_1=1}^{4} \cdots \sum_{i_d=1}^{4} c_{(k_1(s),\ldots,k_d(s))}(i_1,\ldots,i_d) \prod_{j=1}^{d} (s_j - s_{j,k_j(s)})^{i_j-1}, \tag{5.72}$$

where $k(s) = (k_1(s),\ldots,k_d(s))$ is the index of the grid hypercube element containing the point s (i.e. for each $i = 1,\ldots,d$ we have $s_i \in (s_{i,k_i}, s_{i,k_i+1})$ where s_i is the i^{th} component of s, and $s_{i,1} < s_{i,2} < \cdots < s_{i,N}$ are the knot points on the i^{th} coordinate axis). The coefficients $c_{k(s)}(i_1,\ldots,i_d)$ are determined by requiring that $\hat{f}(s)$ interpolate the true function $f(s)$ at each of the grid points and that the first and second derivatives of $\hat{f}(s)$ match up along all of the edges of the grid hypercubes so that the resulting function is twice continuously differentiable over all $s \in S$. However there are two extra degrees of freedom in each dimension due to the fact that the N knots define only $N-1$ intervals in each dimension. The polynomial spline algorithm developed in Johnson *et. al.* 1993 resolves this using a construction due to de Boor 1978 that fits only $N-3$ cubic polynomials in each dimension. Since each cubic polynomial is defined by 4 coefficients, the total number of coefficients $c_{k(s)}(i_1,\ldots,i_d)$ in a multidimensional cubic spline with N knot points on each coordinate axis is $[4(N-3)]^d$.

Johnson 1989 showed that the number of floating point operations required to compute these coefficients is of order $O(4^d N^d)$. After these coefficients have been computed and stored, it takes an additional $(4^d - 1)$ floating operations to compute an interpolated estimate $\hat{V}_t(s)$ at any point $s \in S$. Since this must be done at many points $s \in S$ in order to compute the numerical integrals at each of the N^d grid points on the right hand side of (5.65), it dominates the cpu requirements of the cubic spline DP algorithm. Thus the method is clearly subject to the curse of dimensionality. However even though evaluating the cubic spline in (5.65) requires approximately 2^{d-1} times more work than evaluation of the linear spline in (5.70), the increased accuracy of a cubic spline approximation may allow one to use fewer knot points N in each dimension to obtain an estimate of $\hat{V}_t$ with the same level of accuracy. For example in their numerical comparisons of the linear and cubic spline methods Johnson *et. al* 1993 found that solving a four dimensional

stochastic reservoir test problem to within an error tolerance of approximately 1% of the true solution required $N = 13$ knot points per dimension using linear spline interpolation but only $N = 4$ knot points per dimension with cubic spline interpolation. The linear spline algorithm took 4,043 cpu seconds as opposed to only 26 cpu seconds for the cubic spline algorithm, representing a speedup of 155 times. Part of the speedup is also due to the fact that the cubic spline interpolation is everywhere twice continuously differentiable which enabled the use of faster gradient optimization algorithms to compute the maximizing value of a at each of the grid points, whereas the linear spline interpolation had kinks that required the use of the slower non-gradient Nelder-Mead 1965 optimization algorithm. They concluded that "In a case with a 250-fold reduction in CPU time, this implies that a 10-fold reduction is associated with the use of a quasi-Newton algorithm and a 25-fold reduction is associated with the increased accuracy of the (cubic) spline." (p. 494).

These speedups provide fairly convincing evidence for the superiority of cubic spline interpolation over simple linear interpolation, at least in the class of problems considered. However a drawback of cubic spline interpolation is that it is not guaranteed to preserve concavity or monotonicity properties of the value function. Judd and Solnick 1994 discuss a quadratic spline interpolation procedure of Schumacher 1983 that enables one to impose monotonicity and concavity restrictions relatively easily. They argue that these properties are quite important in a variety of economic problems and their numerical examples demonstrate that the extra effort involved in enforcing these constraints has a signifi cant payoff in terms of enabling one to use a smaller number of knot points to produce approximate solutions of a specifi ed degree of accuracy.

Chebyshev Polynomial Approximation. [60] Chebyshev polynomial approximation is a special case of *polynomial series approximation* which is in turn a special case of the class of linear approximation algorithms introduced in section 3.3. These algorithms approximate the time t value function $V_t(s)$ by a linear combination of the fi rst k elements of an infi nite sequence of polynomial *basis functions* $\{p_1(s), p_2(s), \ldots, p_k(s), \ldots\}$ defi ned over all $s \in S$.[61] By specifying a particular set of coeffi cients $\hat{\theta}_t = (\hat{\theta}_{t,1}, \ldots, \hat{\theta}_{t,k})$ we obtain an estimate $V_{\hat{\theta}_t}$ of the time t value function V_t given by

$$V_{\hat{\theta}_t}(s) = \sum_{i=1}^{k} \hat{\theta}_{i,t} p_i(s). \tag{5.73}$$

[60] This section borrows heavily from papers by Judd 1990 and 1994, and Judd and Solnick, 1994.

[61] We use the term "basis" in a loose sense here. In particular we do not require the basis functions to be orthogonal or linearly independent.

To implement this method we need to specify the basis functions and a method for choosing the "best fitting" coefficient vector $\hat{\theta}_t$. A natural choice for the set of basis functions are the "ordinary" polynomials $p_i(s) = s^i$ since the Weierstrauss approximation theorem shows that if S is compact, then the set of all linear combinations of the form (5.73) is dense in the space $C(S)$ of all continuous functions on S. A natural way to determine $\hat{\theta}_t$ is via the nonlinear least squares approach in equation (5.66). A problem with ordinary least squares estimation of $\hat{\theta}_t$ using the ordinary polynomial basis $\{1, s, s^2, \dots\}$ is that successive terms in the series approximation (5.73) become highly collinear as k gets large and this can create numerical problems (e.g. near-singular moment matrices) in process of computing the least squares estimate of $\hat{\theta}_t$. For example, if the grid points $\{s_1, \dots, s_N\}$ are not well distributed over the entire domain S, the approximate value function $V_{\hat{\theta}_t}$ can display explosive oscillations as k gets large, especially in regions of S where there are few grid points.

These problems motivate the use of uniformly bounded, orthogonal polynomial series approximations. There are (infinitely) many such orthogonal bases for $C(S)$. In addition, different orthogonal bases can be generated depending on how one defines the inner product on elements of $C(S)$. If μ is a measure on S, then an inner product can be constructed using μ as a "weighting function":

$$< f, g >_\mu \equiv \int_S f(s)g(s)d\mu(s)ds. \tag{5.74}$$

The Chebyshev polynomials $\{p_i\}$ are an orthogonal collection of polynomials defined on the domain $S = [-1, 1]$ where orthogonality is defined using the weighting function $d\mu(s) = 1/\sqrt{1-s^2}$.[62] The explicit formula for the $\{p_i\}$ is given by $p_i(s) = \cos(i \cos^{-1}(s))$, but they can also be defined via the recursion:

$$p_i(s) = 2sp_{i-1}(s) - p_{i-2}(s), \tag{5.75}$$

with initial conditions $p_0(s) = 1$ and $p_1(s) = s$. By definition, the Chebyshev polynomials satisfy the continuous orthogonality relation

$$< p_i, p_j >_\mu = \int_{-1}^{1} \frac{p_i(s)p_j(s)}{\sqrt{1-s^2}} ds = 0, \quad i \neq j, \tag{5.76}$$

however the key to Chebyshev approximation is the fact that the $\{p_i\}$ also satisfy the discrete orthogonality relation:

$$\sum_{l=1}^{k} p_i(s_l^k)p_j(s_l^k) = 0, \quad i \neq j, \tag{5.77}$$

[62] The fact that the Chebyshev polynomials are only defined on the $[-1, 1]$ interval does not limit the applicability of this approach, since a function $f(s)$ defined on an arbitrary interval $[a, b]$ can be transformed into an equivalent function on $[-1, 1]$ via the change of variables $s \to 2(s-a)/(b-a) - 1$.

where s_l^k are the the zeros of p_k given by:

$$s_l^k \equiv \cos\left(\frac{(2l-1)\pi}{2k}\right), \quad l = 1, \ldots, k. \tag{5.78}$$

This result implies that the Chebyshev zeros $\{s_1^k, \ldots, s_k^k\}$ can serve as a set of grid points for interpolation of an arbitrary continuous function f. Using these grid points, we can interpolate an arbitrary continuous function f by the function $\hat{f}_{k-1}(s)$ defined by:

$$\hat{f}_{k-1}(s) = -\frac{1}{2}\hat{\theta}_1 + \sum_{i=1}^{k-1} \hat{\theta}_{i+1} p_i(s), \tag{5.79}$$

where the coefficients $\hat{\theta} = (\hat{\theta}_1, \ldots, \hat{\theta}_k)$ are given by:

$$\hat{\theta}_i = \frac{2}{k} \sum_{l=1}^{k} f(s_l^k) p_{i-1}(s_l^k). \tag{5.80}$$

The function $\hat{f}_{k-1}$ defined in equation (5.79) is called the *degree $k-1$ Chebyshev interpolant of the function f*. It is easy to see that the discrete orthogonality relation (5.77) implies that $\hat{f}_{k-1}(s)$ agrees with $f(s)$ at each of the Chebyshev zeros, $s \in \{s_1^k, \ldots, s_k^k\}$, so that $\hat{f}_{k-1}$ is a polynomial of degree $k-1$ that interpolates f at these knot points.[63] The following theorem provides a uniform error bound on the approximation error of Chebyshev interpolation:

Theorem 5.10. *Suppose that $f \in C^r(S)$. Then there is a constant d_r such that for all k we have;*

$$\|f - \hat{f}_k\| \leq \left(\frac{2}{\pi}\log(k+1) + 2\right) \frac{d_r}{k^r} \|f^{(r)}\|. \tag{5.81}$$

The amount of work (i.e. number of additions and multiplications) involved in computing the coefficients $\hat{\theta}_t$ at each stage t using a degree $k-1$ Chebyshev interpolant is $O(k^2)$. Once these coefficients are computed, the effort required to evaluate $\hat{f}_k$ at any $s \in S$ is only $O(k)$. Similar to cubic splines, the smoothness properties of Chebyshev polynomials lead to the presumption that we can obtain good approximations for small values of k: this has been verified in many numerical applications, although all of the numerical examples that we are aware of are for infinite horizon problems so we defer discussion of numerical performance until section 5.4.2.

---

[63] Note that the Chebyshev coefficient estimates $\hat{\theta}$ given in (5.80) are *not* the same as the least squares estimates based on the k Chebyshev polynomials $\{p_0, \ldots, p_{k-1}\}$ and grid points $\{s_1^k, \ldots, s_k^k\}$.

The last remaining issue is whether one can use Chebyshev polynomials to interpolate continuous multidimensional functions $f : S \to R$, where $S \subset R^d$. We can do this using tensor products of the one dimensional Chebyshev polynomials:

$$\hat{f}_{k-1}(s_1, \ldots, s_d) = \sum_{i_1=1}^{k} \cdots \sum_{i_d=1}^{k} \hat{\theta}_{(i_1,\ldots,i_d)} p_{i_1-1}(s_1) \cdots p_{i_d-1}(s_d), \tag{5.82}$$

where the coefficients $\hat{\theta}_{(i_1,\ldots,i_d)}$ can be computed using a d-fold extension of formula (5.80). Besides the Chebyshev polynomials, other commonly used families of orthogonal polynomials include the *Legendre polynomials* (defined on the interval $[-1, 1]$ using the weight function $d\mu(s) = 1$), the *Laguerre polynomials* (defined on the interval $[0, \infty)$ using the weight function $d\mu(s) = se^{-s}$), and the *Hermite polynomials* (defined on $(-\infty, \infty)$ using weight function $d\mu(s) = \exp\{-s^2/2\}$).

Besides the direct formula for $\hat{\theta}$ given by the Chebyshev interpolation formula (5.80) or the indirect nonlinear least squares approach to estimating $\hat{\theta}$ in formula (5.66), the θ coefficients can be determined using a broad class of algorithms that Judd 1992 has referred to as *projection methods.* An example of a projection method is the *Galerkin method* which determines the $k \times 1$ vector $\hat{\theta}_t$ as the solution to the system of k nonlinear equations given by:

$$< V_{\hat{\theta}_t} - \hat{\Gamma}(V_{\hat{\theta}_{t+1}}), p_i >_\mu = 0, \quad i = 1, \ldots, k. \tag{5.83}$$

The Galerkin method is closely connected to least squares: $V_{\hat{\theta}_t} - \hat{\Gamma}(V_{\hat{\theta}_{t+1}})$ is a "residual function", and the Galerkin method requires that the projections of this residual function on each of the k basis functions $\{p_1, \ldots, p_k\}$ should be zero, i.e. the residual function is required to be orthogonal to the linear space spanned by $\{p_1, \ldots, p_k\}$. Unfortunately one can not literally carry out the Galerkin procedure since the inner products in (5.83) involve integrals that must be computed numerically. A problem with the Galerkin procedure relative to least squares or Chebyshev interpolation is that it is not clear what to do if the system (5.83) has no solution or more than one solution. The other problem is that all of these procedures are subject to a compound version of the curse of dimensionality: not only do the number of basis functions k necessary to attain an ϵ-approximation increase exponentially fast in d, but the amount of computer time required to find an ϵ-approximation to a nonlinear system of k equations and unknowns increases exponentially fast on a worst case basis, see Sikorski 1984, 1985.

Approximation by Neural Networks. The key problem with all of the spline and polynomial series approximation methods discussed so far is the curse of dimensionality: in each case the number of θ coefficients k required to obtain an ϵ-approximation to a given function f increases

exponentially fast in the dimension d of the state space S. In section 3.3 we discussed recent results of Barron 1993 and Hornik, Stinchcombe, White and Auer (HSWA) 1992 that show that neural networks can approximate functions of d variables on a worst case basis without using an exponentially increasing number of coefficients. This work suggests that we can break the curse of dimensionality of solving the MDP problem by approximating value functions by single layer feedforward networks of the form:

$$V_{t,\theta}(s) = \lambda_0 + \sum_{i=1}^{k} \lambda_i \phi(\delta_i' s + \rho_i), \tag{5.84}$$

where $\theta_t = (\lambda_0, \ldots, \lambda_k, \delta_1, \ldots, \delta_k, \rho_1, \ldots, \rho_k)$ has a total of $N = (d+2)k + 1$ components (since the δ_i are $d \times 1$ vectors) and $\phi : R \to R$ is a sigmoidal "squashing function" which was described in section 3.3. Unfortunately the complexity bounds presented in section 3.3 show that neural networks cannot succeed in breaking the curse of dimensionality associated with solving the dynamic programming problem. The reason is that although neural nets are capable of achieving very good approximations to functions using relatively few parameters, the Achilles heel is the curse of dimensionality associated with solving the global minimization problem necessary to find a set of neural network parameters $\hat{\theta}_t$ such that the associated network output function $V_{t,\hat{\theta}}$ best fits the true value function V_t. While there are certain tricks that can reduce the computational burden of computing $\hat{\theta}_t$, they do not succeed in circumventing the curse of dimensionality of the global minimization problem.[64] Barron concluded that "We have avoided the effects of the curse of dimensionality in terms of the accuracy of approximation but not in terms of computational complexity." (p. 933). This is exactly what a number of investigators have found in practice: neural nets lead to nonlinear least squares problems that have many local minima, and the effort required to find a local minimizer that yields good approximations to the value or policy function can be extremely burdensome. For example a recent study by Coleman and Liu 1994 used a neural net with $k = 2, 3$ and 5 hidden units and a logistic squashing function ϕ to approximate the solution to a dynamic programming problem of a household's optimal consumption/savings decisions in the presence of incomplete markets. Paraphrasing their results in our notation, they concluded that

[64] Barron showed that the approximation error in the neural network will continue to decrease at rate $1/\sqrt{k}$ if we optimize the neural net parameters sequentially starting with only one hidden unit and minimizing over $(\lambda_1, \delta_1, \rho_1)$ and then successively adding additional hidden units $2, 3, \ldots, k$ treating the parameter values for the previous hidden units as fixed. Due to the fact that the cpu-time required to find an ϵ-approximation to a global minimum of a smooth function increases exponentially in the number of variables on a worst case basis, it follows that it is significantly easier to solve k individual $(d+1)$-dimensional optimization problems than a single $k(d+1)$-dimensional problem.

> "For low values of k the method seems to perform quite poorly. For $k = 2$ the approximation was uniformly above the true solutions, and for $k = 3$ the approximation exhibits an S-shaped pattern around the true solution. To address this latter problem, for the second method we fixed 27 uniformly spaced grid points and selected the neural net that came the closest to the true solution at all these points. This method substantially improves the results, which leads to approximations which are about as good as the Chebyshev polynomial method. We wish to point out, however, that fitting the Chebyshev polynomial was trivial, but fitting the neural net was quite difficult: there evidently existed a variety of distinct local minima that needed to be ruled out." (p. 20)

Regression-based Interpolation and Monte Carlo Simulation Methods. In addition to the curse of dimensionality associated with solving the multivariate function approximation and optimization subproblems of an MDP problem, there is also a curse of dimensionality associated with the multidimensional numerical integration subproblem required to evaluate the Bellman operator $\Gamma(V)(s)$ at any point $s \in S$. As we discussed in the introduction and in section 3, randomized monte carlo integration algorithms succeed in breaking the curse of dimensionality of the integration problem. This suggests that monte carlo simulation methods might be more effective than deterministic methods such as multivariate quadrature for solving the integration subproblem in high-dimensional MDP's. Keane and Wolpin 1994 used this approach in their algorithm for approximating solutions to finite horizon DDP's, i.e. MDP's with finite choice sets $A(s)$. The Keane-Wolpin algorithm computes the alternative-specific value functions $V_t(s,a)$, $t = 0, \ldots, T$ defined by the recursion:

$$V_t(s,a) = u(s,a) + \beta \int \max_{a' \in A(s')} [V_{t+1}(s',a')] p(ds'|s,a), \tag{5.85}$$

with the terminal condition $V_T(s,a) = u(s,a)$. The $V_t(s,a)$ functions are related to the usual value functions $V_t(s)$ by the identity:

$$V_t(s) = \max_{a \in A(s)} [V_t(s,a)]. \tag{5.86}$$

To implement the Keane-Wolpin algorithm, one needs to specify a grid over the state space S, say, $\{s_1, \ldots, s_N\}$. Keane and Wolpin used randomly generated grids, although in principle it could also be deterministically chosen. At each point s_j on the grid, and for each action $a \in A(s_j)$ one draws N realizations from the transition density $p(ds'|s_j,a)$.[65] We denote these realizations

[65] There is no reason why the number of monte carlo draws must equal the number of grid points: they only reason we have imposed this restriction is to simplify notation. In principle one can choose different numbers of realizations $N_{j,a}$ at each pair (s_j, a).

$(\tilde{s}_{1ja}, \ldots, \tilde{s}_{Nja})$ to emphasize the fact that separate realizations are drawn for each conditioning pair (s_j, a). Then the Keane-Wolpin estimate of $V_t(s_j, a)$ is given by:

$$\hat{V}_t(s_j, a) = u(s_j, a) + \frac{\beta}{N} \sum_{i=1}^{N} \max_{a' \in A(\tilde{s}_{ija})} [\hat{V}_{t+1}(\tilde{s}_{ija'}, a')], \quad j = 1, \ldots, N. \tag{5.87}$$

Notice that $\hat{V}_t$ is only defined at the pre-assigned grid points $\{s_1, \ldots, s_N\}$ and not at other points $s \in S$. Since the realizations $(\tilde{s}_{1ja}, \ldots, \tilde{s}_{Nja})$ will generally not fall on the pre-assigned grid points $\{s_1, \ldots, s_N\}$, Keane and Wolpin fit a linear regression using the N simulated values $y \equiv (\tilde{V}_t(s_1), \ldots, \tilde{V}_t(s_N))$ as an $N \times 1$ dependent variable and K powers and cross-products of the d components of the grid points $\{s_1, \ldots, s_N\}$ as the $N \times K$ matrix of regressors, X. The ordinary least squares coefficient estimates $\hat{\beta} = (X'X)^{-1}X'y$ were then used to generate estimates of $\tilde{V}_t(s)$ at arbitrary points $s \in S$ outside of the preassigned grid $\{s_1, \ldots, s_N\}$. In principle one could use a variety of other non-parametric regression procedures (e.g. kernel estimation, nearest neighbor estimation, regression trees etc.) or any of the spline or polynomial interpolation methods discussed above to generate smoothed estimates $\tilde{V}_t(s)$ defined at any $s \in S$. Note that equation (5.87) implicitly assumes that some sort of interpolation procedure has already been carried out on the stage $t+1$ value function $\hat{V}_{t+1}$ so that it can be evaluated at all the random points $\tilde{s}_{ija}$, $i = 1, \ldots, N$, that were drawn at the stage t grid point (s_j, a). A final feature of the Keane-Wolpin is the use of bias-corrections to offset the cumulative upward bias in $V_t(s, a)$ induced by the backward recursion procedure. Note that even if $\tilde{V}_{t+1}$ were an unbiased estimate of V_{t+1} Jensen's inequality implies that $\tilde{V}_t$ will be an upward biased estimate of V_t:

$$\begin{aligned} E\{\tilde{V}_t(s_j, a)\} &= u(s_j, a) + E\left\{ \frac{\beta}{N} \sum_{i=1}^{N} \max_{a' \in A(\tilde{s}_{ija})} [\tilde{V}_{t+1}(\tilde{s}_{ija'}, a')] \right\} \\ &\geq V_t(s_j, a). \end{aligned} \tag{5.88}$$

This implies that each successive stage of the backward induction process will tend to compound the upward biases carried over from previous stages. Keane and Wolpin discovered a simple "bias correction" function that enabled them to generate much closer approximations to the true value function. Although Keane and Wolpin have not derived explicit error bounds or provided a formal proof of the convergence of their method, they did perform extensive numerical tests of their algorithm and concluded that "our approximation method ameliorates Bellman's 'curse of dimensionality' problem, obtaining approximate solutions for problems with otherwise intractably large state space." (p. 35). Although their use of monte carlo integration does break the curse of dimensionality of the integration subproblem, their algorithm does not avoid the curse of dimensionality

of the approximation subproblem. As we showed in section 3.3, the worst case complexity of multivariate function approximation increases exponentially in the problem dimension d regardless of whether deterministic or randomized approximation methods are used.

Smooth Approximation of the Decision Rule. All of the smooth approximation methods considered so far have focused on approximating the value function V. However in many problems the primary interest is in approximating the optimal decision rule α. Given an estimate $\hat{V}$ of V we can compute the implied estimate $\hat{\alpha}(s) = (\hat{\alpha}_0(s), \ldots, \hat{\alpha}_T(s))$ of the optimal decision rule $\alpha(s) = (\alpha_0(s), \ldots, \alpha_T(s))$ at any point $s \in S$ via the dynamic programming recursion:

$$\hat{\alpha}_t(s) = \underset{a \in A(s)}{argmax} \left[u(s,a) + \beta \int \hat{V}_{t+1}(s') p(ds'|s,a) \right], \quad t = 0, \ldots, T-1. \tag{5.89}$$

In practice $\hat{\alpha}_t(s)$ will differ from $\alpha_t(s)$ not only because $\hat{V}_t$ differs from V, but also due to any additional approximation errors introduced by numerical solution of the integration and constrained maximization subproblems in equation (5.89). If we only need to compute $\hat{\alpha}_t(s)$ at a small number of points $s \in S$, the cost of finding accurate approximate solutions to these subproblems might not be too burdensome. However if we need to evaluate $\hat{\alpha}$ at a large number of points in S, say for purposes of stochastic simulation of the optimal decision rule, it will generally be too time-consuming to compute $\hat{\alpha}_t(s)$ at many different points $s \in S$. In these cases it makes sense to incur the up-front cost of using one of the function approximation methods discussed previously (Chebyshev polynomial approximation, piecewise polynomial approximation, neural networks, etc.) to compute an approximate decision rule $\hat{\hat{\alpha}}_t$ given information $I_N = (\hat{\alpha}(s_1), \ldots, \hat{\alpha}(s_N))$ on the optimal decisions computed over a finite grid of points $\{s_1, \ldots, s_N\}$ in S. This results in a smooth approximation $\hat{\hat{\alpha}}$ to the optimal decision rule α that can be quickly evaluated at any $s \in S$.

However an alternative approach, first suggested by Smith 1991, is to develop a solution procedure that approximates α directly, completely bypassing the need to approximate V. [66] Let $\alpha_\theta(s)$ denote a parametric approximation to α, i.e. a smooth mapping $\theta \to A(s)$ depending on a $k \times 1$ vector of parameters θ which are chosen to best approximate the true optimal decision rule α. For example α_θ could be given by a polynomial series approximation or the output of a neural network. If α represents a control variable such as consumption expenditures that can take on only

[66] The Euler equation approach also bypasses the need to solve for V and will be discussed in section 5.4.

nonnegative values it is easy to defi ne parameterizations that automatically enforce this constraint, e.g.

$$\alpha_\theta(s) = \exp\left\{\sum_{i=1}^{k} \theta_i p_i(s)\right\}, \tag{5.90}$$

where the $\{p_i(s)\}$ are some polynomial basis such as the Chebyshev polynomials, etc. Other parameterizations such as splines, etc. can obviously be used as well. The motivation for Smith's procedure is the static formulation of the dynamic optimization introduced in section 2:

$$V(s) = \underset{\alpha=(\alpha_0,\ldots,\alpha_T)}{argmax} E_\alpha\left\{\sum_{t=0}^{T} \beta^t u(s_t, \alpha_t(s_t)) \,|s_0 = s\right\}. \tag{5.91}$$

We have largely avoided consideration of the direct static characterization of α in this chapter in favor of the indirect dynamic programming characterization since the static optimization problem is over the infi nite-dimensional space of all decision rules and none of the standard static optimization procedures can be expected to work particularly well over such large spaces. However the formulation suggests that if there are flexible parameterizations $\theta \to \alpha_\theta$ that provide good approximations to a wide class of potential decision rules using a relatively small number of parameters k, then we can approximate the optimal decision rule α by the parametric decision rule $\alpha_{\hat{\theta}}$ where $\hat{\theta}$ is the solution to the k-dimensional static optimization problem:

$$\hat{\theta} = \underset{\theta \in R^k}{argmax}\, V_{\alpha_\theta}(s), \tag{5.92}$$

and V_{α_θ} is the value function corresponding to the decision rule $\alpha_\theta = (\alpha_{0,\theta}, \ldots, \alpha_{T,\theta})$:

$$V_{\alpha_\theta}(s) \equiv E_{\alpha_\theta}\left\{\sum_{t=0}^{T} \beta^t u(s_t, \alpha_{t,\theta}(s_t))\Big| s_0 = s\right\}. \tag{5.93}$$

For certain MDP problems, the optimal decision rule might be a relatively simple function of s, so that relatively parsimonious parameterizations α_θ with small numbers of parameters k might do a good job of approximating α. In these cases Smith refers to the approximation $\alpha_{\hat{\theta}}$ as a ***simple rule of thumb.***

The diffi culty with this method as it has been stated so far is that we generally do not know the form of the value function $V_{\alpha_\theta}(s)$ corresponding to the decision rule α_θ. Calculation of $V_{\alpha_\theta}(s)$ is a major source of the numerical burden of solving the dynamic programming problem. However monte carlo integration can be used to break the curse of dimensionality of approximating $V_{\alpha_\theta}(s)$.

Let $\{\tilde{s}_t(\theta)\}$ denote a realization of the controlled stochastic process induced by the decision rule α_θ. This process can be generated recursively by

$$\tilde{s}_t(\theta) \sim p(\,\cdot\,|\tilde{s}_{t-1}(\theta), \alpha_{t-1,\theta}(\tilde{s}_{t-1}(\theta))), \tag{5.94}$$

where $\tilde{s}_0(\theta) = s$ is the initial condition for the simulation. Let $\{\tilde{s}_t^m(\theta)|m = 1, \ldots, M\}$ denote M independent simulations of this process. Then we can construct an unbiased simulation estimate of $V_{\alpha_\theta}(s)$ as follows:

$$\tilde{V}_{\alpha_\theta}(s) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=0}^{T}\beta^t u\left(\tilde{s}_t^m(\theta), \alpha_{t,\theta}(\tilde{s}_t^m(\theta))\right). \tag{5.95}$$

Then we can defi ne$\tilde{\theta}_M$ to be the solution to the following optimization problem

$$\tilde{\theta}_M = \underset{\theta\in R^k}{argmax}\,\tilde{V}_{\alpha_\theta}(s). \tag{5.96}$$

However this straightforward simulation approach is subject to a number of severe problems. First, the simulated value function $\tilde{V}_{\alpha_\theta}$ will generally not be a continuous function of θ, complicating the solution to the maximization problem (5.96). Second, we must re-simulate the controlled process in (5.94) each time we choose a new trial value of θ in the process of solving the maximization problem (5.96). This introduces extra noise into the optimization creating a phenomenon known as "chattering" which can prevent the simulation estimate $\tilde{\theta}_M$ from converging to its intended limit $\hat{\theta}$ as $M \to \infty$. The chattering problem can make it impossible to consistently estimate the optimal decision rule α even if the parameterization of the decision rule α_θ (5.90) is suffi ciently flexible to enable it to provide arbitrarily accurate approximations to an arbitrary decision rule as $k \to \infty$.

Smith 1991 showed that we can avoid both of these problems if the MDP is a CDP in the Euler class of Defi nition 2.4. Recall that problems in this class have a continuous decision variable (which we denote by c rather than a) and a partitioned state vector $s = (y, z)$ where z is an exogenous state variable with Markov transition probability $q(z'|z)$ and y is an endogenous state variable with law of motion given by $y' = r(y, c, z', z)$. Given a stochastic realization of the exogenous state variable $(\tilde{z}_1 \ldots, \tilde{z}_T)$ with initial value $z_0 = z$, defi ne the corresponding realization of the endogenous state variables $(\tilde{y}_1(\theta), \ldots, \tilde{y}_T(\theta))$ by the recursion

$$\tilde{y}_{t+1}(\theta) = r(\tilde{y}_t(\theta), \tilde{c}_t(\theta), \tilde{z}_{t+1}, \tilde{z}_t), \tag{5.97}$$

where $\tilde{c}_t(\theta) = \alpha_{t,\theta}(\tilde{y}_t(\theta), \tilde{z}_t)$ and $y_0 = y$ is the initial condition for the endogenous state variable. Smith's algorithm begins by generating M independent realizations of the exogenous state variables

which remain fixed for all subsequent trial values of θ. However each time θ changes we update the M corresponding realizations of the endogenous state variables using the recursion (5.97). Using these simulations we can define a monte carlo estimate $\tilde{V}_{\alpha_\theta}(s)$ of $V_{\alpha_\theta}(s)$ by:[67]

$$\tilde{V}_{\alpha_\theta}(s) = \tilde{V}_{\alpha_\theta}(y,z) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=0}^{T}\beta^t u(\tilde{y}_t(\theta), \tilde{z}_t, \alpha_\theta(\tilde{y}_t(\theta), \tilde{z}_t)). \tag{5.98}$$

It is easy to verify that as long as the function $r(y,c,z',z)$ is a continuously differentiable function of y and c that $\tilde{V}_{\alpha_\theta}(s)$ will be a continuously differentiable function of θ. Furthermore the fact that the realizations of the exogenous state variables remain fixed even though θ changes implies that a fundamental *stochastic equicontinuity* condition is satisfied, eliminating the chattering problem of the naive simulator (5.94). Smith proved the following theorem which bears many similarities to asymptotic results for "simulation estimators" that have arisen recently in econometrics:

Theorem 5.11. *Under certain additional regularity conditions in Smith 1991, as $M \to \infty$ we have:*

$$\begin{aligned} &\tilde{\theta}_M \to \hat{\theta} \quad \textit{w.p.1} \\ &\sqrt{M}\left[\hat{\theta}_M - \hat{\theta}\right] \Longrightarrow \tilde{Z} \sim N(0,\Sigma), \end{aligned} \tag{5.99}$$

where $\tilde{\theta}$ is the solution to maximization problem (5.92) and Σ is the asymptotic covariance matrix of the simulation-based estimate $\hat{\theta}_M$ in equation (5.96).

If the parameterization of the decision rule α_θ is ultimately dense in the space of possible decision rules (i.e. for any possible decision rule α we have $\inf_\theta \|\alpha_\theta - \alpha\| \to 0$ as $k \to \infty$ where k is the dimension of θ), then it seems reasonable to conjecture that Smith's theorem should imply that $\alpha_{\hat{\theta}_M} \to \alpha$ provided that k and M tend to infinity at the "right" rates, although to our knowledge this result has not yet been formally established. However Smith compared his numerical solutions to the analytical solution of the Brock-Mirman model given in equation (2.55) of example 2 of section 2.6. His numerical solutions were based on linear and quadratic rules of thumb for α that depend on $k = 3$ and $k = 6$ unknown parameters, respectively. He also considered a third rule of thumb given by a simple one parameter "partial adjustment model" about the steady state capital stock that would result if the technology shock z_t were assumed to remain indefinitely at its current level. Smith evaluated the quality of these three rules of thumb in terms of the loss in discounted utility relative to the optimal decision rule and in terms of how well the fitted rules approximated the optimal decision rule. He found that:

[67] In practice, Smith uses a procedure known as *antithetic variates* to reduce the variance in the monte carlo estimates of $\hat{V}_{\alpha_\theta}(s)$.

> "from a welfare perspective, the three rules of thumb are nearly indistinguishable from the optimal rule. For example, the welfare loss from using the optimal linear rule rather than the truly optimal rule is equivalent to losing only 0.01% of per period consumption. The corresponding losses for the optimal quadratic rule and the 'partial adjustment of the capital stock' rule are, respectively, $8.3 \times 10^{-5}\%$ and $2.0 \times 10^{-5}\%$. The 'partial adjustment of the capital stock' and the optimal quadratic rule yield capital stock decisions that deviate only rarely by more than 0.5% from optimal behavior. The results of this section show that ... parsimoniously parameterized rules of thumb can mimic optimal behavior very closely, according to a variety of metrics. Indeed, a surprising finding is that a one-parameter family of decision rules (the 'partial adjustment of the capital stock' rule) outperforms a six parameter family (the optimal quadratic rule) along all dimensions considered. The success of the 'partial capital stock adjustment' rule shows that rules of thumb which incorporate some of the economic structure underlying the optimization problem can perform better than 'brute force' polynomial expansions." (p. 18–20).

Geweke, Slonim, and Zarkin (GSZ) 1992 extended Smith's approach to DDP's. Recall that DDP's have discrete rather than continuous control variables and do not have the convenient structure of the Euler class of definition 2.4 that allowed Smith to generate unbiased simulations of $V_{\alpha_\theta}(s)$ without sacrificing smoothness and stochastic equicontinuity in θ. The extra complexity of the DDP framework necessitates a somewhat more complicated backward induction approach to simulating the value function in which the calculations of the best fitting parameter vector θ are done in stages starting from period T and working backward. To make this clear it is useful to partition the parameter vector as $\theta = (\theta_0, \ldots, \theta_T)$ where each of the θ_t subvectors will be computed via a recursive series of maximum likelihood estimation problems to be described below. For simplicity, we will assume that there are $|A|$ possible actions in each state. Then the optimal decision rule is $\alpha(s) = (\alpha_0(s), \ldots, \alpha_T(s))$ where each $\alpha_t : S \to \{1, \ldots, |A|\}$. GSZ suggested approximating each of the α_t by a multinomial probability distribution over $\{1, \ldots, |A|\}$, given by

$$P_{\theta_t}(a|s) = \frac{\exp\left\{\sum_{i=1}^k \theta_t^{a,i} p_i(s)\right\}}{\sum_{a'=1}^{|A|} \exp\left\{\sum_{i=1}^k \theta_t^{a',i} p_i(s)\right\}}. \tag{5.100}$$

where the $\{p_i\}$ are a set of basis functions for a linear series approximation (e.g. Chebyshev polynomials) and $\theta_t^{a,i}$ denotes the component of the vector θ_t corresponding to alternative a and basis function i. In order for the maximum likelihood estimator of θ_t to be well-defined, it is necessary to impose a normalization that $\theta_t^{|A|,i} = 0$, $i = 1, \ldots, k$. This implies that the mapping from $\theta_t \to P_{\theta_t}$ is $1:1$ which is sufficient to identify the $(|A|-1)k$ unrestricted coefficients of θ_t. McFadden 1981 has shown that if the basis functions $(p_1, \ldots, p_k)$ are rich enough to approximate arbitrary continuous functions of s as $k \to \infty$, then the multinomial choice probabilities (5.100)

can provide a uniform ϵ-approximation to any conditional probability distributions $P(a|s)$ on the set $\{1, \ldots, |A|\}$ which is continuous in s over a compact domain S.

In order to describe the recursive procedure for estimating the $T+1$ subvectors of θ, consider the terminal period T. Draw N state vectors $(s_T^1, \ldots, s_T^N)$ at random from S. For each s_T^i, compute the optimal decisions $\alpha(s_T^i)$ by the following formula:

$$\alpha_T(s_T^i) = \underset{a \in A}{argmax}[u(s_T^i, a)]. \tag{5.101}$$

Then using the "data set" consisting of the N pairs $(s_T^i, \alpha_T(s_T^i))$ as "observations", compute $\hat{\theta}_T$ as the solution to the following maximum likelihood estimation problem:

$$\hat{\theta}_T = \underset{\theta_T}{argmax} \prod_{i=1}^{N} P_{\theta_T}(\alpha_T(s_T^i)|s_T^i). \tag{5.102}$$

This likelihood function is globally concave in θ_T so that convergent estimates of $\tilde{\theta}_T$ are easily computed provided that N is suffi ciently large (McFadden, 1973). The estimated choice probability $P_{\tilde{\theta}_T}(a|s)$ can be thought of as a probabilistic approximation to the optimal decision rule $\alpha_T(s)$.

At time step t of the backward induction process we generate N vectors $(s_t^1, \ldots, s_t^N)$ at random from S. For each pair (s_t^i, a), $a = 1, \ldots, |A|$, $i = 1, \ldots, N$ we generate M *IID* sequences $\{\tilde{s}_{t+j}^m(s_t^i, a), \tilde{a}_{t+j}^m(s_t^i, a)|j = 1, \ldots, T-t\}$ by drawing $\tilde{s}_{t+j}^m(s_t^i, a)$ sequentially from the transition probability $p(\cdot\,|\tilde{s}_{t+j-1}^m(s_t^i, a), \tilde{a}_{t+j-1}^m(s_t^i, a))$ and drawing $\tilde{a}_{t+j}^m(s_t^i, a)$ from the estimated choice probability $P_{\tilde{\theta}_{t+j}}(\cdot\,|\tilde{s}_{t+j}^m(s_t^i, a))$, $j = 1, \ldots, T-t$, subject to the initial conditions $\tilde{s}_t^m(s_t^i, a) = s_t^i$ and $\tilde{a}_t^m(s_t^i, a) = a$, $i = 1, \ldots, N$ and $a = 1, \ldots, |A|$. Using these M simulated paths to construct estimates of the time $t+1$ value function, compute estimates of $\hat{\alpha}_t(s_t^i)$, $i = 1, \ldots, N$ as follows:

$$\tilde{\alpha}_t(s_t^i) = \underset{a \in A}{argmax} \left[u(s_t^i, a) + \frac{1}{M} \sum_{m=1}^{M} \sum_{j=1}^{T-t} \beta^j u\left(\tilde{s}_{t+j}^m(s_t^i, a), \tilde{a}_{t+j}^m(s_t^i, a)\right) \right]. \tag{5.103}$$

Using the N pairs $(s_t^i, \tilde{\alpha}_t(s_t^i))$ as data, estimate $\tilde{\theta}_t$ by maximum likelihood using the likelihood function similar to (5.102). The resulting estimated choice probability function $P_{\tilde{\theta}_t}(a|s)$ can be thought of as a probabilistic estimate of $\alpha_t(s)$, $t = 0, \ldots, T$.

GSZ do not provide a formal proof of the convergence of their procedure. Such a proof would have to specify the rates at which M, N and k would have to tend to infi nity in order to guarantee that the choice probabilities converge to the optimal decision rule: i.e. $P_{\tilde{\theta}_t}(a|s) \to I\{a = \alpha_t(s)\}$ with probability 1 uniformly in a and s. However GSZ present an extensive numerical evaluation of the

ability of their method to approximate the solution to a model of retirement behavior with $|A| = 2$ possible decisions in each state ($a = 1$ retire, $a = 0$ continue working), $T = 10$ periods, and a two dimensional continuous state variable $s_t = (y_t, w_t)$ consisting of current income y_t and net worth w_t. Since there is no known exact analytical solution that they could compare their approximate solutions to, GSZ treated the "truth" as the solution generated by their procedure for a large values of N, M and k (i.e. $N = 25,000$, $M = 500$, and $k = 10$, respectively). The polynomial basis functions p_i were chosen to be ordinary polynomials and cross products of polyomials of degree 3 or less in the two components of s_t. They concluded that

> "Over the wide range of parameter settings for the solution algorithm considered, there is little change in expected utility for the illustrative problem. The 'best' solution requires about an hour of Sparc I time, but there are other solutions that produce very little loss in expected utility relative to that entailed in routine specifi cation errors, and statistically indistinguishable decision rules, but require only about fi ve minutes of Sparc I time." (p. 18).

While many of the solution methods presented in this section appear very promising for the solution of small to medium scale continuous MDP problems, especially in their use of monte carlo simulation to break the curse of dimensionality of the high dimensional integration subproblems required to compute $V_{\alpha\theta}$, the achilles heel of all of these methods is the curse of dimensionality of the embedded optimization and approximation problems. For example Smith's method encounters a compound form of the curse of dimensionality since the amount of computer time required to fi nd an ϵ-approximation to the global maximum in (5.96) increases exponentially fast as k increases and the value of k required to guarantee that $\alpha_\theta(s)$ provides an ϵ-approximation to an arbitrary decision rule $\alpha(s)$ increases exponentially fast in the dimension d of the state variables increases. The GSZ method avoids the curse of dimensionality of the optimization problem (since the likelihood function is globally concave in θ and Nemirovsky and Yudin have shown that the worst case complexity of concave optimization problems increases linearly in the number of variables, k). However their method does not avoid the curse of dimensionality of the approximation problem, since the number of basis functions k used in the choice probabilities must increase exponentially fast as d increases in order to guarantee an ϵ-approximation to α. However it is not clear that the curse of dimensionality is really a practical problem for the kinds of MDP's that are currently being solved: both Smith's and GSZ's results show that very parsimonious parameterizations of α can succeed in generating reasonably accurate approximations for problems of current interest in economic applications. If their results carry over to higher dimensional problems, it suggests the possibility that these smooth approximation methods might be able to circumvent the curse of dimensionality, at least on an average case basis.

5.4 Continuous Infinite Horizon MDP's

Most of the basic approaches to approximating continuous infinite horizon MDP's have already been outlined in the previous section on continuous finite horizon MDP's. The main additional complication in the infinite horizon case is that there is no terminal period from which to begin a backward induction calculation: since V is the fixed point to Bellman's equation $V = \Gamma(V)$ we need to employ iterative methods to approximate this fixed point in addition to approximating the infinite-dimensional Bellman operator $\Gamma : \mathcal{B}(S) \to \mathcal{B}(S)$. As in the previous section we present two general approaches to approximating V: 1) discrete approximation, and 2) smooth approximation.

As we noted in the introduction, smooth approximation methods typically do not exploit the fact that Γ is a contraction operator, so proving the convergence of these methods is more difficult. However smooth approximation methods may do a better job of exploiting other features of the MDP problem such as the smoothness of V or the fact that the MDP is in the Euler class for which Euler equations can be derived. However the results of section 3.3 on the curse of dimensionality of the multivariate function approximation problem imply that all of the standard smooth approximation methods for MDP problems are also subject to the curse of dimensionality, at least on a worst case basis.

Discrete approximation methods compute an approximate fixed point $\hat{V}_N$ to an approximate finite-dimensional Bellman operator $\Gamma_N : R^N \to R^N$ that fully preserves the contraction property of Γ. Since these methods involve relatively straightforward extensions of the general methods for contraction fixed points in section 4 and the more specialized methods for discrete infinite horizon MDP's in section 5.2, it has been much easier to derive computable error bounds and relatively tight upper and lower bounds on their computational complexity. However the lack of corresponding error bounds for most of the existing smooth approximation methods for MDP's makes it is difficult to say much in general about the the relative efficiency of discrete versus smooth approximation methods. A starting point for such an analysis is the complexity bound for continuous infinite horizon MDP's derived by Chow and Tsitsiklis 1989. Their complexity bound shows that *no algorithm* is capable of breaking the curse of dimensionality, at least on a worst case basis.

Theorem 5.12:. *The worst case deterministic complexity of the class of continuous infinite horizon MDP problems satisfying regularity conditions* $(A1), \ldots, (A5)$ *of section 5.3.1 using the standard*

information (i.e. function evaluations) on (u, p) *is given by:*

$$comp^{\textit{wor-det}}(\epsilon, d_s, d_a, \beta) = \Theta\left(\frac{1}{((1-\beta)^2\epsilon)^{2d_s+d_a}}\right). \tag{5.104}$$

Recall from section 3 that Θ symbol denotes both an upper and lower bound on the amount of cpu-time required by *any* deterministic algorithm to compute an ϵ-approximation to V on a worst case basis. An heuristic explanation of the exponential lower bound (5.104) is that it is a product of the complexity bounds of three "subproblems" involved in computing an approximate solution to a continuous MDP in (5.65): 1) a $\Theta(1/\epsilon^{d_a})$ bound on the complexity in finding an ϵ-approximation to the global maximum of the constrained optimization problem, 2) a $\Theta(1/\epsilon^{d_s})$ bound on the complexity of finding an ϵ-approximation to the multivariate integral, and 3) a $\Theta(1/\epsilon^{d_s})$ bound on the complexity of the approximation problem, i.e. the problem of finding a (Lipschitz) continuous function $\hat{V}$ that is uniformly within ϵ of the fixed point $V = \Gamma(V)$.[68] Interestingly, in a subsequent paper Chow and Tsitsiklis 1991 show that a simple discrete approximation method — a *multigrid algorithm* — nearly attains the complexity bound (5.104) and hence constitutes an approximately optimal algorithm for the general continuous MDP problem. We are not aware of any corresponding optimality results for smooth approximation methods.

It is important to note three key assumptions underlying the Chow and Tsitsiklis complexity bound: 1) the class of MDP problems considered are CDP's (continuous decision processes), 2) only deterministic algorithms are considered, and 3) complexity is measured on a worst case basis. We review the recent paper of Rust 1994c that shows that another discrete approximation method, a *random multigrid algorithm*, succeeds in breaking the curse of dimensionality of infinite horizon DDP's (i.e MDP's with finite choice sets). This result is due to the fact that 1) essentially all of the work involved in approximating the solution to a DDP is the multivariate integration problem, and 2) randomization breaks the curse of dimensionality of the integration problem. However randomization does not succeed in breaking the curse of dimensionality of the problem of multivariate function approximation. This implies that none of the standard smooth approximation methods are capable of breaking the curse of dimensionality of the DDP problem.

[68] The surprising aspect of the Chow Tsitsiklis complexity bound is that the bound is proportional to the complexity bound for the problem of approximating the function $\Gamma(W)$ for a fixed lipschitz continuous function W. One would expect that the problem of finding a fixed point to Γ to be inherently more difficult than simply evaluating Γ at a particular value W, and we would expect that this extra difficulty ought to be reflected by an extra factor of ϵ representing the extra work involved in finding more accurate approximations to the fixed point. This is certainly the case for standard iterative algorithms such as successive approximations or Newton-Kantorovich iterations in which the number of evaluations of Γ for various trial values of V_k increases to infinity as $\epsilon \to 0$, but not for the optimal algorithm. We will provide further intuition as to why this is true below.

Randomization is also incapable of breaking the curse of dimensionality associated with the maximization subproblem of the MDP problem. In this sense CDP's are inherently more difficult problems than DDP's. The only hope for breaking the curse of dimensionality for CDP's is to focus on subclasses of CDP's with further structure (i.e. CDP's where the maximization subproblem in (5.65) is concave for each $s \in S$), or to evaluate the complexity of MDP problems on an average instead of a worst case basis. However we do not know whether there are algorithms that are capable of breaking the curse of dimensionality of MDP's on an average case basis: although section 3.3 presented recent results of Woźniakowski 1992 that proves that the approximation problem is tractable on an average case basis, it is still an open question whether the optimization problem is tractable an average case basis (Wasilkowski, 1994).

5.4.1 Discrete Approximation Methods

Discrete approximation methods involve replacing the continuous state version of Bellman's equation (5.65) by a discretized version

$$V_N(s_i) = \Gamma_N(V_N)(s_i) = \max_{a \in A(s_i)} \left[u(s_i, a) + \beta \sum_{j=1}^{N} V_N(s_j) p_N(s_j | s_i, a) \right], \qquad (5.105)$$

defined over a finite grid of points $\{s_1, \ldots, s_N\}$ in the state space S. [69] Section 5.3.1 discussed four possible ways of choosing grid points: 1) uniform (deterministic) grids, 2) quadrature grids, 3) uniform (random) grids, and 4) low discrepancy grids. Each of these grids lead to natural definitions of an approximate discrete Markov transition probabilities p_N over the finite state space $\{s_1, \ldots, s_N\}$, which implies that the approximate Bellman operator Γ_N is a well defined contraction mapping from R^N into itself for each N. As we showed in section 5.3.1, each of the "discretized" transition probabilities $p_N(\cdot|s, a)$ are also continuous functions of s and are defined for all $s \in S$. This implies that each of the approximate Bellman operators that we considered in section 5.3.1 also have the property of being *self-approximating*, i.e. $\Gamma_N(V)(s)$ can be evaluated at any $s \in S$ by replacing s_i by s in formula (5.105) without the need to resort to any auxiliary interpolation or approximation procedure. Thus, Γ_N has a dual interpretation: it can be viewed as mapping from R^N into R^N when we restrict attention to its values on the grid $\{s_1, \ldots, s_N\}$, but it can also be viewed as a mapping from $\mathcal{B}(S)$ into $\mathcal{B}(S)$. Thus, quantities such as $\|\Gamma_N(W) - \Gamma(W)\|$

[69] As we discussed in section 5.3.1 discrete approximation methods do not necessarily require discretization of the action sets $A(s_i)$, $i = 1, \ldots, N$. Any one of the range of deterministic or stochastic constrained optimization algorithms including "continuous" optimization methods such as gradient hill climbing algorithms, Nelder-Mead polytope algorithms, or random algorithms such as simulated annealing could be used as well.

are well defined, and quantities such as $\|V_N - V\|$ are well defined provided we interpret V_N as its "extension" to $\mathcal{B}(S)$ via Γ_N, i.e. $V_N(s) = \Gamma_N(V_N)(s)$ for $s \notin \{s_1, \ldots, s_N\}$.

Even though V_N (viewed now as the fixed point to Γ_N in R^N) can be computed exactly (modulo roundoff error) by the policy iteration algorithm described in section 5.2, we will generally make do by calculating an approximate fixed point $\hat{V}_N$ of Γ_N. Using the triangle inequality and Lemma 2.1, we can derive the following bound on the error $\|\hat{V}_N - V\|$ between the true and approximate solution to the MDP problem (where we now treat $\hat{V}_N$ and V_N as their extensions to $\mathcal{B}(S)$ via Γ_N):

$$\begin{aligned} \|\hat{V}_N - V\| &= \|\hat{V}_N + V_N - V_N - V\| \\ &\leq \|\hat{V}_N - V_N\| + \|V_N - V\| \\ &\leq \|\hat{V}_N - V_N\| + \frac{\|\Gamma_N(V) - \Gamma(V)\|}{(1-\beta)}. \end{aligned} \tag{5.106}$$

The "fixed point approximation error" is represented by the quantity $\|\hat{V}_N - V_N\|$ and the "discretization error" is represented by the quantity $\|\Gamma_N(V) - \Gamma(V)\|$. The error bounds derived in section 5.2 allow one to make the fixed point approximation error less than $\epsilon/2$ and the error bounds presented in section 5.3.1 allow one to choose a sufficiently fine grid (i.e. a sufficiently large N) so that the discretization error is less than $(1-\beta)\epsilon/2$ uniformly for all V. It then follows from inequality (5.106) that the maximum error in the approximate solution $\hat{V}_N$ will be less than ϵ. We now turn to descriptions of two broad classes of discrete approximation algorithms: 1) single grid methods and 2) multigrid methods. Multigrid methods reduce the fixed point approximation error and the discretization error in tandem, beginning with a coarse grid in order to reduce the computational burden of getting to a neighborhood of V, but generating successively finer grids to refine the level of accuracy as the algorithm begins to home in on the solution.

Single Grid Methods. Single grid methods consist of two steps: 1) choose a procedure for generating a grid and generate N grid points $\{s_1, \ldots, s_N\}$ in S, 2) use one of the solution methods for discrete infinite horizon MDP's described in section 5.2 to find an approximate fixed point $\hat{V}_N$ to the approximate Bellman operator Γ_N corresponding to this grid. Once one has chosen a method for generating grid points and has determined a number of grid points N to be generated, the level of discretization error (i.e. the second term in inequality (5.106)) is fixed. The only remaining decision is how many iterations of the discrete infinite horizon MDP solution algorithm should be performed to reduce the fixed point approximation error to an appropriate level. A standard choice for the latter solution algorithm is the method of successive approximations, especially in view of the results of section 4 which suggest that successive approximations may be an approximately optimal

algorithm for solving the fi xed point problem, at least if the dimension N is suffi ciently large. Using the upper bound $T(\epsilon, \beta)$ on the number of successive approximation steps from equation (5.3) of section 5.2.1, it follows that the ϵ-complexity of successive approximations is $O(T(\epsilon, \beta)|A|N^2)$, where $|A|$ denotes the amount of cpu-time required to solve the constrained optimization problem in (5.105). Now consider the use of uniformly spaced grid points. Using inequality (5.49) it is easy to see that $N = \Theta(1/[(1-\beta)^2\epsilon]^{d_s})$ grid points are required in order to guarantee that $\|\Gamma_N(V) - \Gamma(V)\| \leq (1-\beta)\epsilon$ uniformly for all V satisfying $\|V\| \leq K/(1-\beta)$, where $K \equiv max_{s \in S}\, max_{a \in A(s)}\, |u(s,a)|$. If we solve the maximization problem in (5.105) by discretizing the d_a-dimensional action sets $A(s)$, it can be shown that a total of $|A| = \Theta(1/[(1-\beta)^2\epsilon]^{d_a})$ points are required to guarantee that $\|\Gamma_N(V) - \Gamma(V)\| \leq (1-\beta)\epsilon$ uniformly for all V satisfying $\|V\| \leq K/(1-\beta)$. Using inequality (5.106) it follows that if $|A|$ and N are chosen this way and if $T(\epsilon, \beta)$ successive approximation steps are performed, then the ϵ-complexity of successive approximations over a uniform grid, $comp^{sa,ug}(\epsilon, \beta, d_a, d_s)$, is given by:

$$comp^{sa,ug}(\epsilon, \beta, d_a, d_s) = \Theta\left(\frac{T(\epsilon, \beta)}{((1-\beta)^2\epsilon)^{2d_s + d_a}}\right). \tag{5.107}$$

Notice that the cpu-time requirement is a factor of $T(\epsilon, \beta)$ higher than the lower bound in (5.104). Thus we see that successive approximations using a single fi xed uniform discretization of the state and action spaces is not even close to being an optimal algorithm for the continuous MDP problem. In view of the additional diffi culty involved in computing the approximate transition probability p_N, (discussed following equation (5.47) in section 5.3.1) this method is *not* recommended for use in practical applications.

A potentially superior choice is the quadrature grid and the simpler Nyström/Tauchen method for forming the approximate fi nite state Markov chain given in equation (5.52) of section 5.3.1. Tauchen 1990 used this approach to approximate the value function to the Brock-Mirman stochastic optimal growth problem (example 2 in section 2.6). Recall that this problem has a 2-dimensional continuous state variable, $s_t = (k_t, z_t)$ where k_t is the endogenous capital stock state variable and z_t is the exogenous technology shock state variable. Tauchen used a 20 point Gauss-Hermite quadrature rule to calibrate a discrete Markov chain approximation to the technology shock transition density $q(z_{t+1}|z_t)$ and 90 equispaced gridpoints for $\log(k_t)$. The resulting grid over the two dimensional state space S contained $N = 20 \times 90 = 1800$ points. Given this grid, Tauchen used successive approximations, terminating when the percentage change in successive iterations was less than $.001$. Using a Compaq 386/25 megahertz computer and the Gauss programming language, it took 46 cpu minutes to compute an approximate solution $\hat{V}_N$. The resulting solution

is extremely accurate: comparing the implied optimal decision rule $\hat{\alpha}_N$ to the analytic solution in the case of log utility and 100% depreciation presented in equation (2.55) of section 2.6, Tauchen found that "The algorithm is only off by one digit or so in the fourth decimal place." (p. 51). Although Tauchen's method simplifies the numerical integration problem involved in computing the normalizing factors in the denominator of the formula for p_N, the use of quadrature grids does not eliminate the curse of dimensionality of the underlying integration problem since it is subject to the same curse of dimensionality as any other deterministic integration algorithm. As a result, the complexity bound for quadrature methods is basically the same as for uniform grids:

$$comp^{sa,qg}(\epsilon, \beta, d_a, d_s) = \Theta(comp^{sa,ug}(\epsilon, \beta, d_a, d_s)). \tag{5.108}$$

where *qg* stands for quadrature grid. More generally, the results of section 3.2 imply that the worst case complexity of *any* single grid method that uses successive approximations and a deterministically chosen grid will be at least as large as for the naive uniform grid.

Rust 1994c showed that a variation of discrete approximation using *random* uniform grids (i.e. N *IID* uniformly distributed draws from S) does succeed in breaking the curse of dimensionality of the integration subproblem. Although we have noted that randomization in incapable of breaking the curse of dimensionality of the maximization subproblem in CDP's, it does break the curse of dimensionality of the DDP problem since essentially all the work involved approximating the Bellman operator to a DDP problem is in the multivariate integration subproblem. Rust proved that successive approximations using the random Bellman operator $\tilde{\Gamma}_N$ defined equation (5.57) of section 5.3.1 succeeds in breaking the curse of dimensionality of solving the DDP problem on a worst case basis. For simplicity, assume that $S = [0,1]^d$, and that a random grid $\{\tilde{s}_1, \ldots, \tilde{s}_N\}$ consists of N *IID* uniform random draws from S. The error bound in Theorem 5.8 shows that randomization breaks the curse of dimensionality of the integration problem. Theorem 5.9 provides an upper bound on the randomized complexity of the DDP problem for finite horizon problems, and the bound extends to infinite horizon problems by substituting $T(\epsilon, \beta)$ for the horizon length T in equation (5.62). For completeness, we state this result as the following corollary to Theorem 5.9:

Corollary. *Under the regularity conditions of Theorem 5.8 randomization breaks the curse of dimensionality of solving infinite horizon DDP problems. An upper bound on the worst case randomized complexity of the infinite horizon DDP problem is given by:*

$$comp^{\textsf{wor-ran}}(\epsilon, \beta, d) = O\left(\frac{\log\left(1/(1-\beta)\epsilon\right) d^4 |A|^3 K_u^4 K_p^4}{|\log(\beta)|(1-\beta)^8 \epsilon^4}\right). \tag{5.109}$$

where the constants K_u and K_p are the Lipschitz bounds for u and p given in Definition 5.1.

Finally, consider the use of low discrepancy grids $\{s_1, \ldots, s_N\}$. These are deterministically chosen grids such as the Hammersley, Halton, or Sobol' points described in section 5.3.1. The approximate discrete Markov chain approximation p_N to the transition density is formed exactly the same as in equation (5.58) for the randomly chosen grid, but with the deterministically chosen grid points $\{s_1, \ldots, s_N\}$ in place of the random grid points $\{\tilde{s}_1, \ldots, \tilde{s}_N\}$. Since low discrepancy grids are deterministically chosen, they will be subject to the same curse of dimensionality that any other deterministic integration method faces on a worst case basis. However the results of Woźniakowski and Paskov discussed in section 3.2 show that low discrepancy grids do succeed in breaking the curse of dimensionality on an average case basis. These theoretical results have been impressively confirmed in result numerical comparisons by Paskov 1994 that show that low discrepancy grids are significantly faster than uniform random grids for solving high dimensional ($d = 360$) integration problems arising in finance. These results suggest that successive approximations based on an approximate Bellman operator Γ_N formed from low discrepancy grids could also succeed in breaking the curse of dimensionality of solving DDP problems on an average case basis. However to our knowledge this conjecture has not been formally proven.

Finally, we have so far only considered the use of successive approximations to calculate an approximate fixed point $\hat{V}_N$ of Γ_N. Clearly any of the other solution methods discussed in section 5.2 could be used as well. Particularly promising algorithms include the Trick-Zin linear programming approach using constraint generation, policy iteration methods that approximately solve the linear system $(I - \beta M_\alpha)V_\alpha = u_\alpha$ using the GMRES algorithm, Puterman's modified policy iteration algorithm, and the state aggregation methods presented in section 5.2. Although the 1987 result of Sikorski-Woźniakowski given in Theorem 4.2 of section 4 suggests that simple successive approximations is an almost optimal algorithm in high dimensional problems (i.e. for large N), we observed that their theorem assumed that the cost of evaluating the Bellman operator is independent of N which is clearly not the case for discrete approximation methods. It is not clear whether their result will continue to hold in a complexity analysis that recognizes that cost of evaluating $\Gamma_N(V)$ increases with N. However, we will now consider the broader class of multigrid algorithms that successively refine the grid over S. It turns out that simple successive approximations is indeed almost optimal within this larger class.

Multigrid Methods. Multigrid methods have attracted substantial attention in the literature on numerical solution of partial differential equations where they have been found to lead to substantially

faster convergence both theoretically and in practice. [70] To our knowledge Chow and Tsitsiklis 1991 were the first to show how multigrid methods can be applied to continuous MDP's. They proposed a "one way" multigrid algorithm that works as follows:

1. Beginning with a coarse uniform grid over S containing $N_1 = (1/h)^{d_s}$ grid points $\{s_1, \ldots, s_{N_1}\}$, (where h is a sufficiently small positive constant), use the implied approximate Bellman operator Γ_{N_1} to carry out T_1 successive approximation steps starting from an arbitrary initial estimate $\hat{V}_0$ of V where T_1 is the smallest number of successive approximation iterations until the following inequality is satisfied:

$$\|\Gamma_{N_1}^{T_1}(\hat{V}_0) - \Gamma_{N_1}^{T_1-1}(\hat{V}_0)\| \leq \frac{2K'h}{\beta(1-\beta)}, \tag{5.110}$$

   where K' is a bounding constant determined by the Lipschitz constants for u and p in assumptions $(A1), \ldots, (A5)$ and h satisfies $h \leq 1/2K'$.

2. At iteration k of the multigrid algorithm generate a new uniform grid with $N_k = 2^{d_s} N_{k-1}$ points. Use the value function $\hat{V}_k = \Gamma_{N_{k-1}}^{T_{k-1}}(\hat{V}_{k-1})$ from the final successive approximation step from the previous grid as the starting point for successive approximations using the new grid. [71] Carry out out T_k successive approximations steps starting from $\hat{V}_k$ where T_k is the smallest number of successive approximations steps until the following inequality is satisfied:

$$\|\Gamma_{N_k}^{T_k}(V_k) - \Gamma_{N_k}^{T_k-1}(V_k)\| \leq \frac{2K'h}{2^{k-1}\beta(1-\beta)}, \tag{5.111}$$

3. Terminate the outer grid generation iterations $k = 1, 2, 3, \ldots$ at the smallest value $\hat{k}$ satisfying:

$$\frac{2K'h}{2^{\hat{k}-1}\beta(1-\beta)} \leq \epsilon \tag{5.112}$$

   where ϵ is a predefined solution tolerance.

---

[70] See Hackbusch 1985 for a survey, and the Web server `http://info.desy.de/pub/ww/projects/MG.html` for a comprehensive and up to date bibliography of multigrid methods and applications.

[71] Actually Chow and Tsitsiklis suggested a better estimate of $\hat{V}_k$, namely the average of the upper and lower McQueen-Porteus error bounds after T_{k-1} successive approximation steps, see equation (5.6) in section 5.2.1. We have used the value of the last successive approximation step only to simplify notation.

Chow and Tsitsiklis proved that the multigrid algorithm terminates with a value function $\hat{V}_{\hat{k}+1}$ that satisfi es $\|\hat{V}_{\hat{k}+1} - V\| \leq \epsilon$. Furthermore they proved that the multigrid algorithm is almost optimal in the sense that its complexity $comp^{sa,mg}(\epsilon, \beta, d_s, d_a)$ (where *mg* denotes "multigrid") is within a factor of $1/|\log(\beta)|$ of the lower bound on the complexity of the continuous MDP problem given in equation (5.104) of Theorem 5.12:

Theorem 5.13. *Under assumptions* $(A1), \ldots, (A5)$ *the multigrid algorithm outlined in steps 1 to 3 above is almost optimal, i.e. its complexity,*

$$comp^{sa,mg}(\epsilon, \beta, d_s, d_a) = O\left(\frac{1}{|\log(\beta)|}\left[\frac{1}{((1-\beta)^2)^{2d_s+d_a}}\right]\right), \tag{5.113}$$

is within a factor of $1/|\log(\beta)|$ *of the lower bound on the worst case complexity of the MDP problem.*

This result formalizes the sense in which simple successive approximations is "almost optimal" for solving the MDP problem. The multigrid algorithm can obviously be generalized to allow other deterministic methods for generating grid points, including quadrature grids and low discrepancy grids. Similar to our results for single grid methods, the worst case complexity bounds for multigrid methods using other reasonable deterministic methods for generating grids (e.g. quadrature grids) will be similar to the complexity bound for the multigrid method using uniform grids of Chow and Tsitsiklis.

Rust 1994c presented a "random multigrid algorithm" for DDP's that is basically similar to the deterministic multigrid algorithm of Chow and Tsitsiklis, with the exception that it does not require an exponentially increasing number of grid points in order to obtain accurate approximations to the integrals underlying the random Bellman operator $\tilde{\Gamma}_N$. In particular, at each stage k of the random multigrid algorithm we only need to increase the number of grid points by a factor of 4 as opposed to the factor 2^{d_s} required by the deterministic multigrid algorithm. The following steps lay out the details of the random multigrid algorithm:

1. Beginning with a coarse uniform grid over S containing an arbitrary number of grid points, $N_1 \geq 1$, use the implied random Bellman operator $\tilde{\Gamma}_{N_1}$ to carry out T_1 successive approximation steps starting from an arbitrary initial estimate V_0 of V where T_1 is the smallest number of successive approximation iterations until the following inequality is satisfi ed:

$$\|\tilde{\Gamma}_{N_1}^{T_1}(V_0) - \tilde{\Gamma}_{N_1}^{T_1-1}(V_0)\| \leq \frac{K}{\sqrt{N_1}(1-\beta)}, \tag{5.114}$$

where K is a bounding constant determined by the Lipschitz constants for u and p in assumptions $(A1), \ldots, (A5)$, $K = \gamma(d)|A|K_u K_p$, and $\gamma(d)$ is given in equation (5.60) of Theorem 5.8.

2. At iteration k of the multigrid algorithm generate a new uniform grid with $N_k = 4N_{k-1}$ points. Use the value function $\tilde{V}_k = \tilde{\Gamma}_{N_{k-1}}^{T_{k-1}}(\tilde{V}_{k-1})$ from the final successive approximation step from the previous grid as the starting point for successive approximations using the new grid. [72] Carry out out T_k successive approximations steps starting from $\tilde{V}_k$ where T_k is the smallest number of successive approximations steps until the following inequality is satisfied:

$$\|\tilde{\Gamma}_{N_k}^{T_k}(\tilde{V}_k) - \tilde{\Gamma}_{N_k}^{T_k-1}(\tilde{V}_k)\| \leq \frac{K}{\sqrt{N_k}(1-\beta)}, \tag{5.115}$$

3. Terminate the outer grid generation iterations $k = 1, 2, 3, \ldots$ at the smallest value $\tilde{k}$ satisfying:

$$N_{\tilde{k}} \geq \frac{K^2}{(1-\beta)^4 \epsilon^2} \tag{5.116}$$

where ϵ is a predefined solution tolerance.

Since N_k is increasing by a factor of 4 at each iteration k of the random multigrid algorithm, Theorem 5.8 implies that the expected error is being halved at each iteration, and therefore the random multigrid algorithm terminates with a finite value of $\tilde{k}$ with probability 1. Rust showed that the multigrid approach is faster than the single grid random successive approximations algorithm by a factor of $\log(1/(1-\beta)\epsilon)$, leading to the following upper bound on the randomized complexity of infinite horizon DDP's:

Theorem 5.14. *An upper bound on the worst case complexity of infinite horizon DDP problems is given by:*

$$comp^{\textit{wor-ran}}(\epsilon, \beta, d) = O\left(\frac{|A|^3 d^4 K_u^4 K_p^4}{|\log(\beta)|(1-\beta)^8 \epsilon^4}\right). \tag{5.117}$$

Note that Theorem 5.13 is an upper bound on the randomized complexity of the DDP problem. The lower bound is not yet known. However in view of Bakvalov's result that simple monte carlo integration is an almost optimal algorithm when the dimension d is large relative to the degree of

[72] As per our previous footnote, we actually take the the average of the upper and lower McQueen-Porteus error bounds after T_{k-1} successive approximation steps, see equation (5.6) in section 5.2.1.

smoothness r of the integrand (see Theorem 3.1 in section 3.2), it is reasonable to conjecture that the upper bound on the randomized complexity of the DDP problem given in Theorem 5.13 is not far (i.e. within a factor of $1/|\log(\beta)|$) of the lower bound on the randomized complexity of this problem since $r = 1$ for the class of Lipschitz continuous value functions implied by Definition 5.1.

It is easy to define multigrid algorithms corresponding to the other discrete solution algorithms presented in section 5.2, such as multigrid policy iteration, multigrid linear programming, and so forth. We do not yet have any computational experience with these methods, so it is not clear whether they will outperform multigrid successive approximations.[73] However the near optimality of the successive approximations multigrid algorithm suggests that these other methods may not do much better: at most they might eliminate the remaining factor of $1/|\log(\beta)|$ difference between the complexity of the multigrid successive approximations algorithm and Chow and Tsitsiklis's lower bound on deterministic complexity of MDP's given in Theorem 5.12.

5.4.2 Smooth Approximation Methods

Section 5.3.2 presented a large number of different smooth approximation methods for continuous finite horizon MDP's. We begin this section by observing that all of the methods in section 5.3.2 extend in a straightforward manner to smooth approximation methods for continuous infinite horizon MDP's: one simply uses these methods to compute a finite horizon approximation to the infinite horizon problem with $T(\epsilon, \beta)$ periods, where $T(\epsilon, \beta)$ is the maximum number of successive approximation steps required to ensure that $\|\Gamma^t(V_0) - V\| \leq \epsilon$ for any starting estimate V_0. The drawback of this approach is that it suffers from the slow rate of convergence of the method of successive approximations (discussed in sections 4 and 5.2), which is especially problematic when β is close to 1. In addition there are difficult unsolved problems involved in establishing sufficient conditions for these "approximate successive approximations" methods to converge at all. In this section we focus on linear approximation algorithms of the form:

$$V_\theta(s) = \sum_{i=1}^{k} \theta_i p_i(s), \tag{5.118}$$

[73] This statement is not quite true, since the "adaptive grid generation" approach of Trick and Zin described in section 5.2.3 can be viewed as a multigrid linear programming algorithm for continuous MDP's. Unfortunately their method has not yet been compared to the multigrid successive approximation algorithm.

where $\{p_1, p_2, \dots\}$ are a pre-specified sequence of basis functions such as the Chebyshev polynomials. Theorem 3.8 of section 3.3 showed that approximation algorithms of this form are optimal error algorithms for approximating functions in any class F that is convex and symmetric. The first class of methods we discuss, introduced by Kortum 1992, might be called "parametric successive approximations" since it converts the successive approximations iterations for V into successive approximations iterations for the coefficient vector θ. Next we consider a potentially faster method that we refer to as "parametric policy iteration" that was first suggested by Schweitzer and Seidman 1985: it is the standard policy iteration algorithm except that each policy evaluation step uses a linear approximation $V_{\hat{\theta}_t}$ to the exact value function V_{α_t} implied by the trial decision rule α_t. Both of these approaches use the method of ordinary least squares to compute the best fitting parameter estimates $\hat{\theta}_t$ at each stage t. Thus, these approaches can be viewed as iterative versions of the class of *minimum residual methods* and the closely related class of *projection methods* that are described in much more generality by Judd in chapter 6. The basic idea behind these methods is to find a parameter $\hat{\theta}$ such that the *residual function*

$$R(V_\theta)(s) = V_\theta(s) - \Gamma(V_\theta)(s), \tag{5.119}$$

as close to the 0 function as possible. Finally we review "Euler equation methods" for CDP's in the Euler class of definition 2.4. These methods focus on direct approximation of the optimal decision rule α as a fixed point of the Euler operator Ψ (defined in section 2.6) rather than approximating V as a fixed point of the Bellman operator Γ. Just as in the case of the value function, we consider linear approximations to the decision rule of the form

$$\alpha_\theta(s) = \sum_{i=1}^{k} \theta_i p_i(s), \tag{5.120}$$

and use any one of the approaches described above for approximating the value function including successive approximations, minimum residual, and projection methods to search for a parameter $\hat{\theta}$ that makes the associated *Euler equation residual function* $R(\alpha_{\hat{\theta}})$ as close to the zero function as possible. Finally instead of using linear approximations to α, we can develop methods based on linear approximations to certain conditional expectation functions entering the Euler equation residual function $R(\alpha)$ for α. This class of methods, referred the *parameterized expectations approach* (PEA), was introduced by Marcet 1990.

Parametric Successive Approximations. In continuous MDP's the standard method of successive approximations $V_t = \Gamma^t(V_0)$, $t = 1, 2, \dots$ generates a globally convergent sequence of iterations in the function space $\mathcal{B}(S)$ as described in section 4. We wish to illustrate a number of difficulties

involved in proving the convergence of successive approximations when one is simultaneously trying to approximate the functions $\Gamma^t(V_0)$ by any of the smooth approximation methods outlined in section 5.3.1. We will focus on linear approximations V_θ of the form (5.118) due to their optimality properties noted above and also because it simplifies notation. If we consider approximations V_θ using a fixed number k of basis functions $\{s_1, \ldots, s_k\}$ then unless the true value function V lies within the subspace spanned by $\{p_1, \ldots, p_k\}$ we have $\inf_{\theta \in R^k} \|V_\theta - \Gamma(V_\theta)\| > 0$, which implies that we will be unable to obtain arbitrarily accurate approximations to V. However if the sequence $\{p_1, p_2, \ldots\}$ is ultimately dense in the space B of all possible value functions in the sense of equation (1.2) in the introduction, then inequality (2.22) of Lemma 2.2 implies that we can find a k sufficiently large to make $\inf_{\theta \in R^k} \|V_\theta - \Gamma(V_\theta)\|$ as small as desired. Suppose we choose k large enough to guarantee that $\inf_{\theta \in R^k} \|V_\theta - \Gamma(V_\theta)\| \leq (1-\beta)\epsilon$, where ϵ is some predefined solution tolerance. Then inequality (2.24) of Lemma 2.2, implies that there exists a $k > 0$ and a $\hat{\theta} \in R^k$ such that the estimate $V_{\hat{\theta}}$ is uniformly within ϵ of the true solution V. The problem now is to describe a feasible algorithm that that is capable of finding the required k and$\hat{\theta}$ for any pre-specified solution tolerance ϵ. Consider first the problem of finding$\hat{\theta}$ when k is fixed at some arbitrary value.

Kortum 1992 suggested the method of successive approximations in θ-space. Starting from an arbitrary initial guess $\hat{\theta}_0$ (corresponding to an initial approximate value function, $V_{\hat{\theta}_0}$), we form updated estimates $\{\hat{\theta}_t\}$ according to the formula:

$$\hat{\theta}_t = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}_t, \tag{5.121}$$

where the $N \times 1$ vector $\boldsymbol{y}_t = (\boldsymbol{y}_t(1), \ldots, \boldsymbol{y}_t(N))'$, and the $N \times K$ matrix $\boldsymbol{X} = \{\boldsymbol{X}(i,j)\}$ are defined by:

$$\begin{aligned} \boldsymbol{y}_t(i) &= \hat{\Gamma}(V_{\hat{\theta}_{t-1}})(s_i) \\ \boldsymbol{X}(i,j) &= p_j(s_i), \end{aligned} \tag{5.122}$$

where $i = 1, \ldots, N$, $j = 1, \ldots, k$, $\hat{\Gamma}$ is an approximation to the Bellman operator Γ induced by the need to numerically approximate the maximization and integration subproblems, and $\{s_1, \ldots, s_N\}$ is a grid of $N \geq k$ points in S. The interpretation of equation (5.121) is that $\hat{\theta}_t$ is the ordinary least squares estimate that minimizes the sum of squared errors of the residual function $V_\theta - \hat{\Gamma}(V_{\hat{\theta}_{t-1}})$:

$$\hat{\theta}_t = \underset{\theta \in R^k}{argmin} \sum_{i=1}^{N} \left| V_\theta(s_i) - \hat{\Gamma}(V_{\hat{\theta}_{t-1}})(s_i) \right|^2. \tag{5.123}$$

It is not essential that ordinary least squares be used to compute $\hat{\theta}_t$: any linear or nonlinear approximation method that is capable of yielding arbitrarily accurate approximations to functions

of the form $\Gamma(W)$ will do as well. Any of the methods discussed in section 5.3.1 could be used: for example $\hat{\theta}_t$ could be calculated by the Chebyshev interpolation formula (5.80) using a grid $\{s_1, \ldots, s_N\}$ consisting of the first $N = k$ Chebyshev zeros, or $\hat{\theta}_t$ could be calculated by the Galerkin method in equation (5.83), or by any of the broader class of projection methods which we will discuss in more detail below. Finally θ_t could be calculated by least squares but using a nonlinear-in-parameters functional form for V_θ such as the neural network approximation given in equation (5.84) of section 5.3.2.

A more important question is when to terminate the successive approximations iterations. Kortum 1992 suggested termination when successive changes in the parameter vector is less than some predefined solution tolerance, $\|\hat{\theta}_t - \hat{\theta}_{t-1}\| \leq \epsilon$. The triangle inequality implies that

$$\|V_{\hat{\theta}_t} - V_{\hat{\theta}_{t-1}}\| \leq \|\hat{\theta}_t - \hat{\theta}_{t-1}\| \sum_{i=1}^{k} \|p_i\|, \tag{5.124}$$

where $\|\hat{\theta}_t - \hat{\theta}_{t-1}\|$ denotes the maximum change in k coefficients from iteration $t-1$ to t. Thus, if the $\hat{\theta}_t$ coefficients change by a sufficiently small amount the corresponding value functions $V_{\hat{\theta}_t}$ will also change by only a small amount. However it is not clear that this is the relevant notion for convergence since we are interested in finding $\hat{\theta}_t$ so that the maximum absolute value of the residual function $\|V_{\hat{\theta}_t} - \Gamma(V_{\hat{\theta}_t})\|$ is as close to zero as possible. A more sensible criterion would be to stop would be to stop when $\|V_{\hat{\theta}_t} - \hat{\Gamma}(V_{\hat{\theta}_t})\|$ is as small as possible. Unfortunately, the mapping defining the successive approximation iterates $\{\hat{\theta}_t\}$ is not guaranteed to be a contraction so there is no assurance that $\|V_{\hat{\theta}_t} - \Gamma(V_{\hat{\theta}_t})\|$ is a monotonically declining sequence as is the case when the method of successive approximations is performed on the value functions directly. In addition, if the number of basis functions k is not sufficiently large, it may not be possible to find $\hat{\theta}$ satisfying $\|V_{\hat{\theta}} - V\| \leq \epsilon$. In the absence of computable error bounds for $\|V_{\hat{\theta}_t} - V\|$ that could serve as a termination criterion for this method, we have to settle for the weaker assurance that this difference will converge to zero provided that the number of successive approximation steps t the number of basis functions k and the number of grid points N tend to infinity at the right rates. One can imagine adjusting these parameters iteratively in a multi-level algorithm akin to the multigrid algorithm described in section 5.4.1. Letting $i = 1, 2, 3, \ldots$ denote the outer level iteration for this algorithm, we could imagine a multilevel algorithm that starts with small values for k_1 and N_1 (defining relatively coarse approximations to V and the integrals defining Γ the L_2 norm for R_θ) and takes T_1 successive approximation steps using equations (5.121) and (5.122) resulting in a coefficient vector $\hat{\theta}_{T_1}$. Then at iteration $i = 2$ we set larger values for k_2 and N_2 and

use $\hat{\theta}_0^2 = (\hat{\theta}_{T_1}^1, \mathbf{0}')$ as the starting value for T_2 successive approximations steps in the larger $k_2 \times 1$ coeffi cient vectors$\hat{\theta}_t^2$ where $\mathbf{0}$ denotes a $(k_2 - k_1) \times 1$ vector of zeros. We continue this way until the smallest value $\hat{i}$ such that $\|V_{\hat{\theta}_{T_{\hat{i}}}} - \Gamma(V_{\hat{\theta}_{T_{\hat{i}}}})\| \leq (1-\beta)\epsilon$. Given the error bounds in Lemma 2.2, it follows that this algorithm terminates with an estimate of the value function that satisfi es $\|V_{\hat{\theta}_{T_{\hat{i}}}} - V\| \leq \epsilon$.

Unfortunately we are not aware of any formal proofs of the convergence of this algorithm and we are unaware of numerical applications of the method. To date most numerical applications of the successive approximations approach to solving continuous infi nite horizon MDP's have not attempted to generate continuous approximations $\hat{V}_{\hat{\theta}_t}$ at each step of the backward induction process. Instead, most of the applications of which we are aware (including Christiano 1990 and Tauchen 1990) have adopted the discrete approximation approach, carrying out all of the successive approximation steps on a discretized fi nite state MDP and using multilinear interpolation or some other approximation method to generate a smooth extension $\hat{V}$ defi ned over all of S from "data" consisting of an $N \times 1$ approximate fi xed point$\hat{V}_N$ to an approximate Bellman operator defi ned over a fi nite grid $\{s_1, \ldots, s_N\}$.

Parametric Policy Iteration. Schweitzer and Seidman 1985 and Kortum 1922 independently proposed a parametric version of the Bellman-Howard policy iteration algorithm described in section 5.2.1. Their method differs from the standard implementation of policy iteration in its use of a polynomial series approximation V_θ given in equation (5.118) to approximate the value function V_α corresponding to a given stationary policy α. Recall from section 2.4 that V_α is the solution to the linear system $V_\alpha = u_\alpha - \beta M_\alpha V_\alpha$ where M_α is the Markov operator corresponding to the policy α. This linear system must be solved at each "policy evaluation" step of the policy iteration algorithm, and in the case of continuous MDP's it is actually a linear functional equation whose solution can only be approximated. Let $\{s_1, \ldots, s_N\}$ be a grid over the state space S. The parametric policy iteration algorithm works as follows: given a guess of a policy α_t at stage t of the policy iteration algorithm we approximate the true value function V_{α_t} corresponding to the policy α_t by the value function $V_{\hat{\theta}_t}$ where $\hat{\theta}_t$ is given by:

$$\hat{\theta}_t = (\boldsymbol{X}_t' \boldsymbol{X}_t)^{-1} \boldsymbol{X}_t' \boldsymbol{y}_t, \tag{5.125}$$

where the $N \times k$ matrix $\boldsymbol{X}_t$ and the $N \times 1$ vector $\boldsymbol{y}_t$ are defi ned by

$$\begin{aligned} \boldsymbol{y}_t(i) &= u(s_i, \alpha_t(s_i)) \\ \boldsymbol{X}_t(i,j) &= p_j(s_i) - \beta \int p_j(s') p(ds'|s_i, \alpha_t(s_i)). \end{aligned} \tag{5.126}$$

Given $\hat{\theta}_t$ we form an updated policy α_{t+1} defined by:

$$\alpha_{t+1}(s_i) = \underset{a \in A(s_i)}{argmax} \left[u(s_i, a) + \beta \int V_{\hat{\theta}_t}(s') p(ds'|s_i, a) \right], \quad i = 1, \ldots, N. \tag{5.127}$$

We continue iterating on equations (5.126) and (5.127) until $max_{i=1,\ldots,N} |\alpha_{t+1}(s_i) - \alpha_t(s_i)| \leq \epsilon$ in which case we can use $V_{\hat{\theta}_{t+1}}$ as the corresponding estimate of V. Just as in the case of parametric successive approximations, the formula for $\hat{\theta}_t$ in equation (5.125) correspond to the ordinary least squares estimate:

$$\begin{aligned} \hat{\theta}_t &= \underset{(\theta_1,\ldots,\theta_k)}{argmin} \sum_{i=1}^{N} \left| u(s_i, \alpha_t(s_i)) + \beta M_{\alpha_t}(V_\theta)(s_i) - V_\theta \right|^2 \\ &= \underset{(\theta_1,\ldots,\theta_k)}{argmin} \sum_{i=1}^{N} \left| u(s_i, \alpha_t(s_i)) + \beta \sum_{j=1}^{k} \theta_j \int p_j(s') p(ds'|s_i, \alpha_t(s_i)) - \sum_{j=1}^{k} \theta_j p_j(s_i) \right|^2. \end{aligned} \tag{5.128}$$

Similar to the previous case, there is no particular reason to restrict attention to OLS: any of the other linear or nonlinear approximation methods considered in section 5.3.2 could be used as well.

In certain cases the integrals appearing in equations (5.128) and (5.127) can be computed analytically. For example Kortum shows that when the basis functions $\{p_1, \ldots, p_k\}$ are polynomials the integrals are the derivatives of the moment generating function of the transition probability $p(ds'|s, a)$, and in some cases there are analytic expressions for these moment generating functions. If not, then we need to invoke some sort of numerical integration method such as the quadrature or monte carlo integration methods discussed in section 5.3.1. Similar to the parametric successive approximations method described above, the parametric policy iteration algorithm does not converge monotonically to the true solution V, so special care must be taken to prevent cycling. We are not aware of error bounds or formal proofs of the convergence of this method. Just as in the case of parametric successive approximations, a proof of convergence will require specification of the rates at which the number of iterations t the number of basis functions k and the number of grid points N tend to infinity in order to guarantee that the sequence of value functions $\{V_{\hat{\theta}_t}\}$ converges to V. We can imagine multilevel level algorithms that iteratively adjust t, k, and N until a certain convergence tolerance is attained. Although we are not aware of any formal procedures to insure convergence, Schweitzer and Seidman and Kortum report very favorable experience with the method in numerical experiments. Kortum used the method to approximate the value and policy functions for a MDP model of a firm's optimal investment in research and development. He used a

basis consisting of the first $k = 8$ Chebyshev polynomials.[74] Paraphrasing his conclusions in our terminology, he found that:

> "The algorithm often requires less than six iterations before it converges to a tolerance of $.0000001$ for the policy function. Running on an IBM-compatible computer with and 80486 processor, the algorithm requires 5-60 seconds to converge. The algorithm was successful in matching the closed form solutions for $V(s)$ and $\alpha(s)$, even in cases where V was not a polynomial." (p. 14)

Parametric Linear Programming. In view of the LP characterization of the value function to an infinite horizon MDP given in equation (5.24), Schweitzer and Seidman 1985 proposed approximating V by $V_{\hat{\theta}}$ where $\hat{\theta}$ is the solution to the following linear programming problem over a grid $\{s_1, \ldots, s_N\}$ of points in S:

$$\begin{aligned} \hat{\theta} &= \underset{\theta \in R^k}{argmin} \sum_{i=1}^{k} \theta_i \sum_{j=1}^{N} p_i(s_j) \\ u(s_j, a) &\leq \sum_{i=1}^{k} \theta_i \left[p_i(s_j) - \beta \int p_i(s') p(ds'|s_j, a) \right], \end{aligned} \tag{5.129}$$

where the constraints hold for each $a \in A(s_j)$ for for each grid point s_i, $i = 1, \ldots, N$. Schweitzer and Seidman showed that this problem has a feasible solution, and that the objective function (5.129) is bounded below, which implies that solutions to both the primal and dual problems exist and are equal. In this case solution of the dual to (5.129) is computationally preferable since it will have only k constraints regardless of the number N of points in the state space grid or then number of possible (discrete) actions in the $A(s_j)$. We are not aware of any applications of this approach, although it does bear many similarities to the linear programming approach implemented by Trick and Zin in section 5.2.3. Indeed, we described Trick and Zin's method as applying to the class of discrete infinite horizon MDP's, but it was actually used to solve a continuous infinite horizon MDP problem, the Brock-Mirman stochastic growth model of example 2 of section 2.6. The main difference between Schweitzer and Seidman's approach and Trick and Zin's approach is that the latter is based on discrete approximation methods rather than the smooth linear approximations of the form (5.118).

---

[74] Kortum did not report the number N of grid points used in his regressions for $\hat{\theta}_t$ and his procedure for choosing $\{s_1, \ldots, s_N\}$.

Minimum Residual Methods. These methods are based on the equivalent formulation of the Bellman equation as determining V as the unique zero to the residual function, i.e. the nonlinear operator $R(V) = [I - \Gamma](V) = 0$. Given a parametric approximation V_θ, these methods approximate V by $V_{\hat{\theta}}$ where $\hat{\theta}$ is chosen to make $R(V_\theta)$ as close to zero as possible in the L_p norm:

$$\hat{\theta} = \underset{\theta \in R^k}{argmin} \, \|R(V_\theta)\|_p \equiv \left[\int |V_\theta(s) - \Gamma(V_\theta)(s)|^p \, \mu(ds) \right]^{\frac{1}{p}} \quad p \in [1, \infty]. \tag{5.130}$$

The function μ entering (5.130) represents a probability measure reflecting the analyst's view of the relative importance of deviations in the residual function at various values of s. For this reason, the MR method is also known as the method of minimum *weighted* residuals (MWR). The important point is that MR methods convert the infinite-dimensional fixed problem into a finite-dimensional minimization problem.

In theory we are interested in minimizing the sup-norm of $R(V_\theta)$ which corresponds to the choice $p = \infty$. However in practice the sup-norm is difficult to work with since it leads to objective function that is generally non-differentiable in θ. Therefore standard practice is to use the L_2 norm. It is important to note that in general the L_2 and L_∞ norms generate different topologies on the space $\mathcal{B}(S)$. In particular the fact that $\hat{V}$ is close to V in the L_2 norm does not necessarily imply the $\hat{V}$ is close in the L_∞ form. However if V is known to live in certain compact subspaces B of $\mathcal{B}(S)$ (such as the Sobolev space of all functions whose r^{th} derivative have uniformly bounded L_p norm), then the Rellich-Kondrachev Theorem (Adams, 1975, Theorem 6.2) provides sufficient conditions for the L_p norm to be equivalent to the L_2 norm, $p \geq 2$, i.e. conditions under which there exists a constant $c_p > 0$ such that

$$\|\hat{V} - V\|_p \leq c_p \|\hat{V} - V\|_2, \quad p \geq 2, \quad V, \hat{V} \in B. \tag{5.131}$$

We will henceforth assume that these sufficient conditions hold for the particular problem of interest and work with the L_2 norm. The maximization operator in the definition of Γ also leads to a non-differentiable objective function. This can be solved by using the smoothed Bellman operator Γ_σ defined in equation (4.1) of section 4.[75] The smoothness of the resulting objective function enables one to use much more efficient gradient hill-climbing algorithms to compute $\hat{\theta}$. Finally, let $\hat{\Gamma}_\sigma$ denote an approximate version smoothed Bellman operator that employs some numerical integration method to approximately calculate the integrals entering Γ_σ. Numerical

[75] In CDP's, the summation term over $a \in A(s)$ inside the $\log$ term in equation (4.1) must be replaced by an integral over the continuum of possible actions in $A(s)$.

integration must also be used to approximate the integrals defi ning the L_2 norm. Nearly all of the standard numerical integration algorithms can be represented as a weighted average of the integrand over a grid of points $\{s_1, \ldots, s_N\}$ in the domain S using weights $\{w_1, \ldots, w_N\}$. For example Gaussian quadrature has this representation, and monte carlo integration has this representation for a randomly chosen grid $\{\tilde{s}_1, \ldots, \tilde{s}_N\}$ and uniform weights $w_i = 1/N$, $i = 1, \ldots, N$. Using this representation for the L_2 norm in (5.130) leads to the defi nition of the MR estimate of$\hat{\theta}$:

$$\hat{\theta} = \underset{\theta \in R^k}{argmin} \sum_{i=1}^{N} \left| V_\theta(s_i) - \hat{\Gamma}_\sigma(V_\theta)(s_i) \right|^2 w_i. \tag{5.132}$$

One obtains a large number of variations of the basic MR method depending on how one specifi es the basis functions $\{p_1, \ldots, p_k\}$ determining the approximation method for V_θ and the grid points $\{s_1, \ldots, s_N\}$ and weights $\{w_1, \ldots, w_N\}$ determining the numerical integration used to approximate integrals entering Γ_σ and the L_2 norm for the objective function in (5.130) (we assume that N is the same in both cases only for notational simplicity). Assuming that we have chosen procedures that can generate arbitrarily accurate estimates for suffi ciently large values of k and N, there are 3 basic parameters governing the ability of the MR estimate of the value function to approximate the true solution V: k and N and the value of the smoothing parameter σ. By letting σ tend to zero and k and N tend to infi nity at the right rates, it is possible to show that $V_{\hat{\theta}}$ converges to V. Since we are not aware of a general proof of the convergence of the MR estimate of the value function in the literature, we state the following convergence theorem (omitting precise statements of various technical regularity conditions) and provide a sketch of its proof.

Theorem 5.15. *Assume that the basis $\{p_1, p_2, \ldots\}$ is such that linear approximations V_θ of the form (5.130) can provide arbitrarily accurate uniform approximations to functions $V \in B \subset \mathcal{B}(S)$ where B is the Sobolev space of functions whose r^{th} derivative have uniformly bounded sup-norms. Assume that the integration weights $\{w_1, \ldots, w_N\}$ and grid points $\{s_1, \ldots, s_N\}$ are chosen so that $\hat{\Gamma}_\sigma \to \Gamma_\sigma$ as $N \to \infty$ uniformly for all $V \in B$ satisfying $|V| \leq K/(1-\beta)$ where $K = \sup_{s \in S} \sup_{a \in A(s)} |u(s,a)|$. Finally, assume that as $N \to \infty$ that*

$$\sum_{i=1}^{N} \left| V(s_i) - \hat{\Gamma}_\sigma(V)(s_i) \right|^2 w_i \to \|V - \Gamma_\sigma(V)\|_2, \tag{5.133}$$

uniformly for $V \in B$ satisfying $|V| \leq K/(1-\beta)$. Then under certain regularity conditions including regularity conditions $(A1), \ldots, (A5)$ of Definition 5.1, for each $\epsilon > 0$ there exist k

and N sufficiently large and σ sufficiently small such that if $\hat{\theta}$ is the solution to (5.132) we have $\|V_{\hat{\theta}} - V\| \leq \epsilon$.

Proof (sketch). First, we show that it is possible to choose $\hat{\theta}$ so that $\|V_{\hat{\theta}} - \hat{\Gamma}_\sigma(V_{\hat{\theta}})\| \leq (1-\beta)\epsilon/2$. We do this by choosing N sufficiently large so that the numerical approximation to the L_2 norm in (5.132) is within $(1-\beta)\epsilon/4c_\infty$ of its limit $\|V_\theta - \hat{\Gamma}_\sigma\|_2$, uniformly for θ in a compact set where c_∞ is the constant from inequality (5.131) bounding the sup norm in terms of the L_2 norm. The triangle inequality implies that if $\hat{\theta}$ is chosen so that (5.132) is less than $(1-\beta)\epsilon/4c_\infty$ then the true L_2 norm of the residual function $\|V_{\hat{\theta}} - \hat{\Gamma}_\sigma(V_{\hat{\theta}})\|_2$ will be less than $(1-\beta)\epsilon/2c_\infty$. Inequality (5.131) then implies the desired inequality in terms of the sup-norm.

Now we show that by choosing a sufficiently small smoothing parameter σ and a sufficiently precise numerical integration method for $\hat{\Gamma}_\sigma$ we can guarantee that $\|V_{\hat{\theta}} - V\| \leq \epsilon$. Let $\hat{V}_\sigma$ denote the exact fixed point of the approximate smoothed Bellman operator $\hat{\Gamma}_\sigma$. Using $\hat{V}_\sigma$, a second application of the triangle inequality, and the error bounds in Lemmas 2.1 and 2.2, we obtain the following bound on the maximum error between the MR estimate $V_{\hat{\theta}}$ and the true fixed point V:

$$\begin{aligned} \|V_{\hat{\theta}} - V\| &\leq \|V_{\hat{\theta}} - \hat{V}_\sigma\| + \|\hat{V}_\sigma - V\| \\ &\leq \frac{\|V_{\hat{\theta}} - \hat{\Gamma}_\sigma(V_{\hat{\theta}})\|}{(1-\beta)} + \frac{\|\hat{\Gamma}_\sigma(V) - \Gamma(V)\|}{(1-\beta)}. \end{aligned} \tag{5.134}$$

By choosing a sufficiently small smoothing parameter σ and a sufficiently large number of points N in the numerical integrals entering $\hat{\Gamma}_\sigma$ we can guarantee that $\|\hat{\Gamma}_\sigma(V) - \Gamma(V)\| \leq (1-\beta)\epsilon/2$ uniformly for all V satisfying $\|V\| \leq K/(1-\beta)$ just as we did for the approximate Bellman operators in section 5.3.1. Combining this inequality and the inequality $\|V_{\hat{\theta}} - \hat{\Gamma}_\sigma(V_{\hat{\theta}})\| \leq (1-\beta)\epsilon/2$ derived above, inequality (5.134) implies that $\|V_{\hat{\theta}} - V\| \leq \epsilon$.

Theorem 5.15 is unsatisfactory in the sense that it does not provide a constructive algorithm for finding values of k, N and σ guaranteeing that $\|V_{\hat{\theta}} - V\| \leq \epsilon$. We are not aware of any applications of the MR approach to approximate value functions to continuous infinite horizon MDP's. However, there have been numerous applications of the MR approach to approximate the optimal decision rule α for CDP's in the Euler class of definition 2.4. We will describe these applications in our discussion of Euler equation methods below.

Projection Methods. We introduced projection methods in our discussion of Chebyshev approximation in in section 5.3.2. Our discussion in this section will be brief since Judd 1994 (chapter 6) discusses projection methods in detail, showing how they apply to many different numerical

problems in addition to MDP's. The basic idea of projection methods is that the $k \times 1$ parameter vector $\hat{\theta}$ should be chosen to set set the projections of the residual function $R(V_{\hat{\theta}})$ against a set of k "projection directions" $\{\phi_1, \ldots, \phi_k\}$ equal to zero:

$$< R(V_{\hat{\theta}}), \phi_i >_\mu \equiv \int \left[V_{\hat{\theta}}(s) - \hat{\Gamma}_\sigma(V_{\hat{\theta}})(s) \right] \phi_i(s) \mu(ds) = 0, \quad i = 1, \ldots, k, \tag{5.135}$$

where μ is a probability measure defi ning the weighted inner product $< f, g >_\mu$. Equation (5.135) is a nonlinear system of k equations in the k unknowns θ, and we assume that at least one solution $\hat{\theta}$ exists. Since the true residual function $R(V)(s) = V(s) - \Gamma(V)(s)$ is identically zero, its projection on any set projection directions $\{\phi_1, \ldots, \phi_k\}$ will be zero. Therefore it seems reasonable to assume that if $V_{\hat{\theta}}$ is a good approximation to V and if $\hat{\Gamma}_\sigma$ is a good approximation to Γ, then (5.135) should have a solution $\hat{\theta}$.

Judd 1994 (chapter 6) describes many different possible choices for the projection directions: here we only summarize the most prominent choices. The Galerkin method is a special case when $\phi_i = p_i$ where p_i are the basis functions for the linear approximation to V_θ in equation (5.118). The interpretation of the Galerkin method is that since the residual function $R(V_{\hat{\theta}})$ represents the "error" in the approximation $V_{\hat{\theta}}$, we can make the error small by forcing it to be orthogonal to each of the fi rst k basis functions. It is easy to see that the MR method can be viewed as a special case of a projection method with projection directions given by

$$\phi_i(s) = \left[\partial V_\theta / \partial \theta(s) - \hat{\Gamma}'_\sigma(V_\theta)(\partial V_\theta / \partial \theta)(s) \right] \Big|_{\theta = \hat{\theta}} . \tag{5.136}$$

A fi nal variant of projection method is the *collocation method* which is a special case where μ has its support on a fi nite grid $\{s_1, \ldots, s_k\}$ and the projection directions are the indicator functions for these points, i.e. $\phi_i(s) = I\{s = s_i\}$ where $I\{s = s_i\} = 1$ if $s = s_i$, and 0 otherwise. Thus, the collocation method amounts to choosing a grid of k points in S, $\{s_1, \ldots, s_k\}$, and choosing $\hat{\theta}$ so that the residual function is zero at these k points:

$$R(V_{\hat{\theta}})(s_i) = 0, \quad i = 1, \ldots, k. \tag{5.137}$$

In addition to the four types of grids described in section 5.3.1, another choice for $\{s_1, \ldots, s_k\}$ are the Chebyshev zeros, or more generally the zeros of the basis functions $\{p_1, \ldots, p_k\}$. Judd refers to the resulting method as *orthogonal collocation* and provides the following rationale for choosing this particular grid in the special case where the p_i are the Chebyshev polynomials:

> "As long as $R(V_\theta)(s)$ is smooth in s, the Chebyshev Interpolation Theorem says that these zero conditions force $R(V_\theta)(s)$ to be close to zero for all $s \in S$, and that these are the best possible points to use if we are to force $R(V_\theta)$ to be close to the zero function. Even after absorbing these considerations, it is not certain that even orthogonal collocation is a reliable method; fortunately, its performance turns out to be surprisingly good." (p. 42)

The main conceptual problems with projection methods are: 1) determining what to do if the system (5.135) doesn't have a solution, 2) determining which solution $\hat{\theta}$ to choose if the system (5.135) has more than 1 solution, 3) deriving bounds on the difference between the approximate and exact solution $\|V_{\hat{\theta}} - V\|$. We are not aware of error bounds or even proofs that $V_{\hat{\theta}} \to V$ as $k \to \infty$ and $\hat{\Gamma}_\sigma \to \Gamma$, at least for the case of MDP's. To date we are not aware of any applications of projection methods to approximating the value function V as a fixed point to the Bellman operator Γ. Krasnosel'skii *et. al.* 1972 and Ziedler 1993 present proofs of the convergence of projection methods for certain classes of nonlinear operators. It seems likely that these approaches could be modified to establish convergence proofs in the case of MDP's. However there have been many applications of projection methods to approximating the optimal decision rule α as a fixed point to the Euler operator Ψ as we show in the next section.

Euler Equation Methods. These are specialized methods for CDP's in the Euler class of definition 2.4. The basic idea is to approximate the optimal decision rule α directly from the solution to the Euler equation rather than indirectly from an approximate solution $\hat{V}$ to Bellman's equation and the identity (2.8) determining α from V. A general form of the Euler equation was derived in equation (2.48) of section 2.6. MDP's in the Euler class have optimal decision rules α that can be characterized in two mathematically equivalent ways: as a fixed point to the *Euler operator* $\alpha = \Psi(\alpha)$ where Ψ is implicitly defined from equation (2.48), or as a zero to the *Euler equation*, $R(\alpha)(s) = 0$, where $R(\alpha)$ is the residual function defined by equation (2.48). This section reviews solution methods based on both characterizations of α, including successive approximation methods, and MR and projection methods. These latter methods use parametric linear approximations α_θ given in equation (5.120) and seek a parameter vector $\hat{\theta} \in R^k$ such that the Euler equation residual $\hat{R}(\alpha_{\hat{\theta}})$ is as close to the zero function as possible, where $\hat{R}$ denotes some computable approximation to the residual function. A final class of methods, known as the parameterized expectations approach (PEA), parametrize a conditional expectation function entering $R(\alpha)$ rather than the decision rule α. Similar to MR and projection methods, PEA methods seek a parameter $\hat{\theta}$ that sets the Euler equation residual function as close as possible to the zero function. This results in an indirect parametric approximation of the decision rule $\alpha_{\hat{\theta}}$.

A great deal of research effort has been expended on Euler equation methods over the past 10 years. There are several reasons why economists are especially interested in this class of problems. First MDP's in the Euler class occur frequently in economic theory and macroeconomic applications (see, e.g. Stokey and Lucas, 1989). Second, Euler equation methods form the basis for a popular econometric estimation and testing methodology developed by Hansen and Singleton 1982 which allows one to test the empirical validity of a CDP model by assessing whether a quadratic form of the Euler equation residuals is statistically significantly different from zero.[76] The final reason for the interest in Euler equation methods is the fact that they have performed well in numerical comparisons of alternative methods for solving the Brock-Mirman stochastic growth model in the 1990 *Journal of Business and Economic Statistics* (see Taylor and Uhlig 1990 for an introduction to the special issue). Due to the fact that the Brock-Mirman growth model has been a focal point for most the the numerical comparisons of Euler equation methods, the rest of this section will describe how Euler equation methods are used to solve the Brock Mirman model (see equation (2.53) of example 2 of section 2.6) since the notation required to present the methods for problems in the general Euler class in definition 2.4 becomes rather cumbersome. We will see that it is straightforward to generalize the methods presented for the Brock Mirman special case to arbitrary problems in the Euler class.

We begin by considering successive approximations methods that approximate α as a fixed point to Ψ, an approach pioneered by Coleman 1990 and Hussey 1989. The operator Ψ is defined pointwise at each (k,z) as the value $\Psi(\alpha)(k,z)$ that sets the Euler equation residual identically equal to zero:

$$0 = u'(\Psi(\alpha)(k,z)) - \beta \int u'\left(\alpha\left(f(k,z)+k-\Psi(\alpha)(k,z),z'\right)\right)\left[1+\frac{\partial f}{\partial k}(f(k,z)+k-\Psi(\alpha)(k,z),z')\right] q(dz'|z) \tag{5.138}$$

If we were able to evaluate (5.138) exactly for all (k,z), we could carry out a function space version of successive approximations:

$$\alpha_t = \Psi(\alpha_{t-1}) = \Psi^t(\alpha_0). \tag{5.139}$$

Although Ψ is not a contraction mapping, Coleman 1991 provided sufficient conditions for successive approximations to converge to a unique fixed point of Ψ. This result motivates the use of "approximate successive approximation" iterations of the form

$$\hat{\alpha}_t = \hat{\Psi}(\hat{\alpha}_{t-1}) = \hat{\Psi}^t(\hat{\alpha}_0), \tag{5.140}$$

[76] Perhaps the main reason why these Euler equation tests are so popular is that they can be implemented without completely solving the CDP problem: see Hansen and Singleton, 1982.

where $\hat{\Psi}$ denotes a computable approximation to Ψ, and $\hat{\alpha}_t$ denotes the corresponding approximate decision rule after t approximate successive approximation steps. Coleman and Hussey discretized the state space and used bilinear interpolation to approximate $\hat{\alpha}_t(k,z)$ at points of the grid. Hussey used an N point quadrature grid approach discussed in section 5.3.1 yielding an approximate Euler operator $\hat{\Psi}_N$ similar to the way Tauchen 1990 formed the approximate Bellman operator $\hat{\Gamma}_N$ using a quadrature grid. Specifically, let $\{z_1,\ldots,z_{N_1}\}$ denote the quadrature abscissa for the exogenous state variable z and $q_{N_1}(z_j|z_i)$ denote the corresponding Markov chain defined over these grid points via formula (5.52) (Coleman used a uniform grid over z instead of quadrature abscissa). Let $\{k_1,\ldots,k_{N_2}\}$ denote a grid over the endogenous state variable k. This results in a total of $N = N_1N_2$ points in the grid for this problem. Given this grid we can approximate the successive approximations step $\alpha_t = \Psi(\alpha_{t-1})$ by the corresponding approximate successive approximations step $\hat{\alpha}_t = \hat{\Psi}_N(\hat{\alpha}_{t-1})$ defined by the pointwise solution to:

$$\begin{aligned} 0 &= u'(\hat{\alpha}_t(k_i,z_j)) \\ &\quad - \beta \sum_{l=1}^{N_1} u'(\hat{\alpha}_{t-1}\left(f(k_i,z_j)+k_i-\hat{\alpha}_t(k_i,z_j),z_l\right)) \left[1+\frac{\partial f}{\partial k}(f(k_i,z_j)+k_i-\hat{\alpha}_t(k_i,z_j),z_l)\right] q_{N_1}(z_l|z_j). \end{aligned} \tag{5.141}$$

Note that equation (5.141) only defines $\hat{\alpha}_t$ at the N grid points (k_i,z_j), $i=1,\ldots,N_2$, $j=1,\ldots,N_1$. We must use Newton's method or some other nonlinear equation solver to compute each of the N solutions $\hat{\alpha}_t(k_i,z_j)$. Note that in the process of searching for a value $\hat{\alpha}_t(k_i,z_j)$ that approximately solves equation (5.141) we will have to evaluate the previous decision rule $\hat{\alpha}_{t-1}$ at various points $k' = f(k_i,z_j)+k_i-a_m$, where a_m denotes a trial value for $\hat{\alpha}_t(k_i,z_j)$ generated at iteration m of Newton's method. The points k' will generally not lie on the preassigned capital grid $\{k_1,\ldots,k_{N_2}\}$ so at each step t of the successive approximations iterations we need to employ some sort of function approximation procedure to yield an estimate $\hat{\alpha}_t$ that is defined at all possible values of k. Hussey and Coleman used simple linear interpolation, although this creates kinks that can make it difficult for Newton's method to find a solution $\hat{\alpha}_t(k_i,z_j)$. In terms of our earlier notation, the Euler operator $\hat{\Psi}_N$ is not self-approximating in the k variable. However $\hat{\Psi}_N$ is self-approximating in the z variable: i.e. since the quadrature formula guarantees that $q_{N_1}(z_l|z)$ is a continuous function of its conditioning variable, one simply solves equation (5.141) to generate an estimate of $\hat{\alpha}_t(k_i,z)$ at any point z off of the preassigned grid. Despite these numerical problems, the approximate decision rule $\hat{\alpha}$ (computed via successive approximations with a tolerance of $\epsilon = .0001$) was quite close to the decision rules calculated by Tauchen 1990 and Christiano 1990 using successive approximations of the Bellman operator to compute an approximate value function (see tables 6 and 10 in Taylor and Uhlig, 1990). It is difficult to compare the relative cpu-times of

successive approximations based on the Euler versus the Bellman operator since Coleman used a 38 megaflop Amdahl mini-supercomputer rated at 38 million instructions per second and Tauchen used a 386 PC rated at .25 megaflops. Making a rough adjustment for the relative speeds of these computers, Coleman's Euler equation algorithm would require approximately 17,000 cpu-seconds on a 386 PC whereas Tauchen's Bellman equation algorithm took approximately 2,800 cpu-seconds. The extra cpu consumption of the Euler equation method is probably due to the computational burden of using Newton's method to solve equation (5.141) at the N points on the grid in order to evaluate the approximate Euler operator $\hat{\Psi}_N(\alpha)$ whereas only simple algebraic operations are required in order to to evaluate the approximate Bellman operator $\hat{\Gamma}_N(V)$.

The main advantage of performing successive approximations using the Euler operator as opposed to the Bellman operator in this context is that the former method produces an approximate decision rule $\hat{\alpha}$ whose Euler equation residual $\hat{R}(\hat{\alpha})$ is generally closer to zero. This should not be surprising in view of the fact that this is the objective of Euler equation methods, whereas the objective of value function methods is to make the value function residual $\hat{R}(\hat{V})$ close to zero. However it is somewhat surprising to find that even though successive approximations using the Bellman operator yields fairly accurate approximations to α, the method can yield relatively inaccurate implied approximations to the Euler equation residuals $R(\hat{\alpha})$. For example Tauchen 1990 concluded that

> "Even though the algorithm approximates the decision rule closely, it still might appear to do poorly on criteria that test for statistical violations of orthogonality conditions implied by the Euler equation. The reason is that a value-function approach does not explicitly impose the Euler equation on the discrete model. Thus, for a fixed discretization, tests of orthogonality conditions will be rejected with probability tending to 1 as the length of the simulated realization increases." (p. 49)

This conclusion is confirmed by the results in table 11 of Taylor and Uhlig 1990 which compares the "accuracy statistic" suggested by den Haan and Marcet 1994 for various solution methods. The accuracy statistic is a χ^2 test of the hypothesis that the Euler equation residual function is identically zero, where the χ^2 statistic is computed from simulated realizations of $\{k_t, z_t\}$ drawn from the exact distribution of technology shocks $q(z_{t+1}|z_t)$ and the approximate optimal decision rule $\alpha_{\hat{\theta}}$. Taylor and Uhlig presented accuracy statistics for 10 different versions of the Brock-Mirman model for various parameter values for the discount factor β, etc. The statistics (which are distributed as $\chi^2(11)$ with a 97.5% critical value of 22 under the null hypothesis that $R(\hat{\alpha}) = 0$) range from 150 to 580 for Tauchen's approximations to α compared to 11 to 24 for Coleman's approximations to α. This finding suggests that if one is more interested in the Euler

equation residual rather than the value function residual as the appropriate metric of the accuracy of an approximate solution, one should consider using Euler equation methods.[77] However it turns out that there are much faster methods than successive approximations for approximating $\hat{\alpha}$ via the Euler equation. We describe these methods now.

Judd 1992 was the first to employ projection methods to compute approximate decision rules from the Euler equation. Recall that projection methods use linear approximations to the decision rule α_θ of the general form given in equation (5.120). Since the Brock-Mirman model has a two-dimensional state variable, Judd used tensor products of the Chebyshev polynomials to create his basis functions

$$\begin{aligned} \alpha_\theta(k,z) &= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\theta_{ij}p_i(k)p_j(z) \\ p_i(k) &= \cos\left(i\cos^{-1}\left(2\frac{k-\underline{k}}{\overline{k}-\underline{k}}-1\right)\right) \\ p_j(z) &= \cos\left(i\cos^{-1}\left(2\frac{z-\underline{z}}{\overline{z}-\underline{z}}-1\right)\right), \end{aligned} \tag{5.142}$$

where $\overline{k}$, $\underline{k}$, $\overline{z}$ and $\underline{z}$ are pre-specified bounds on k and z, representing a truncation of the original unbounded state MDP problem to an approximate problem over a compact state space S. He worked with a transformed, "linearized" version of the Euler equation given by

$$\begin{aligned} 0 &= \alpha_\theta(k,z) \\ &\quad -(u')^{-1}\left(\beta\int u'(\alpha_\theta\left(f(k,z)+k-\alpha_\theta(k,z),z'\right)\left[1+\frac{\partial f}{\partial k}(f(k,z)+k-\alpha_\theta(k,z),z')\right]q(dz'|z)\right). \end{aligned} \tag{5.143}$$

The rationale for this transformation is that "the $(u')^{-1}$ operation in (5.143) will unwrap some of the nonlinearity arising from the u' operation inside the integral, hopefully leaving us with a more linear problem." (p. 25). Judd then exploited the particular details of the transition probability $q(z'|z)$, specifically that $\log(z') = \rho\log(z)+\epsilon$ where $\epsilon \sim N(0,\sigma^2)$, to develop an efficient Gauss-Hermite quadrature rule for approximating the numerical integral in (5.143). This results in the

[77] Note the Euler equation methods do not necessarily lead to small Euler equation residuals over the entire state space, and the precise details of the way these methods are implemented might lead to approximate decision rules with larger Euler equation residuals than methods that derive $\hat{\alpha}$ from an approximate value function $\hat{V}$. For example, the Euler equation approach of Baxter *et. al.* 1990 solves the Euler equation over a relatively coarse grid and restricts $\alpha(k,z)$ to have values on this grid. The χ^2 statistics for the Euler equation residuals for this method were generally higher than those for Tauchen's value function based method.

following expression for the approximate residual function $\hat{R}(\alpha_\theta)$:

$$\hat{R}(\alpha_\theta)(k,z) = \alpha_\theta(k,z) - (u')^{-1}\left(\beta \sum_{i=1}^{N} u'(h_\theta(z,k,z_i))w_i\right)$$
$$h_\theta(k,z,z_i) \equiv \alpha_\theta\left(f(k,z)+k-\alpha_\theta(k,z), z^\rho e^{\sigma z_i}\right)\left)\left[1+\frac{\partial f}{\partial k}\left(f(k,z)+k-\alpha_\theta(k,z), z^\rho e^{\sigma z_i}\right)\right]. \tag{5.144}$$

where $\{z_1,\ldots,z_N\}$ and $\{w_1,\ldots,w_N\}$ are the Gauss-Hermite quadrature abscissa and weights, respectively.[78] With the approximate residual function $\hat{R}(\alpha_\theta)$ defined, it is then straightforward to apply the projection methods described above. Judd used a Galerkin method where the projection directions $\{\phi_{ij}\}$ were set equal to the tensor products of the basis functions in equation (5.142) and Gauss-Chebyshev quadrature with N_1 points for k and N_2 points for z were used to approximate the integrals defining the inner products (see equation (5.135)). Judd also used the orthogonal collocation method described above, choosing $\hat{\theta}$ so that $\hat{R}(\alpha_{\hat{\theta}})(k_i,z_j)=0$, $i=1,\ldots,m_1$, $j=1,\ldots,m_2$, where the grid points $\{k_i,z_j\}$ are the zeros to the Chebyshev polynomials defined in equation (5.142). Judd computed numerical solutions for various values of m_1 and m_2 ranging from 2 to 10, and an $N=8$ Gauss-Hermite quadrature rule for the numerical integral in equation (5.144). The numerical performance of Judd's projection method is impressive: the method is extremely fast and highly accurate, where accuracy was measured both in terms of the magnitude of the approximation error $\|\alpha_{\hat{\theta}}-\alpha\|$ (where α is an analytic solution to the special case of 100% depreciation in equation (2.55) of section 2.6), or in terms of the sup-norm of the normalized residual function, $\|\hat{R}(\alpha_{\hat{\theta}})/\alpha_{\hat{\theta}}\|$ which expresses the magnitude of the approximation error as a fraction of current consumption. Judd concluded that:

> "First, note that the errors are rather small. Even for the $m_1=2$, $m_2=2$ case, the errors are roughly one dollar per hundred. Second, as we allow the approximation to use more terms the errors fall until in the $m_1=10$, $m_2=6$ case, we often find optimization errors of less than one dollar per million. Third, the various norms of the residual function have very similar values, indicating that the errors are uniformly small. In particular, the similarity in values for the norms of the [average and maximum absolute deviations] indicates that the solutions is almost as good at the edges of the state space as in the middle. Fourth these methods are fast. The solutions in the $m_1=2$ $m_2=2$ case were solved in .2 to .4 cpu seconds [on a Compaq 386 PC running at 20 megahertz] and in the $m_1=4$ $m_2=3$ case in 1.1 to 2 seconds. The speed advantage of using orthogonal collocation is demonstrated by the fact that the $m_1=7$ $m_2=5$ cases generally took 8 to 18 seconds, whereas the [corresponding Galerkin method using $N_1=20$ and $N_2=12$ quadrature points to evaluate the integrals defining the

[78] Actually, due to a change of variables involved in applying the Gauss-Hermite integration rule, the weights need to be multiplied by $1/\sqrt{\pi}$ and the abscissa need to be multiplied by $\sqrt{2}$.

> inner products] took three times as long, which is expected since the projections were integrals using 240 points instead of 35. Fifth, the orthogonal collocation method does remarkably well given the small amount of computation. This is indicated by the small optimization errors reported and the fact that by going to Galerkin procedures which use many more points, very little is any accuracy is gained." (p. 30–31)

Judd's findings have been confirmed in independent work by Ueda 1994. She also found that Judd's measure of numerical accuracy, the sup-norm of the normalized residual function $\|\hat{R}(\alpha_{\hat{\theta}})/\alpha_{\hat{\theta}}\|$, provides a much better gauge of the true accuracy in the numerical solution than den-Haan and Marcet's χ^2 test of the residual function.

Parameterized Expectations Approach. This approach was introduced by Marcet 1988 and was used to approximate the decision rule to the Brock-Mirman model by den Haan and Marcet 1990. It is similar to the "parametric successive approximations" method described above in the sense that it yields a sequence of successive approximations for the parameter vector $\{\hat{\theta}_i\}$ that are designed to converge to a fixed point θ^*. However instead of parametrizing the value function or the policy function, the PEA method makes a (log) linear parameterization of the conditional expectation function $\mathcal{E}(k,z)$ entering the Euler equation. In the case of the Brock Mirman model, $\mathcal{E}(k,z)$ is given by:

$$\mathcal{E}(k,z) = \beta \int u'\left(\alpha(f(k,z)+k-\alpha(k,z),z')\right)\left[1+\frac{\partial f}{\partial k}\left(f(k,z)+k-\alpha(k,z),z'\right)\right] q(dz'|z). \tag{5.145}$$

Since it is straightforward to show that $\mathcal{E}(k,z) > 0$, den Haan and Marcet used the following log-linear approximation to this function

$$\mathcal{E}_\theta(k,z) = \exp\left\{\sum_{i=1}^{m} \theta_i p_i(k,z)\right\}, \tag{5.146}$$

where $\{p_1,\ldots,p_m\}$ are basis functions such as polynomials, Chebyshev polynomials, etc. The parameterization of the conditional expectation implies a corresponding parameterization for α by enforcing the constraint that the Euler equation residual is identically zero:

$$\alpha_\theta(k,z) = (u')^{-1}(\mathcal{E}_\theta(k,z)). \tag{5.147}$$

Given the parameterizations for $\alpha(k,z)$ and $\mathcal{E}(k,z)$ the PEA method works as follows:

1. Generate a $T+1$-period simulation of the exogenous technology shock process $\{\tilde{z}_0,\ldots,\tilde{z}_T\}$. This series is drawn only once and remains fixed as $\hat{\theta}_i$ is updated.

2. Starting with an arbitrary initial guess for $\hat{\theta}_0$ at iteration $i = 0$, use $\{\tilde{z}_0, \ldots, \tilde{z}_T\}$ and the estimate $\hat{\theta}_i$ to derive simulated realizations for consumption and capital $\{\tilde{c}_t(\hat{\theta}_i), \tilde{k}_t(\hat{\theta}_i)\}$ implied by $\hat{\theta}_i$ using the parametric decision rule $c_t = \alpha_{\hat{\theta}_i}(k_t, z_t)$ from equation (5.147) and the law of motion for the capital stock, $k_{t+1} = f(k_t, z_t) + k_t - c_t$.

3. Compute an updated estimate $\hat{\theta}_{i+1}$ by nonlinear least squares using the simulated realization $\{\tilde{c}_t(\hat{\theta}_i), \tilde{k}_t(\hat{\theta}_i)\}$ as data:

$$\hat{\theta}_{i+1} = \underset{\theta \in R^k}{argmin} \sum_{t=0}^{T-1} \left| u'(\tilde{c}_{t+1}(\hat{\theta}_i)) \left[1 + \frac{\partial f}{\partial k}(\tilde{k}_{t+1}(\hat{\theta}_i), \tilde{z}_{t+1})\right] - \mathcal{E}_\theta(\tilde{k}_t(\hat{\theta}_i), \tilde{z}_t) \right|^2. \quad (5.148)$$

4. If $\|\hat{\theta}_{i+1} - \hat{\theta}_i\| \leq \epsilon$, stop, otherwise increase i by 1 and return to step 2.

The idea behind the PEA method is that if $\mathcal{E}_\theta(k, z)$ were correctly parameterized, i.e. if there is a θ^* such that $\mathcal{E}_{\theta^*}(k, z)$ equals the true conditional expectation $\mathcal{E}(k, z)$, then the decision rule α_{θ^*} will coincide with the true optimal decision rule α and the simulations $\{\tilde{c}_t(\theta^*), \tilde{k}_t(\theta^*)\}$ will be realizations from the true controlled stochastic process. Since $\mathcal{E}(k, z)$ is a conditional expectation, it follows that the nonlinear least squares estimate from equation (5.148) will converge to the best fi tting parameter estimates as $T \to \infty$. However the best fi tting parameter estimate is θ^* since the defi nition of conditional expectation implies that θ^* is the fi xed point to the least squares problem

$$\theta^* = H(\theta^*) \equiv \underset{\theta \in R^k}{argmin}\, E\left\{ \left| \mathcal{E}_\theta(k, z) - u'(\tilde{c}_{t+1}(\theta^*)) \left[1 + \frac{\partial f}{\partial k}(\tilde{k}_{t+1}(\theta^*), \tilde{z}_{t+1})\right] \right|^2 \right\}. \quad (5.149)$$

This heuristic explanation suggests that a suffi ciently large number of basis functions m will ensure that there exists θ^* such that $\mathcal{E}_{\theta^*}(k, z)$ is a very good approximation to the true conditional expectation function $\mathcal{E}(k, z)$. Therefore with a suffi ciently large number of simulations T, the sequence of successive approximations $\{\hat{\theta}_i\}$ should converge to θ^*, which implies that the resulting approximate decision rule α_{θ^*} will be close to the true optimal decision rule α. Proving this turns out to be quite diffi cult since one must impose suffi cient regularity conditions to guarantee that the nonstationary simulated process $\{\tilde{c}_t(\hat{\theta}_i), \tilde{k}_t(\hat{\theta}_i)\}$ has suffi cient ergodicity to guarantee that least squares estimates $\hat{\theta}_i$ will converge to well defi ned probability limits. A recent paper by Marshall and Marcet provides suffi cient conditions for the convergence of the PEA method, but their suffi cient conditions appear quite diffi cult to verify in practical problems. Indeed it is sometimes diffi cult to guarantee that the successive approximations $\{\hat{\theta}_i\}$ will converge, especially in cases where the time series simulations for $\{\tilde{c}_t(\theta), \tilde{k}_t(\theta)\}$ become explosively nonstationary for certain values of θ.

Nevertheless the numerical experience with the PEA in a wide range of problems has been very encouraging. Using relatively few parameters ($m = 4$) and the ordinary polynomials as basis functions, den Haan and Marcet were able to compute approximate decision rules $\alpha_{\hat{\theta}}$ that were comparable in accuracy to approximations produced by successive approximations of the Bellman operator using the quadrature grid approach of Tauchen 1990 or the uniform grid of Christiano 1990. However the PEA method is not as fast as the projection method of Judd described above. Judd's 1992 results show the "Our four parameter approximation was always 200-500 times faster than the three parameter approximation procedure in den Haan and Marcet." (p. 37). Judd also criticized the PEA method's reliance on repeated monte carlo simulations to generate the simulated controlled process $\{\tilde{c}_t(\theta), \tilde{k}_t(\theta)\}$ for various trial values of θ:

> "This simulation approach has many undesirable properties relative to standard quadrature methods. First, to compensate for the low accuracy of Monte Carlo methods, long simulations must be run. den Haan and Marcet report taking about three minutes on a Compaq 386/25 to a solve for a solution with three free parameters. Second, the points at which the errors are calculated depend on the guess for θ. This endogeneity combined with the simulation aspects means that parameterized expectations evaluates the residual only at states which are frequently visited. This makes it inappropriate for many problems. For example, parameterized expectations cannot be applied to dynamic games since off-equilibrium paths behavior is critical in the definition of subgame perfect equilibria." (p. 36–37).

Since optimal decision rules to MDP's are subgame-perfect equilibria to games against nature, Judd's comment suggests that the PEA may not be an accurate solution method for MDP's. However Marshall and Marcet 1994 respond to these criticisms by claiming that

> "In fact, endogenous oversampling is an important feature of many algorithms in numerical analysis, and is likely to be an essential element in any algorithm for solving models with many state variables. By eliminating the integral computations involved in the calculation of the residual, the PEA further reduces the computation time, since standard quadrature integration is often very costly even in two dimensional problems. The use of Monte-Carlo integration means that the integrals can be calculated even in models with a large number of stochastic shocks." (p. 16–17).

Christiano and Fisher 1994 have made important progress towards resolving this debate in their independent numerical comparison of PEA and projection methods for the Brock-Mirman model. They also showed how to apply these methods to approximate optimal decision rules to a version of the Brock Mirman model with *irreversible investment* which imposes the additional constraint that gross investment is non-negative: $k_{t+1} \geq k_t$. One can derive a version of the

Euler equation that handles this additional constraint by introducing a state-dependent Lagrange multiplier $\lambda(k, z)$ resulting in an augmented Euler equation of the form

$$R(\alpha)(k, z) - \lambda(k, z) = 0, \tag{5.150}$$

where $R(\alpha)$ is the residual function to the unconstrained, reversible investment version of the Brock-Mirman model given by the right hand side of equation (5.138). The functions α and λ must also satisfy the Kuhn-Tucker complimentary slackness conditions:

$$\begin{aligned} f(k, z) &\geq \alpha(k, z) \\ 0 &= \lambda(k, z) \left[f(k, z) - \alpha(k, z) \right]. \end{aligned} \tag{5.151}$$

The addition of the extra constraint complicates the approximate solution of the problem since there are now essentially two functions that need to be parameterized, the decision rule α and the Lagrange multiplier λ, and these functions must satisfy the additional constraints in equation (5.151).

Christiano and Fisher compared six alternative solution methods for the reversible and irreversible investment versions of the Brock Mirman model. Since neither version of the model they consider admits a closed-form solution, they use the approximate solution to an optimal decision rule $\hat{\alpha}$ derived from an approximate value function $\hat{V}$ computed by successive approximations of an approximate Bellman operator defi ned over a very fi ne discretization of the capital stock variable k_t (the technology shock variable z_t is assumed to to be a two state Markov chain in their version of the Brock-Mirman model). The six approximation methods considered are

1. ***Conventional PEA.*** This is the method originally proposed by Marcet described above with only minor modifi cations (e.g. the expectation function $\mathcal{E}_\theta(k, z)$ is parameterized as the exponential of a linear combination of the orthogonal Legendre polynomials rather than the ordinary polynomials used by den Haan and Marcet 1990).

2. ***PEA with Exogenous Oversampling.*** In order to respond to Judd's criticism of the endogenous oversampling features of the conventional PEA, Christiano and Fisher included a method that oversamples infrequently visited regions of the state space. They did this by drawing J independent realizations of the exogenous technology shock process $\{\tilde{z}_0^j, \ldots, \tilde{z}_T^j\}$, $j = 1, \ldots, J$, which remain fi xed throughout the subsequent iterative process of searching for a fi xed point in θ-space. For each trial value $\hat{\theta}_i$ in the successive approximations iterations, they generated J corresponding simulations of $\{\tilde{k}_t^j(\hat{\theta}_i), \tilde{c}_t^j(\hat{\theta}_i)\}$, $t = 0, \ldots, T$ from a grid of initial capital stocks $\{k_0^1, \ldots, k_0^J)\}$ that are well dispersed over the range of possible values

of k. The successive approximations iterations for $\hat{\theta}_i$ are then computed by nonlinear least squares updating formula:

$$\hat{\theta}_{i+1} = \underset{\theta \in R^k}{argmin} \sum_{j=1}^{J} \sum_{t=1}^{T} \left| u'(\tilde{c}_t^j(\hat{\theta}_i) \left[1 + \frac{\partial f}{\partial k}(\tilde{k}_t^j(\hat{\theta}_i), \tilde{z}_t^j) \right] - \mathcal{E}_\theta(\tilde{k}_{t-1}^j(\hat{\theta}_i), \tilde{z}_{t-1}^j) \right|^2 . \quad (5.152)$$

which are terminated when the iterations satisfy $\|\hat{\theta}_{i+1} - \hat{\theta}_i\| \leq \epsilon$.

3. ***PEA-Collocation.*** This approach, introduced by Christiano and Fisher, represents an alternative successive approximations procedure for fi nding an approximate fi xed point $\theta^* = H(\theta^*)$ to the mapping H defi ned in equation (5.149). The advantage of their method is that it eliminates the time-consuming process of repeated long time series simulations of the controlled process $\{\tilde{k}_t(\hat{\theta}_i), \tilde{c}_t(\hat{\theta}_i)\}$ each time one computes an updated estimate of $\hat{\theta}_{i+1}$ by nonlinear least squares. This method can be regarded as an iterative form of a projection/collocation method that generates an updated estimate $\hat{\theta}_{i+1}$ from the current estimate $\hat{\theta}_i$ as the solution to the following nonlinear system of m equations in the m unknown parameters θ:

$$0 = \mathcal{E}_\theta(k,z) - \beta \int u' \left(\alpha_{\hat{\theta}_i}(f(k,z) + k - \alpha_{\hat{\theta}_i}(k,z), z') \right) \left[1 + \frac{\partial f}{\partial k} \left(f(k,z) + k - \alpha_{\hat{\theta}_i}(k,z), z' \right) \right] q(dz'|z), \quad (5.153)$$

where $\alpha_{\hat{\theta}_i}$ is the parametric decision rule implied by $\mathcal{E}_{\hat{\theta}_i}$ given in equation (5.147). In general the value of m needed to obtain a good estimate of $\hat{\theta}_{i+1}$ using Christiano and Fisher's projection method is much smaller than the length of the time-series simulations T needed to reliably estimate $\hat{\theta}_{i+1}$ using Marcet's nonlinear least squares approach, especially when there are multicollinearity problems arising from the use of the standard polynomial basis for $\mathcal{E}_\theta$. Christiano and Fisher used the orthogonal Chebyshev polynomials for the basis functions for the parameterization of $\mathcal{E}_\theta$ in equation (5.146). Using the discrete orthogonality property of the Chebyshev polynomials (see equation (5.77) of section 5.3.2) and the fact that there are only two possible values for z, they obtained the following explicit representation for $\hat{\theta}_{i+1}$

$$\hat{\theta}_{i+1}(z) = \frac{1}{m} \sum_{j=1}^{m} p_j(k_j) \log \left[h_{\hat{\theta}_i}(k_j, z, z_1) q(z_1|z) + h_{\hat{\theta}_i}(k_j, z, z_2) q(z_2|z) \right]$$

$$h_\theta(k,z,z') \equiv \beta u' \left(\alpha_\theta(f(k,z) + k - \alpha_\theta(k,z), z') \right) \left[1 + \frac{\partial f}{\partial k} \left(f(k,z) + k - \alpha_\theta(k,z), z' \right) \right]. \quad (5.154)$$

4. ***Galerkin-Chebyshev.*** This is the same projection method used by Judd (1992) described above, modified to use only a one-dimensional Chebyshev approximation for $\alpha_\theta(k,z)$ rather than the two-dimensional tensor product representation given in equation (5.142) due to the fact that z takes on only two possible values. Thus, two separate $m \times 1$ coefficient vectors $\{\theta(z_1), \theta(z_2)\}$ (corresponding to the two possible values of z) are computed from two $m \times 1$ sets of projection equations:

$$0 =< \hat{R}(\alpha_{\theta(z)}), p_i >, \quad i = 1, \ldots . m. \quad z = z_1, z_2. \tag{5.155}$$

   The integrals in equation (5.155) are approximated by Gauss-Legendre quadrature.

5. ***Galerkin-Spline.*** This approach, advocated by McGrattan 1993, uses a basis for α_θ consisting of spline functions with knots at points $\{k_1, \ldots, k_m\}$ rather than the first m Chebyshev polynomials. As in the case of the Galerkin-Chebyshev method, two separate $m \times 1$ coefficient vectors $\theta(z_i)$ are computed, so that the approximation α_θ takes the form

$$\alpha_\theta(k,z) = \sum_{i=1}^{m} \theta_i(z)\phi_i(k). \tag{5.156}$$

   The spline basis $\{\phi_1, \ldots, \phi_m\}$ consists of "tent functions" defined by

$$\phi_i(k) = \begin{cases} \dfrac{k - k_{i-1}}{k_i - k_{i-1}} & k_{i-1} \le k \le k_i \\ \dfrac{k_{i+1} - k}{k_{i+1} - k_i} & k_i \le k \le k_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{5.157}$$

   The integrals defining the inner products in equation (5.155) were approximated by N-point Gauss-Legendre quadrature. Otherwise the method is the same as the Galerkin-Chebyshev method described above.

6. ***Approximate Successive Approximations of the Euler Operator.*** This method is essentially the same as the Coleman-Hussey approach described in equations (5.140) and (5.141) above.

Christiano and Fisher described modifications to these six approximation methods to handle the Kuhn-Tucker conditions arising from the irreversible investment constraints in equation (5.151). Accomodating these constraints is a relatively straightforward matter in the PEA framework. In PEA, one approximates the conditional expectation function $\mathcal{E}(k,z)$ the same as in the unrestricted

model. However in deriving the corresponding parametric approximation to α one imposes the irreversible investment constraint:

$$\alpha_\theta(k,z) = max\left[k, (u')^{-1}\left(\mathcal{E}_\theta(k,z)\right)\right], \tag{5.158}$$

where the corresponding approximation to the multiplier function λ is given by:

$$\lambda_\theta(k,z) = u'(\alpha_\theta(k,z)) - \mathcal{E}_\theta(k,z). \tag{5.159}$$

It is easy to see that this parameterization for α and λ automatically satisfies the Kuhn-Tucker constraints (5.151).

Accomodating the irreversible investment constraint is more involved in the other frameworks. Christiano and Fisher exploit the special feature of the two state specification of the technology shock process by assuming that 1) the irreversible investment constraint will never be binding in the good state z_2 (i.e. the state with the higher value of z), and 2) in the bad state z_1 if the irreversible investment constraint becomes binding some threshold level $\overline{k}$ it will remain binding at higher levels of k as well. These assumptions (which were subsequently verified to hold at the numerically computed solution) imply that $\lambda(k,z_2) = 0$ so that there are only three functions that define the solution to the MDP problem, the lagrange multiplier function in the bad state $\lambda(k,z_1)$ and the consumption decision rules in the good and bad states $\alpha(k,z)$, $z \in \{z_1, z_2\}$. Christiano and Fisher made the following parametric approximations to $\alpha(k,z_1)$ and $\lambda(k,z_1)$ to ensure that the Kuhn-Tucker conditions in equation (5.151) are enforced:

$$\begin{aligned}
\alpha_\theta(k,z_1) &= \begin{cases} min\left[a_\theta(k,z_1), f(k,z_1)\right] & k < \overline{k} \\ f(k,z_1) & k \geq \overline{k} \end{cases} \\
\lambda_\theta(k,z_1) &= \begin{cases} 0 & k < \overline{k} \\ max\left[l_\theta(k,z_1), 0\right] & k \geq \overline{k} \end{cases}
\end{aligned} \tag{5.160}$$

Thus, the functions that are actually parameterized in equation (5.160) are the unrestricted functions $a_\theta(k,z_1)$ and $l_\theta(k,z_1)$ which are specified to be linear combinations of a set of basis functions. The threshold value $\overline{k}$ is determined from the solution to the equation $f(\overline{k},z_1) = a_\theta(\overline{k},z_1)$. Otherwise the method proceeds exactly like the Galerkin method for the reversible investment version of the Brock Mirman model, with the exception that one uses the augmented Euler residuals in equation (5.150) as the residual function entering the orthogonality conditions (5.155) defining the projection equations. The method of successive approximations can also incorporate the irreversible investment constraints by this approach, using the Euler operator Ψ defined from the

augmented Euler equation (5.150) which is an operator of two arguments, $\Psi(\alpha, \lambda)$. One then uses successive approximations over a grid of capital values to compute an approximate fixed point $(\alpha, \lambda) = \Psi(\alpha, \lambda)$. Christiano and Fisher used a penalty function approach to assure that the Kuhn-Tucker constraints were approximately satisfied in the Galerkin-spline method.

Tables 5.3 and 5.4 reproduce the key results of Christiano and Fisher's numerical comparison of the methods (programmed in Gauss with certain key subroutines imported from Fortran) using a Gateway 486 DX2/66 computer. The results show that the fastest methods that the fastest and most accurate methods are the PEA-collocation and the Galerkin-Chebyshev methods. Their results also confirm our previous conclusion that the method of successive approximations using the approximate Euler operator $\hat{\Psi}$ is dominated by the projection methods.[79] In addition to evaluating accuracy in terms of the sup norm of the Euler equation residual function, they compared the decision rules $\alpha_{\hat{\theta}}$ generated the methods to the "true" decision rule α computed from V, where V was approximated by successive approximations of the Bellman operator Γ using a very fine grid for the capital stock (which took several cpu hours to converge). They found that the PEA-collocation and the Galerkin-Chebyshev decision rules to be virtually indistinguishable from the true decision rule, and therefore from each other.

Method	m, N, T	CPU Seconds	sup norm of Euler Residuals
Conventional PEA	$m = 3$, $T = 10,000$	155.1	1.3×10^{-4}
PEA-Exogenous oversampling	$m = 3$, $T = 10,000$	151.9	1.6×10^{-4}
PEA-Collocation	$m = 3$	0.6	2.1×10^{-6}
Galerkin-Chebyshev	$m = 3$	3.5	4.5×10^{-4}
Galerkin-Chebyshev	$m = 5$	6.3	6.2×10^{-7}
Galerkin-Spline	$N = 18$	8.8	3.7×10^{-4}
Galerkin-Spline	$N = 36$	17.9	7.2×10^{-5}
Successive Approximations of Ψ	$N = 36$	253.1	1.0×10^{-4}
Successive Approximations of Ψ	$N = 72$	557.6	3.3×10^{-5}

Table 5.3 Comparison of Methods for Reversible Investment Version of Brock Mirman Model

[79] The exception is the Galerkin-spline method which is slower in the irreversible investment case. However this is not a result of the projection method *per se* but "reflects the fact that this algorithm involves repeatedly solving the model for higher values of the penalty function parameter." (p. 28).

Method	m, N, T	CPU Seconds	sup norm of Euler Residuals
Conventional PEA	$m = 3, T = 10,000$	174.4	1.7×10^{-4}
PEA-Exogenous oversampling	$m = 3, T = 10,000$	181.7	1.9×10^{-4}
PEA-Collocation	$m = 3$	2.5	9.9×10^{-5}
Galerkin-Chebyshev	$(5, 3, 2)$	17.6	7.3×10^{-4}
Galerkin-Chebyshev	$(9, 5, 4)$	27.9	8.8×10^{-5}
Galerkin-Spline	$N = 18$	212.5	2.4×10^{-2}
Galerkin-Spline	$N = 36$	1783.9	3.7×10^{-4}
Successive Approximations of Ψ	$N = 36$	522.9	1.0×10^{-4}
Successive Approximations of Ψ	$N = 72$	996.2	3.4×10^{-5}

Table 5.4 Comparison of Methods for Irreversible Investment Version of Brock Mirman Model

Although Christiano and Fisher focus on PEA-collocation as the fastest method, they did not compare it to the orthogonal collocation method which Judd 1992 found to be just as accurate and approximately three times faster than the Galerkin-Chebyshev projection method. However the approximately 10-fold speed advantage of the PEA-collocation method over Galerkin-Chebyshev in the irreversible investment version of the Brock-Mirman model suggests that it may still be several times faster than the best projection method. The increased speed advantage in the irreversible investment case is probably largely due to the fact that it is easier to handle constraints in the PEA.[80] Their findings confirm Judd's criticisms of conventional PEA as being less accurate and orders of magnitude slower than projection methods. However the fact that PEA with exogenous oversampling does not significantly speed up the solution to the problem suggests that the main computational burden of both versions of the PEA is the cost of repeatedly generating the $T = 10,000$ time series observations and solving the nonlinear least squares problem. Since conventional PEA is just about as accurate as PEA with exogenous oversampling, Judd's criticisms about the potential inaccuracies created by endogenous oversampling are not relevant in this case. As Christiano and Fisher conclude, the main difficulty is not lack of numerical accuracy but rather lack of numerical efficiency:

[80] The parameters for the Galerkin-Chebyshev method listed in table 5.4 denote the number of basis functions m in the parameterizations of $\alpha_\theta(k, z_1)$, $a_\theta(k, z_2)$ and $l_\theta(k, z_1)$, respectively.

> "While Marcet's PEA seems to be the easiest to implement, we had difficulties with conventional versions of it. A key component of those versions is a cumbersome nonlinear regression step, potentially involving tens of thousands of observations. One reason for the large number of observations is that the explanatory variables are inefficiently concentrated in a narrow range. We devised an alternative (PEA-collocation), in which the regression step is linear, the explanatory variables are orthogonal, and the required number of observations is the regression is very small: no more than sixteen in our experiments. This method produced results as accurate as the best other method, and is orders of magnitude faster. Although it is clear that PEA-collocation is the best solution method for our example, that does not guarantee that it will dominate in higher dimensional cases. Here, there are at least two considerations. First, do the linearity and orthogonality properties of the PEA-collocation survive into multidimensional settings? In Appendix 2 we define multidimensional PEA-collocation and show that these properties do indeed survive in general. The second consideration involves the mapping from a parameterized expectations function to policy and multiplier functions, which is at the heart of any PEA. In our example, this mapping is trivial, but in higher dimensions it involves solving nonlinear equations. In principle, there could be examples in which this is very costly in programmer and/or computer time, in which case perhaps an alternative method might dominate. Here, it should be born in mind that PEA's have been applied routinely in high-dimensional models." (p. 35–36).

While a number of the smooth approximation methods described in this section appear to be effective for approximating solutions to relatively low-dimensional MDP problems (particularly the PEA-collocation and projection methods based on Chebyshev polynomial approximations and numerical quadrature), all of these methods are subject to the curse of dimensionality that makes it virtually impossible to use these methods in high-dimensional problems. In particular, multidimensional approximation methods such as the tensor product versions of Chebyshev polynomial or other forms of approximations require the number of basis functions m to increase exponentially fast in d in order to maintain an ϵ-approximation to the function to be approximated. Many of the methods we have described, including the minimum residual and projection methods, also involve compound forms of the curse of dimensionality since they require solution to minimization problems of dimension m or solution to systems of nonlinear equations of dimension m, but the amount of computer time required to find ϵ-approximations to these solutions increases exponentially fast in m on a worst case basis. Furthermore randomized procedures for solving these approximation and optimization subproblems cannot break the curse of dimensionality as we noted in section 3.

6. Conclusion

This chapter has surveyed an almost overwhelmingly large literature on numerical methods for MDP's that has developed over the last 40 years. Although the field of numerical dynamic programming could probably be described as "mature", there is no sign that research in this area is slowing down. If anything, contributions to this literature seem to have accelerated over the last five years. The combination of improved solution methods and rapid improvements in computer hardware have enabled economists to greatly reduce computation time, improve accuracy, and solve larger and more realistic MDP models. We have focused on "general purpose" methods that are capable of providing arbitrarily accurate approximations to generic MDP problems. Due to the space constraints in this already long chapter, we have not been able to describe a number of more specialized but potentially promising methods that have performed well in specific applications. These methods include the linear-quadratic approximation method of Christiano 1990 and McGrattan 1990, the "backsolving" approach of Sims 1990 and Ingram 1990, the "extended path method" of Fair and Taylor (see Gagnon 1990), and the Lagrange multiplier approach of Chow 1992, 1993a,b, 1994. Space constraints have also forced us to ignore an interesting literature on learning algorithms for approximating solutions to MDP's with incomplete information about (u, p). We refer the reader to Narendra and Wheeler 1986, Barto, Bradtke and Singh, 1993, and Tsitsiklis, 1994 for studies that show how learning algorithms are able to eventually "discover" optimal strategies. Rust 1989b provides references to an older literature on adaptive control.

We have focused most of our attention on continuous MDP's since these problems arise quite frequently in economic applications and methods for discrete MDP's have been adequately covered elsewhere. As we have seen, there is a controversy over the relative efficiency of discrete versus continuous approximation methods for approximating the solutions to continuous MDP's. We have appealed to the theory of computational complexity to provide a framework for analyzing the speed and accuracy of the large number of discrete and smooth approximation methods that have been proposed. Complexity theory allows us to formalize the well known "curse of dimensionality", that has prevented us from approximating solutions to large scale MDP problems. Complexity theory enables us to determine whether the curse of dimensionality is an inherent aspect of the MDP problem, or whether there exist solution methods that are capable of breaking it. Complexity theory has also enabled us to characterize the form of optimal or nearly optimal algorithms for the general continuous-state/continuous-action infinite horizon MDP problem. We have used this theory to argue that smooth approximation methods are subject to an inherent curse of dimensionality inherited from the curse of dimensionality of the underlying problem of multivariate function approximation.

Although discrete approximation methods are also subject to the curse of dimensionality (at least on a worst case basis for MDP's with continuous action spaces known as CDP's), we have shown that a discrete approximation method, the multigrid/successive approximations algorithm of Chow and Tsitsiklis 1991, is a nearly optimal algorithm for the general infi nite horizon continuous MDP problem. We are not aware of any corresponding optimality results for smooth approximation methods, and indeed at present we do not even have convergence proofs or error bounds for most of these methods. For the subclass of MDP's with fi nite action spaces (i.e. DDP's) we have shown that there exist randomized versions of discrete approximation methods that do succeed in breaking the curse of dimensionality. An example of such a method is the random multigrid/successive approximations algorithm of Rust 1994c.

This chapter has dicusssed three numerical subproblems involved in solving continuous MDP problems: 1) the integration subproblem, 2) the approximation subproblem, and 3) the constrained maximization subproblem. We have provided a fairly detailed dicussion of alternative algorithms for solving the integration subproblem since it comprises a large share of the work involved in solving continuous MDP problems, and virtually all the work for subclasses such as DDP's. Complexity theory has shown that randomized algorithms succeed in breaking the curse of dimensionality of the integration subproblem and this is the basic reason why randomization breaks the curse of dimensionality of solving the DDP problem. Recent progress on "quasi monte carlo" or low-discrepancy integration methods suggests that these methods could lead to substantial speed-ups relative to traditional numerical integration methods such as quadrature in moderate to high-dimensional MDP problems. However while several of these deterministic low discrepancy methods have been proven to break the curse of dimensionality of the integration problem on an average case basis, it is presently only a conjecture on our part that these methods also break the curse of dimensionality of the DDP problem on an average case basis.

We have also devoted a fair amount of attention to the approximation problem, since multivariate function approximation is a major part of the computational burden involved in the continuous MDP problem. Unlike integration, complexity theory shows that randomization is incapable of breaking the curse of dimensionality of multivariate function approximation. However this conclusion seems to contradict the previous conclusion that randomization does succeed in breaking the curse of dimensionality of approximating the class of multivariate value functions $V(s) = \Gamma(V)(s)$ for DDP problems. It turns out that there is no contradiction here: the reason that randomization does not break the curse of dimensionality of the general approximation problem discussed in section 3.3 is due to the fact that complexity bound for the general approximation problem is

defined in terms of a much broader class of functions F (e.g. the class of all Lipschitz continuous functions with uniformly bounded Lipschitz norm). Randomization succeeds in breaking the curse dimensionality of approximating value functions since they are elements of a restricted subset $B \subset F$ that can be represented as the maximum of a finite number of linear integral operators. On the other hand, smooth approximation methods based on general multivariate approximation algorithms such as tensor products of Cheybshev polynomialas, splines, neural networks, etc. are designed to approximate the much wider class F and therefore fail to exploit the special structure of the set B of value functions. Thus, our conclusion that smooth approximation methods are subject to an inherent curse of dimensionality irregardless of whether deterministic or random algorithms are used only applies to the class of *standard* smooth approximation methods (i.e. those based on approximation methods that are capable of approximating arbitrary Lipschitz continuous functions in F rather than exploiting the special structure of the class of value functions V). It is entirely possible that a smooth approximation algorithm that exploits the structure of this subset can succeed in breaking the curse of dimensionality of the DDP problem. Indeed, as we noted in section 5, that discrete approximation methods such as the random multigrid algorithm can also be viewed as smooth approximation methods due to the self-approximating property of the random Bellman operator $\tilde{\Gamma}_N$. This implies that certain *nonstandard* smooth approximation methods do succeed in breaking the curse of dimensionality of approximating value functions to certain classes of MDP problems such as DDP's.

We have devoted relatively little space to the maximization subproblem beyond the observation that when there are a continuum of actions, the maximization subproblem is subject to an inherent curse of dimensionality regardless of whether deterministic or random algorithms are used. This result immediately implies that randomization cannot break the curse of dimensionality involved in solving the general CDP problem, so in this sense CDP's are inherently more difficult problems than DDP's. Indeed we view the practical possibilities for solving general CDP's that have complicated multidimensional constraint sets $A(s)$ to be fairly hopeless. Most of the constrained optimization algorithms that we are aware of require a good deal of "babying" to ensure convergence, and there is generally no guarantee that these methods will converge to a global as opposed to local maximum. Given that these constrained maximization problems must be solved hundreds and thousands of times in the course of computing an approximate solution to the MDP problem, it seems reasonably clear that given the current state of the art we must either resort to discretization of the constraint sets and determine the approximate maximizing value $\hat{\alpha}(s)$ by simple enumeration, or we must focus on highly structured problems such as convex optimization

problems where fast globally convergent algorithms exist. Since convex constrained optimization problems can be solved in polynomial time on a worst case basis, it may even be possible to extend the random Bellman opeator approach of Rust 1994c to break the curse of dimensionality of the class of *convex CDP's* on a worst case basis. The other possibility is to focus on CDP's in the Euler class and to adopt the PEA-collocation method described in section 5.4.2. Even though this method is also subject to a curse of dimensionality on a worst case basis, practical experience with the method suggests that it works fairly well provided the problem dimension is not too high and the constraints involved are not too complicated. It is an open theoretical question whether there exist optimization algorithms that succeed in breaking the curse of dimensionality on an average case basis (see, e.g. Wasilkowski, 1994). At present there seems to be little cause for hope for a breakthrough in this area.

The fact that solving general CDP problem of any significant size seems to be a hopeless task may lead one to question whether it is worthwhile to even worry about characterizing optimal algorithms for this problem. Furthermore, even though it may be feasible to use these methods in sufficiently low dimensional problems, some readers might be skeptical of the practical relevance of results such as the near optimality of the Chow-Tsitsiklis 1991 multigrid algorithm since it is based on a seemingly "naive" uniform discretization of the state space. A standard criticism of worst case or minimax optimality results is that they are too conservative and do not reflect the special structure of problems typically encountered in practice. This view implies that we should be more interested in algorithms that perform well for "typical" problems even though they may peform poorly in artificially constructed worst cases. To find such algorithms we need to conduct the complexity analysis on an averge rather than worst case basis, specifying a Bayesian prior formalizing the notion of "typical problem". However Nemirovsky and Yudin 1983 point out some of the problems in specifying a meaningful prior over infinite dimensional spaces such as the space of possible MDP problems:

> "The motivation for our choice of the minimax method for characterizing methods on classes 'according to the worst case' undoubtedly requires some explanation. There is a widespread belief that the 'minimax approach' is too pessimistic; it is considered more sensible to average the characteristics of the methods on individual problems of the class according to some *a priori* distribution. Such a 'Bayesian approach' postulates that 'in life' problems of a given type are distributed in a definite way. We point out, however, that for arbitrary broad classes of problems there is no way, justified in any degree, of giving such an *a priori* distribution over the class of problems. Hopes of an 'experimental determination' of such a distribution are unfounded: if the class of problems is parametric, with 50 parameters say, then any reliable 'direct' construction of their joint distribution would require a selection of fantastic size, which is certainly unrealizable. So, even in the simplest linear problems, an empirical approach to the

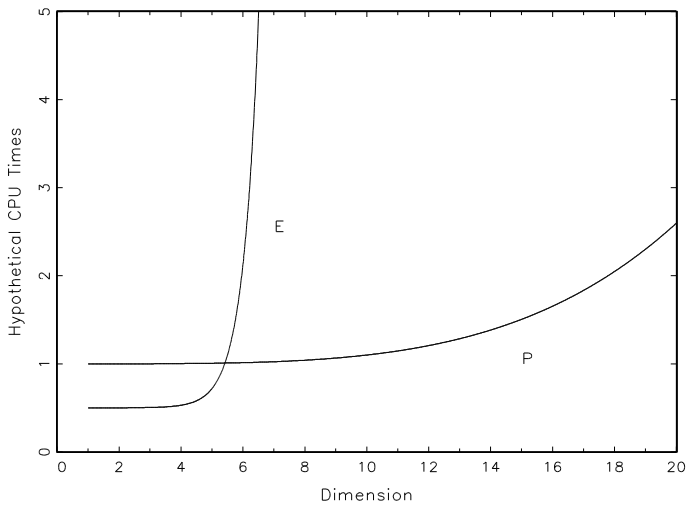

Figure 6.1: Hypothetical Cost Functions for Exponential and Polynomial Time Algorithms

> construction of the *a priori* distribution is hopeless. Thus the 'Bayesian' approach to the study of methods of solving arbitrarily wide classes of problems has no future in practice; the recommended methods would have to work well with an *arbitrary a priori* distribution; but then they would also be good in the minimax sense." (p. 26–27).

While this view certainly has theoretical merit, we have presented several examples in this chapter of less "conservative" algorithms that seem to work well in practice even though they can be proven to work quite poorly on a worst case basis. The classic example is the simplex algorithm of linear programming which in its modified 'constraint generation' form has been successfully used to solve fairly large scale MDP's by Trick and Zin 1993. The other example is the low discrepancy numerical integration approach discussed in section 5.3.1. Even though low discrepancy integration methods are deterministic algorithms the numerical results of Paskov 1994 show that they significantly outperform monte carlo integration in terms of speed and accuracy even in fairly high dimensional problems ($d = 360$). Paskov's work provides practical evidence for our conjecture that a modified version of the random multigrid algorithm that substitutes low discrepancy grid points for random grid points may work well in typical MDP problems even though it is theoretically true that there are worst case problems for which the method does poorly.

While these are useful general results, complexity theory (whether done on a worst or average case basis) can take us only so far. Although the theory tells us a lot about the *the rate of growth* in cpu time as the problem dimension d increases and the permissible error tolerance ϵ decreases, the fact that complexity bounds typically depend on unspecified proportionality constants make it quite difficult to determine the exact value of the complexity bound which we need in order to

characterize the optimal algorithm for any fixed values of d and ϵ. Indeed it is entirely possible that an exponential-time time algorithm E might be faster for use in approximating solutions to low-dimensional MDP's than a polynomial-time algorithm P. This is illustrated in figure 6.1 which plots hypothetical cost functions for algorithm E which increases exponentially fast in d and algorithm P which increases at a rate proportional to the fourth power of d. The figure shows that the exponential-time algorithm is actually twice as fast for solving low-dimensional problems than the polynomial-time algorithm. The polynomial-time algorithm is preferable only for problems of dimension $d \geq 6$ since the exponential-time algorithm "blows up" after this point. It follows that the mere fact that an algorithm is subject to the curse of dimensionality does not necessarily imply that it is a bad method if we are only interested in solving relatively small scale problems. Indeed given that the computational complexity function $comp(\epsilon)$ is the lower envelope of all the algorithm-specific cost curves, it is entirely possible that the exponential-time algorithm E could actually be an optimal algorithm for sufficiently low-dimensional problems. In the case of the DDP problem algorithm E might represent a smooth approximation method such as a Galerkin method that approximates the value function by a linear combination of Chebyshev polynomials and P might represent the random multigrid algorithm.

The phenomenon illustrated in figure 6.1 may be part of the source of the paradoxical findings in section 5.4 that discrete approximation methods such as the method of successive approximations using approximate Euler or Bellman operators defined over a finite grid of points are orders of magnitude slower than smooth approximation methods such as Galerkin-Chebyshev or PEA-collocation. Note also that the discrete approximation methods that have been considered so far are deterministic single grid methods: the theoretical results of section 5.4.1 suggest that the random multigrid or low discrepancy multigrid methods will be orders of magnitude faster than the single methods that have been investigated to date. Unfortunately these approaches have yet to be implemented and compared to existing methods.

We conclude that although complexity theory has suggested a number of useful algorithms, the theory has relatively little to say about important practical issues such as determining the point at which the exponential time algorithm starts to blow up, making it optimal to switch from algorithm E to algorithm P. It seems that the only way to obtain this information is via "hands on" practical experience with the various methods. This kind of experience has been accumlated for other numerical problems. For example in a section entitled "The Present State of the Art" in their book on numerical integration, Davis and Rabinowitz 1984 identified optimal methods for different ranges of d. They concluded that product quadrature rules were the best for low

dimensional problems (i.e. $d \leq 4$) and that for high dimensional problems ($d > 15$) "Sampling or equidistribution methods are indicated. They are time consuming, but some care, are reliable." (p. 417). What we need is a similar statement of the state of the art for MDP's.

In the final analysis there is no substitute for rigorous numerical comparisons of alternative methods such as we have presented in section 5. However we have only presented results for a limited number of methods for two low dimensional test problems — the auto replacment problem (a DDP) and the Brock-Mirman stochastic growth model (a CDP). In future work it will be essential to provide numerical comparisons of a much broader range of methods over a much broader range of test problems including problems of moderate to high dimensionality. The curse of dimensionality has been stumbling block to such a comparison in the past. Perhaps an equally serious problem has been the lack of analytic solutions to multidimensional problems. This has made it difficult to compare the relative accuracy of various numerical methods. Even when an analytic solution exists there are many ways to judge the accuracy of an approximate solution: do we measure it in terms of the error in the value function, the policy function, the Euler equation residual, or in the stochastic properties of the simulated state and control variables? All of these problems are already present in the relatively simple two dimensional Brock-Mirman problem, leading Taylor and Uhlig to conclude that "The differences among the methods turned out to be quite substantial for certain aspects of the growth model. Therefore researchers might want to be careful not to rely blindly on the results of any chosen numerical solution method in applied work." (p. 1). An important direction for future research will be to develop a suite of multidimensional test problems with analytic solutions and a comprehensive set of criteria for measuring the accuracy of solutions along multiple dimensions and to carry out a systematic numerical comparison of the various approaches, including some of the promising new methods suggested in this chapter. At that point we will have a much better picture of the true state of the art in numerical dynamic programming.

7. References

Adams, R.A. (1975) *Sobolev Spaces* Academic Press, New York.

Anselone, P.M. (1971) *Collectively Compact Operator Approximation Theory and Application to Integral Equations* Prentice Hall Series on Automatic Computation, Englewood Cliffs, New Jersey.

Anselone, P.M. and R. Ansorge (1981) "A Unifi ed Framework for the Discretization of Nonlinear Operator Equations" *Numerical Functional Analysis and Optimization* **4-1** 61–99.

Arrow, K.J. (1968) "Applications of Control Theory to Economic Growth" In G.B. Dantzig and A.F. Veinott (eds.) *Mathematics for Decision Sciences* Part 2 American Mathematical Society, Providence.

Arrow, K.J. Harris, T. Marschak, J. (1951) "Optimal Inventory Policy" *Econometrica* **19–3** 250–272.

Barron, A.R. (1993) "Universal Approximation Bounds for Superpositions of a Sigmoidal Function" *IEEE Transactions on Information Theory* **39-3** 930–945.

Barto, A.G. Bradtke, S.J. and S.P. Singh (1993) "Learning to Act using Real-Time Dynamic Programming" manuscript, Department of Computer Science, University of Massachusetts.

Baxter, M. Crucini, M.J. and Rouwenhorst, K.G. (1990) "Solving the Stochastic Growth Model by a Discrete State Space, Euler Equation Approach" *Journal of Business and Economic Statistics* **8-1** 19–21.

Bellman, R. (1955) "Functional Equations in the Theory of Dynamic Programming: Positivity and Quasilinearity" *Proceedings of the National Academy of Sciences* **41** 743–746.

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

Bellman, R. and S. Dreyfus (1962) *Applied Dynamic Programming* Princeton University Press.

Bellman, R. Kalaba, R. and B. Kotkin (1963) "Polynomial Approximation: A New Computational Technique in Dynamic Programming: Allocation Processes" *Mathematics of Computation* **17** 155–161.

Bertsekas, D. (1987) *Dynamic Programming Deterministic and Stochastic Models* Prentice Hall, New York.

Bertsekas, D. (1977) "Monotone Mappings with Application in Dynamic Programming" *SIAM Journal of Control and Optimization* **15-3** 438–464.

Bertsekas, D. (1975) "Convergence of Discretization Procedures in Dynamic Programming" *IEEE Transactions on Automatic Control* **20** 415–419.

Bertsekas, D. Castañon, D. (1989) "Adaptive Aggregation Methods for Infi nite Horizon Dynamic Programming" *IEEE Transactions on Automatic Control* **34-6** 589–598.

Bertsekas, D.P. Shreve, S.E. (1978) *Stochastic Optimal Control: The Discrete Time Case* Academic Press, New York.

Bertsekas, D.P. and Tsitsiklis, J.N. (1993) "Simulated Annealing" *Statistical Science* **8** 1–16.

Bhattacharya, R.N. Majumdar, M. (1989) "Controlled Semi-Markov Models – The Discounted Case" *Journal of Statistical Planning and Inference* **21** 365–381.

Blackwell, D. (1962) "Discrete Dynamic Programming" *Annals of Mathematical Statistics* **33** 719–726.

Blackwell, D. (1965) "Discounted Dynamic Programming" *Annals of Mathematical Statistics* **36** 226–235.

Blackwell, D. (1967) "Positive Dynamic Programming" *Proceedings of the* 5^{th} *Berkeley Symposium on Mathematical Statistics and Probability* **1** 415–418.

Brown, P.N. and Y. Saad (1990) "Hybrid Krylov Methods for Nonlinear Systems of Equations" *SIAM Journal on Scientific and Statistical Computing* **11-3** 450–481.

Brock, W.A. (1982) "Asset Prices in a Production Economy" in J.J. McCall (ed.) *The Economics of Information and Uncertainty* University of Chicago Press.

Brock, W.A. Mirman, L.J. (1972) "Optimal Economic Growth Under Uncertainty: The Discounted Case" *Journal of Economic Theory* **4** 479–513.

Chow, C.S. and Tsitsiklis, J.N. (1989) "The Complexity of Dynamic Programming" *Journal of Complexity* **5** 466–488.

Chow, C.S. and Tsitsiklis, J.N. (1991) "An Optimal Multigrid Algorithm for Continuous State Discrete Time Stochastic Control" *IEEE Transactions on Automatic Control* **36-8** 898–914.

Chow, G.C. (1975) *Analysis and Control of Dynamic Economic Systems* John Wiley, New York.

Chow, G.C. (1979) "Optimum Control of Stochastic Differential Equation Systems" *Journal of Economic Dynamics and Control* 143–175.

Chow, G.C. (1981) *Econometric Analysis by Control Methods* John Wiley, New York.

Chow, G.C. (1992) "Dynamic Optimization without Dynamic Programming" *Economic Modelling* January, 3–9.

Chow, G.C. (1993a) "Optimal Control without Solving the Bellman Equation" *Journal of Economic Dynamics and Control* **17** 621–630.

Chow, G.C. (1993b) "Computation of Optimum Control Functions by Lagrange Multipliers" in D.A. Belsley (ed.) *Computational Techniques for Econometrics and Economic Analysis* 65–72, Kluwer, Dordrecht.

Chow, G.C. (1994) "The Lagrange Method of Optimization in Finance" Research Memorandum 369, Econometric Research Program, Princeton University.

Chew, S.H. Epstein, L.G. (1989) "The Structure of Preferences and Attitudes Towards the Timing and Resolution of Uncertainty" *International Economic Review* **30-1** 103–118.

Chou, A.W. (1987) "On the Optimality of the Krylov Information" *Journal of Complexity* **3** 545–559.

Christiano, L.J. (1990) "Linear-Quadratic Approximation and Value Function Iteration: A Comparison" *Journal of Business and Economic Statistics* **8-1** 99–113.

Christiano, L.J. and J.D.M. Fisher (1994) "Algorithms for Solving Dynamic Models with Occasionally Binding Constraints" manuscript, Department of Economics, University of Western Ontario.

Coleman, W.J. (1990) "Solving the Stochastic Growth Model by Policy-Function Iteration" *Journal of Business and Economic Statistics* **8** 27–29.

Coleman, W. J. (1991) "Equilibrium in a Production Economy With Income Tax" *Econometrica* **59-4** 1091–1104.

Coleman, W.J. (1993) "Solving Nonlinear Dynamic Models on Parallel Computers" *Journal of Business and Economic Statinstics* **11-3** 325–330.

Coleman, W.J. and M. Liu (1994) "Incomplete Markets and Inequality Constraints: Some Theoretical and Computational Issues" manuscript, Fuqua School of Business, Duke University.

Cryer, C.W. (1982) *Numerical Functional Analysis* Clarendon Press, Oxford.

Daniel, J.W. (1976) "Splines and Effi ciency in Dynamic Programming" *Journal of Mathematical Analysis and Applications* **54** 402–407.

Dantzig, G.B. Harvey, R.P. Landowne, Z.F. and McKnight, R.D. (1974) *DYGAM – A Computer System for the Solution of Dynamic Programs* Control Analysis Corporation, Palo Alto, California.

Davis, P.J. Rabinowitz, P. (1975) *Methods of Numerical Integration* Academic Press, New York.

de Boor, C. (1978) *A Practical Guide to Splines* Springer, New York.

den Haan, W. and A. Marcet (1990) "Solving the Stochastic Growth Model by Parameterizing Expectations" *Journal of Business and Economic Statistics* **8-1** 31–34.

Denardo, E.V. (1967) "Contraction Mappings Underlying the Theory of Dynamic Programming" *SIAM Review* **9** 165–177.

Dudley, R.M. (1989) *Real Analysis and Probability* Wadsworth & Brooks/Cole, Pacifi c Groves, California.

Dynkin, E.B. Juskevic, A.A. (1979) *Controlled Markov Processes* Springer-Verlag, New York.

Eckstein, Z, Wolpin, K. (1989) "The Specifi cation and Estimation of Dynamic Stochastic Discrete Choice Models" *Journal of Human Resources* **24-4** 562–598.

Epstein, L.G. Zin, S.E. (1989) "Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: A Theoretical Framework" *Econometrica* **57-4** 937–970.

Epstein, L.G. Zin, S.E. (1990) "'First-Order Risk Aversion and the Equity Premium Puzzle" *Journal of Monetary Economics* **26** 387–407.

Evtushenko, Yurij G. (1985) *Numerical Optimization Techniques* Optimization Software Division, Springer-Verlag, New York.

Fleming, W.H. and H. Mete Soner (1993) *Controlled Markov Processes and Viscosity Solutions* Springer Verlag, New York.

Gagnon, J.E. (1990) "Solving the Stochastic Growth Model by Deterministic Extended Path" *Journal of Business and Economic Statistics* **8-1** 35–36.

Garey, M.R. Johnson, D.S. (1983) *Computers and Intractibility: A Guide to the Theory of NP-completeness* W.H. Freeman, New York.

Geman, S. and C. Hwang (1986) "Diffusions for Global Optimization" *SIAM Journal on Control and Optimization* **24-5** 1031–1043.

Geweke, J. Slonim, R. and Zarkin, G. (1992) "Econometric Solution Methods for Dynamic Discrete Choice Problems" manuscript, University of Minnesota.

Gihman, I.I. Skorohod, A.V. (1974, 1975, 1979) *The Theory of Stochastic Processes* Volumes I, II, III. Springer-Verlag, New York.

Goldfarb, D. and M.J. Todd (1989) "Linear Programming" in A.H.G. Rinooy Kan and M.J. Todd (eds.) *Handbooks in Operations Research Volume One: Optimization* North Holland.

Gihman, I.I. Skorohod, A.V. (1979) *Controlled Stochastic Processes* Springer-Verlag, New York.

Goffe, W.L. Ferrier, G.D. and J. Rogers (1992) "Global Optimization of Statistical Functions with Simulated Annealing" manuscript, University of North Carolina at Wilmington.

Hackbusch, W. (1985) *Multi-Grid Methods and Applications* Springer-Verlag, New York.

Hakansson, N. (1970) "Optimal Investment and Consumption Strategies Under Risk for a Class of Utility Functions" *Econometrica* **38** 587–607.

Hammersley, J.J. and D.C. Handscomb (1992) *Monte Carlo Methods* Chapman and Hall, London.

Hansen, L.P. Sargent, T.J. (1980a) "Formulating and Estimating Dynamic Linear Rational Expectations Models" *Journal of Economic Dynamics and Control* **2–1** 7–46.

Hansen, L.P. Sargent, T.J. (1980b) "Linear Rational Expectations Models for Dynamically Interrelated Variables" in R.E. Lucas, Jr. and T.J. Sargent (eds.) *Rational Expectations and Econometric Practice* Minneapolis, University of Minnesota Press.

Hansen, L.P. Sargent, T.J. (1994) *Dynamic Models of Linear Economies* manuscript, Hoover Institution.

Hansen, L.P. Singleton, K. (1982) "Generalized Instrumental Variables Estimation of Nonlinear Rational Expectations Models" *Econometrica* **50** 1269–1281.

Hansen, L.P. Singleton, K. (1983) "Stochastic Consumption, Risk Aversion, and the Temporal Behavior of Asset Returns" *Journal of Political Economy* **91-2** 249–265.

Hawkins, D. and H.A. Simon (1949) "Note: Some conditions for Macroeconomic Stability" *Econometrica* **17** 245–248.

Hinderer, K. (1970) *Foundations of Nonstationary Dynamic Programming with Discrete Time Parameter* Springer-Verlag, Berlin.

Hornik, K.M. Stinchcombe, M. and H. White (1990) "Universal Approximation of An Unknown Function and Its Derivatives Using Multilayer Feedforward Networks" *Neural Networks* **3** 551–560.

Hornik, K. M. Stinchcombe, H. White, and P. Auer (1993) "Degree of Approximation Results for Feedforward Networks Approximating Unknown Mappings and Their Derivatives" manuscript, Department of Economics, University of California-San Diego.

Howard, R. (1960) *Dynamic Programming and Markov Processes* J. Wiley, New York.

Howard, R. (1971) *Dynamic Probabilistic Systems: Volume 2 — Semi-Markov and Decision Processes* J. Wiley, New York.

Hwang, C. (1980) "Laplace's Method Revisited: Weak Convergence of Probability Measures" *Annals of Probability* **8-6** 1177–1182.

Imrohoroğlu, A. and S. Imrohoroğlu (1993) "A Numerical Algorithm for Solving Models with Incomplete Markets" *International Journal of Supercomputer Applications* **7-3** 212–230.

Ingram, B.F. (1990) "Solving the Stochastic Growth Model by Backsolving with an Expanded Shock Space" *Journal of Business and Economic Statistics* **8-1** 37–38.

Johnson, S.A. *et. al.* (1993) "Numerical Solution of Continuous-State Dyanmic Programs Using Linear and Spline Interpolation" *Operations Research* **41-3** 484–500.

Judd, K. (1990) "Minimum Weighted Residual Methods in Dynamic Economic Models" manuscript, Hoover Institution.

Judd, K. (1991) *Numerical Methods in Economics* manuscript, Hoover Institution.

Judd, K. (1993) "Projection Methods for Solving Aggregate Growth Models" *Journal of Economic Theory* **58** 410–452.

Judd, K. (1994a) "Comments on Marcet, Rust and Pakes" in C. Sims (ed.) *Advances in Econometrics Sixth World Congress,* Vol II. 261–276, Cambridge University Press.

Judd, K. (1994b) "Approximation Methods and Projection Methods in Economic Analysis" manuscript, Hoover Institution.

Judd, K. and A. Solnick (1994) "Numerical Dynamic Programming with Shape-Preserving Splines" manuscript, Hoover Institution.

Kantorovich, L.V. Akilov, G.P. (1982) *Functional Analysis* Pergamon Press, Oxford.

Keane, M. P. Wolpin, K. I. (1994) "The Solution and Estimation of Discrete Choice Dynamic Programming Models by Simulation: Monte Carlo Evidence" Staff Report 181, Federal Reserve Bank of Minneapolis.

Klee, V. Minty, G.J. (1972) "How Good is the Simplex Algorithm?" in O. Shisha (ed.) *Inequalities III* 159–175.

Kortum, S. (1992) "Value Function Approximaton in an Estimation Routine" manuscript, Boston University.

Krasnosel'skii, M.A. Vainikko, G.M. Zabreiko, P.P. Rutitskii, Ya. B. and V. Ya. Stetsenko (1972) *Approximate Solution of Operator Equations* D. Louvish, translator. Wolters-Noordhoff Publishing, Groningen.

Kreps, D.M. Porteus, E.L. (1978) "Temporal Resolution of Uncertainty and Dynamic Choice Theory" *Econometrica* **46-1** 185–200.

Kreps, D.M. Porteus, E.L. (1979) “Dynamic Choice Theory and Dynamic Programming” *Econometrica* **47-1** 91–100.

Kushner, H.J. (1990) “Numerical Methods for Stochastic Control Problems in Continuous Time” *SIAM Journal on Control and Optimization* **28-5** 999–1048.

Kushner, H.J. and A.J Kleinman (1971) “Accelerated Procedures for the Solution of Discrete Markov Control Problems” *IEEE Transactions on Automatic Control* **16-2** 147–152.

Kushner, H.J. and P.G. Dupuis (1992) *Numerical Methods for Stochastic Control Problems in Continuous Time* Springer Verlag, New York.

Kydland, F. Prescott, E.C. (1982) “Time to Build and Aggregate Fluctuations” *Econometrica* **50** 1345–1370.

Leland, H. (1974) “Optimal Growth in a Stochastic In A Stochastic Environment” *Review of Economic Studies* **41** 75–86.

Leonteif, W. (1966) *Input-Output Economics* Oxford University Press, London.

Lettau, M. and H. Uhlig (1994) “Rules of Thumb and Dynamic Programming” manuscript, Department of Economics, Princeton University.

Levhari, D. Srinivasan, T. (1969) “Optimal Savings Under Uncertainty” *Review of Economic Studies* **36** 153–163.

Long, J.B. Plosser, C. (1983) “Real Business Cycles” *Journal of Political Economy* **91-1** 39–69.

Lucas, R.E. Jr. (1978) “Asset Prices in an Exchange Economy” *Econometrica* **46** 1426–1446.

Lucas, R.E. Jr. Prescott, E.C. (1971) “Investment Under Uncertainty” *Econometrica* **39–5** 659–681.

Luenberger, D.G. (1969) *Optimization by Vector Space Methods* Wiley, New York.

Machina, M. J. (1987) “Choice Under Uncertainty: Problems Solved and Unsolved” *Journal of Economic Perspectives* **1-1** 121–154.

Marcet, A. (1994) “Simulation Analysis of Dynamic Stochastic Models: Applications to Theory and Estimation” in C. Sims (ed.) *Advances in Econometrics Sixth World Congress*, Vol II. 91-118, Cambridge University Press.

Marcet, A and D.A. Marshall (1994) “Solving Nonlinear Rational Expectations Models by Parameterized Expectations: Convergence to Stationary Solutions” mnuscript, Universitat Pompeu Fabra, Barcelona.

McGrattan, E.R. (1990) “Solving the Stochastic Growth Model by Linear-Quadratic Approximation” *Journal of Business and Economic Statistics* **8-1** 41–44.

McGrattan, E.R. (1993) “Solving the Stochastic Growth Model with a Finite Element Method” manuscript, Federal Reserve Bank of Minneapolis.

Merton, R.C. (1969) “Lifetime Portfolio Selection Under Uncertainty: The Continuous-time Case” *Review of Economics and Statistics* **51** 247–257.

Nelder, J. and R. Mead (1965) “A Simplex Method for Function Minimization” *Computational Journal* **7** 308–313.

Niederreiter, H. (1992) *Random Number Generation and Quasi-Monte Carlo Methods* SIAM CBMS-NSF volume 63, Philadelphia.

Novak, E. (1988) *Deterministic and Stochastic Error Bounds in Numerical Analysis* Springer-Verlag Lecture Notes in Mathematics, volume 1349, Berlin.

Ortega, J.M. (1988) *Introduction to Parallel and Vector Solution of Linear Systems* Plenum Press, New York. Ortega, J.M. and W.C. Rheinboldt (1970) *Iterative Solution on Nonlinear Equations in Several Variables* Academic Press, New York.

Ortega, J.M. Voigt, R.G. (1985) "Solution of Partial Differential Equations on Vector and Parallel Processors" *SIAM Review* **27-2** 149–240.

Ozaki, H. and P. Streufert (1992) "Nonlinear Dynamic Programming for Nonlinear Stochastic Objectives" manuscript, University of Western Ontario.

Pakes, A. and P. McGuire (1994) "Computing Markov-Perfect Nash Equilibria: Numerical Implications of a Dynamic Differentiated Products Model" **25-4** 555–589.

Pan, V. Reif, J. (1985) "Effi cient Parallel Solution of Linear Systems" *Transactions of the ACM* 143-152.

Papadimitriou, C.H. and J.N. Tsitsiklis (1987) "The Complexity of Markov Decision Processess" *Mathematics of Operations Research* **12-3** 441–450.

Paskov, S.H. (1993) "Average Case Complexity of Multivariate Integration for Smooth Functions" *Journal of Complexity* **9** 291–312.

Paskov, S.H. (1994) "Computing High Dimensional Integrals with Application to Finance" Technical Report, Columbia University Department of Computer Science.

Phelps, E. (1962) "Accumulation of Risky Capital" *Econometrica* **30** 729–743.

Phelan, C. Rust, J. (1991) "U.S. Social Security Policy: A Dynamic Analysis of Incentives and Self-Selection" manuscript University of Wisconsin.

Pinkus, A. (1985) *n-Widths in Approximation Theory* Spinger-Verlag, Berlin.

Porteus, E. L. (1980) "Overview of Iterative Methods for Discounted Finite Markov and Semi-Markov Decision Chains" in R. Hartley et. al. (eds.) *Recent Developments in Markov Decision Processes,* Academic Press.

Powell, M.J.D. (1981) *Approximation Theory and Methods* Cambridge University Press.

Press, W.B.P. Flannery, S.A. Teukolsky and W.T. Vetterling (1986) *Numerical Recipies* Cambridge University Press, Cambridge.

Puterman, M.L. (1990) "Markov Decision Processes" in D.P. Heyman and M.J. Sobel (eds.) *Handbooks in Operations Research and Management Science* Volume 2, North-Holland/Elsevier, Amsterdam.

Puterman, M.L. (1994) *Markov Decision Processes* Wiley, New York.

Rall, L.B. (1969) *Computational Solution of Nonlinear Operator Equations* Wiley, New York.

Rivlin, T.J. (1969) *An Introduction to the Approximation of Functions* Blaisdell Publishing Company, Waltham Massachusetts.

Rust, J. (1985) "Stationary Equilibrium in a Market for Durable Assets" *Econometrica* **53-4** 783–806.

Rust, J. (1986) "When Is It Optimal to Kill Off the Market for Used Durable Goods?" *Econometrica* **54-1** 65–86.

Rust, J. (1987) "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher" *Econometrica* **55-5** 999–1033.

Rust, J. (1988a) "Statistical Models of Discrete Choice Processes" *Transportation Research* **22B-2** 125–158.

Rust, J. (1988b) "Maximum Likelihood Estimation of Discrete Control Processes" *SIAM Journal on Control and Optimization* **26-5** 1006–1023.

Rust, J. (1989a) "A Dynamic Programming Model of Retirement Behavior" in D. Wise (ed.) *The Economics of Aging* University of Chicago Press, 359–398.

Rust, J. (1989b) "Comment on 'Optimal Collection of Information by Partially Informed Agents"' *Econometric Reviews* 170–178.

Rust, J. (1994a) "Structural Estimation of Markov Decision Processes" forthcoming in D. McFadden and R. Engle (eds.) *Handbook of Econometrics* **volume 4** North Holland.

Rust, J. (1994b) "Estimation of Dynamic Structural Models, Problems and Prospects: Discrete Decision Processes" in C. Sims (ed.) *Advances in Econometrics Sixth World Congress* Volume II, 119–170. Cambridge University Press.

Rust, J. (1995a) "Do People Behave According to Bellman's Principle of Optimality?" manuscript, University of Wisconsin.

Rust, J. (1995b) "Using Randomization to Break the Curse of Dimensionality", manuscript, University of Wisconsin.

Rust, J. (1996) *Stochastic Decision Processes: Theory, Computation, and Estimation*, manuscript, University of Wisconsin.

Saad, Y. (1984) "Practical Use of Some Krylov Subspace Methods for Solving Indefi nite and Nonsymmetric Linear Systems" *SIAM Journal on Scientific and Statistical Computing* **5-1** 203–228.

Saad, Y. (1989) "Krylov Subspace Methods on Supercomputers" *SIAM Journal on Scientific and Statistical Computing* **10-6** 1200–1232.

Saad, Y. and M.H. Schultz (1986) "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems" *SIAM Journal on Scientific and Statistical Computing* **7-3** 856–869.

Samuelson, P.A. (1969) "Lifetime Portfolio Selection by Dynamic Stochastic Programming" *Review of Economics and Statistics* **51** 239–246.

Sargent, T.J. (1978) "Estimation of Dynamic Labor Demand Schedules Under Rational Expectations" *Journal of Political Economy* **86–6**, 1009–1044.

Sargent, T.J. (1981) "Interpreting Economic Time Series" *Journal of Political Economy* **89–2**, 213–248.

Schumacher, L.L. (1983) "On Shape Preserving Quadratic Spline Interpolation" *SIAM Journal of Numerical Analysis* **20-4** 854–864.

Schweitzer, P.J. and A. Seidmann (1985) "Generalized Polynomial Approximations in Markovian Decision Processes" *Journal of Mathematical Analysis and Applications* **110** 568–582.

Semmler, W. (1994) "Solving Nonlinear Dynamic Models by Iterative Dynamic Programming" manuscript, New School for Social Research, New York.

Sikorski, K. (1984) "Optimal Solution of Nonlinear Equations Satisfying a Lipschitz Condition" *Numerische Matematik* **43** 225–240.

Sikorski, K. (1985) "Optimal Solution of Nonlinear Equations" *Journal of Complexity* **1** 197–209.

Simon, H.A. (1956) "Dynamic Programming Under Uncertainty with a Quadratic Criterion Function" *Econometrica* **24** 74-81.

Sims, C.A. "Solving the Stochastic Growth Model by Backsolving with a Particular Nonlinear Form for the Decision Rule" *Journal of Business and Economic Statistics* **8-1** 45–47.

Smith, A. A. Jr. (1991) "Solving Stochastic Dynamic Programming Problems Using Rules of Thumb" Discussion Paper 816, Institute for Economic Research, Queen's University, Ontario, Canada.

Solis, F.J. and R.J. Wets (1981) "Minimization by Random Search Techniques" *Mathematics of Operations Research* **6-1** 19–30.

Stern, S. (1991) "Approximate Solutions to Stochastic Dynamic Programming Problems", manuscript, University of Virginia.

Stock, J. Wise, D. (1990) "Pensions, The Option Value of Work and Retirement" *Econometrica* **58–5**, 1151–1180.

Stokey, N.L. Lucas, R.E. Jr. (with Prescott, E.C.) (1989) *Recursive Methods in Economic Dynamics* Harvard University Press, Cambridge, Massachusetts.

Stone, C.J. (1982) "Optimal Global Rates of Convergence for Nonparametric Regression" *Annals of Statistics* **10-4** 1040-1053.

Tapiero, C.S. and A. Sulem (1994) "Computational Aspects in Applied Stochastic Control" forthcoming in *Computational Economics.*

Tauchen, G. (1990) "Solving the Stochastic Growth Model by Using Quadrature Methods and Value Function Iterations" *Journal of Business & Economic Statistics* **8-1** 49–51.

Tauchen, G. Hussey, R. (1991) "Quadrature-Based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models" *Econometrica* **59-2** 371–396.

Taylor, J.B. and H. Uhlig (1990) "Solving Nonlinear Stochastic Growth Models: A Comparison of Alternative Solution Methods" *Journal of Business and Economic Statistics* **8-1** 1–17.

Temlyakov, V.N. (1987) "Approximate Recovery of Periodic Functions of Several Variables" *Math. USSR Sbornik* **56** 249–261.

Traub, J.F. (1994) “Breaking Intractability” *Scientific American* **270-1** 102–107.

Traub, J.F. Wasilkowski, G.W. and Woźniakowski, H. (1988) *Information-based Complexity* Academic Press.

Traub, J.F. and Woźniakowski, H. (1980) *A General Theory of Optimal Algorithms* Academic Press, New York.

Traub, J.F. and Woźniakowski, H. (1984) “On the Optimal Solution of Large Linear Systems” *Journal of the Association for Computing Machinery* **31-3** 545–559.

Traub, J.F. and Woźniakowski, H. (1991) “Information-Based Complexity: New Questions for Mathematicians” *The Mathematical Intelligencer* **13-2** 34–43.

Trick, M.A. and S.E. Zin (1993) “A Linear Programming Approach to Solving Stochastic Dynamic Programs” manuscript, Carnegie-Mellon University.

Tsitsiklis, J.N. (1993) “Asynchronous Stochastic Approximation and Q-Learning” manuscript, Laboratory for Information and Decision Systems, MIT.

Tsitsiklis, J.N. (1994) “Complexity-Theoretic Aspects of Problems in Control Theory” manuscript, Center for Intelligent Control Systems, MIT. van Dijk, N.M. (1980) *Controlled Markov Processes: Time Discretization* CWI Tract 11, Mathematische Centrum, Amsterdam.

Wald, A. (1947) *Sequential Analysis* Wiley, New York.

Wasilkowski, G.W. (1994) “On the Average Complexity of Global Optimization Problems” manuscript, Department of Mathematics, University of Kentucky.

Wershculz, A.G. (1991) *The Computational Complexity of Differential and Integral Equations* Oxford University Press, New York.

Wheeler, R.M. Jr. and K.S. Narendra (1986) “Decentralized Learning in Finite Markov Chains” *IEEE Transactions on Automatic Control* **AC-31** 519–526.

Whittle, P. (1982) *Optimization Over Time: Dynamic Programming and Stochastic Control* Volumes I and II. J. Wiley, New York.

Woźniakowski, H. (1991) “Average Case Complexity of Multivariate Integration” *Bulletin of the American Mathematical Society* **24** 185–194.

Woźniakowski, H. (1992) “Average Case Complexity of Linear Multivariate Problems” *Journal of Complexity* **8** (Parts I and II) 372–392.

Zaremba, S.K. (1968) “The Mathematical Basis of Monte Carlo and Quasi Monte Carlo Methods” *SIAM Review* **10** 303–314.

Zeidler, E. (1993) *Nonlinear Functional Analysis and its Applications* Volumes I and IIA Springer-Verlag, New York.

Zijm, W.H.M. (1980) *Nonnegative Matrices in Dynamic Programming* CWI Tract 167, Mathematische Centrum, Amsterdam.