Regression Methods for Pricing Complex American-Style Options

John N. Tsitsiklis, *Fellow, IEEE,* and Benjamin Van Roy

***Abstract*—We introduce and analyze a simulation-based approximate dynamic programming method for pricing complex American-style options, with a possibly high-dimensional underlying state space. We work within a finitely parameterized family of approximate value functions, and introduce a variant of value iteration, adapted to this parametric setting. We also introduce a related method which uses a single (parameterized) value function, which is a function of the time-state pair, as opposed to using a separate (independently parameterized) value function for each time. Our methods involve the evaluation of value functions at a finite set, consisting of "representative" elements of the state space. We show that with an arbitrary choice of this set, the approximation error can grow exponentially with the time horizon (time to expiration). On the other hand, if representative states are chosen by simulating the state process using the underlying risk-neutral probability distribution, then the approximation error remains bounded.**

I. Introduction

An important problem in financial intermediation is that of pricing and hedging American-style options—i.e., options with flexible exercise features. Such contracts—ranging from American equity and fixed income options to convertible bonds—arise in virtually all major financial markets. Their analysis typically entails solving problems of optimal stopping. For simple contracts, including "vanilla options" such as American puts and calls, the relevant optimal stopping problems can be solved efficiently by traditional numerical methods. However, the computational requirements associated with such methods become prohibitive as the number of uncertainties influencing the value of a contract grows.

A simple approach to solving optimal stopping problems—and therefore pricing and hedging contracts with flexible exercise features—involves "backward induction," i.e., the dynamic programming value iteration algorithm. The process starts by computing a value function at the expiration date, and then recursively works backward, computing value functions for preceding time periods. Each value function maps states to expected future payoffs, where the expectation is taken with respect to "risk-neutral probabilities." Optimal stopping decisions (i.e., exercising decisions) can be made by comparing the payoff for stopping (i.e., the intrinsic value of a contract) against the expected future payoff.

Manuscript received August 15, 2000; revised February 20, 2001. This work was supported by the Merrill Lynch-MIT partnership, NSF CAREER Grant ECS-9985229, and ONR Grant MURI N00014-00-1-0637.

J. N. Tsitsiklis is with the Massachusetts Institute of Technology (MIT), Cambridge, MA 02139 USA.

B. Van Roy is with Stanford University, Stanford, CA 94305 USA.

Publisher Item Identifier S 1045-9227(01)05023-8.

Many complex contracts involve contingencies on multiple sources of uncertainty, each represented as a state variable. The size of the state space grows exponentially in the number of variables involved, and consequently, storage and manipulation of functions over the state space—as entailed in value iteration—becomes intractable. This phenomenon—known as the curse of dimensionality—creates a need for parsimonious approximation schemes.

One simple approximation method—dating all the way back to 1959 [3]—is approximate value iteration. Similar to value iteration, this approximation algorithm proceeds recursively, beginning with a value function at expiration and recursively computing value functions for preceding time periods. However, instead of computing and storing each value function at every state, values are computed only at a representative sample of states and a linear combination of basis functions is fit to the data via least-squares regression, in order to approximate the value function over the entire state space.

In earlier work [20], we studied versions of approximate value iteration for infinite horizon optimal stopping problems, where the recursive procedure is iterated with the hope of converging to an approximate value function for the infinite horizon problem (e.g., a perpetual option). As established earlier for related algorithms [19], approximate value iteration can diverge. However, we were able to develop special variants of approximate value iteration that make use of simulated trajectories from the underlying Markov chain and are guaranteed to converge to an approximation of the desired value function.

In this paper, we bring the line of analysis developed in [20] to bear on finite horizon problems, which are more relevant to real-world financial contracts (such contracts almost exclusively prescribe finite expiration times). A key observation in this context is that errors in approximate value iteration can grow exponentially in the problem horizon (i.e., time to expiration). This phenomenon closely relates to the fact that value iteration can diverge when applied to infinite-horizon problems, as observed in [20]. However, using simulated trajectories in a spirit similar to algorithms from [20], one can design versions of approximate value iteration for which the error is uniformly bounded over all horizons.

Let us note that other researchers have independently developed a similar algorithm that benefits from the same property. In particular, Longstaff and Schwartz [14] have also proposed the use of simulated trajectories in approximate value iteration. Empirical results reported in [14] are promising, but the authors do not offer an analysis relating performance to the use of simulated trajectories. The main result presented in [14] is asymptotic, establishing only that approximations converge to the cor-

rect value function as the number of basis functions and trajectories used by the algorithm grow. In addition to assymptotic results of this kind, the theory we provide in this paper offers theoretical support for the apparent effectiveness of such algorithms when using a limited number of basis functions.

An additional contribution of this paper is a version of approximate value iteration that uses basis functions that generalize over both the state space and time. In particular, as an alternative to generating a new approximation to the value function at each time period, we consider a single approximation made up of basis functions each taking as arguments both state and time. Such approximations tend to be far more parsimonious, since the number of parameters computed and stored need not grow linearly with the number of time periods.

Other versions of approximate value iteration have also been proposed in the options pricing literature. Some involve partitioning the state space and computing one value per partition [18], [2]. This can be viewed as a version of approximate value iteration involving piece-wise constant approximations, which tend to be somewhat restrictive. Another notable approach involves "stochastic mesh" methods [1], [7]–[9], [16]. These methods can be thought of as variants of Rust's algorithm [17], specialized to the context of optimal stopping. Values are approximated at points in a finite mesh over the state space in a spirit similar to traditional grid techniques. The difference, however, is that the mesh includes a tractable collection of randomly sampled states, rather than the intractable grid that would arise in standard state space discretization. Though using a stochastic mesh can curtail computational requirements in significant ways, such algorithms generally require a number of samples that grows exponentially in the dimension of the state space, except for some cases that satisfy unrealistically restrictive assumptions such as those presented in [17].

This paper is organized as follows. In Section II, we introduce the optimal stopping problem associated with the option pricing problem, and the corresponding value iteration (dynamic programming) algorithm. In Section III, we introduce a few variants of value iteration that work within a parameterized family of value functions, and carry out computations at a finite set of "representative" elements of the state space. In Section IV, we show by means of examples, that such methods can have a large approximation error (exponential in the time horizon of the problem). In Section V, we show that if representative states are chosen by simulating the state process using the underlying risk-neutral probability distribution, then the approximation error remains bounded. In Section VI, we introduce a related method which uses a single (parameterized) value function, which is a function of the time-state pair, as opposed to using a separate (independently parameterized) value function for each time. Finally, Section VII contains some brief conclusions.

II. Pricing via Value Iteration

The problem of pricing an American-style option amounts to one of optimal stopping. A reward equal to the intrinsic value of the option, discounted at the risk-free rate, is received at termination. The price of the option is the expected reward under an optimal exercising strategy, where the expectation is taken with respect to a risk-neutral distribution. Hence, the price is given by

$$\sup_{\tau \in [0,\mathcal{T}]} E[e^{-r\tau} g(x_\tau)]$$

where

$\{x_t \in \Re^d \,\vert\, 0 \le t \le \mathcal{T}\}$	risk-neutral state process, assumed to be Markov;
r	risk-free interest rate, assumed to be a known constant;
$g(x)$	intrinsic value of the option when the state is x;
$\mathcal{T}$	expiration time, and the supremum is taken over stopping times that assume values in $[0, \mathcal{T}]$.

Naturally, we consider stopping times with respect to the filtration $\{\mathcal{F}_t \,\vert\, 0 \le t \le \mathcal{T}\}$, where $\mathcal{F}_t$ is the σ-field generated by $\{x_s \,\vert\, 0 \le s \le t\}$. We will assume for simplicity that the risk-neutral process is time-homogeneous, as is true for most problems solved in practice.

It is sometimes convenient to consider discrete-time versions of the aforementioned optimal stopping problem. This modification can be thought of as a requirement that the option be exercised at certain prespecified intervals—in other words, the option is treated as a Bermudan. The restriction on exercise times diminishes the value of the option, but under mild technical conditions, the difference in value is small and vanishes as the difference between allowable exercise times goes to zero.

Without loss of generality, let us assume that the expiration time $\mathcal{T}$ is equal to an integer N, and that the allowable exercise times are separated by a time interval of unit length. The price of the resulting Bermudan option is then

$$\sup_{\tau} E[\alpha^\tau g(x_\tau)]$$

where $\alpha = e^{-r}$, and where τ ranges over the set of stopping times (with respect to $\{\mathcal{F}_t \,\vert\, 0 \le t \le N\}$) that take values in $\{0, 1, \ldots, N\}$. In this discrete-time and Markovian formulation, the dynamics of the risk-neutral process can be described by a transition operator P, defined by

$$(PJ)(x) = E[J(x_{n+1}) \,\vert\, x_n = x].$$

Note that the above expression does not depend on n, since the process is assumed time-homogeneous.

A primary motivation for this discretization is that it facilitates exposition of computational procedures, which typically entail discretization. The value iteration algorithm, for example, provides a means for options pricing when time is discrete. This algorithm generates a sequence $J_N, J_{N-1}, \ldots, J_0$ of value functions, where $J_n(x)$ is the price of the option at time n, if x_n is equal to x. The value functions are generated iteratively according to

$$J_N = g$$

and

$$J_n = \max(g, \alpha P J_{n+1}), \qquad n = N-1, N-2, \ldots, 0$$

where the maximum is taken pointwise. The initial price of the option is then $J_N(x_0)$.

In principle, value iteration can be used to price any Bermudan option. However, the algorithm suffers from the "curse of dimensionality"—that is, the computation time grows exponentially in the number d of state variables. This difficulty arises because computations involve discretization of the state space, and such discretization leads to a grid whose size grows exponentially in dimension. Since one value is computed and stored for each point in the grid, the computation time exhibits exponential growth. For complex American options such as those involving path dependencies (e.g., Asian options) or multi-factor interest rate models, the number of state variables can be substantial and the computational requirements of value iteration become prohibitive.

III. Approximations

Unless the dimension of the state space is small, the pricing problem becomes intractable and calls for approximation of the value functions. The first step is to introduce a parameterized value function $\tilde{J} : \Re^d \times \Re^K \mapsto \Re$, which assigns values $\tilde{J}(x,r)$ to states x, where $r \in \Re^K$ is a vector of free parameters. The objective then becomes to choose, for each n, a parameter vector r_n so that

$$\tilde{J}(x, r_n) \approx J_n(x).$$

For this to be possible, a suitably rich parameterization has to be chosen, so that the "approximation architecture" has the capability of closely approximating the functions of interest. This choice typically requires some practical experience or theoretical analysis that provides rough information about the shape of the function to be approximated. Furthermore, we need algorithms for computing appropriate parameter values. We will present variants of value iteration designed to accommodate the latter need.

A. Features and Approximation Architectures

In choosing a parameterization to approximate the value function for a particular problem, it is useful to consider the notion of a *feature*. Let us define a *feature* as a function mapping the state space $\Re^d$ to $\Re$. Given a problem, one may wish to define several features $\phi_1, \ldots, \phi_K$. Then, to each state $x \in \Re^d$, we associate the feature vector $\phi(x) = (\phi_1(x), \ldots, \phi_K(x))'$. Such a feature vector is meant to represent the most salient properties of a given state.

In a feature-based parameterization, $\tilde{J}(x,r)$ depends on x only through $\phi(x)$. Hence, for some function $f : \Re^K \times \Re^K \mapsto \Re$, we have $\tilde{J}(x,r) = f(\phi(x), r)$. In problems of interest, the value functions can be complicated, and a feature-based parameterization attempts to break their complexity into less complicated mappings f and ϕ. There is usually a trade-off between the complexity of f and ϕ, and different choices lead to drastically different structures. As a general principle, the feature extraction function ϕ is usually hand crafted and relies on whatever human experience or intelligence is available. The function f represents the choice of an *architecture* used for approximation.

In this paper, we will restrict attention to linearly parameterized architectures, of the form

$$\tilde{J}(x,r) = \sum_{k=1}^{K} r(k)\phi_k(x)$$

i.e., the value function is approximated by a linear combination of features. The simplicity of this architecture makes it amenable to analysis, and we will discuss theoretical results pertaining to approximate value iteration in this context. To emphasize the linear dependence on parameters and as shorthand notation, we define an operator Φ (that maps vectors in $\Re^K$ to real-valued functions of the state) by

$$(\Phi r)(x) = \sum_{k=1}^{K} r(k)\phi_k(x).$$

Many standard function approximators can be thought of as linear feature-based parameterizations. Among these are radial basis function networks, wavelet networks, polynomials, and more generally all approximators that involve a fixed set of basis functions.

The architecture, as described by f, could also be a nonlinear mapping such as a feedforward neural network (multilayer perceptron) with weights r. The feature extraction mapping ϕ could be either entirely absent or it could be included to facilitate the job of the neural network. Such approximation architectures may offer gains in practical performance. Unfortunately, there is not much that can be said analytically in this context, and we will not study such architectures in this paper.

B. Approximate Value Iteration

Given a choice of parameterization $\tilde{J}$, the computation of appropriate parameters calls for a numerical algorithm. In this paper, we study versions of approximate value iteration, which generate a sequence of parameters $r_{N-1}, r_{N-2}, \ldots, r_0$, leading to approximations $\tilde{J}(\cdot, r_{N-1}), \ldots, \tilde{J}(\cdot, r_0)$ to the true value functions $J_{N-1}, \ldots, J_0$.

The simplest form of approximate value iteration involves a single projection matrix Π (acting on the space of value functions) that projects onto the span of $\phi_1, \ldots, \phi_K$, with respect to a weighted quadratic norm $\|J\|_\pi$, defined by

$$\|J\|_\pi = \left(\int_{x \in \Re^d} J^2(x)\pi(dx) \right)^{1/2}$$

where π is a probability measure on $\Re^d$. In other words, the projection operator is characterized by

$$\Pi J = \operatorname*{argmin}_{\Phi r} \|J - \Phi r\|_\pi.$$

The algorithm generates iterates satisfying

$$\tilde{J}(\cdot, r_{N-1}) = \Pi \max(g, \alpha P g)$$

and

$$\tilde{J}(\cdot, r_n) = \Pi \max(g, \alpha P \tilde{J}(\cdot, r_{n+1})), \qquad n = N-2, \ldots, 0.$$

Note that the range of the projection is the same as that of Φ and therefore, for any function J with $||J||_\pi < \infty$, there is a weight vector $r \in \Re^K$ such that

$$\Pi J = \Phi r = \tilde{J}(\cdot, r).$$

Clearly, the approximation algorithm generates value functions by mimicking value iteration, while sacrificing exactness in order to maintain functions within the range of the approximator (the span of the features).

A more sophisticated version of the algorithm involves projections that are dependent on the time period. In particular, one can define a sequence of probability measures, $\pi_0, \ldots, \pi_{N-1}$ as well as projection operators $\Pi_0, \ldots, \Pi_{N-1}$ that project with respect to the norms $\|\cdot\|_{\pi_0}, \ldots, \|\cdot\|_{\pi_{N-1}}$, respectively. Then, the approximate value iteration algorithm would generate iterates according to

$$\tilde{J}(\cdot, r_{N-1}) = \Pi_{N-1} \max(g, \alpha P g)$$

and

$$\tilde{J}(\cdot, r_n) = \Pi_n \max(g, \alpha P \tilde{J}(\cdot, r_{n+1})), \qquad n = N-2, \ldots, 0.$$

C. Sample-Based Projection and Q-Values

The approximation algorithm that we have described offers advantages over value iteration because it uses a parsimonious representation. Only K numerical values need to be stored at each stage. However, we have not discussed the computation of these parameters, which can turn out to be time-intensive. In this section, we address this issue and offer an alternative version of approximate value iteration that facilitates projection.

Exact computation of a projection is not generally viable. However, one can approximate a projection Π effectively by sampling a collection of states $y_1, \ldots, y_m \in \Re^d$ according to the probability measure π, and then defining an approximate projection operator

$$\hat{\Pi} J = \operatorname*{argmin}_{\Phi r} \sum_{i=1}^{m} (J(y_i) - (\Phi r)(y_i))^2.$$

As m grows, this approximation becomes close to exact, in the sense that $\|\hat{\Pi} J - \Pi J\|_\pi$ converges to zero with probability 1.

Given the above approximate projection operator $\hat{\Pi}$, one can define a modified version of approximate value iteration, generating parameters according to

$$\tilde{J}(\cdot, r_{N-1}) = \hat{\Pi} \max(g, \alpha P g)$$

and

$$\tilde{J}(\cdot, r_n) = \hat{\Pi} \max(g, \alpha P \tilde{J}(\cdot, r_{n+1})), \qquad n = N-2, \ldots, 0.$$

As opposed to the original version of the algorithm, in which projections posed a computational burden, this new variant involves the solution of a linear least squares problem, with K free parameters, and admits efficient computation of projections, as long as the number of samples m is reasonable. However, there is an additional obstacle that we must overcome, as we now discuss.

For each sample y_i, and any function J, evaluating $\max(g(y_i), \alpha(PJ)(y_i))$ entails the computation of an expectation $(PJ)(y_i) = E[J(x_{k+1}) \,|\, x_k = y_i]$. This expectation is over a potentially high-dimensional space $\Re^d$ and can therefore pose a computational challenge. Monte Carlo simulation offers one way to address this task. In particular, for each sample y_i, we can simulate independent samples $z_{i1}, \ldots, z_{il}$ from the transition distribution, conditioned on the current state being y_i, and then define an approximate expectation operator $\hat{P}$ by

$$(\hat{P}J)(y_i) = \frac{1}{l} \sum_{j=1}^{l} J(z_{ij}).$$

Then, a modified version of approximate value iteration is given by

$$\tilde{J}(\cdot, r_{N-1}) = \hat{\Pi} \max(g, \alpha \hat{P} g)$$
$$\tilde{J}(\cdot, r_n) = \hat{\Pi} \max(g, \alpha \hat{P} \tilde{J}(\cdot, r_{n+1})) \qquad n = N-2, \ldots, 0.$$

Though this approach is viable, there is an alternative that makes implementation even more convenient. This alternative, which we describe next, relies on single-sample estimates of the desired expectations.

D. Using Q-values and single-sample estimates

Define for each $n = 0, \ldots, N-1$, a Q-function

$$Q_n = \alpha P J_{n+1}.$$

Note that $Q_n(x)$ represents the expected discounted payoff at time n conditioned on a decision not to exercise. A version of value iteration can be used to produce Q-values directly

$$Q_{N-1} = \alpha P g,$$
$$Q_n = \alpha P \max(g, Q_{n+1}), \qquad n = 1, 2, \ldots, N-1.$$

Furthermore, given a parameterization $\tilde{Q}(\cdot, r) = \Phi r$ and a projection operator Π, a version of approximate value iteration is defined by

$$\tilde{Q}(\cdot, r_{N-1}) = \alpha \Pi P g$$
$$\tilde{Q}(\cdot, r_n) = \alpha \Pi P \max(g, \tilde{Q}(\cdot, r_{n+1})) \qquad n = N-2, \ldots, 0. \tag{1}$$

Once again, we sample states in order to approximate the projection and expectation. For the projection, as before, we sample a collection of states $y_1, \ldots, y_m$, distributed according to π. The approximate projection operator is then given by

$$\hat{\Pi} J = \operatorname*{argmin}_{\Phi r} \sum_{i=1}^{m} (J(y_i) - (\Phi r)(y_i))^2.$$

We will now base our approximation of the expectation on only a single sample. In particular, for each y_i, we generate one successor state z_i by simulation. Then, the approximate expectation operator is defined by

$$(\hat{P}J)(y_i) = J(z_i).$$

The resulting version of approximate value iteration is of the form

$$\tilde{Q}(\cdot, r_{N-1}) = \alpha \hat{\Pi} \hat{P} g$$
$$\tilde{Q}(\cdot, r_n) = \alpha \hat{\Pi} \hat{P} \max(g, \tilde{Q}(\cdot, r_{n+1})) \qquad n = N-2, \ldots, 0. \tag{2}$$

In full detail, a typical iteration of the algorithm proceeds as follows. Given the parameter vector r_{n+1}, and the resulting approximation $\tilde{Q}(\cdot, r_{n+1})$ of Q_{n+1}, defined by

$$\tilde{Q}(x, r_{n+1}) = \sum_{k=1}^{K} r_{n+1}(k)\phi_k(x)$$

we select m independent random samples of the state, $y_1, \ldots, y_m$, according to the probability measure π, and for each y_i, we simulate a successor state z_i. Then, the vector r_n is found by minimizing

$$\sum_{i=1}^{m} \left(\alpha \max \left\{ g(z_i), \sum_{k=1}^{K} r_{n+1}(k)\phi_k(z_i) \right\} - \sum_{k=1}^{K} r(k)\phi_k(y_i) \right)^2$$

with respect to $r(1), \ldots, r(K)$.

Given a sample state y_i, the expected value (with respect to the random next state z_i) of $(\hat{P}J)(y_i)$ is just $(PJ)(y_i)$, for any function J. For this reason, $\alpha \hat{\Pi} \hat{P} \max(g, \tilde{Q}(\cdot, r_n))$ is an unbiased estimate of $\alpha \hat{\Pi} P \max(g, \tilde{Q}(\cdot, r_n))$. This was made possible because $\hat{P}$ enters linearly and effectively allows for the noise (in the next state z_i) to be averaged out. Such unbiasedness would not be possible with the original version of approximate value iteration that produced iterates $\tilde{J}(\cdot, r_n)$, because the dependence on $\hat{P}$ was nonlinear.

Of course, there is no need for employing the same probability measure π at each iteration n. In a more general version of the algorithm, we introduce probability measures $\pi_0, \ldots, \pi_{N-1}$. For each time n, we generate m random states $y_{n1}, \ldots, y_{nm}$, sampled independently according to π_n, leading to an approximate projection operator $\hat{\Pi}_n$, and the algorithm

$$\tilde{Q}(\cdot, r_{N-1}) = \alpha \hat{\Pi}_{N-1} \hat{P} g, \tag{3}$$
$$\tilde{Q}(\cdot, r_n) = \alpha \hat{\Pi}_n \hat{P} \max(g, \tilde{Q}(\cdot, r_{n+1})) \qquad n = N-2, \ldots, 0. \tag{4}$$

In studying the behavior of the algorithms considered here, we need to address two distinct issues.

1) We need to study the approximation error of an idealized algorithm such as (1).
2) We need to determine whether the use of an approximate projection and a single-sample estimate of the expectation [as in (2)] leads to a significant difference from the results of the idealized algorithm.

The second issue is not hard to resolve using laws of large numbers. The first one is more subtle and constitutes our main focus.

IV. The Accumulation of Error

There is an appealing simplicity to the approximate value iteration algorithms defined in the previous section. Unfortunately, such algorithms often lead to errors that grow exponentially in the problem horizon. In this section, we provide examples demonstrating this phenomenon. A remedy to this undesirable state of affairs is offered in Sections V and VI.

We focus in this section on the version of approximate value iteration that involves Q-functions and exact computation of expectations and projections. In other words, we consider the algorithm in (1)

$$\tilde{Q}(\cdot, r_{N-1}) = \alpha \Pi P g,$$
$$\tilde{Q}(\cdot, r_n) = \alpha \Pi P \max(g, \tilde{Q}(\cdot, r_{n+1})), \qquad n = N-2, \ldots, 0.$$

Our observations, however, apply to other forms of approximate value iteration, including those that make use of sample estimates, as well as those that are based on value functions $\tilde{J}(\cdot, r_n)$.

A. A Simple Example

Consider a vanilla American put with strike price 1, and N periods until expiration. The intrinsic value given a current stock price x is

$$g(x) = \max(0, 1 - x).$$

We consider risk-neutral price dynamics given by

$$x_{n+1} = \begin{cases} ux_n & \text{w.p. } p \\ dx_n & \text{w.p. } 1-p \end{cases}$$

with $u > 1 > d$.

Consider using a single basis function $\phi(x) = g(x)$, which is identical to the intrinsic value of the put. Hence, we will generate for each n a scalar r_n such that $r_n\phi(x)$ represents an approximation to the present value of the option when the time is n and the current stock price is x.

For the purposes of approximate value iteration, we define a distribution π over the state space. To keep our computations as simple as possible, we will choose a trivial distribution, which assigns all probability to a single state $1 - \Delta$, for some small positive scalar Δ. Hence, the corresponding projection operator Π ensures that there is an exact fit at the state $1 - \Delta$, that is

$$(\Pi J)(1 - \Delta) = J(1 - \Delta)$$

for any function J.

Let us now study how the parameters r_n evolve as approximate value iteration progresses. Recall that the iteration under consideration is

$$\tilde{Q}(\cdot, r_n) = \alpha \Pi P \max(g, \tilde{Q}(\cdot, r_{n+1})), \qquad n = N-2, \ldots, 0.$$

For any $r \geq 0$, we have

$$(P \max(g, r\phi))(1 - \Delta) \geq (P \max(g, r\phi))(1)$$
$$\geq (1-p)(1-d)r.$$

Given the simple form of the projection, we have

$$\begin{aligned}(\Pi P\max(g,\tilde{Q}(\cdot,r_{n+1})))(1-\Delta)\\ =(P\max(g,\tilde{Q}(\cdot,r_{n+1})))(1-\Delta)\\ =(1-p)(1-d)r_{n+1}.\end{aligned}$$

It follows that

$$\Delta r_n \geq \alpha(1-p)(1-d)r_{n+1}$$

and dividing both sides of the inequality by Δ, we have

$$r_n \geq \frac{\alpha}{\Delta}(1-p)(1-d)r_{n+1}.$$

Hence, for $\Delta < \alpha(1-p)(1-d)$, the sequence r_n grows at an exponential rate, as n progresses from N to 0. Furthermore, the growth rate can be made arbitrarily large by reducing Δ.

B. A Numerical Example

Let us now consider somewhat more realistic computations, again involving the same American put. As before, we take the strike price to be 1 and consider the same binomial model for stock price movements

$$x_{n+1}=\begin{cases}ux_n & \text{w.p. } p\\ dx_n & \text{w.p. } 1-p\end{cases}$$

with $u > 1 > d$. Particular values of the variables employed in our computations are provided in the table below

strike	$\overline{x}$	1
high return	u	3/2
low return	d	2/3
probability of high return	p	0.4121
discount factor	α	0.99

We will approximate the value function using a quadratic. For each n, the weights are given by a three-dimensional vector $r_n \in \Re^3$, and the approximation is defined by

$$\tilde{Q}(x,r_n)=r_n(1)+r_n(2)x+r_n(3)x^2.$$

In the previous section, to keep computations as simple as possible, we employed a distribution π that assigned all probability to the single point $1-\Delta$. Let us now consider a situation more likely to arise in a practical implementation. Let the distribution π assign probabilities uniformly to points in a discrete grid spread over a segment of the state space. In particular, we consider the set

$$S=\{0.1,0.2,0.3,\ldots,1.8,1.9,2.0\}$$

and focus on approximating values at these points. Hence, the projection is with respect to the norm defined by

$$\|Q\|_\pi=\left(\frac{1}{20}\sum_{x\in S}Q^2(x)\right)^{1/2}.$$

As before, the iteration under consideration is

$$\tilde{Q}(\cdot,r_n)=\alpha\Pi P\max(g,\tilde{Q}(\cdot,r_{n+1})),\qquad n=N-2,\ldots,0$$

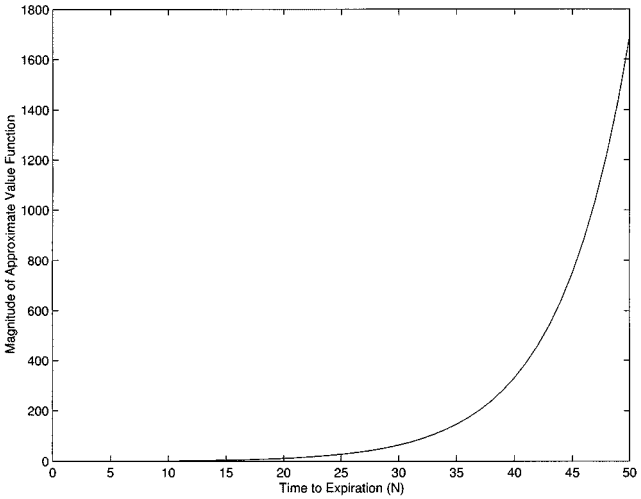

Fig. 1. Exponential growth of the error in the example of Section IV-B.

initialized with $\tilde{Q}(\cdot,r_{N-1})=\alpha\Pi Pg$. Plotted in Fig. 1 are the magnitudes $\|\tilde{Q}(\cdot,r_0)\|_\pi$ of approximations $\tilde{Q}(\cdot,r_0)$ generated by the algorithm, for various horizons N. The magnitudes apparently grow exponentially with N. Note that the present value of the put with N periods until expiration is bounded above by α^N. Hence, the exponential growth of the norm, as depicted in Fig. 1, implies exponential growth in the error between the true and approximated Q-functions, as measured by the norm $\|\cdot\|_\pi$.

V. The Use of Trajectory Distributions

In this section, we show that if the sample states used in the approximate projection are obtained by simulating trajectories of the process x_n, then the approximation error grows no faster than $\sqrt{N}\epsilon$ (instead of growing exponentially in N), where ϵ is the best possible approximation error under the chosen approximation architecture.

Let $\pi_0,\pi_1,\ldots,\pi_N$ be the probability measures on $\Re^d$ that describe the probability distribution of $x_0,x_1,\ldots,x_N$, respectively, under the risk-neutral dynamics of the process. We take x_0 to be known, so that π_0 is concentrated on that single point. We make the following assumption.

Assumption 1:

1) For every n, we have
$$\|g\|_{\pi_n}^2=E[g^2(x_n)]<\infty.$$
2) For every n and every k, the feature ϕ_k satisfies
$$\|\phi_k\|_{\pi_n}^2=E[\phi_k^2(x_n)]<\infty.$$
3) (Linear independence of the features.) For every n and every $r\neq 0$, the random variable $r'\phi(x_n)$ is nonzero with positive probability. Equivalently, the matrix $E[\phi(x_n)\phi'(x_n)]$ is positive definite.

Let Π_n be the projection operator with the respect to the norm $\|\cdot\|_{\pi_n}$. As a consequence of Assumption 1, and for any function J (with $\|J\|_{\pi_n}<\infty$), the projection $\Pi_n J$ is of the form $\Pi_n J=\Phi r$, for a unique choice of r, given by

$$r=(E[\phi(x_n)\phi'(x_n)])^{-1}E[\phi(x_n)J(x_n)].$$

We start by comparing the exact algorithm

$$Q_n = \alpha P \max(g, Q_{n+1})$$

with the idealized algorithm (1) (which does not involve any state sampling) given by

$$\tilde{Q}(\cdot, r_n) = \alpha \Pi_n P \max(g, \tilde{Q}(\cdot, r_{n+1})).$$

The two algorithms are initialized with

$$Q_{N-1} = \alpha P g, \qquad \tilde{Q}(\cdot, r_{N-1}) = \alpha P_{N-1} g.$$

Let

$$\tilde{e}_n = \|\tilde{Q}(\cdot, r_n) - Q_n\|_{\pi_n}$$

which is the approximation error of the idealized algorithm, and let

$$e_n^* = \min_r \|\tilde{Q}(\cdot, r) - Q_n\|_{\pi_n} = \|\Pi_n Q_n - Q_n\|_{\pi_n}$$

which the best possible error under the chosen approximation architecture.

Our first and main result states that if the approximation architecture is capable of delivering a close approximation (i.e., if the e_n^* are small), then the algorithm will succeed (i.e., $\tilde{e}_n$ will be small).

Theorem 1: We have $\tilde{e}_{N-1} = e_{N-1}^*$ and

$$\tilde{e}_n^2 \le \alpha^2 \tilde{e}_{n+1}^2 + (e_n^*)^2, \qquad n = 0, 1, \ldots, N-2.$$

In particular, if $e_n^* \le \epsilon$ for every n, then

$$\tilde{e}_n \le \frac{\epsilon}{\sqrt{2}\sqrt{1-\alpha}}$$

and

$$\tilde{e}_n \le \sqrt{N}\epsilon.$$

Proof: Note that

$$\begin{aligned}\tilde{e}_{N-1} &= \|\alpha \Pi P g - Q_{N-1}\|_{\pi_{N-1}} \\ &= \|\Pi Q_{N-1} - Q_{N-1}\|_{\pi_{N-1}} = e_{N-1}^*.\end{aligned}$$

Using the Pythagorean theorem, we have

$$\begin{aligned}&\|\tilde{Q}(\cdot, r_n) - Q_n\|_{\pi_n}^2 \\ &\quad = \|\tilde{Q}(\cdot, r_n) - \Pi_n Q_n\|_{\pi_n}^2 + \|\Pi_n Q_n - Q_n\|_{\pi_n}^2 \\ &\quad \le \alpha^2 \|\Pi_n P(\max(g, \tilde{Q}(\cdot, r_{n+1})) \\ &\quad - \max(g, Q_{n+1}))\|_{\pi_n}^2 + (e_n^*)^2.\end{aligned}$$

Note that for any function J, we have the following properties (as long as the norms involved are finite):

$$\|\Pi_n J\|_{\pi_n} \le \|J\|_{\pi_n}$$

(this is a general property of projections), and

$$\|PJ\|_{\pi_n} \le \|J\|_{\pi_{n+1}}.$$

To verify the second property, we argue as in [19], [21]. We have, using the definition of P and Jensen's inequality

$$\begin{aligned}\|PJ\|_{\pi_n}^2 &= E[((PJ)(x_n))^2] \\ &= E[E[J(x_{n+1}) \mid x_n]^2] \\ &\le E[E[J^2(x_{n+1}) \mid x_n]] \\ &= E[J^2(x_{n+1})] \\ &= \|J\|_{\pi_{n+1}}^2.\end{aligned}$$

Using the above noted properties, we obtain

$$\begin{aligned}&\|\tilde{Q}(\cdot, r_n) - Q_n\|_{\pi_n}^2 \\ &\quad \le \alpha^2 \|\max(g, \tilde{Q}(\cdot, r_{n+1})) \\ &\quad - \max(g, Q_{n+1})\|_{\pi_{n+1}}^2 + (e_n^*)^2 \\ &\quad \le \alpha^2 \|\tilde{Q}(\cdot, r_{n+1}) - Q_{n+1}\|_{\pi_{n+1}}^2 + (e_n^*)^2 \\ &\quad = \alpha^2 \tilde{e}_{n+1}^2 + (e_n^*)^2.\end{aligned}$$

This establishes the main part of the theorem. The other two inequalities are straightforward corollaries. **Q.E.D.**

It now remains to study the additional error introduced by random sampling of the state space, as opposed to the exact calculation of expectations and projections in the idealized algorithm. Let us first provide a self-contained and precise description of the algorithm to be discussed.

Starting from the given initial state x_0, we simulate m independent trajectories of the process x_n. Let x_n^i be the sample of x_n obtained during the ith simulated trajectory. Thus, the random variables x_n^i, for $i = 1, \ldots, m$, are i.i.d., drawn from the distribution π_n. The algorithm produces an approximation $\tilde{Q}(\cdot, \hat{r}_n) = \Phi \hat{r}_n$ of Q_n, where $\hat{r}_n$ is defined by

$$\begin{aligned}\hat{r}_n = \operatorname*{argmin}_r \sum_{i=1}^m &(\alpha \max(g(x_{n+1}^i), \hat{r}_{n+1}' \phi(x_{n+1}^i)) \\ &- r' \phi(x_n^i))^2.\end{aligned}$$

The approximation error for this algorithm is $\|\tilde{Q}(\cdot, \hat{r}_n) - Q_n\|_{\pi_n}$, which is bounded by

$$\|\tilde{Q}(\cdot, \hat{r}_n) - \tilde{Q}(\cdot, r_n)\|_{\pi_n} + \|\tilde{Q}(\cdot, r_n) - Q_n\|_{\pi_n}$$

where r_n are the parameters in the idealized algorithm. An upper bound for the second term above is provided by Theorem 1. We can therefore concentrate on the first term, which can be bounded as follows:

$$\begin{aligned}\|\tilde{Q}(\cdot, \hat{r}_n) - \tilde{Q}(\cdot, r_n)\|_{\pi_n} &= \|\Phi(\hat{r}_n - r_n)\|_{\pi_n} \\ &\le \|\hat{r}_n - r_n\| \cdot \|\Phi\|_{\pi_n} \le C \|\hat{r}_n - r_n\|\end{aligned}$$

where C is an absolute constant, $\|\cdot\|$ is the standard Euclidean norm, and

$$\|\Phi\|_{\pi_n} = \max_{\|r\|=1} \frac{\|\Phi r\|_{\pi_n}}{\|r\|}.$$

We have the following result, in which we choose our notation in a way that emphasizes the dependence of the sample-based algorithm on the sample size m.

Theorem 2: Let $\hat{r}_n(m)$ (a random variable) be the vector of time n parameters, obtained on the basis of m simulated trajectories. As $m \to \infty$, the sequence $\hat{r}_n(m)$ converges (with probability 1) to the parameter vector r_n obtained by the idealized algorithm.

Proof: Let $J : \Re^d \mapsto \Re$ be the function defined by $J(x) = \max(g(x), \tilde{Q}(x, r_{n+1}))$. Then

$$r_n = \alpha(E[\phi(x_n)\phi'(x_n)])^{-1}E[\phi(x_n)J(x_{n+1})]$$

and we also define

$$\tilde{r}_n(m) = \alpha\left(\sum_{i=1}^{m}\phi(x_n^i)\phi'(x_n^i)\right)^{-1}\sum_{i=1}^{m}\phi(x_n^i)J(x_{n+1}^i).$$

We use the inequality

$$||\hat{r}_n(m) - r_n|| \le ||\hat{r}_n(m) - \tilde{r}_n(m)|| + ||\tilde{r}_n(m) - r_n||.$$

Let $\delta_n(m) = ||\tilde{r}_n(m) - r_n||$, which is seen to converge to zero, almost surely, by the strong law of large numbers.

Next, we consider the term $||\hat{r}_n(m) - \tilde{r}_n(m)||$. We define

$$\hat{J}(x) = \max(g(x), \tilde{Q}(x, \hat{r}_{n+1}(m)))$$

and note that

$$\hat{r}_n(m) = \alpha\left(\sum_{i=1}^{m}\phi(x_n^i)\phi'(x_n^i)\right)^{-1}\sum_{i=1}^{m}\phi(x_n^i)\hat{J}(x_{n+1}^i).$$

Thus,

$$\hat{r}_n(m) - \tilde{r}_n(m) = \alpha\left(\sum_{i=1}^{m}\phi(x_n^i)\phi'(x_n^i)\right)^{-1} \cdot \sum_{i=1}^{m}\phi(x_n^i)(\hat{J}(x_{n+1}^i) - J(x_{n+1}^i)). \quad (5)$$

Let $\beta_n(m) = ||\hat{r}_n(m) - r_n||$. For $n \le N-2$, and any x, we have

$$\begin{aligned}|\hat{J}(x) - J(x)| &= |\max(g(x), \tilde{Q}(x, \hat{r}_{n+1}(m)) \\ &\quad - \max(g(x), \tilde{Q}(x, r_{n+1}))| \\ &\le |\tilde{Q}(x, \hat{r}_{n+1}(m)) - \tilde{Q}(x, r_{n+1})| \\ &\le ||\phi(x)|| \cdot ||\hat{r}_{n+1}(m) - r_{n+1}|| \\ &= \beta_{n+1}(m)||\phi(x)||.\end{aligned}$$

Using this inequality and (5), we obtain

$$||\hat{r}_n(m) - \tilde{r}_n(m)|| \le A_n(m)\beta_{n+1}(m)$$

where $A_n(m)$ are random variables that remain bounded as $m \to \infty$.

To summarize, for $n \le N-2$, we have

$$\beta_n(m) \le \delta_n(m) + A_n(m)\beta_{n+1}(m).$$

For $n = N-1$, a simpler argument shows that $\beta_{N-1}(m)$ converges to zero. (Compare the formulas for r_n and $\hat{r}_{N-1}$, when both J and $\tilde{J}$ are equal to g.) As $m \to \infty$, the sequence $\delta_n(m)$ goes to zero, while $A_n(m)$ remains bounded. It follows that $\beta_n(m)$ converges (almost surely) to zero for any fixed n. **Q.E.D.**

VI. Generalizing Over Time

Until now, we have been using a separate parameter vector r_n for each n, in order to approximate Q_n. In other words, we have aimed at approximating N separate functions, each with domain $\Re^d$. When N is large (e.g., if the time to expiration is substantial and/or a fine time discretization is used), this results in a large number of parameters to be computed. Note, however, that the functions Q_n and Q_{n+1} are often "close." In this section, we exploit this fact to reduce the number of parameters. In particular, we consider generating, as an approximation, a single function over the domain

$$S = \Re^d \times \{0, 1, \ldots, N-1\}.$$

This function is meant to approximate simultaneously every element in the sequence $Q_0, Q_1, \ldots, Q_{N-1}$ of Q-functions.

To enable a concise description of the algorithm, let us first define some abstract notation. For any function $J : \Re^d \mapsto \Re$, we define the operator F by $FJ = \alpha P \max(g, J)$. (As in previous sections, P is the transition operator.) As before, we have $Q_n = FQ_{n+1}$, $n = 0, 1, \ldots, N-1$, with the convention $Q_N = g$. This relation can be rewritten as

$$(Q_0, Q_1, \ldots, Q_{N-1}) = (FQ_1, \ldots, FQ_N)$$

or, more abstractly, as

$$Q = HQ$$

where $Q = (Q_0, Q_1, \ldots, Q_{N-1})$, and H is an operator acting on functions defined on S.

As before, let π_n be the probability measure describing the distribution of the random variable x_n, and consider the norm $||J||_{\pi_n} = E[J^2(x_n)]$, where the expectation is with respect to π_n. Let us define a new norm, on the set of functions with domain S. For any function $Q : S \mapsto \Re$, we let

$$||Q||^2 = \frac{1}{N}\sum_{n=0}^{N-1}||Q(\cdot, n)||_{\pi_n}^2$$

or, in probabilistic terms

$$||Q||^2 = \frac{1}{N}\sum_{n=0}^{N-1}E[Q(x_n, n)^2].$$

The operator H is a contraction with respect to this norm. To see this, note that

$$\begin{aligned}||HQ - H\overline{Q}||^2 &= ||H(Q_0, \ldots, Q_{N-1}) - H(\overline{Q}_0, \ldots, \overline{Q}_{N-1})||^2 \\ &= \frac{1}{N}\sum_{n=0}^{N-1}||FQ_{n+1} - F\overline{Q}_{n+1}||_{\pi_n}^2 \\ &\le \frac{\alpha}{N}\sum_{n=0}^{N-1}||Q_{n+1} - \overline{Q}_{n+1}||_{\pi_{n+1}}^2 \\ &\le \frac{\alpha}{N}\sum_{n=0}^{N-1}||Q_n - \overline{Q}_n||_{\pi_n}^2\end{aligned}$$

$$= \alpha \|Q - \overline{Q}\|^2.$$

We have used here the property (established in Section V that F is a contraction, as well as the convention $Q_N = g = \overline{Q}_N$.

The approximation architecture to be employed uses K feature functions $\phi_k : S \mapsto \Re$ and approximate Q-functions of the form

$$\tilde{Q}_n(x, r) = r'\phi(x, n), \qquad n = 0, 1, \ldots, N-1$$

where $r \in \Re^K$ is a vector of free parameters.

We now define an operator Π that corresponds to the projection (with respect to the norm $\|\cdot\|$) onto the set of functions that can be represented by our approximation architecture. In particular, given a function $Q = (Q_0, Q_1, \ldots, Q_{N-1})$, its projection is of the form $r'\phi(\cdot, \cdot)$, where r is chosen to minimize

$$\sum_{n=0}^{N-1} E[(r'\phi(x_n, n) - Q_n(x))^2].$$

The minimizing r is given by the closed-form formula

$$r = \left(\sum_{n=0}^{N-1} E[\phi(x_n, n)\phi'(x_n, n)] \right)^{-1} \cdot \sum_{n=0}^{N-1} E[\phi(x_n, n) Q_n(x_n)].$$

An exact solution to the pricing problem corresponds to computing the function Q^* (defined on S) which is the unique solution of the equation $HQ^* = Q^*$. Given the approximation architecture, the closest possible approximation to Q^* is given by ΠQ^*. However, it is difficult to compute this function since Q^* itself is unknown. We will therefore settle for the solution to the fixed point equation $Q = \Pi H Q$. This equation has a unique solution because H is a contraction (as shown earlier), and Π is a nonexpansion (this is a generic property of projection operators). One desirable characteristic of this fixed point is that its associated approximation error is within a constant factor from the best possible approximation error $\|\Pi Q^* - Q^*\|$, as we now establish.

Theorem 3: Let κ be the contraction factor of the operator ΠH. Let Q be the unique fixed point of ΠH. Then

$$\|Q - Q^*\| \leq \frac{1}{\sqrt{1 - \kappa^2}} \|\Pi Q^* - Q^*\|.$$

Proof: We have, using the Pythagorean theorem

$$\begin{aligned} \|Q - Q^*\|^2 &= \|Q - \Pi Q^*\|^2 + \|\Pi Q^* - Q^*\|^2 \\ &= \|\Pi H Q - \Pi H Q^*\|^2 + \|\Pi Q^* - Q^*\|^2 \\ &\leq \kappa^2 \|Q - Q^*\|^2 + \|\Pi Q^* - Q^*\|^2 \end{aligned}$$

and the desired conclusion follows. **Q.E.D.**

The fixed point of ΠH can be obtained, in principle, by carrying out the iteration $Q := \Pi H Q$. In more detail, a typical iteration uses an available parameter vector r_k, resulting in an approximate Q function of the form $\tilde{Q}_n(x, r) = r_k'\phi(x, n)$, and calculates a new parameter vector r_{k+1} according to

$$r_{k+1} = \left(\sum_{n=0}^{N-1} E[\phi(x_n, n)\phi'(x_n, n)] \right)^{-1} \cdot \sum_{n=0}^{N-1} E[\phi(x_n, n) \max(g(x_{n+1}), r_k'\phi(x_{n+1}, n))]. \tag{6}$$

An implementable version of this iteration is obtained by simulating a number m of trajectories. Let x_n^i, $i = 1, \ldots, m$, be the sample value of x_n obtained during the ith simulated trajectory. Let $\hat{r}_k$ be the parameter vector after k iterations. We then generate a new parameter vector $\hat{r}_{k+1}$ according to the formula

$$\hat{r}_{k+1} = \left(\sum_{n=0}^{N-1} \sum_{i=1}^{m} \phi(x_n^i, n)\phi'(x_n^i, n) \right)^{-1} \cdot \sum_{n=0}^{N-1} \sum_{i=1}^{m} \phi(x_n^i, n) \max(g(x_{n+1}^i), \hat{r}_k'\phi(x_{n+1}^i, n)). \tag{7}$$

In effect, we are replacing the expectation (with respect to the underlying probability measure) by an expectation with the respect to the empirical measure provided by the simulations. Starting with the same parameter vector r, i.e., if $r_0 = \hat{r}_0$, the strong law of large numbers implies that, as the sample size m increases, the vector $\hat{r}_k$ produced by the kth iteration of the simulation-based algorithm (7) approaches the vector r_k produced by the exact algorithm (6).

We note that there are two variants of the above described algorithm.

1) During each iteration, simulate and use a *new* set of trajectories. In that case, the parameter vector $\hat{r}_k$ evolves as a time-homogeneous Markov process. Eventually, this Markov process reaches steady-state, but the variance of $\hat{r}_k$ remains positive, and $\hat{r}_k$ does not converge to a constant. Let r_∞ be a random variable distributed according to the steady-state distribution of this Markov process. Also, let r^* be the parameter vector associated with the fixed point of ΠH. We conjecture that as the number m of simulated trajectories grows to infinity, the random variable r_∞ converges to r^*, in probability. We do not pursue this issue any further, because the variant we discuss next is more natural and economical.
2) We simulate a number m of trajectories once and for all. These *same* trajectories are used at each iteration of the algorithm (7). Then, the sequence $\hat{r}_k$ is guaranteed to converge. The reason is that the algorithm (7) is identical to the deterministic algorithm $Q := \Pi H Q$ applied to a new problem in which the probability measure associated with the process x_n is replaced by the empirical measure provided by the simulation. The contraction property of ΠH remains true, and $\hat{r}_k$ therefore converges. The limit of the sequence $\hat{r}_k$, denoted by $\hat{r}_\infty$, is of course a random variable, since it is affected

by the randomly simulated trajectories. As m grows to infinity, the empirical measure "converges" to the true measure, which suggests that $\hat{r}_\infty$ converges to r^*, in probability.

VII. Conclusion

We have introduced certain simulation-based methods, of the value iteration type, for pricing complex American-style options. We have provided convergence results and error bounds that establish that such methods are viable, as long as state sampling is carried out by simulating the natural distribution of the underlying state process. This provides theoretical support for the apparent effectiveness of this particular form of state sampling.

References

[1] V. Averbukh, "Pricing American Options Using Monte Carlo Simulation," Ph.D. dissertation, Cornell Univ., Ithaca, NY, 1997.

[2] J. Barraquand and D. Martineau, "Numerical valuation of high dimensional multivariate American securities," *J. Financial Quantitative Anal.*, vol. 30, pp. 383–405, 1995.

[3] R. Bellman and S. Dreyfus, "Functional approximations and dynamic programming," *Math. Tables and Other Aids Comp.*, vol. 13, pp. 247–251, 1959.

[4] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Belmont, MA: Athena Scientific, 1995.

[5] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximation*. Berlin, Germany: Springer-Verlag, 1990.

[6] M. J. Brennan and E. S. Schwartz, "Convertible bonds: Valuation and optimal strategies for call and conversion," *J. Finance*, vol. 32, no. 5, pp. 1699–1715, 1977.

[7] M. Broadie and P. Glasserman, "Pricing American-style securities using simulation," *J. Economic Dyn. Contr.*, vol. 21, pp. 1323–1352, 1997.

[8] ——, *Monte Carlo Methods for Pricing High-Dimensional American Options: An Overview*. New York: Columbia Univ., 1997.

[9] M. Broadie, P. Glasserman, and G. Jain, "Enhanced Monte Carlo estimates for American option prices," *J. Derivatives*, vol. 5, pp. 25–44, 1997.

[10] J. M. Harrison and D. M. Kreps, "Martingales and arbitrage in multiperiod securities markets," *J. Economic Theory*, vol. 20, pp. 381–408, 1979.

[11] J. M. Harrison and S. R. Pliska, "Martingales and stochastic integrals in the theory of continuous trading," *Stochastic Processes and Their Applications*, vol. 11, pp. 261–271, 1981.

[12] D. Heath, R. Jarrow, and A. Morton, "Bond pricing and the term structure of interest rates," *Econometrica*, vol. 60, pp. 77–106, 1992.

[13] I. Karatzas, "On the pricing of American options," *Appl. Math. Optimization*, vol. 17, pp. 37–60, 1988.

[14] F. A. Longstaff and E. S. Shwartz, "Valuing American options by simulation: A simple least-squares approach," *Rev. Financial Studies*, vol. 14, no. 1, pp. 113–147, 2001.

[15] R. C. Merton, "The theory of rational option pricing," *Bell J. Economics Management Sci.*, vol. 4, pp. 141–183, 1973.

[16] S. Raymar and M. Zwecher, "A Monte Carlo valuation of American call options on the maximum of several stocks," *J. Derivatives*, vol. 5, pp. 7–23, 1997.

[17] J. Rust, "Using randomization to break the curse of dimensionality," *Econometrica*, vol. 65, no. 3, pp. 487–516, 1996.

[18] J. A. Tilley, "Valuing American options in a path simulation model," *Trans. Soc. Actuaries*, vol. 45, pp. 83–104, 1993.

[19] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Automat. Contr.*, vol. 42, pp. 674–690, 1997.

[20] ——, "Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives," *IEEE Trans. Automat. Contr.*, vol. 44, no. 10, pp. 1840–1851, Oct. 1999.

[21] B. Van Roy, "Learning and Value Function Approximation in Complex Decision Processes," Ph.D. dissertation, Massachusetts Inst. Technol., Cambridge, 1998.

John N. Tsitsiklis (F'99) was born in Thessaloniki, Greece, in 1958. He received the B.S. degree in mathematics in 1980, and the B.S., M.S., and Ph.D. degrees, in electrical engineering, all from the Massachusetts Institute of Technology (MIT), Cambridge, in 1980, 1981, and 1984, respectively.

During the academic year 1983 to 1984, he was an Acting Assistant Professor of Electrical Engineering at Stanford University, Stanford, CA. Since 1984, he has been with the Massachusetts Institute of Technology, where he is currently Professor of Electrical Engineering and Computer Science. His research interests include systems, optimization, control, and operations research. He is a coauthor of *Parallel and Distributed Computation: Numerical Methods* (Englewood Cliffs, NJ: Prentice-Hall, 1989), *Neuro-Dynamic Programming* (Belmont, MA: Athena Scientific, 1996), and *Introduction to Linear Optimization* (Belmont, MA: Athena Scientific, 1997).

Dr. Tsitsiklis has been a recipient of an IBM Faculty Development Award in 1983, an NSF Presidential Young Investigator Award in 1986, an Outstanding Paper Award by the IEEE Control Systems Society, the MIT Edgerton Faculty Achievement Award in 1989, the Bodossakis Foundation Prize in 1995, and the INFORMS/CSTS prize in 1997. He has served as Acting Co-Director of the MIT Laboratory for Information and Decision Systems (Spring 1996 and 1997). He has also been a visitor with the Department of EECS at the University of California at Berkeley, and the Institute for Computer Science in Iraklion, Greece. He was a plenary speaker at the 1992 IEEE Conference on Decision and Control. He has been an associate editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, A*utomatica*, and A*pplied* M*athematics* L*etters*.

Benjamin Van Roy received the B.S. degree in computer science and engineering and the M.S. and Ph.D. degrees in electrical engineering and computer science, all from the Massachusetts Institute of Technology (MIT), Cambridge, in 1993, 1995, and 1998, respectively.

He is currently an Assistant Professor at Stanford University, Stanford, CA, in the Departments of Electrical Engineering and Management Science and Engineering, with a courtesy appointment in the Department of Computer Science.

Dr. Van Roy has received an NSF CAREER Award, the MIT George M. Sprowls Award for the best doctoral dissertation in computer science, the MIT Morris J. Levin Memorial Award for an outstanding Master's thesis, and the MIT George C. Newton Award for the best undergraduate laboratory project. At Stanford, he has been named Terman Fellow and Morgenthaler Faculty Scholar.