Journal of Computational Finance, Vol.4, No.3, 39-88, Spring 2001.

Pricing American Options: A Comparison of Monte Carlo Simulation Approaches*

Michael C. Fu, Scott B. Laprise, Dilip B. Madan, Yi Su, Rongwen Wu
University of Maryland at College Park

September 1999; revised December 1999, March 2000, April 2000, June 2000

Abstract

A number of Monte Carlo simulation-based approaches have been proposed within the past decade to address the problem of pricing American-style derivatives. The purpose of this paper is to empirically test some of these algorithms on a common set of problems in order to be able to assess the strengths and weaknesses of each approach as a function of the problem characteristics. In addition, we introduce another simulation-based approach that parameterizes the early exercise curve and casts the valuation problem as an optimization problem of maximizing the expected payoff (under the martingale measure) with respect to the associated parameters, the optimization problem carried out using a simultaneous perturbation stochastic approximation (SPSA) algorithm.

Keywords: American options, Monte Carlo simulation, options pricing, stochastic approximation, early exercise.

*This work was supported in part by the National Science Foundation under Grant DMI-9713720, and by the Semiconductor Research Corporation under Grant 97-FJ-491. Earlier versions of this paper were presented at the *2nd Annual Conference on Computational and Quantitative Finance*, New York City, 1999 and the *10th Derivatives Securities Conference*, Boston, 2000.

1 Introduction

In the second edition of his book on financial derivatives, Hull writes (1993, p.363):

> "Monte Carlo simulation can only be used for European-style options."

Numerous papers since then have refuted this claim, and Hull himself in the 4th edition of his book tempers the assertion to, "... cannot easily handle situations where there are early exercise opportunities." (Hull 2000, p.408) Nonetheless, pricing American-style options via Monte Carlo simulation still remains a very challenging problem. The problem lies in the estimation of the early exercise decisions available in American-style derivative contracts. Standard simulation programs are forward algorithms, which means that the paths of state variable are simulated forward in time. Most path-dependent featured options are easily priced through the simulation of sample paths. However, pricing options with "early-exercise" features, such as American options, generally requires a backward algorithm. By working backwards from the maturity date of the option via dynamic programming, the optimal exercise strategy and option price can be estimated. The problem in using simulation to price American-style options results from the inconsistency of applying an inherently forward-based procedure to a problem that requires a backwards procedure to solve. If the usual backwards induction algorithm is applied on a single sample path basis, the resulting future value is derived from perfect foresight, rather than based only on the expectation. Therefore, the estimated optimal early exercise strategy based on a sample path via a backwards algorithm will overestimate the true option value.

As a result of these problems in applying simulation, the primary methods for pricing American-style options are binomial trees and other lattice methods (such as trinomial trees), and finite difference methods to solve the associated boundary value partial differential equations (PDEs). Broadie and Detemple (1996) provide a recent comparison of various (non-simulation-based) existing methods in pricing standard American call and put options written on a single underlying dividend-paying asset. In general (not just for American-style derivatives), the computational speed of these methods is significantly better than that of simulation methods for simple models and contracts. However, the major drawback of these methods is that they can often only handle one or two sources of uncertainty: for more state variables, they become computationally prohibitive, with computation times typically increasing exponentially with the number of state variables.

Since the convergence rate of Monte Carlo methods is generally independent of the number of state variables, it is clear that they become viable as the underlying models (asset prices and volatilities, interest rates) and derivative contracts themselves (defined on path-dependent functions or multiple assets) become more complicated. Our goal is to use a small computational testbed of problems to compare and contrast the strengths and weaknesses of some recently proposed simulation-based algorithms along the various dimensions of these additional complications. For more on the use of Monte Carlo simulation for derivatives pricing, see Boyle (1977), and Boyle, Broadie, and Glasserman (1997); see also Broadie and Detemple (1997b) for a recent review of all numerical methods.

We consider three main classes of algorithms. Since the American option pricing problem can be cast in the framework of a stochastic dynamic programming problem (for discrete time; stochastic control for continuous time), the first class of algorithms is based on the idea of attempting to mimic the basic backwards induction algorithm of stochastic dynamic programming. Tilley (1993) was the first to attempt to apply simulation to American option pricing, using a bundling technique and a backward induction algorithm. His approach is a "single pass" algorithm, in that all simulations are carried out first before the algorithm is applied. In contrast, the approach of Grant, Vora, Weeks (1996, 1997) is sequential in its use of simulation, proceeding inductively backwards to

approximate the exercise boundary at each early exercise point, finally estimating the price in a forward simulation based on the obtained boundaries. In it original form, a "brute-force" approach (enumeration) is used to find the exercise boundary, the set of critical points at which the immediate exercise payoff equals the expected holding value. Ibanez and Zapatero (1999) use a fixed-point algorithm to find these critical points, whereas we use a secant method. The second class of algorithms is based on the approach of explicitly characterizing the early exercise curve through a parameterization, and then using gradient-based simulation to maximize the expected payoff with respect to the parameters. Fu and Hu (1995) treat a simple American call option with discrete dividends by casting the pricing problem as an optimization problem of maximizing the expected payoff of the contract (under the risk-neutral measure) with respect to the early exercise thresholds. They use direct gradient estimates that could be derived for the particular problem. Here, we generalize the approach to arbitrary parameterized early exercise boundaries that are optimized with respect to the parameters through the use of an efficient algorithm based on the simultaneous perturbation stochastic approximation (SPSA) approach of Spall (1992), which was used by Fu and Hill (1997) for discrete-event systems. Bossaerts (1989) proposed some similar ideas, but as far as we know, there was never any implementation of particular computational algorithms. The third approach is based on finding efficient upper and lower bounds from simulated paths. Broadie and Glasserman (1997a,1998) propose two algorithms – one based on a nonrecombining tree and another based on a stochastic mesh – which provide high and low estimates for the option price that converge asymptotically to the true price. These two algorithms and the Tilley algorithm are also discussed in greater detail in Broadie and Glasserman (1997b).

Improvements on the basic idea of Tilley include Carriere (1995), who uses a sequential nonlinear regression algorithm on spline functions to approximate conditional expectations for the holding values at early exercise points, and Longstaff and Schwartz (1999), who use least-squares regression on polynomials to approximate the early exercise boundary. Another approach not considered here is state aggregation, where a subset of the full state definition is used to make early exercise decisions. This can result in significant computational savings for high-dimensional problems, but the accuracy depends on how successfully the aggregated state captures the important features of the complete state, in terms of the early exercise decision. Barraquand and Martineau (1995) apply such a state aggregation technique to partition the space of underlying assets into a tractable number of one-dimensional cells via a stratification function based on the payoff function, e.g., for an American max option, only the maximum is retained as the aggregate state. To improve upon this approximation, Raymar and Zwecher (1997) suggest using a two-factor aggregation for the American max option, partitioning with respect to the maximum price and a second factor such as the second highest price.

The rest of this paper is organized as follows. The testbed cases are first described. Then, each of the three approaches is described in more detail, with two explicit algorithms given for each approach. The algorithms are then compared on the testbed problems, in terms of implementation ease, memory, and accuracy versus computational expense. Summary and conclusions based on the experiments are then presented.

2 Computational Testbed

The testbed consists of American call options on a single and multiple underlying assets, an American-Asian call option on a single stock, and an American put option on a single stock modeled by the Merton jump-diffusion model. Many of these test cases have been used by Broadie and Glasserman (1997a,1998), Fu and Hu (1995), and Grant, Vora, and Weeks (1996,1997) in testing

their own proposed algorithms. Throughout, the expiration date will be denoted by T, the strike price by K, the interest rate by r, the volatility by σ, and the stock price at time t by S_t (starting price S_0), with an argument $S_t(k)$ indicating a particular sample path k and superscripts S_t^i when there are multiple stocks. Throughout the interest rate and volatility are assumed constant, but these can easily be made stochastic for all the procedures with little difficulty, a usual advantage of Monte Carlo methods. Under the scenario, the price of an American-style option can be written as the solution to the following optimal stopping problem:

$$\sup_{\tau} E^Q \left[e^{-r\tau} L_\tau(\{S_t : t \in [0,\tau]\}) \right], \tag{1}$$

where Q denotes the appropriate risk-neutral (martingale) measure, $L_t(\cdot)$ represents the payoff at time t, which itself could be path dependent (as in an Asian option), and the supremum is over all stopping times $\tau \leq T$. All cases will have a finite number of discrete exercise opportunities, at points $\{t_0 = 0, t_1, ..., t_N = T\}$, where $N+1$ gives the total number of exercise opportunities. Therefore, the supremum becomes a maximization operator. Richardson extrapolation can be used for extension to continuous early exercise opportunities (Geske and Johnson 1984).

Testbed Set 1: Call Option with Discrete Dividends

Testbed Set 1 is an American call option on a single stock that distributes dividend D_j at time $t_j = \sum_{i=1}^{j} \tau_i \ \ (\tau_i > 0)$, $j = 1, \ldots, N$, where τ_i is the time between the distribution of dividend $i-1$ and i, and N is the number of dividends distributed during the lifetime of the call contract, the last dividend distributed on the expiration date. Thus,

$$L_t = (S_t - K)^+. \tag{2}$$

The dividend amounts $\{D_j\}$ can be deterministic or stochastic, but following standard models, we assume that the stock price drops after ex-dividend by the amount of the dividend, i.e.,

$$S_{t_j^+} = S_{t_j^-} - D_j, \quad j = 1, \ldots, N.$$

The stock price model adopted here follows that of Stoll and Whaley (1993) and others, in considering the dynamics of the stock price net of the present value of escrowed dividends, given by

$$\tilde{S}_t = h(Z; \tilde{S}_0, t, \mu, \sigma),$$

for some random vector Z independent of the parameters. In particular,

$$\tilde{S}_{t_j} = h(Z_j; \tilde{S}_{t_{j-1}}, \tau_j, \mu, \sigma), \qquad j = 1, ..., N,$$

with independent $Z_j \sim f_j$, probability density functions. The actual stock price process is then given by

$$S_t = \tilde{S}_t + \sum_{i=j+1}^{N} D_i e^{-r(t_i - t)}, \quad \text{for } t_j \leq t < t_{j+1}, \quad j = 0, 1, ..., N.$$

At exercise decision points,

$$S_{t_j^-} = \tilde{S}_{t_j} + \tilde{D}_j, \quad \tilde{D}_j = \sum_{i=j}^{N} D_i \exp\left(-r \sum_{k=j+1}^{i} \tau_k \right), \quad j = 1, ..., N, \tag{3}$$

where $\tilde{D}_j$ represents the net present value (at t_j^-) of all future dividends $D_j, ..., D_N$. Specifically, we will take geometric Brownian motion

$$dS_t = S_t[\mu dt + \sigma dZ_t],$$

which under the risk-neutral measure $\mu = r$ leads to the lognormal distribution of stock prices:

$$f_j(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}, \quad h(Z_j; S, t, r, \sigma) = Se^{(r-\sigma^2/2)t+\sigma\sqrt{t}Z_j}, \tag{4}$$

i.e., Z_j are i.i.d. standard $N(0,1)$ normal r.v.s.

Although the American feature allows exercise of the option at any time up to and including the expiration date, under the assumption of a frictionless market, the call option should only be exercised – if at all – right before an ex-dividend date or at the expiration date. Thus, the potential early exercise dates are the ex-dividend dates $\{t_j, j = 1, ..., N\}$.

Testbed Set 2: Call Option with Continuous Dividends

Testbed Set 2 is a call option on a single stock (payoff again given by (2)) that pays continuous dividends at a rate δ, but early exercise is restricted to the discrete points $\{t_j, j = 0, 1, ..., N\}$. We assume that the stock price net of the present value of escrowed dividends changes continuously according to

$$S_{t_j} = S_{t_j^-} = S_{t_j^+} = h(Z; S_{t_{j-1}}, \tau_j, \mu - \delta, \sigma), \quad j = 1, ..., N.$$

Note that this stock model incorporates the dividends directly into the stock price dynamics. In particular, we will again assume that the risk-neutral version of the underlying stock price process follows geometric Brownian motion:

$$dS_t = S_t[(r - \delta)dt + \sigma dZ_t], \tag{5}$$

so that simulation would use the expressions given by (4), with r simply replaced by $r - \delta$. We note that in this setting, McDonald and Schroder (1998) give a relationship between the put and call option values.

Testbed Set 3: Max Option Multiple Underlying Assets

Testbed Set 3 is a call option on the maximum of n stocks ($n = 2, 5$), i.e., the payoff function upon exercise at time t is

$$L_t = (\max_{i=1,...,n} S_t^i - K)^+, \tag{6}$$

where again early exercise is restricted to the discrete points $\{t_j, j = 0, 1, ..., N\}$. The individual stock prices are assumed to follow the correlated geometric Brownian motion processes

$$dS_t^i = S_t^i[(r - \delta_i)dt + \sigma_i dZ_t^i],$$

where Z_t^i is a standard Brownian motion process and the instantaneous correlation of Z^i and Z^j is ρ_{ij}. In our numerical examples, we follow Broadie and Glasserman (1997ab) by taking $\delta_i = \delta$ and $\rho_{ij} = \rho$ for all $i, j = 1, ..., n$ and $i \neq j$. This type of option was also considered by Broadie and Detemple (1997a) and Raymar and Zwecher (1997).

The first two approaches to be discussed require some characterization of the early exercise boundary. For n non-identical stocks, one intuitively appealing boundary that is consistent with

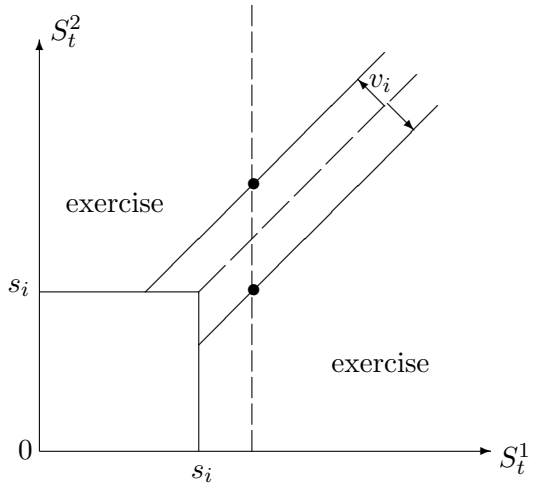

Figure 1: Early exercise boundary for option of maximum of 2 identical assets.

the analytical findings of Broadie and Detemple (1997a) is to set a threshold level for each individual stock plus another threshold level for each pairwise absolute (or squared) difference. This gives $n + \begin{pmatrix} n \\ 2 \end{pmatrix}$ threshold levels to set. By symmetry, when the stocks are assumed to follow identical (but not necessarily independent) price processes, the thresholds should be the same, so the number of thresholds is reduced to 2, and the pairwise differences can be handled as an aggregated sum of squares of differences for ease of implementation.

Specifically, one could specify the hold region at date t_i as a hypercube appended with a (hyper)cylinder as follows:

$$\left\{ (S_{t_i}^1, S_{t_i}^2, ..., S_{t_i}^n) : \max_{j=1,...,n} S_{t_i}^j < s_i, \text{ or } \sum_{j,k=1,...,n} v_i^{j,k} (S_{t_i}^k - S_{t_i}^j)^2 < v_i \right\},$$

where $\{s_i, v_i, v_i^{j,k}\}$ are the parameters characterizing the region. If $S^1, S^2, ..., S^n$ are assumed to follow the same distribution, though possibly correlated with a common correlation coefficient (the particular cases we will consider in our numerical experiments), we define the following metric on the vector of stock prices $\mathbf{S}_t \triangleq (S_t^1, S_t^2, ..., S_t^n)$:

$$||\mathbf{S}_t|| = \sum_{j,k:\ j<k\leq n} (S_t^j - S_t^k)^2, \tag{7}$$

and specify the hold region by (see Figure 1)

$$\{(S_{t_i}^1, S_{t_i}^2, ..., S_{t_i}^n) : \max_{j=1,...,n} S_{t_i}^j < s_i, \text{ or } ||\mathbf{S}_{t_i}|| < v_i\}, \tag{8}$$

which gives a two-parameter model $\{s_i, v_i\}$ different from the two-factor model of Raymar and Zwecher (1997), who use the maximum and the second highest stock value or the median. Figure 1 displays the early exercise boundary for an option on the maximum of two identical underlying assets. The dashed vertical line in Figure 1 indicates that for a given value of $S_t^1 > s_i$, there are three distinct regions based on the value of S_t^2, going from exercise to hold to exercise.

Testbed Set 4: Asian Option

Testbed Set 4 is an American-Asian call option with payoff defined by

$$L_t = (\bar{S}_t - K)^+, \quad \bar{S}_t = \frac{1}{t - t' + 1} \sum_{i=t'}^{t} S_i, \tag{9}$$

where $\bar{S}_t$ is the discrete arithmetic average of the stock prices from a pre-specified date t' up to the exercise time t. Although it might be optimal to early exercise this option at any time, we again restrict the early exercise opportunities to the discrete points $\{t_j, j = 1, ..., N\}$. The underlying stock price process follows geometric Brownian motion with continuous dividends according to Equation (5), as in the previous two cases.

Again, the first two approaches to be discussed require some characterization of the early exercise boundary, which here would depend on the joint state variables $(S_t, \bar{S}_t)$. Following Grant, Vora, and Weeks (1997), we will assume a piecewise linear boundary that is characterized by two parameters at each exercise point.

Testbed Set 5: Put Option on a Jump-Diffusion Asset

The last test case, Testbed Set 5, is a put option defined on a single underlying stock, i.e.,

$$L_t = (K - S_t)^+. \tag{10}$$

This time the stock price process is assumed to follow a jump-diffusion model:

$$S_{t+\tau} = S_t e^{(r - \sigma^2/2)\tau + \sigma\sqrt{\tau} Z_0 + \sum_{i=1}^{N(\tau)} (\delta Z_i - \delta^2/2)}, \tag{11}$$

where $Z_i \sim N(0, 1)$ iid, $N(\tau) \sim Poisson(\lambda\tau)$, and the jump sizes are i.i.d. lognormally distributed $LN(\mu_i, \sigma_i^2)$, with $\mu_i = -\delta^2/2$ and $\sigma_i = \delta$ $(\delta > 0)$. (Note: in Grant, Vora, Weeks 1996, there are typographical errors in the specification given on p.222, Equation (6), with an inadvertent $\sqrt{\tau}$ in the jump process portion and omission of the negative $\delta^2/2$ term.)

3 Description of the Algorithms

3.1 Dynamic Programming

The approaches considered here are the bundling method of Tilley (1993) and the sequential method of Grant, Vora, and Weeks (1996, 1997). As mentioned earlier, both approaches mimic the backwards induction solution method of stochastic dynamic programming. The implemented algorithms will be indicated by "DP1" and "DP2" for the adaptation of Tilley (1993) and Grant, Vora, and Weeks (1996, 1997), respectively.

3.1.1 Bundling Algorithm

In Tilley (1993), the bundling algorithm is clearly stated and easy to implement for one-dimensional problems. The algorithm computes an estimate for the option's holding value by using backwards induction and a bundling technique. At time t, paths whose stock prices are similar can be grouped together to obtain an estimate of the one period ahead option value. Proceeding recursively from the option's expiration date, an optimal exercise policy can be estimated.

We first describe the one-dimensional algorithm, followed by the higher-dimensional extensions. In place of the double indexing notation used in Tilley (1993), where "$A(k, t)$" represented the value of some variable A at time t of path k, we will use "$A_t(k)$", as specified earlier for the stock price process.

One-Dimensional Algorithm

For notational convenience and without loss of generality, the time points will be indexed $0, 1, ..., N$. The discount factor from time t to time $t+1$ of path k is denoted $d_t(k)$, with the discount factor from time 0 to time t denoted by $D_t(k)$, where $D_t(k) = \prod_{i=0}^{t-1} d_i(k)$. Let X_t denote the exercise (strike) price at time t, $t = 0, 1, ..., N$, i.e., in general the strike prices may depend on the date of exercise. In all of our testbed cases, the strike price is fixed: $K = X_i \ \forall i$.

The simulation procedure entails the random generation and storing of a finite sample of R stock price paths from time $t = 0$ to time $t = T$. The kth path of stock prices in the simulation sample is represented by the sequence $\{S_0, S_1(k), \ldots, S_N(k)\}$. The backwards induction algorithm begins at the last early exercise date t_{N-1}. At epoch $t \in \{t_0, t_1, \ldots, t_{N-1}\}$, we proceed inductively as follows:

ALGORITHM DP1

Step 1: Reorder the stock price paths by the stock price at time t: from highest to lowest for a put option, and lowest to highest for a call option. Partition the ordered paths into Q distinct bundles of P paths each ($QP = R$): the first P paths in the first bundle, the second P paths in the second bundle, etc. Let $B_t(k)$ be the set of paths in the bundle containing k at epoch t.

Step 2: For each path k, compute $I_t(k)$, the intrinsic value of the option:

$$I_t(k) = \begin{cases} (S_t(k) - X_t)^+ & \text{for a call option;} \\ (X_t - S_t(k))^+ & \text{for a put option.} \end{cases}$$

Step 3: Compute the option's holding value, $H_t(k)$, defined as the present value of the expected one period ahead option value $V_t(k)$:

$$H_t(k) = \frac{d_t(k)}{P} \sum_{\forall\, j \,\in B_t(k)} V_{t+1}(j),$$

where $V_t(k)$ is defined in Step 7 below; $V_N(k) = I_N(k) \ \forall \ k$.

Step 4: For each path k, define a "tentative" exercise-or-hold indicator variable $x_t(k)$:

$$x_t(k) = \begin{cases} 1 & \text{if } I_t(k) > H_t(k) \quad \text{Exercise option;} \\ 0 & \text{if } I_t(k) \leq H_t(k) \quad \text{Hold option,} \end{cases}$$

Step 5: Examine the sequence $\{x_t(k) : k = 1, \ldots, R\}$. Define the "transition zone" between hold and exercise as the sequence of 0's and 1's that begins with the first 1 and ends with the last 0. Determine a sharp boundary between the hold and exercise decisions as the start of the first string of 1's whose length exceeds that of every subsequent string of 0's. The path index of this leading 1 is denoted k_t^*.

Step 6: Define a "new" exercise-or-hold indicator variable, $y_t(k)$, that incorporates the sharp boundary:

$$y_t(k) = \begin{cases} 1 & \text{for } k \geq k_t^* \quad \text{Exercise option;} \\ 0 & \text{for } k < k_t^* \quad \text{Hold option.} \end{cases}$$

Step 7: For each path k, define the current value, $V_t(k)$, of the option:

$$V_t(k) = \begin{cases} I_t(k) & \text{if } y_t(k) = 1; \\ H_t(k) & \text{if } y_t(k) = 0. \end{cases}$$

Step 8: Let $t = t - 1$ and return to Step 1.

After the algorithm has been processed backwards to epoch 0, define $z_t(k)$, the exercise-or-hold indicator variable:

$$z_t(k) = \begin{cases} 1 & \text{if } y_t(k) = 1 \text{ and } y_s(k) = 0 \ \forall \ s < t; \\ 0 & \text{otherwise.} \end{cases}$$

Note: For each path k, either the option is never exercised ($z_t(k) = 0 \ \forall \ t$), or the option is exercised at epoch t^* ($z_{t^*}(k) = 1$ and $z_t(k) = 0 \ \forall \ t \neq t^*$).

The option price estimator is defined by:

$$\frac{1}{R} \sum_{k=1}^{R} \sum_{t=1}^{N} z_t(k) D_t(k) I_t(k). \tag{12}$$

Imprecision in the estimator given by (12) is a result of R (thus P and Q) being finite. This imprecision arises from two factors: (1) the continuous distribution of stock prices at each epoch is not sampled finely enough; and (2) the mathematical expectation in Step 3 above is approximated by an average over a finite number of paths. Imprecision from the first type can be reduced by increasing Q, the number of bundles; and imprecision of the second type can similarly be reduced by increasing P, the number of paths per bundle. However, for a fixed R, Q and P are inversely related, implying a tradeoff between the first and second types of imprecision.

Extension to Higher Dimensions

Tilley (1993) does not include an extension of his algorithm to higher dimensional options. In fact, this extension is not obvious, as noted in numerous papers (e.g., Broadie and Glasserman 1997a, p.1326, 1997b, p.17; Grant, Vora, and Weeks 1997, p.591; Clewlow and Strickland 1998, p.174). There are two main considerations: the bundling procedure and the sorting procedure. We propose an extension of the one-dimensional bundling algorithm to higher dimensional options; in particular, options on the maximum of multiple assets and American-Asian options. Many of the procedures we adopt are straightforward adaptations of the one dimensional algorithm. However, those which are unique to this higher dimensional problem are detailed below.

Extension to Max Option on Multiple Underlying Assets

Simulation: The simulation procedure involves the random generation of a finite sample of R stock price path groupings. Each grouping consists of n correlated stock price paths, where n is the number of stocks considered. The k^{th} path grouping of stock prices in the simulation sample is represented by the sequence $\{\mathbf{S}_0, \mathbf{S}_1(k), \ldots, \mathbf{S}_N(k)\}$, where $\mathbf{S}_i(k) = (S_i^1(k), \ldots, S_i^n(k))$ and $S_i^j(k)$ is the stock price of the j^{th} stock of path grouping k at epoch i ($1 \leq j \leq n$). Note: $\mathbf{S}_0 = (S_0^1, \ldots, S_0^n)$.

Step 1:
(i) At epoch t, compute the average stock price over all R path groupings of each of the n stocks. Reorder the path groupings by the stock price of the stock with the highest average. Partition the path groupings into Q^1 distinct bundles of P^1 path groupings each ($Q^1 P^1 = R$).
(ii) Consider the first of the Q^1 bundles. Compute the average stock price over all P^1 path groupings of each of the n stocks, not including the stock which was used to sort in the initial Step (i). Reorder the path groupings by the stock price of the stock with the highest average. Partition

the path groupings into Q^2 distint bundles of P^2 path groupings each ($Q^2P^2 = P^1$). Repeat for each of the Q^1 bundles.
(iii) Consider the first of the Q^1Q^2 bundles. Compute the average stock price over all P^2 path groupings of each of the n stocks, not including the two stocks which were sorted on previously. Reorder the path groupings by the stock price of the stock with the highest average. Partition the path groupings into Q^3 distinct bundles of P^3 path groupings each ($Q^3P^3 = P^2$). Repeat for each of the Q^1Q^2 bundles, etc.

After sorting over all n stocks, we have $Q = \prod_{j=1}^{n} Q^j$ distinct bundles of $P = P^n$ paths each ($QP = R$). Define $B_t(k)$ to be the set of path groupings in the bundle containing k at epoch t. Note: the above detailed ordering is done from highest to lowest for a put option, and from lowest to highest for a call option.

Step 2: For each path grouping k, compute $I_t(k)$, the intrinsic value of the option, defined appropriately in the following:

$$I_t(k) = \begin{cases} (\max_i S_t^i(k) - X_t)^+ & \text{for a call option;} \\ (X_t - \max_i S_t^i(k))^+ & \text{for a put option.} \end{cases}$$

Step 4:
(i) For those path groupings k "in the money" at time t, i.e., $I_t(k) > 0$, compute $||\mathbf{S}_t(k)||$, the metric given by (7) for path grouping k. Order these path groupings with respect to this measure from highest to lowest. Recombine all the path groupings by attaching this ordering to the end of the "out of the money" path groupings.
(ii) For each path grouping k, define a "tentative" exercise-or-hold indicator variable $x_t(k)$:

$$x_t(k) = \begin{cases} 1 & \text{if } I_t(k) > H_t(k) \quad \text{Exercise option;} \\ 0 & \text{if } I_t(k) \leq H_t(k) \quad \text{Hold option.} \end{cases}$$

All other steps not mentioned proceed exactly as in the one-dimensional algorithm.

Computational time in the above multi-dimensional algorithm can be reduced by a simplification of Step 1. In particular, sorting can be done on the n stocks in a predetermined order rather than with respect to average stock prices. Also, it can be argued that sorting on less than the n stocks is sufficient, especially if sorting is done with respect to the average stock prices. Regardless, in any sorting procedure, the choices of $\{Q^1, Q^2, \ldots\}$ and $\{P^1, P^2, \ldots\}$, the size and number of the bundles, is quite flexible.

Application to the American-Asian Option

The above multidimensional algorithm can be easily applied to the American-Asian option on one stock. In this setting, with $S(k) = \{S_0, S_1(k), \ldots, S_N(k)\}$ representing the stock price path of the underlying asset, let $\bar{S}(k) = \{\bar{S}_0(k), \bar{S}_1(k), \ldots, \bar{S}_N(k)\}$ represent the "path" (sequence) of the stock price averages. The above multi-dimensional algorithm is now applied for n=2, with the following modifications:

(1) In Step 1, the stock path groupings are first sorted with respect to the average stock path, $\{\bar{S}(k)\}$, and secondly with respect to the stock price path, $\{S(k)\}$.
(2) $I_t(k)$ is defined by

$$I_t(k) = \begin{cases} (\bar{S}_t(k) - X_t)^+ & \text{for a call option;} \\ (X_t - \bar{S}_t(k))^+ & \text{for a put option.} \end{cases}$$

(3) The procedure for determining the sequencing of $x_t(k)$ is the same as that in the one-dimensional algorithm.

3.1.2 Sequential Dynamic Programming Algorithm

The sequential approach of Grant, Vora, and Weeks (1996, 1997) is relatively straightforward in principle, attempting to characterize the exercise boundary at each potential early exercise point through a procedure that closely mimics the backwards induction technique of dynamic programming, and then simulating forward to estimate the option price. The exercise boundaries in their examples are characterized by *critical prices (thresholds)* that completely determine the option holder's early exercise strategy.

We apply this approach to three types of American options. First, we consider the single asset American option and follow the estimation procedure outlined in Grant, Vora and Weeks (1996). Second, we consider the American-Asian option as described in Grant, Vora, and Weeks (1997). While this procedure is generally straightforward, the paper lacks the details to make implementation obvious and certain assumptions had to be made. Third, we extend the American-Asian technique and consider higher dimensional options; in particular, options on the maximum of multiple assets.

One-Dimensional Algorithm

We consider an American call on a single stock; the procedure for a put is identical except for the payoff function. As in the bundling algorithm, we assume the exercise opportunities occur at $t = 0, 1, \ldots, N$, with a fixed time span, τ, between exercise dates (τ is written as a fraction of a year). Let $P_t(S)$ denote the (conditional) expected option payoff at time t, given the asset price S. The option value is then given by $P_0(S_0)$. For this option, the optimal exercise boundary at each point t_i is simply a threshold $s_i^* (\geq K)$, called a *critical price*, such that the option is exercised if $S_i > s_i^*$.

At the expiration date T, where the option payoff is $P_N(S_N) = (S_N - K)^+$, the critical price is $s_N^* = K$. Beginning at the last early exercise date t_{N-1}, we use a backwards induction algorithm to estimate the critical prices at the early exercise dates. At date t_{N-1}, the value of the call is given by:

$$P_{N-1}(S_{N-1}) = \begin{cases} S_{N-1} - K & \text{if } S_{N-1} > s_{N-1}^*; \\ e^{-r\tau} E[P_N | S_{N-1}] & \text{if } S_{N-1} \leq s_{N-1}^*. \end{cases} \tag{13}$$

Thus, the critical price at time t, s_{N-1}^*, is the price at which the two quantities in (13) are equal, i.e., $s_{N-1}^* - K = e^{-r\tau} E[P_N | S_{N-1} = s_{N-1}^*]$. In other words, we need to approximate the zero of the function $f(s) = s - K - e^{-r\tau} E[P_N | S_{N-1} = s]$, where $E[P_N | S_{N-1} = s]$ is estimated by simulating $\{S_N^i\}_{i=1}^n$ conditional on $S_{N-1} = s$. Using some root-finding algorithm such as the bisection or secant method, we can bracket the root of $f(s)$ to some predetermined tolerance and then interpolate linearly to obtain an estimate for the root.

Proceeding recursively, obtaining an estimate for s_i^* for $i < N-1$ follows an analogous procedure, based upon estimates for $\{s_{i+1}^*, \ldots, s_{N-1}^*, s_N^* = K\}$ already obtained. The value of the call at time i with stock price S_i is given by:

$$P_i(S_i) = \begin{cases} S_i - K & \text{if } S_i > s_i^*; \\ e^{-r\tau} E[P_{i+1} | S_i] & \text{if } S_i \leq s_i^*; \end{cases}$$

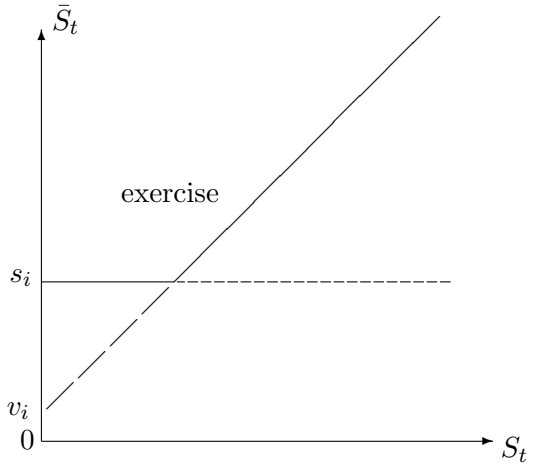

Figure 2: Early exercise boundary for Asian option.

where $s_i^* - K = e^{-r\tau}E[P_{i+1}|S_i = s_i^*]$. In order to find s_i^*, we must estimate $E[P_{i+1}|S_i = s]$ by simulating $\{S_j, j = i+1, ...\}$ until $S_j > s_j^*$ or expiration, giving the explicit estimator:

$$e^{-r(N-(i+1))\tau}(S_N - K)^+ \mathbf{1}\left\{\bigcap_{j=i+1}^{N-1} S_j \le s_j^*\right\} + e^{-r(k-(i+1))\tau}(S_k - K)\mathbf{1}\left\{\bigcap_{j=i+1}^{k-1} S_j \le s_j^*, S_k > s_k^*\right\}, \quad (14)$$

where $\mathbf{1}\{\cdot\}$ is the indicator function, with the first condition indicating no early exercise and the second condition indicating early exercise at time k.

Once estimates for the critical prices at all exercise points are obtained, the value of the option is estimated through a simulation initiated at time 0.

American-Asian Algorithm

For the American-Asian option, we follow the algorithm as detailed in Grant, Vora, Weeks (1997). Here, the early exercise boundary must be based on both S_t and $\bar{S}_t$. As it turns out, there is a critical average price $\bar{S}_t^*$ for *each* stock price S_t. Therefore, the early exercise conditions are expressed as a *locus* of critical prices, $\bar{S}_t^*(S_t)$. At each t, to estimate this locus of critical prices, we first select a representative sample of possible stock prices covering a reasonable range. For each S_t in this sample, we estimate the critical average price $\bar{S}_t^*(S_t)$ in an identical manner as the one-dimensional algorithm. Notice that since the representative sample of stock prices can never be exhaustive, linear interpolation must be employed to find the critical average price at every possible stock price.

To simplify the identification of threshold levels, a linear approximation of the exercise boundary is introduced. For the call option considered, the hold region is taken as follows (see Figure 2):

$$\left\{\bar{S}_t < s_i \text{ or } \bar{S}_t - S_t < v_i\right\}, \quad (15)$$

so the exercise boundary at t_i is given by the piecewise linear function (indicated by the solid line in Figure 2, with $\bar{S}_t$ a function of S_t):

$$\phi_i(x) = \begin{cases} s_i & \text{if } x \le s_i - v_i; \\ v_i + x & \text{if } x > s_i - v_i; \end{cases} \quad (16)$$

and the problem reduces to finding at each of the early exercise dates t_i the optimal values of the parameters (s_i, v_i) that maximize the (continuation) value of the option.

Again, we use backwards induction to estimate the set of optimal parameters. At early exercise point t_i, we already have estimates for the optimal parameters at all of the subsequent early exercise dates, $\{(s_{i+1}, v_{i+1}), ..., (s_{N-1}, v_{N-1})\}$, as well as an obvious exercise policy at the expiration date. By the properties of a call option, the optimization parameters should be monotone with respect to the time until the expiration date. With this in mind, we first prepare a parameter grid (s_i, v_i) with the smallest value of s_i and v_i equal to s^*_{i+1} and v^*_{i+1}, respectively. The range of the grid should reasonably cover the "expected" optimal parameters, and the size of the grid is flexible: in general, more parameters will lead to a more accurate optimization. We next prepare a price grid $(S_t, \bar{S}_t)$ that approximately covers the possible threshold regions (a function of the parameter grid). We choose a pair of parameters from the parameter grid and use this pair for the exercise threshold at time t. Then following this rule and the previously obtained decision rules at the subsequent parameters, we use simulation from each of the pairs of the price grid to calculate an average value for the option. Repeating this procedure for each of the pairs in the parameter grid, we choose (s^*_i, v^*_i) to be the pair that produces the highest (continuation) option value.

Once optimal parameters have been estimated at each early exercise date, we estimate the value of the option through a simulation initiated at time 0. Early exercise occurs on the first date such that the stock price/stock average pair falls in the exercise region.

Extension to Higher Dimensions

We next extend the algorithm by applying it to higher dimensions. In general, this would require specification of complicated exercise/hold boundaries at each early exercise point. For the cases considered here, the American call option on the maximum of n assets, characterization of the early exercise boundary was already discussed; in particular, since all of the stocks are identically distributed with a common correlation coefficient, the region specified by (8) will be used, with exercise/hold decisions based on the two quantities $\max_i S^i_t$ and $||\mathbf{S}_t||$, the latter metric defined by (7). With this dimensional reduction, the optimization of the parameters (s_i, v_i) at each early exercise epoch and the subsequent valuation of the option price otherwise follows the procedure for the American-Asian case. Note, however, that in contrast to the Asian case, where there is a single exercise threshold for $\bar{S}_t$ for each value of S_t (see Figure 2), for the max-option on two assets, there are *two* critical values for S^2_t for each value of $S^1_t > s_i$, as shown by the dashed vertical line indicated with two bullets in Figure 1.

3.2 Parameterizing the Exercise Boundary

The main idea of this approach is to parameterize the exercise boundary, and then maximize the expected discounted payoff with respect to the parameters. The difference between this approach and the previous approach is that no dynamic programming is involved, i.e., the procedure simultaneously optimizes all parameters by iteration, instead of sequentially by backwards recursion. A similar approach was applied to American option pricing problems in an analytical optimization framework by Welch and Chen (1991). Here, the optimization is carried out using Monte Carlo simulation through an iterative algorithm called stochastic approximation (refer to Kushner and Yin 1997; see also Fu 1994 for a review of techniques for simulation optimization), which we now briefly describe.

Suppose the optimal stopping problem given by Equation (1) has been transformed to the

following parametric optimization problem:

$$\max_{\theta \in \Theta} E[L(\theta, \omega)], \tag{17}$$

where $\theta \in \Theta \subset R^p$ is the p-dimensional vector of interest, $L(\theta, \omega)$ is the (sample) objective function of interest (in our application, the payoff of an American option), Θ a compact set in R^p, and ω an element in the probability space of interest, e.g., a sample path in simulation. Stochastic approximation attempts to find the solution to (17) by mimicking steepest-decent algorithms from the deterministic domain of non-linear programming using the following iterative search scheme:

$$\theta_{j+1} = \Pi_\Theta(\theta_j + a_j \hat{g}_j), \tag{18}$$

where $\theta_j = ((\theta_j)_1, ..., (\theta_j)_p)$ represents the jth iterate, $\hat{g}_j$ represents an estimate of the gradient of $E[L]$ with respect to the parameter vector θ at θ_j, $\{a_j\}$ is a positive sequence of numbers converging to 0, and Π_Θ denotes a projection on Θ. Key to the success of the method is the availability of a gradient estimate. Direct gradients can be obtained on a case-by-case basis using techniques such as perturbation analysis (PA) (see Fu and Hu 1995, 1997; Glasserman 1991; Broadie and Glasserman 1996), and their use generally gives the best convergence rate for the SA algorithm given by (18). When this is not practical, simultaneous perturbations stochastic approximation (SPSA) (see Spall 1992) —- which uses only estimates of the objective function — may be generally and quite easily applied. Here, we apply PA to a couple of the cases and SPSA to all of the cases. The implemented algorithms will be indicated by "PASA" for the stochastic approximation algorithm based on PA gradient estimates, and "SPSA" for the SPSA algorithm.

3.2.1 SPSA Algorithm

We present and contrast the usual finite difference estimators with the simultaneous perturbation estimator. Let e_i denote the unit vector in the ith direction of $\mathcal{R}^p$, $(\widehat{g}_j)_i$ the ith component of $\widehat{g}_j$, and $\{c_j\}$ a positive sequence converging to 0. The usual symmetric difference (SD) estimator perturbs each parameter one at a time:

$$(\widehat{g}_j)_i = \frac{L(\theta_n + c_j e_i, \omega_j^{i+}) - L(\theta_n - c_j e_i, \omega_j^{i-})}{2c_j}, \tag{19}$$

where ω_j^{i+} and ω_j^{i-} denote the pair of sample paths used for the ith component of the jth iterate of the algorithm. If the method of common random numbers is employed, then $\omega_j^{i-} = \omega_j^{i+} = \omega_j$.

Let $\{\Delta_j\}$ be an i.i.d. vector sequence of perturbations of i.i.d. components $\{(\Delta_j)_i, i = 1, ..., p\}$ symmetrically distributed about 0 with $E\left|(\Delta_j)_i^{-2}\right|$ uniformly bounded (see Spall 1992). Then the simultaneous perturbation (SP) estimator perturbs the entire parameter vector simultaneously once:

$$(\widehat{g}_j)_i = \frac{L_j^+ - L_j^-}{2c_j(\Delta_j)_i}, \tag{20}$$

$$\text{where} \quad L_j^+ = L(\theta_n + c_j\Delta_j, \omega_j^+), \tag{21}$$

$$L_j^- = L(\theta_n - c_j\Delta_j, \omega_j^-), \tag{22}$$

where ω_j^+ and ω_j^- denote the pair of sample paths used for the jth iterate of the algorithm, and L_j^+ and L_j^- are performance estimates at the parameter value θ_n *simultaneously* perturbed in all directions. Again, if the method of common random numbers is employed, then $\omega_j^+ = \omega_j^- = \omega_j$.

The key point to note is that each estimate $L(\theta, \omega)$ is computationally expensive relative to the generation of Δ. The SD estimator (19) requires a different pair of estimates in the *numerator* for each parameter to estimate the gradient, thus requiring $2p$ simulations, whereas the SP estimator (20) uses the *same* pair of estimates in the numerator for all parameters, and instead the *denominator* changes. Thus, SP requires only **two** simulations at each SA iteration to form a gradient estimate.

3.2.2 PASA Algorithm

As we remarked earlier, in general the PASA algorithm would be preferred, because it will lead to a faster convergence rate than the SPSA algorithm. However, application of the PASA algorithm is problem dependent, in that an appropriate PA estimator must be found for each setting. Sometimes this is not difficult, as in going from the discrete to continuous dividends case here, but oftentimes it is not at all straightforward, as in the multiple assets cases. Here we have the PA estimators for the two single stock American call and American-Asian cases, i.e., Testbed Sets 1, 2 and 4.

For the American call option defined on a single stock with discrete early exercise opportunities, the optimal early exercise boundary simply consists of a set of early exercise thresholds, which we denote by $\{s_j(\geq K), j = 0, 1, ..., N\}$, such that the option is exercised if $S_{t_j^-} > s_j$. The optimal values of the thresholds, i.e., the settings that maximize the expected payoff and hence give the option value, are denoted as before by s_j^*. The sample performance is the net present (at time 0) value of the option payoff (not the option value), and can be written as

$$L = \sum_{i=0}^{N} \mathbf{1}\left\{\bigcap_{j=0}^{i-1} S_{t_j^-} \leq s_j, S_{t_i^-} > s_i\right\}\left(S_{t_i^-} - K\right)e^{-rt_i} + \mathbf{1}\{S_{t_1^-} \leq s_1, ..., S_{t_N^-} \leq s_N\}(S_T - K)^+ e^{-rT}, \quad (23)$$

which is analogous to (14). The indicator functions simply indicate whether or not the option has been exercised at each potential exercise point.

In Fu, Wu, Gürkan, and Demir (2000), the following PA estimator was derived for the derivative of the expected discounted payoff with respect to a parameter in the model, i.e., $dE[L]/d\theta$:

$$\sum_{i=0}^{N} \mathbf{1}\left\{\bigcap_{j=0}^{i-1} S_{t_j^-} \leq s_j\right\} \frac{\partial h^{-1}(y_i^*)}{\partial \theta} f_i(h^{-1}(y_i^*)) \left\{E\left[L \middle| \bigcap_{j=0}^{i-1} S_{t_j^-} \leq s_j, S_{t_i^-} = s_{i_-}\right] - (s_i - K)e^{-rt_i}\right\}$$

$$+ \sum_{i=0}^{N} \mathbf{1}\left\{\bigcap_{j=0}^{i-1} S_{t_j^-} \leq s_j, S_{t_i^-} > s_i\right\} \frac{\partial}{\partial \theta}\left[(S_{t_i^-} - K)e^{-rt_i}\right]$$

$$+ \mathbf{1}\{S_{t_1^-} \leq s_1, ..., S_{t_N^-} \leq s_N\} \frac{\partial}{\partial \theta}\left[(S_T - K)^+ e^{-rT}\right], \quad (24)$$

where $$y_i^* = (s_i - \tilde{D}_i; \tilde{S}_{t_{i-1}}, \tau_i, \mu, \sigma), \quad i = 0, ..., N. \quad (25)$$

with the stock price process as specified earlier and $\tilde{D}_i$ given by (3). For the pricing problem, we are interested in the case where θ is the vector of thresholds $s_j, j = 0, ..., N$.

For geometric Brownian motion under the risk-neutral measure, we have the expressions for $h_i = h\ \forall i$ and $f_i = f\ \forall i$ provided by (4), with the inverse $h^{-1}(y; S, t, r, \sigma) = (\ln(y/S) - (r - \sigma^2/2)t)/(\sigma\sqrt{t})$, so that for $dE[L]/ds_i$, (24) becomes the following estimator:

$$\mathbf{1}\left\{\bigcap_{j=0}^{i-1} S_{t_j^-} \leq s_j\right\} \frac{e^{-\left((\ln((s_i - \tilde{D}_i)/\tilde{S}_{t_{i-1}}) - (r - \sigma^2/2)\tau_i)/(\sigma\sqrt{\tau_i})\right)^2/2}}{(s_i - \tilde{D}_i)\sigma\sqrt{2\pi\tau_i}} \left\{E\left[L \middle| \bigcap_{j=0}^{i-1} S_{t_j^-} \leq s_j, S_{t_i^-} = s_{i_-}\right] - (s_i - K)e^{-rt_i}\right\}.$$

For the continuous dividends case, the same estimator given by Equation (24) is applicable with the change in the definition of y_i^* given by (25) to the following:

$$y_i^* = (s_i; S_{t_{i-1}}, \tau_i, r-\delta, \sigma), \quad i = 0, ..., N,$$

which gives the explicit estimator:

$$\mathbf{1}\left\{\bigcap_{j=0}^{i-1} S_{t_j^-} \le s_j\right\} \frac{e^{-\left((\ln(s_i/S_{t_{i-1}})-(r-\delta-\sigma^2/2)\tau_i)/(\sigma\sqrt{\tau_i})\right)^2/2}}{s_i\sigma\sqrt{2\pi\tau_i}} \left\{E\left[L\left|\bigcap_{j=0}^{i-1} S_{t_j^-} \le s_j, S_{t_i^-} = s_{i_-}\right.\right] - (s_i - K)e^{-rt_i}\right\}.$$

Remarks. Note that Fu, Wu, Gürkan, and Demir (2000) corrects the expressions given in Fu and Hu (1995, 1997). These and other typographical errors in Fu and Hu (1997) can be found at `http://alexandra.bmgt.umd.edu/mss/mfu/book/`.
These derivative estimators have also been used in conjunction with nonlinear programming for option pricing in a technique called sample path optimization (Gürkan, Özge, and Robinson 1996).

Estimator for American-Asian Options

PA estimators for American-Asian options were derived in Wu and Fu (2000). For geometric Brownian Motion model, using the piecewise linear approximation of the exercise boundary given by (16), the estimator with respect to s_i is given by

$$\mathbf{1}\left\{\bigcap_{j=1}^{i-1} \overline{S}_{t_j} < \phi_j(S_{t_j}), \overline{S}_{t_i-\tau_i} > s_i + \frac{v_i}{n_i - 1}\right\} \times$$
$$\left\{\mathbf{1}\left\{n_i s_i > (n_i-1)\overline{S}_{t_i-\tau_i}\right\} \frac{n_i e^{-\left((\ln((n_i s_i-(n_i-1)\overline{S}_{t_i-\tau_i})/S_{t_i-\tau_i})-(r-\sigma^2/2)\tau_i)/(\sigma\sqrt{\tau_i})\right)^2/2}}{\sigma\sqrt{2\pi\tau_i}\left(n_i s_i - (n_i-1)\overline{S}_{t_i-\tau_i}\right)} \times \right.$$
$$\left.\left[E\left[\mathcal{L}\left|\bigcap_{j=1}^{i-1} \overline{S}_{t_j} < \phi_j(S_{t_j}), \overline{S}_{t_i-\tau_i}, S_{t_i} = \left(n_i s_i - (n_i-1)\overline{S}_{t_i-\tau_i}\right)_-\right.\right] - (s_i - K)e^{-rt_i}\right]\right\},$$

and the estimator with respect to v_i is given by

$$\mathbf{1}\left\{\bigcap_{j=1}^{i-1} \overline{S}_{t_j} < \phi_j(S_{t_j}), \overline{S}_{t_i-\tau_i} > s_i + \frac{v_i}{n_i - 1}\right\} \times$$
$$\left\{\frac{-n_i e^{-\left((\ln(((n_i-1)\overline{S}_{t_i-\tau_i}-n_i v_i)/((n_i-1)S_{t_i-\tau_i})))-(r-\sigma^2/2)\tau_i)/(\sigma\sqrt{\tau_i})\right)^2/2}}{\sigma\sqrt{2\pi\tau_i}\left(n_i v_i - (n_i-1)\overline{S}_{t_i-\tau_i}\right)} \times \right.$$
$$\left.\left[E\left[\mathcal{L}\left|\bigcap_{j=1}^{i-1} \overline{S}_{t_j} < \phi_j(S_{t_j}), \overline{S}_{t_i-\tau_i}, S_{t_i} = \left(\overline{S}_{t_i-\tau_i} - \frac{n_i v_i}{n_i-1}\right)_+\right.\right] - (\overline{S}_{t_i-\tau_i} - \frac{v_i}{n_i-1} - K)e^{-rt_i}\right]\right\}.$$

3.3 Bounds from Simulated Paths

Here we present two algorithms proposed by Broadie and Glasserman (1997a,1998), simulated tree and stochastic mesh, that are based on simulated paths, and which lead to biased low and biased high estimators that converge to the true value in the appropriate limit. These algorithms apply only for cases where the number of exercise opportunities is finite, and the simulated tree is practical only when the number is relatively small (less than five). The simulated tree results in a

tree generated for each path, whereas the stochastic mesh is constrained to a mesh of points. The implemented algorithms will be indicated by "ST" for the simulated tree algorithm, and "SM" for the stochastic mesh algorithm.

3.3.1 Simulated Tree Algorithm

In contrast to the standard Monte Carlo approach for valuing European options, in the simulated tree (ST) algorithm the evolution of stock prices is simulated using random trees rather than just sample paths. The number of branches at each node is b, where b is called the branching parameter. As before, we let d_t denote the discount factor from time t to time $t+1$ (if the time steps are uniformly of size 1 and the interest rate constant as in the testbed of cases we consider, then this is simply equal to e^{-r}), and $L_t(s)$ the payoff from exercise at time t in state s. The evolution of the simulated trees can be described as follows: From the initial state S_0, we generate b independent nodes $S_1^1, S_1^2 ... S_1^b$ at time $t = 1$. (Here superscripts denote branches, not different stocks.) Note that the nodes at a fixed time appear according to the order in which they are generated, not according to their node values, as would be the case in a lattice method. From each of these b nodes, $S_1^{i_1}, i_1 = 1, 2, ...b$, in turn, generate b new nodes, $S_2^{i_1 1}, ...S_2^{i_1 b}$ at time $t = 2$. Generally, from each node $S_t^{i_1 i_2 ... i_t}$ at time t, we can generate b nodes $S_{t+1}^{i_1 i_2 ... i_t j}$ at time $t+1, j = 1, ..., b$, conditionally independent of each other given $S_t^{i_1 i_2 ... i_t}$ and each having the same distribution given $S_t^{i_1 i_2 ... i_t}$. Thus, $i_1 i_2 ... i_t$ specifies a simulated path of the tree up to time t.

The simulated trees can be used to derive two estimators, one biased high and one biased low. The high ST estimator $\Psi \stackrel{\triangle}{=} \Psi_0$ is defined by the following backwards recursion:

$$\begin{aligned} \Psi_T^{i_1 i_2 ... i_T} &= L_T(S_T^{i_1 i_2 ... i_T}), \\ \Psi_t^{i_1 i_2 ... i_t} &= \max \left\{ L_t(S_t^{i_1 i_2 ... i_t}), \frac{1}{b} \sum_{j=1}^{b} d_t \Psi_{t+1}^{i_1 i_2 ... i_t j} \right\}, \ t = 0, ...T-1. \end{aligned} \tag{26}$$

At any node, therefore, the high estimate is the maximum of the immediate exercise value and the average of the discounted high estimators at successor nodes.

The high estimator uses all branches emanating from a node to approximate both the optimal decision and the node values. The low estimators differs in that it separates the branches used to determine the decision from those used to determine the continuation value. The low ST estimator $\psi \stackrel{\triangle}{=} \psi_0$ is defined by the following backwards recursion:

$$\begin{aligned} \psi_T^{i_1 i_2 ... i_T} &= L_T(S_T^{i_1 i_2 ... i_T}), \\ \psi_t^{i_1 i_2 ... i_t} &= \frac{1}{b} \sum_{j=1}^{b} \eta_t^{i_1 i_2 ... i_t j}, \ t = 0, 1, ...T-1, \end{aligned} \tag{27}$$

where

$$\eta_t^{i_1 i_2 ... i_t j} = \begin{cases} L_t(S_t^{i_1 i_2 ... i_t}) & \text{if } L_t(S_t^{i_1 i_2 ... i_t}) > \frac{1}{b-1} \sum_{i=1, i \neq j}^{b} d_t \psi_{t+1}^{i_1 i_2 ... i_t i}; \\ d_t \psi_{t+1}^{i_1 i_2 ... i_t j} & \text{otherwise.} \end{cases}$$

Enhancements to this method are contained in Broadie, Glasserman, and Jain (1997).

Extension to American-Asian Options

To apply the algorithm to American-Asian options with a finite number of exercise points, all time points that enter into the average payoff function, not just the early exercise points, must be simulated. To be specific, for each replication of the simulated tree, a single sample path is generated

from time 0 to time t', the first averaging date. From t', b independent paths are generated to the next averaging date. From each of these b nodes, a single sample path is generated that includes all the time steps required in the averaging until the frist exercise date, t_1, is reached. Again, at t_1, b independent paths are generated to the next averaging date and then a single sample path is generated to t_2. This process continues until the paths reach the final exercise date. In other words, at each early exercise point t_i, b independent nodes are generated to the next averaging date, from which a single sample path is simulated forward until the next early exercise point is reached.

3.3.2 Stochastic Mesh Algorithm

The stochastic mesh (SM) method was designed for pricing high-dimensional American options when there is a finite, but possibly large, number of exercise opportunities. As in the simulated tree algorithm, high and low estimators are derived. The procedure starts by generating random vectors $X_t(i)$ for $i = 1, 2, ..., b$, and $t = 1, ...T$, where now b is a fixed number of nodes at each time step, instead of a branching parameter. Let $f_t(x, \cdot)$ denote the density of S_{t+1} given $S_t = x$. For $t = 1, ..., T$, the vectors $X_t(i), i = 1, ..., b$ are generated as i.i.d. samples from the mesh density function $g_t(\cdot)$. The choice of $g_t(\cdot)$ is critical to the practical success of the method. The high SM estimator $\Psi \stackrel{\triangle}{=} \Psi_0$ is defined by the following backwards recursion ($i = 1, 2, ..., b$):

$$\begin{aligned} \Psi_T(X_T(i)) &= L_T(X_T(i)), \\ \Psi_t(X_t(i)) &= \max\left(L_t(X_t(i)), \frac{1}{b}\sum_{j=1}^{b} \Psi_{t+1}(X_{t+1}(j)) w_t(X_t(i), X_{t+1}(j)) \right), \ t = T-1, ...0, \end{aligned} \tag{28}$$

where

$$w_t(x, X_{t+1}(j)) = \frac{f_t(x, X_{t+1}(j))}{g_{t+1}(X_{t+1}(j))}.$$

A choice recommended by Broadie and Glasserman (1998) for the mesh density function $g_t(\cdot)$ is given by

$$\begin{aligned} g_1(u) &= f_0(S_0, u), \\ g_t(u) &= \frac{1}{b}\sum_{j=1}^{b} f_{t-1}(X_{t-1}(j), u), \ t = 2, ..., T. \end{aligned}$$

Using the density function to generate the mesh can be interpreted and implemented in the following way. Suppose from each of the mesh nodes $X_{t-1}(j), j = 1, ..., b$, we generate exactly one successor $X_t(j)$ according to the transition density $f_{t-1}(X_{t-1}(j), \cdot)$. Then we choose a value randomly and uniformly from $\{X_{t-1}(1), ..., X_{t-1}(b)\}$. The value chosen is then distributed according to the density $g_t(\cdot)$, conditional on $\{X_{t-1}(1), X_{t-1}(2)..., X_{t-1}(b)\}$. This is called stratified sampling.

The low SM estimator is based on n_p independent sample paths, each obtained by simulating a trajectory of the underlying process $\{S_t\}$ until the exercise region, determined by the mesh, is reached. Denote the simulated path by $S = (S_0, S_1...S_T)$. The path is simulated according to the density function of the process S_t, i.e., given $S_t = x$, the density of S_{t+1} is $f_t(x, \cdot)$. The approximate optimal policy determined by the mesh will exercise at

$$\tau(S) = \min\left\{ t : L_t(S_t) \geq \frac{1}{b}\sum_{j=1}^{b} \Psi_{t+1}(S_{t+1}) w_t(S_t, S_{t+1}) \right\}, \quad \Psi_T(S_T) = L_T(S_T). \tag{29}$$

For sample path i, $i = 1, ..., n_p$, we denote by S^i the stock price sequence, τ_i the corresponding optimal stopping time defined by (29), and define the path estimator in the obvious way by

$$\psi_i = L_\tau(S_\tau).$$

The low SM estimator is then defined by taking the sample mean over a number of simulated sample paths:

$$\psi = \frac{1}{n_p} \sum_{i=1}^{n_p} \psi_i.$$

Applicability to American-Asian Options

It appears that the stochastic mesh method cannot be directly applied to American-Asian options, because the density of $(S_{t+1}, \bar{S}_{t+1})$ given $(S_t, \bar{S}_t)$ is zero with probability one.

3.3.3 Confidence Intervals

For both the ST and SM estimators, Broadie and Glasserman (1997a,1998) provide proofs that the high estimator Ψ is biased high and the low estimator ψ is biased low. Furthermore, both of them converge to the true price as $b \to \infty$. Based on the high and low estimators, a single point estimator can be obtained simply by taking the average of the two:

$$\frac{\max\{(S_0 - K)^+, \psi\} + \Psi}{2},$$

where the lower limit is truncated at the immediate exercise value of the option, which is a trivial lower bound on the true price. One can also obtain a conservative confidence interval. For example, based on m independent replications, an approximate $(1 - \alpha/2)100\%$ confidence interval for an American call is given by

$$\left[\max\left\{(S_0 - K)^+, \psi - z_{\alpha/2} s_\psi(m)/\sqrt{m}\right\}, \Psi + z_{\alpha/2} s_\Psi(m)/\sqrt{m}\right],$$

where $z_{\alpha/2}$ is the 1-$\alpha/2$ quantile of the standard normal distribution, and $s_\psi(m)$ and $s_\Psi(m)$ are the sample standard deviations of ψ and Ψ, respectively.

4 Computational Results

In this section, we present computational results from implementation of the various algorithms on the testbed of cases. The purpose of these numerical experiments is to try to confirm and/or determine where the strengths of each is likely to lie, in terms of ease of implementation, applicability and flexibility, and computational efficiency. The testbed of cases described in Section 2 was used with the following specific parameter settings:

1. Call options on single underlying dividend-paying stock:
   time horizon: $T = 0.5, 1.0, 2.0, 3.0$ years;
   dividends: $D = 1.0, 2.5$ (for $T = 0.5$, where $\tau_i = 0.25$); $2.0, 5.0$ (for $T = 1, 2, 3$, where $\tau_i = 0.5$);
   $\delta = 0.04, 0.10$ for continuous dividend cases;
   interest rates and volatility: $r = 0.05, 0.09$; $\sigma = 0.20$;
   starting price and strike prices: $S_0 = 100$; $K = 80, 90, 100, 110, 120$.

2. Call options on the maximum of n identically distributed, but correlated with a common correlation coefficient ρ, dividend-paying stocks (Broadie and Glasserman 1997a,1998):
   $n = 2, 5; T = 1.0; r = 0.05; \sigma = 0.20; \rho = 0.30; \delta = 0.10; S_0 = 70, 80, 90, 100, 110, 120, 130; K = 100$.

3. American Asian call option on a single non-dividend-paying stock (Grant, Vora, and Weeks 1997):
   $T = t_N = 120$ days; $r = 0.09$; $\sigma = 0.20, 0.30; S_0 = 100$; $K = 90, 95, 100, 105, 110$; with the averaging beginning at time $t' = 91$ days, but exercise allowed only beginning at $t_1 = 105$. Only the cases $N = 2, 4, 6$ were considered, with $t_2 = 110, t_3 = 115$ for $N = 4$ and $t_2 = 108, t_3 = 111, t_4 = 114, t_5 = 117$ for $N = 6$.

4. Put option on a single stock modeled by a jump-diffusion model (Grant, Vora, Weeks 1996):
   $T = 0.5$; $S_0 = K = 100$; $r = 0.10$; $\lambda = 2$; $\delta = 0.2$; $\sigma = \sqrt{0.08} \approx 0.2828427$, gives an overall volatility of $\sqrt{\lambda\delta^2 + \sigma^2} = 0.4$.

The parameter values for the last three testbed cases follow the settings used in the indicated papers from which the examples were taken.

Parameterizations

The parameterizations of the exercise boundaries used by PASA, SPSA, and DP2 for the testbed cases are as follows:
(i) The exercise boundary for all American options on a single stock with discrete exercise opportunities — Testbed Sets 1, 2, and 5 — is parameterized by the vector $\theta = (s_0, s_1, ..., s_N)$ corresponding to the early exercise thresholds at exercise points $t_i, i = 0, 1, ..., N$, respectively.
(ii) For the American-Asian option, at each exercise date t_i, we follow Grant, Vora, and Weeks (1997), and use the linear exercise boundary given earlier by (15);thus the parameter vector to be optimized is $\theta = (s_0, v_0, s_1, v_1, ..., s_N, v_N)$.
(iii) For the American-max call option on n underlying assets, the exercise boundary implied by the hold region given by (8) leads to the same form of the parameter vector as for the previous case: $\theta = (s_0, v_0, s_1, v_1, ..., s_N, v_N)$.

Implementation

The following is a summary of the various implementation parameters for each of the algorithms, some (but not all) of which have been defined already in the description of the algorithms:

DP1	R, P, Q
DP2	$R, n_p, \lvert s\rvert, \lvert v\rvert$
SPSA	$R, n_p, \eta, \theta_0, \{a_n\}, \{c_n\}, \{\Delta_n\}$
PASA	$R, n_p, \eta, \theta_0, \{a_n\}$
ST	R, b
SM	$R, b, n_p, g_t(\cdot)$

Table 1: Implementation Parameters for Algorithms.

The parameters R and n_p have slightly different interpretations for each of the algorithms. In SPSA, PASA, and DP2, R is the number of simulation replications used to price the option after

all the early exercise boundaries have been obtained, whereas it is the total number of replications for ST and DP1. (Note that the same number of "replications" in ST — which generate branching trees and not single paths — will involve significantly more computation than for DP1.) For SM, R is the number of replications for the upper bound, and n_p is the number of additional paths per replication to estimate the lower estimate; i.e., n_pR is the number of additional paths that must be generated for the lower bound. In SPSA and PASA, n_p is the number of replications per gradient estimate (i.e., per iteration of θ, where η is the total number of iterations), whereas it is the number of replications per point estimate in the search procedure for each induction step of DP2.

For the multiple dimension cases of DP2, $|s|$ and $|v|$ represent the size of the the parameter grid that is searched to find the optimal (s_i^*, v_i^*) at each early exercise date. There are additional implementation considerations for DP1 and DP2; in particular, a bundling/sorting procedure must be specified for DP1, and a search procedure (which could be a brute-force grid in multiple dimensions) must be specified for DP2. The particular procedures adopted have been described earlier. Also, the technique of common random numbers was incorporated for SPSA, so that only a single sample path was used to estimate the gradient in each SPSA iteration given by (18).

The values of the implementation parameters used were as follows:

- **DP1** — For single asset options (Testbed Sets 1, 2, and 5), $R = 7225, P = Q = 85$; for American-Asian option, $R = 13440$, $P = 70$, and $Q = (16\ 12)$; for max options, $R = 13440$, $P = 70$, $Q = (16\ 12)$ for $n = 2$, and $R = 13440$, $P = 70$, $Q = (6\ 4\ 2\ 2\ 2)$ for $n = 5$.

- **DP2** — $R = 20,000$; $n_p = 2,000$ for Testbed Cases 1 and 2; $n_p = 4,000$ for Testbed Cases 3, 4, and 5; $|s| = 4, |v| = 9$ for Asian (as in Grant, Vora, Weeks (1997)), $|s| = 4, |v| = 9$ for $n = 2$ max option, $|s| = 8, |v| = 18$ for $n = 5$ max option.

- **SPSA** — $R = 20,000; c_j = 10/j^{0.25}; n_p = 100; \eta = 500$. For Testbed 1, $a_j = 2000/j^{0.751}$ ($a_j = 5000/j^{0.751}$ when $r = .09$, $D = 2$, $T = 3$); for Testbed 2, $a_j = 1000/j^{0.751}$ ($a_j = 2000/j^{0.751}$ when $\delta = .04$, $T = 2, 3$); for Testbed 3 and 4, $a_j = 80/j^{0.751}$; for Testbed 5, $a_j = 2000/j^{0.751}$. $\{(\Delta_n)_i\}$ i.i.d. symmetric Bernoulli random variables (± 1 w.p. 1/2). θ_0 (initial iterate): for Testbed 1,2,5, $\theta_0 = \left(Ke^{rT}, Ke^{r(T-t_1)}, ..., K\right)$; for American Asian, $\theta_0 = \left(Ke^{rT}, 0, Ke^{r(T-t_1)}, 0, ..., K, 0\right)$; for max, $\theta_0 = \left(Ke^{rT}, 10n, Ke^{r(T-t_1)}, 10n, ..., K, 10n\right)$.

- **PASA** — $R = 20,000$ for all cases; for Testbed Cases 1 and 2, $a_j = 100/j, n_p = 100, \eta = 500, \theta_0 = (K, K, ..., K)$, 100 replications are used to estimate the portion of conditional expectation in PA estimators; for American-Asian option, $a_j = 50/j; n_p = 50; \eta = 10; \theta_0 = (K, 0, K, 0, ..., K, 0)$, 10 replications are used to estimate the portion of conditional expectation in PA estimators.

- **ST** — $R = 100$, except Asian, where $R = 10,000$; $b = 50$ for all cases except Asian, where $b = 5$; single asset $T = 2$, where $b = 20$ and $T = 3$, where $b = 10$, and jump-diffusion $N = 4$, where $b = 20$.

- **SM** — $R = 50$; $b = 160, n_p = 1600$ for single asset $T = 0.5, 1, 2$; and $b = 700, n_p = 500$ otherwise; $g_t(\cdot)$ recommended by Broadie and Glasserman (1998).

We note that in using ST for derivative contracts such as the American-Asian option here, one needs more replications (higher R) than in other cases in order to get good path estimates, because the payoff depends very heavily on the averaging time points prior to the first branching node.

Results

The results are shown in Tables 2 through 21. The last row for each case in a table contains an approximate "true" value, obtained in a variety of ways, as there was no common approach to handle all of the cases. For the discrete dividends testbed, analytical results are available in principle, though in practice they are not easy to obtain for more than two dividends; hence, a binomial lattice method (indicated by "lattice" in the tables) was used for both the discrete and continuous dividends cases. For all lattices, 1000 time steps were used, with the exception of T=3.0 cases, where 996 time steps were used, because 1000 is not divisible by 6. For the maximum of two assets, analytical values obtained in Broadie and Glasserman (1997a) are reported in the table, indicated by "true". In the remaining cases, no such analytical results are available and lattice methods are impractical, so estimates from the papers where the parameter values were taken are reported, indicated by GVW96, GVW97, and BG, where the accuracy of the latter was greatly enhanced through the use of control variates.

Shown in the tables are the point estimates, the measure of precision, and the CPU times (in seconds). The measures of precision were as follows: two standard errors in the case of SPSA, PASA, DP1 and DP2 and half the (very conservative) 90% confidence interval lengths for ST and SM. For DP1, the standard errors are based on the option values of the R stored paths; however, since the early exercise decisions are based solely on these stored paths, the paths are correlated and do not constitute a final independent replication as in the other methods. In Tables 12, 13, and 16, there are also no C.I. entries for the K=80 cases for SPSA, PASA, DP1, and DP2; this anomaly is a result of the fact that the exercise boundary always results in immediate exercise. All algorithms were implemented in C. Execution was carried out on a Sun Ultra5 running Solaris OS.

Our experiments indicate that where applicable PASA and DP2 seem to provide the best accuracy for a given amount of computation time. However, PASA is by far the most problem dependent. Furthermore, for both of the first two approaches — which includes the SPSA, PASA, DP1, and DP2 algorithms, convergence to the exact price is achieved only when the form of the parameterized exercise boundary coincides with the true optimal form, as in the one-dimensional case; in general, the algorithms converge to a value that is a lower bound on the price, oftentimes reasonably tight when the price is relatively insensitive to a precise specification of the exercise boundary. Among the first two approaches, SPSA is a very fast and flexible pricing method that is easy to apply to a wide variety of cases, once an exercise boundary is specified. From our experiments, we find that both PASA and SPSA are slower to converge for the multiple dividend cases, with PASA converging faster as expected. This indicates that whenever the PASA method is available, PASA would be preferred. For the SM method, the point estimate obtained simply by averaging the low and high estimates appears to have a substantial upward bias for a number of the cases considered.

The exponential growth of the ST algorithm is also apparent in the CPU times reported. For comparison with CPU times reported here, computation times reported in Broadie and Glasserman (1997a) were 2 CPU minutes per entry on a 133 MHz Pentium PC for the two-asset case (not reported for five-asset case), with $T = 1$ year, $b = 50$, $N = 3$, and 100 replications. Also claimed there was that for single asset cases, a one standard error half-width — which corresponds to 68% confidence interval — actually contained 96% of the true values, and 90% confidence levels actually contained the true value 99% of the time. Computation times reported using Microsoft FORTRAN under DOS on a 100MHz Pentium PC, for standard errors under a penny, were about 1 CPU minute (55 seconds) for the Asian case in Grant, Vora, and Weeks (1997) and 2 CPU minutes, 10 seconds, for the jump-diffusion case with 20 early exercise points, 1,000 replications per induction step, and 200,000 replications for the final option valuation step.

5 Summary, Conclusions, and Potential Areas for Future Research

Our experiments confirm the assessment implied by Broadie and Glasserman (1997b) that the most flexible and easily implementable procedure is the simulated tree (ST) algorithm, its primary drawback being its exponential growth in computational requirements with respect to the number of exercise opportunities. In retrospect, the dynamic programming and stochastic approximation approaches are quite similar in implementation, which becomes more obvious in higher-dimensional problems, in that both require an explicit characterization of the exercise boundary for higher dimensions. (In principle, the dynamic programming approach could proceed without this, but in practice this would lead to computational intractability.) The dynamic programming approach proceeds sequentially by recursively optimizing a parameter vector θ_i by backwards induction ($i = N-1, ..., 1$), whereas the parametric optimization approach simultaneously optimizes $(\theta_1, ..., \theta_n)$ by iterative updates via a stochastic approximation algorithm. Among the four algorithms considered in these two classes, it seems that SPSA and DP2 are the most general and easily implementable, with good computational properties, as well. One clear drawback of DP1 is the absence of an accurate measure of precision, as discussed above.

A non-exhaustive list of some other research issues worth further pursuing:

- Control variates were not implemented here. Whether they will improve all algorithms at relatively the same level is not at all obvious a priori, and they seemed to be critical to the practicality of SM, as reported in Broadie and Glasserman (1998).

- For the parameterizing algorithms DP2, SPSA, and PASA, further substantial computational improvements can be realized by employing ordinal optimization ideas such as those considered in Su (2000) for the SPSA case.

- For the stochastic approximation algorithms PASA and SPSA, instead of using a fixed number of iterations, as done here, a good stopping rule might be lead to a more efficient use of computation time. Other implementation issues that are also common to stochastic approximation applications in general and not just to American option pricing problems include determining parameter choices such as the starting point and starting step size. Starting from a set of different starting points could lessen the inherent problem of gradient search algorithms of falling into local minimums.

- For the dynamic programming algorithms DP1 and DP2, further investigation of implementation issues is warranted. In particular, determining a good bundling/sorting procedure seems to be a difficult problem in higher dimensions for DP1, and more efficient search procedures — perhaps incorporating design of experiments methodology — is key to making DP2 a viable alternative.

- Dimension reduction was implemented for the first two approaches for the max-option testbed cases. This is clearly one way to make the corresponding algorithms more computationally efficient, but it must generally be pursued on a case-by-case basis with caution.

Characterizing the early exercise curve of a high-dimensional problem with a lower-dimensional model has also been pursued by Clewlow and Strickland (1998) for interest rate derivatives. This and other related papers can be found (most reprinted, e.g., Tilley 1993) in the volume edited by Dupire (1998).

		t_i=0, 0.25, 0.5 T=0.5, D=1.0			t_i=0, 0.5, 1.0 T=1.0, D=2.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	21.220	0.150	0.49	22.363	0.187	0.43
	PASA	21.201	0.148	0.40	22.732	0.211	0.41
	ST	21.359	0.582	0.85	22.858	0.714	0.86
	SM	21.792	0.863	13.70	23.357	0.943	13.92
	DP1	21.236	0.312	0.38	22.993	0.408	0.38
	DP2	21.207	0.179	0.22	22.861	0.214	0.20
	lattice	21.211		0.40	22.729		0.39
K=90	SPSA	12.596	0.148	0.56	15.093	0.213	0.75
	PASA	12.669	0.159	0.43	15.058	0.213	0.43
	ST	12.653	0.396	0.87	15.034	0.498	0.87
	SM	12.727	0.241	13.85	15.163	0.397	13.84
	DP1	12.733	0.270	0.38	15.122	0.334	0.39
	DP2	12.671	0.169	0.22	15.042	0.230	0.23
	lattice	12.640		0.38	15.025		0.39
K=100	SPSA	6.297	0.126	0.60	9.123	0.181	0.60
	PASA	6.287	0.127	0.43	9.086	0.182	0.45
	ST	6.233	0.224	0.88	9.065	0.327	0.89
	SM	6.380	0.295	13.80	9.303	0.465	13.85
	DP1	6.250	0.210	0.38	9.120	0.301	0.38
	DP2	6.228	0.130	0.21	9.072	0.192	0.23
	lattice	6.257		0.39	9.097		0.38
K=110	SPSA	2.559	0.081	0.60	4.973	0.127	0.59
	PASA	2.526	0.084	0.44	5.038	0.142	0.44
	ST	2.545	0.109	0.88	5.046	0.205	0.89
	SM	2.667	0.236	13.75	5.121	0.283	13.79
	DP1	2.514	0.141	0.38	4.985	0.241	0.39
	DP2	2.509	0.085	0.21	4.833	0.144	0.26
	lattice	2.541		0.38	5.057		0.38
K=120	SPSA	0.835	0.045	0.59	2.612	0.101	0.59
	PASA	0.838	0.048	0.44	2.613	0.106	0.44
	ST	0.864	0.058	0.85	2.617	0.129	0.87
	SM	0.852	0.071	13.71	2.584	0.131	13.74
	DP1	0.900	0.083	0.37	2.579	0.173	0.38
	DP2	0.854	0.049	0.26	2.600	0.109	0.26
	lattice	0.856		0.38	2.606		0.38

Table 2: American call option on single asset: r=0.05.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, D=2.0			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, D=2.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	23.807	0.230	0.93	25.115	0.293	1.49
	PASA	23.763	0.250	1.16	24.851	0.267	1.98
	ST	24.453	1.579	56.99	25.088	2.920	379.12
	SM	25.235	1.729	26.97	25.605	0.990	424.84
	DP1	24.148	0.500	1.77	24.908	0.507	4.20
	DP2	24.254	0.281	0.61	24.906	0.349	1.13
	lattice	23.970		0.39	24.986		0.39
K=90	SPSA	17.266	0.262	1.42	18.964	0.296	1.68
	PASA	17.125	0.249	1.28	18.892	0.280	2.26
	ST	17.641	1.091	57.34	19.091	2.040	377.3
	SM	18.205	1.281	28.24	19.656	0.899	428.69
	DP1	17.722	0.479	1.79	19.713	0.602	4.28
	DP2	17.339	0.293	0.72	18.864	0.350	1.40
	lattice	17.419		0.38	19.145		0.38
K=100	SPSA	11.950	0.231	1.15	14.094	0.267	1.72
	PASA	12.103	0.237	1.39	14.230	0.269	2.42
	ST	12.269	0.750	57.34	14.245	1.504	378.10
	SM	12.695	1.063	28.84	14.672	0.766	431.74
	DP1	12.207	0.434	1.78	14.752	0.513	4.28
	DP2	12.149	0.272	0.71	14.274	0.310	1.20
	lattice	12.203		0.38	14.382		0.38
K=110	SPSA	8.291	0.221	1.22	10.276	0.231	1.75
	PASA	8.271	0.209	1.42	10.527	0.251	2.55
	ST	8.261	0.503	56.97	10.488	1.107	374.23
	SM	8.585	0.722	29.01	10.830	0.523	431.79
	DP1	8.335	0.359	1.77	10.317	0.458	4.27
	DP2	8.167	0.230	0.71	10.703	0.300	1.34
	lattice	8.277		0.36	10.628		0.38
K=120	SPSA	5.400	0.170	1.20	7.434	0.199	2.16
	PASA	5.421	0.175	1.46	7.622	0.225	2.62
	ST	5.415	0.336	56.47	7.632	0.815	369.84
	SM	5.644	0.508	29.18	7.959	0.406	432.48
	DP1	5.621	0.316	1.79	7.328	0.367	4.27
	DP2	5.415	0.193	0.86	7.764	0.251	1.24
	lattice	5.463		0.38	7.746		0.38

Table 3: American call option on single asset: r=0.05.

		t_i=0, 0.25, 0.5 T=0.5, D=2.5			t_i=0, 0.5, 1.0 T=1.0, D=5.0		
		Price	C.I.	CPU	Price	C.I.	CPU
	SPSA	21.110	0.133	0.49	22.017	0.178	0.41
	PASA	21.040	0.133	0.40	22.144	0.177	0.40
	ST	20.930	0.549	0.82	22.074	0.487	0.82
K=80	SM	21.080	0.468	13.37	22.177	0.550	13.61
	DP1	21.026	0.223	0.38	22.288	0.299	0.37
	DP2	20.979	0.133	0.13	22.190	0.178	0.14
	lattice	21.049		0.38	22.184		0.38
	SPSA	11.929	0.122	0.46	13.675	0.161	0.47
	PASA	11.994	0.126	0.41	13.720	0.164	0.40
	ST	11.937	0.395	0.85	13.611	0.467	0.85
K=90	SM	12.047	0.371	13.78	13.678	0.362	13.72
	DP1	12.158	0.212	0.38	13.809	0.280	0.39
	DP2	12.002	0.127	0.14	13.818	0.166	0.14
	lattice	12.002		0.38	13.729		0.38
	SPSA	5.470	0.099	0.56	7.529	0.133	0.55
	PASA	5.509	0.105	0.43	7.568	0.141	0.42
	ST	5.488	0.230	0.87	7.546	0.318	0.87
K=100	SM	5.612	0.306	13.77	7.711	0.449	13.76
	DP1	5.517	0.182	0.38	7.519	0.248	0.37
	DP2	5.435	0.107	0.16	7.584	0.141	0.15
	lattice	5.498		0.37	7.560		0.38
	SPSA	2.016	0.064	0.58	3.797	0.104	0.57
	PASA	2.059	0.071	0.43	3.820	0.112	0.44
	ST	2.070	0.108	0.87	3.831	0.196	0.88
K=110	SM	2.087	0.126	13.72	3.933	0.261	13.73
	DP1	2.144	0.132	0.37	3.738	0.176	0.38
	DP2	2.092	0.075	0.21	3.793	0.113	0.18
	lattice	2.061		0.39	3.821		0.39
	SPSA	0.574	0.034	0.58	1.764	0.073	0.58
	PASA	0.636	0.041	0.44	1.794	0.081	0.45
	ST	0.654	0.050	0.85	1.818	0.119	0.86
K=120	SM	0.630	0.043	13.67	1.798	0.109	13.71
	DP1	0.595	0.066	0.39	1.668	0.134	0.38
	DP2	0.647	0.041	0.18	1.814	0.085	0.20
	lattice	0.644		0.39	1.806		0.38

Table 4: American call option on single asset: r=0.05.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, D=5.0			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, D=5.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	22.018	0.173	0.61	22.023	0.153	0.73
	PASA	22.062	0.160	0.95	22.067	0.144	1.66
	ST	21.942	1.129	55.66	21.547	1.549	368.65
	SM	22.163	0.939	18.30	22.020	0.197	350.00
	DP1	22.197	0.271	1.78	21.961	0.236	4.27
	DP2	22.169	0.162	0.31	22.048	0.145	1.46
	lattice	22.132		0.39	22.042		0.38
K=90	SPSA	13.702	0.153	0.77	12.359	0.190	1.49
	PASA	13.760	0.157	1.14	13.466	0.142	1.92
	ST	13.760	0.861	56.27	13.044	1.539	368.15
	SM	13.891	1.031	24.25	13.437	0.221	377.91
	DP1	13.519	0.258	1.78	13.619	0.235	4.27
	DP2	13.730	0.156	0.39	13.466	0.144	1.51
	lattice	13.783		0.38	13.452		0.38
K=100	SPSA	8.119	0.160	1.09	7.506	0.170	1.68
	PASA	8.099	0.150	1.35	7.968	0.144	2.36
	ST	8.136	0.599	56.37	7.932	0.946	366.22
	SM	8.485	0.910	27.77	7.959	0.342	418.30
	DP1	8.009	0.258	1.80	7.961	0.256	4.27
	DP2	8.136	0.152	0.46	7.780	0.150	1.40
	lattice	8.098		0.38	7.881		0.38
K=110	SPSA	4.723	0.128	1.15	4.916	0.147	1.78
	PASA	4.764	0.132	1.43	4.979	0.139	2.52
	ST	4.765	0.367	56.43	4.897	0.653	366.96
	SM	4.926	0.462	28.56	4.984	0.277	430.29
	DP1	4.698	0.223	1.79	4.770	0.221	4.26
	DP2	4.764	0.135	0.50	4.795	0.140	1.58
	lattice	4.753		0.38	4.847		0.38
K=120	SPSA	2.740	0.103	1.17	3.037	0.112	1.81
	PASA	2.787	0.109	1.47	3.123	0.119	2.60
	ST	2.759	0.224	55.48	3.100	0.432	359.67
	SM	3.066	0.479	29.04	3.199	0.225	432.82
	DP1	3.068	0.214	1.80	3.059	0.202	4.27
	DP2	2.925	0.116	0.52	3.216	0.127	1.01
	lattice	2.791		0.39	3.075		0.38

Table 5: American call option on single asset: r=0.05.

		t_i=0, 0.25, 0.5 T=0.5, D=1.0			t_i=0, 0.5, 1.0 T=1.0, D=2.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	22.475	0.161	0.55	24.379	0.191	0.57
	PASA	22.739	0.189	0.45	25.500	0.258	0.44
	ST	22.588	0.534	0.88	25.300	0.588	0.88
	SM	23.053	0.735	13.94	25.760	0.782	13.94
	DP1	22.385	0.318	0.38	24.780	0.408	0.38
	DP2	22.764	0.194	0.28	25.523	0.266	0.32
	lattice	22.696		0.38	25.456		0.39
K=90	SPSA	13.968	0.160	0.60	17.302	0.216	0.59
	PASA	14.100	0.174	0.45	17.666	0.240	0.44
	ST	13.954	0.360	0.88	17.476	0.432	0.89
	SM	14.177	0.429	13.91	17.889	0.721	13.89
	DP1	13.952	0.292	0.37	17.256	0.398	0.38
	DP2	14.123	0.177	0.43	17.782	0.246	0.33
	lattice	14.049		0.39	17.600		0.38
K=100	SPSA	7.241	0.130	0.61	11.015	0.191	0.60
	PASA	7.325	0.139	0.45	11.269	0.208	0.44
	ST	7.263	0.214	0.89	11.145	0.307	0.89
	SM	7.306	0.203	13.80	11.175	0.297	13.84
	DP1	7.204	0.234	0.38	11.094	0.343	0.39
	DP2	7.136	0.137	0.31	11.166	0.209	0.33
	lattice	7.312		0.38	11.223		0.39
K=110	SPSA	3.173	0.094	0.82	6.545	0.155	0.60
	PASA	3.142	0.095	0.45	6.610	0.166	0.45
	ST	3.139	0.119	0.87	6.579	0.209	0.88
	SM	3.245	0.260	13.76	6.873	0.507	13.83
	DP1	3.174	0.160	0.37	6.463	0.276	0.37
	DP2	3.078	0.094	0.28	6.536	0.167	0.33
	lattice	3.148		0.38	6.602		0.38
K=120	SPSA	1.114	0.053	0.80	3.540	0.116	0.59
	PASA	1.117	0.056	0.45	3.589	0.124	0.44
	ST	1.134	0.067	0.85	3.610	0.147	0.88
	SM	1.078	0.021	13.71	3.563	0.149	13.76
	DP1	1.156	0.094	0.37	3.349	0.202	0.38
	DP2	1.104	0.056	0.34	3.662	0.127	0.34
	lattice	1.129		0.39	3.607		0.38

Table 6: American call option on single asset: $r = 0.09$.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, D=2.0			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, D=2.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	28.015	0.295	1.09	31.315	0.380	1.78
	PASA	28.507	0.324	1.34	31.408	0.380	2.40
	ST	28.692	1.325	57.67	30.695	2.641	381.07
	SM	29.725	2.010	28.44	31.890	1.196	430.68
	DP1	29.224	0.614	1.79	31.581	0.698	4.27
	DP2	29.117	0.361	0.87	31.645	0.429	1.39
	lattice	28.791		0.38	31.628		0.39
K=90	SPSA	21.281	0.283	1.14	24.982	0.364	1.83
	PASA	21.959	0.318	1.42	25.340	0.375	2.56
	ST	21.914	0.992	59.07	24.826	2.009	381.41
	SM	22.804	1.578	28.99	25.586	0.786	432.65
	DP1	21.963	0.555	1.79	25.149	0.663	4.27
	DP2	21.657	0.331	0.77	25.816	0.412	1.98
	lattice	22.004		0.38	25.549		0.38
K=100	SPSA	15.576	0.257	1.17	19.978	0.364	1.88
	PASA	16.237	0.294	1.46	20.186	0.358	2.70
	ST	16.185	0.728	60.58	19.678	1.575	380.88
	SM	16.899	1.467	29.05	20.444	0.752	433.30
	DP1	16.049	0.500	1.78	20.393	0.625	4.28
	DP2	16.323	0.304	0.73	20.150	0.380	1.62
	lattice	16.263		0.38	20.267		0.38
K=110	SPSA	11.304	0.239	1.34	15.187	0.291	1.79
	PASA	11.597	0.259	1.47	15.802	0.336	2.75
	ST	11.591	0.523	57.74	15.383	1.210	379.01
	SM	11.845	0.902	29.41	16.091	0.772	433.50
	DP1	11.515	0.442	1.78	15.655	0.575	4.26
	DP2	11.603	0.267	0.80	16.051	0.349	1.54
	lattice	11.663		0.38	15.820		0.38
K=120	SPSA	7.896	0.202	1.20	11.840	0.280	1.81
	PASA	8.182	0.224	1.48	12.200	0.302	2.77
	ST	8.070	0.374	57.24	11.853	0.944	380.21
	SM	8.444	0.906	32.05	12.267	0.550	433.56
	DP1	7.923	0.370	1.80	12.154	0.505	4.26
	DP2	8.273	0.234	0.73	12.052	0.312	2.41
	lattice	8.141		0.39	12.181		0.38

Table 7: American call option on single asset: r=0.09.

		t_i=0, 0.25, 0.5 T=0.5, D=2.5			t_i=0, 0.5, 1.0 T=1.0, D=5.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	21.324	0.185	0.66	23.500	0.180	0.41
	PASA	21.829	0.134	0.40	23.709	0.179	0.39
	ST	21.747	0.596	0.84	23.640	0.578	0.83
	SM	22.115	0.593	13.79	24.013	0.737	13.77
	DP1	21.994	0.226	0.37	23.536	0.296	0.38
	DP2	21.715	0.134	0.14	23.873	0.180	0.15
	lattice	21.843		0.38	23.727		0.39
K=90	SPSA	12.819	0.125	0.47	15.256	0.165	0.48
	PASA	12.916	0.135	0.41	15.400	0.177	0.42
	ST	12.940	0.423	0.87	15.444	0.505	0.86
	SM	13.337	0.678	13.83	15.668	0.575	13.80
	DP1	13.234	0.236	0.39	15.636	0.313	0.38
	DP2	12.983	0.141	0.17	15.420	0.177	0.18
	lattice	12.927		0.38	15.423		0.39
K=100	SPSA	6.371	0.118	0.60	9.234	0.174	0.62
	PASA	6.383	0.122	0.44	9.216	0.167	0.44
	ST	6.363	0.238	0.89	9.207	0.360	0.88
	SM	6.474	0.236	13.79	9.368	0.428	13.77
	DP1	6.459	0.217	0.37	9.288	0.268	0.38
	DP2	6.282	0.122	0.18	9.043	0.170	0.19
	lattice	6.357		0.38	9.189		0.38
K=110	SPSA	2.593	0.078	0.59	4.912	0.117	0.58
	PASA	2.575	0.082	0.45	5.068	0.135	0.44
	ST	2.581	0.116	0.92	5.077	0.214	0.89
	SM	2.652	0.215	13.78	5.284	0.405	13.75
	DP1	2.637	0.144	0.39	5.260	0.228	0.36
	DP2	2.650	0.088	0.22	5.021	0.138	0.17
	lattice	2.580		0.38	5.075		0.38
K=120	SPSA	0.865	0.046	0.59	2.446	0.086	0.58
	PASA	0.850	0.047	0.44	2.575	0.101	0.44
	ST	0.875	0.058	0.91	2.609	0.132	0.88
	SM	0.854	0.037	13.72	2.744	0.302	13.72
	DP1	0.901	0.083	0.38	2.688	0.173	0.38
	DP2	0.890	0.050	0.22	2.690	0.108	0.25
	lattice	0.866		0.39	2.598		0.38

Table 8: American call option on single asset: r=0.09.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, D=5.0			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, D=5.0		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	23.100	0.222	0.84	22.217	0.252	1.42
	PASA	23.814	0.164	0.93	23.589	0.149	1.53
	ST	23.917	1.219	56.23	23.533	2.027	372.30
	SM	24.021	1.364	22.56	23.699	0.574	360.94
	DP1	23.543	0.292	1.77	23.761	0.249	4.27
	DP2	23.789	0.166	0.56	23.634	0.155	1.79
	lattice	23.764		0.39	23.683		0.39
K=90	SPSA	15.497	0.254	1.17	15.892	0.234	1.61
	PASA	15.890	0.185	1.18	16.130	0.179	1.95
	ST	16.335	1.028	56.85	16.314	1.686	376.81
	SM	16.738	1.364	26.90	16.257	0.849	416.56
	DP1	16.005	0.301	1.78	16.380	0.359	4.26
	DP2	15.899	0.196	0.65	16.064	0.189	1.92
	lattice	15.999		0.39	16.045		0.39
K=100	SPSA	10.617	0.201	1.14	11.221	0.237	1.81
	PASA	10.647	0.191	1.36	11.196	0.201	2.27
	ST	10.828	0.745	57.15	11.337	1.290	377.86
	SM	11.329	0.972	28.50	11.597	0.644	428.85
	DP1	10.569	0.346	1.77	11.338	0.414	4.26
	DP2	10.605	0.203	0.47	11.136	0.214	2.47
	lattice	10.647		0.39	11.203		0.38
K=110	SPSA	7.000	0.177	1.18	7.881	0.197	1.79
	PASA	7.000	0.172	1.42	7.993	0.191	2.44
	ST	7.068	0.468	56.92	7.980	0.915	377.46
	SM	7.480	0.832	28.85	8.234	0.520	432.67
	DP1	6.961	0.319	1.79	8.163	0.372	4.27
	DP2	6.954	0.187	0.54	8.158	0.219	2.49
	lattice	7.012		0.38	7.953		0.38
K=120	SPSA	4.260	0.132	1.17	5.547	0.175	2.39
	PASA	4.501	0.146	1.46	5.642	0.172	2.55
	ST	4.507	0.299	56.40	5.614	0.641	371.94
	SM	4.896	0.635	29.11	5.884	0.357	433.89
	DP1	4.579	0.273	1.77	5.706	0.294	4.29
	DP2	4.471	0.164	0.79	5.646	0.195	1.57
	lattice	4.530		0.38	5.647		0.39

Table 9: American call option on single asset: r=0.09.

		t_i=0, 0.25, 0.5 T=0.5, δ=0.04			t_i=0, 0.5, 1.0 T=1.0, δ=0.04		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	20.080	0.138	0.42	20.806	0.188	0.45
	PASA	20.297	0.153	0.41	21.112	0.213	0.41
	ST	20.441	0.526	0.85	20.966	0.941	0.86
	SM	20.571	0.550	12.05	21.364	0.732	12.79
	DP1	20.472	0.306	0.39	21.344	0.393	0.38
	DP2	20.312	0.151	0.12	21.089	0.228	0.16
	lattice	20.311		0.39	21.103		0.38
K=90	SPSA	11.751	0.134	0.54	13.263	0.171	0.51
	PASA	11.918	0.155	0.43	13.708	0.204	0.43
	ST	11.870	0.383	0.88	13.643	0.453	0.88
	SM	12.045	0.391	13.58	13.799	0.462	13.69
	DP1	11.855	0.240	0.35	13.644	0.354	0.36
	DP2	11.904	0.160	0.16	13.437	0.208	0.15
	lattice	11.877		0.38	13.663		0.38
K=100	SPSA	5.797	0.112	0.57	7.854	0.145	0.55
	PASA	5.781	0.122	0.44	8.103	0.172	0.44
	ST	5.734	0.195	0.86	8.072	0.288	0.88
	SM	5.805	0.226	13.53	8.206	0.435	13.64
	DP1	5.688	0.206	0.39	7.949	0.296	0.37
	DP2	5.708	0.122	0.14	8.186	0.180	0.15
	lattice	5.759		0.39	8.106		0.38
K=110	SPSA	2.294	0.074	0.59	4.241	0.114	0.58
	PASA	2.285	0.080	0.42	4.429	0.133	0.44
	ST	2.278	0.097	0.88	4.395	0.185	0.87
	SM	2.283	0.120	13.50	4.458	0.263	13.59
	DP1	2.320	0.140	0.38	4.336	0.226	0.36
	DP2	2.323	0.084	0.18	4.466	0.139	0.15
	lattice	2.294		0.38	4.432		0.38
K=120	SPSA	0.686	0.039	0.59	2.172	0.086	0.58
	PASA	0.741	0.045	0.44	2.239	0.096	0.4
	ST	0.754	0.052	0.86	2.238	0.112	0.86
	SM	0.757	0.052	13.45	2.270	0.169	13.58
	DP1	0.750	0.075	0.38	2.164	0.157	0.38
	DP2	0.781	0.049	0.15	2.188	0.098	0.15
	lattice	0.759		0.38	2.250		0.38

Table 10: American call option on single asset: r=0.05.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, δ=0.04			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, δ=0.04		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	22.241	0.214	0.82	24.060	0.277	1.38
	PASA	22.558	0.253	1.14	24.096	0.277	2.01
	ST	22.641	1.553	56.83	24.250	2.868	375.06
	SM	23.864	1.799	27.45	24.794	1.305	432.41
	DP1	22.548	0.480	1.70	23.792	0.454	4.17
	DP2	22.694	0.289	0.44	24.008	0.311	0.89
	lattice	22.743		0.39	24.115		0.39
K=90	SPSA	16.030	0.224	1.02	17.983	0.292	1.69
	PASA	16.341	0.248	1.27	18.183	0.278	2.20
	ST	16.127	1.074	57.08	18.576	1.882	374.70
	SM	17.204	1.510	28.29	18.550	0.742	435.87
	DP1	16.389	0.421	1.77	18.299	0.487	4.21
	DP2	16.381	0.269	0.78	18.351	0.304	0.63
	lattice	16.292		0.39	18.232		0.38
K=100	SPSA	11.024	0.196	1.08	13.686	0.288	1.78
	PASA	11.327	0.227	1.34	13.648	0.262	2.34
	ST	11.106	0.734	57.04	13.763	1.434	372.92
	SM	12.039	1.279	28.80	13.940	0.667	440.80
	DP1	11.304	0.410	1.79	13.389	0.480	4.24
	DP2	11.427	0.253	0.53	13.326	0.288	0.92
	lattice	11.280		0.39	13.552		0.38
K=110	SPSA	7.203	0.165	1.13	9.758	0.221	2.15
	PASA	7.587	0.195	1.38	9.99	0.239	2.48
	ST	7.431	0.490	56.66	10.062	1.070	370.33
	SM	7.986	0.916	28.93	10.323	0.541	432.25
	DP1	7.478	0.339	1.76	10.727	0.448	4.24
	DP2	7.823	0.209	0.45	9.963	0.263	1.41
	lattice	7.585		0.38	9.936		0.37
K=120	SPSA	4.791	0.149	1.18	7.157	0.213	1.83
	PASA	4.939	0.164	1.410	7.262	0.212	2.53
	ST	4.840	0.331	56.08	7.248	0.806	367.12
	SM	5.170	0.565	29.13	7.461	0.432	443.13
	DP1	5.091	0.298	1.67	7.234	0.406	4.26
	DP2	5.012	0.177	0.35	7.213	0.233	1.41
	lattice	4.968		0.39	7.197		0.38

Table 11: American call option on single asset: r=0.05.

		t_i=0, 0.25, 0.5 T=0.5, δ=0.10			t_i=0, 0.5, 1.0 T=1.0, δ=0.10		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	20.000		0.41	20.000		0.50
	PASA	20.000		0.40	20.000		0.40
	ST	20.056	0.092	0.82	20.050	0.091	0.84
	SM	20.052	0.092	7.72	20.017	0.036	7.34
	DP1	20.000		0.38	20.000		0.37
	DP2	20.000		0.10	20.000		0.11
	lattice	20.000		0.38	20.000		0.38
K=90	SPSA	10.150	0.125	0.51	10.459	0.152	0.49
	PASA	10.167	0.124	0.41	10.584	0.159	0.42
	ST	10.272	0.335	0.85	10.491	0.535	0.86
	SM	10.249	0.290	11.21	10.731	0.672	12.4
	DP1	10.237	0.209	0.37	10.546	0.273	0.37
	DP2	10.288	0.124	0.12	10.551	0.162	0.12
	lattice	10.180		0.38	10.611		0.38
K=100	SPSA	4.471	0.096	0.58	5.668	0.128	0.57
	PASA	4.463	0.098	0.43	5.651	0.129	0.40
	ST	4.413	0.185	0.89	5.584	0.240	0.87
	SM	4.511	0.257	13.52	5.741	0.354	13.66
	DP1	4.438	0.159	0.36	5.727	0.217	0.37
	DP2	4.541	0.097	0.14	5.779	0.131	0.12
	lattice	4.447		0.39	5.634		0.38
K=110	SPSA	1.561	0.057	0.58	2.766	0.093	0.58
	PASA	1.578	0.062	0.45	2.743	0.095	0.44
	ST	1.576	0.087	0.87	2.725	0.146	0.87
	SM	1.560	0.089	13.48	2.772	0.173	13.62
	DP1	1.621	0.105	0.37	2.752	0.167	0.36
	DP2	1.584	0.064	0.14	2.794	0.097	0.13
	lattice	1.586		0.38	2.747		0.38
K=120	SPSA	0.457	0.032	0.58	1.259	0.066	0.58
	PASA	0.468	0.035	0.45	1.226	0.066	0.44
	ST	0.474	0.040	0.84	1.252	0.088	0.86
	SM	0.499	0.047	13.45	1.251	0.089	13.56
	DP1	0.462	0.056	0.37	1.225	0.115	0.38
	DP2	0.468	0.035	0.13	1.239	0.068	0.12
	lattice	0.473		0.38	1.252		0.38

Table 12: American call option on single asset: r=0.05.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, δ=0.10			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, δ=0.10		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	20.000		0.75	20.000		0.98
	PASA	20.000		1.03	20.000		1.84
	ST	20.268	0.354	55.90	20.557	0.735	367.82
	SM	20.105	0.166	13.40	20.004	0.011	324.56
	DP1	20.000		1.74	20.000		4.31
	DP2	20.000		0.24	20.000		0.43
	lattice	20.000		0.39	20.000		0.38
K=90	SPSA	11.751	0.182	0.98	12.225	0.194	1.44
	PASA	11.603	0.173	1.19	12.362	0.182	2.19
	ST	11.445	0.888	58.21	11.945	1.625	366.59
	SM	11.710	0.922	25.75	12.234	0.349	404.70
	DP1	11.721	0.290	1.73	12.124	0.309	4.31
	DP2	11.696	0.173	0.28	12.122	0.181	0.65
	lattice	11.654		0.38	12.244		0.38
K=100	SPSA	6.935	0.165	1.13	7.780	0.156	1.56
	PASA	7.018	0.152	1.32	8.016	0.168	2.40
	ST	6.853	0.521	55.97	7.839	0.924	364.04
	SM	7.114	0.644	27.80	7.830	0.394	422.49
	DP1	7.213	0.247	1.69	7.986	0.300	4.25
	DP2	7.059	0.151	0.29	7.770	0.164	0.76
	lattice	7.060		0.38	7.863		0.38
K=110	SPSA	4.150	0.124	1.39	4.926	0.135	1.71
	PASA	4.136	0.127	1.39	5.154	0.145	2.58
	ST	4.009	0.345	55.54	5.019	0.645	360.99
	SM	4.417	0.723	28.74	5.063	0.387	428.80
	DP1	4.188	0.221	1.77	5.201	0.258	4.26
	DP2	4.167	0.127	0.34	5.097	0.147	0.56
	lattice	4.183		0.39	5.040		0.37
K=120	SPSA	2.273	0.086	1.15	3.206	0.115	1.78
	PASA	2.433	0.102	1.42	3.248	0.123	2.63
	ST	2.336	0.214	54.89	3.199	0.449	357.95
	SM	2.559	0.388	29.05	3.298	0.225	431.38
	DP1	2.380	0.182	1.76	3.254	0.226	4.24
	DP2	2.446	0.103	0.34	3.185	0.124	0.56
	lattice	2.441		0.39	3.234		0.38

Table 13: American call option on single asset: r=0.05.

		t_i=0, 0.25, 0.5 T=0.5, δ=0.04			t_i=0, 0.5, 1.0 T=1.0, δ=0.04		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	20.882	0.139	0.41	22.989	0.201	0.50
	PASA	21.755	0.185	0.45	23.644	0.252	0.44
	ST	21.604	0.532	0.87	23.468	0.579	0.87
	SM	22.039	0.729	13.77	23.843	0.732	13.72
	DP1	21.625	0.314	0.37	23.578	0.423	0.36
	DP2	21.777	0.193	0.20	23.377	0.261	0.18
	lattice	21.735		0.39	23.625		0.39
K=90	SPSA	12.953	0.143	0.57	15.736	0.207	0.60
	PASA	13.272	0.169	0.44	16.093	0.230	0.45
	ST	13.139	0.328	0.87	15.911	0.406	0.87
	SM	13.250	0.357	13.63	16.022	0.499	13.71
	DP1	13.356	0.283	0.37	15.954	0.392	0.38
	DP2	13.257	0.172	0.22	15.991	0.235	0.19
	lattice	13.231		0.38	16.046		0.38
K=100	SPSA	6.725	0.127	0.61	9.718	0.171	0.59
	PASA	6.764	0.133	0.44	10.075	0.196	0.44
	ST	6.688	0.194	0.89	9.949	0.278	0.88
	SM	6.671	0.178	13.56	10.035	0.377	13.65
	DP1	6.728	0.227	0.36	10.052	0.333	0.37
	DP2	6.763	0.136	0.15	9.997	0.199	0.19
	lattice	6.751		0.38	10.039		0.38
K=110	SPSA	2.849	0.086	0.60	5.702	0.139	0.59
	PASA	2.845	0.090	0.44	5.808	0.157	0.44
	ST	2.826	0.103	0.88	5.756	0.183	0.90
	SM	2.954	0.239	13.54	5.807	0.289	13.61
	DP1	2.830	0.146	0.36	5.689	0.258	0.37
	DP2	2.897	0.092	0.20	5.697	0.156	0.21
	lattice	2.850		0.38	5.805		0.39
K=120	SPSA	0.930	0.046	0.58	2.973	0.102	0.72
	PASA	0.997	0.053	0.44	3.111	0.116	0.45
	ST	0.997	0.057	0.86	3.099	0.126	0.89
	SM	1.046	0.108	13.46	3.134	0.201	13.61
	DP1	1.029	0.091	0.36	3.162	0.196	0.38
	DP2	1.031	0.056	0.16	3.113	0.119	0.20
	lattice	1.002		0.38	3.121		0.39

Table 14: American call option on single asset: r=0.09.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, δ=0.04			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, δ=0.04		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	25.354	0.235	0.87	28.604	0.340	1.62
	PASA	26.634	0.311	1.28	29.184	0.352	2.22
	ST	26.403	1.371	57.37	29.421	2.466	378.27
	SM	27.519	1.574	27.89	30.172	1.302	439.54
	DP1	26.940	0.573	1.80	29.711	0.681	4.21
	DP2	27.057	0.346	0.53	29.504	0.407	2.81
	lattice	26.887		0.39	29.473		0.39
K=90	SPSA	19.625	0.264	1.09	23.360	0.341	1.79
	PASA	20.322	0.299	1.35	23.689	0.352	2.44
	ST	19.908	1.013	57.57	23.489	2.018	378.50
	SM	21.077	1.590	28.67	24.002	0.919	439.53
	DP1	20.395	0.526	1.73	24.013	0.647	4.20
	DP2	20.453	0.322	0.41	23.604	0.391	0.77
	lattice	20.340		0.38	23.599		0.39
K=100	SPSA	14.466	0.252	1.16	18.338	0.318	1.82
	PASA	14.861	0.274	1.38	18.607	0.333	2.52
	ST	14.538	0.725	57.59	18.481	1.568	377.77
	SM	15.769	1.681	29.13	18.944	0.684	430.94
	DP1	15.278	0.470	1.72	18.373	0.558	4.24
	DP2	15.128	0.296	0.56	18.660	0.362	1.72
	lattice	14.885		0.38	18.562		0.39
K=110	SPSA	10.461	0.237	1.20	14.144	0.284	2.24
	PASA	10.495	0.241	1.40	14.368	0.303	2.53
	ST	10.282	0.534	57.39	14.298	1.227	375.88
	SM	11.056	1.149	29.12	14.614	0.721	430.89
	DP1	10.591	0.423	1.78	14.281	0.532	4.29
	DP2	10.358	0.257	0.41	14.385	0.331	2.81
	lattice	10.576		0.39	14.381		0.39
K=120	SPSA	7.095	0.193	1.20	10.345	0.233	1.78
	PASA	7.276	0.205	1.42	10.914	0.272	2.60
	ST	7.089	0.384	56.86	10.911	0.959	373.13
	SM	7.380	0.673	29.13	11.106	0.467	431.21
	DP1	7.089	0.356	1.78	11.203	0.485	4.27
	DP2	7.431	0.222	0.56	10.954	0.299	2.71
	lattice	7.318		0.39	10.992		0.39

Table 15: American call option on single asset: r=0.09.

		t_i=0, 0.25, 0.5 T=0.5, δ=0.10			t_i=0, 0.5, 1.0 T=1.0, δ=0.10		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	20.000		0.45	20.000		0.48
	PASA	20.000		0.40	20.000		0.41
	ST	20.184	0.242	0.84	20.240	0.317	0.84
	SM	20.236	0.328	9.54	20.365	0.501	10.80
	DP1	20.000		0.37	20.000		0.38
	DP2	20.000		0.11	20.000		0.13
	lattice	20.000		0.38	20.000		0.39
K=90	SPSA	11.011	0.126	0.49	11.926	0.159	0.49
	PASA	11.095	0.134	0.42	12.184	0.174	0.42
	ST	10.968	0.536	0.86	12.069	0.496	0.86
	SM	11.125	0.467	12.52	12.250	0.513	13.53
	DP1	11.244	0.224	0.38	12.190	0.299	0.37
	DP2	11.163	0.136	0.13	12.096	0.173	0.12
	lattice	11.084		0.39	12.165		0.38
K=100	SPSA	5.220	0.110	0.60	6.996	0.153	0.59
	PASA	5.216	0.110	0.42	6.950	0.148	0.43
	ST	5.164	0.189	0.88	6.894	0.271	0.87
	SM	5.328	0.305	13.54	6.869	0.174	13.62
	DP1	5.247	0.194	0.37	6.885	0.232	0.36
	DP2	5.265	0.112	0.14	6.895	0.152	0.14
	lattice	5.180		0.38	6.929		0.39
K=110	SPSA	1.963	0.066	0.59	3.701	0.116	0.59
	PASA	1.989	0.072	0.46	3.634	0.114	0.45
	ST	1.982	0.091	0.87	3.625	0.164	0.87
	SM	1.971	0.083	13.49	3.656	0.192	13.59
	DP1	2.032	0.124	0.37	3.453	0.180	0.39
	DP2	2.021	0.075	0.16	3.608	0.114	0.14
	lattice	1.992		0.38	3.645		0.38
K=120	SPSA	0.618	0.037	0.58	1.775	0.079	0.59
	PASA	0.619	0.040	0.45	1.767	0.082	0.44
	ST	0.636	0.046	0.85	1.781	0.104	0.87
	SM	0.646	0.058	13.46	1.799	0.127	13.57
	DP1	0.644	0.067	0.37	1.808	0.145	0.36
	DP2	0.604	0.041	0.15	1.825	0.086	0.13
	lattice	0.638		0.39	1.787		0.38

Table 16: American call option on single asset: r=0.09.

		t_i=0, 0.5, 1.0, 1.5, 2.0 T=2.0, δ=0.10			t_i=0, 0.5, 1.0, ..., 3.0 T=3.0, δ=0.10		
		Price	C.I.	CPU	Price	C.I.	CPU
K=80	SPSA	20.059	0.231	0.93	20.514	0.248	1.72
	PASA	20.149	0.199	1.04	20.805	0.205	1.84
	ST	20.748	0.898	56.27	21.340	1.586	371.70
	SM	20.861	1.052	21.79	20.728	0.659	420.90
	DP1	20.399	0.347	1.74	21.162	0.379	4.27
	DP2	20.420	0.203	0.26	20.762	0.223	0.65
	lattice	20.281		0.38	20.844		0.39
K=90	SPSA	13.638	0.180	0.88	14.416	0.253	1.65
	PASA	13.727	0.199	1.21	14.861	0.211	2.13
	ST	13.497	1.013	56.63	14.808	1.680	370.88
	SM	14.411	1.522	27.32	14.806	0.542	422.53
	DP1	13.845	0.356	1.76	14.799	0.355	4.28
	DP2	13.704	0.203	0.28	14.563	0.216	0.54
	lattice	13.776		0.38	14.796		0.38
K=100	SPSA	9.030	0.191	1.14	10.268	0.228	1.76
	PASA	9.111	0.183	1.32	10.638	0.202	2.33
	ST	8.856	0.651	56.63	10.507	1.122	369.33
	SM	9.323	0.720	28.01	10.560	0.592	432.79
	DP1	9.204	0.297	1.76	10.468	0.348	4.21
	DP2	9.033	0.183	0.34	10.487	0.207	0.55
	lattice	9.071		0.38	10.397		0.39
K=110	SPSA	5.680	0.151	1.14	7.327	0.200	2.23
	PASA	5.830	0.156	1.37	7.418	0.184	2.44
	ST	5.671	0.426	56.21	7.326	0.819	366.57
	SM	6.132	0.715	28.85	7.401	0.456	437.05
	DP1	5.771	0.283	1.79	7.517	0.316	4.27
	DP2	5.822	0.162	0.33	7.253	0.188	0.57
	lattice	5.835		0.38	7.261		0.38
K=120	SPSA	3.570	0.119	1.16	4.946	0.171	1.83
	PASA	3.676	0.130	1.41	5.174	0.161	2.53
	ST	3.564	0.273	55.69	5.057	0.606	367.28
	SM	3.850	0.493	29.08	5.190	0.359	439.58
	DP1	3.594	0.225	1.75	5.109	0.290	4.27
	DP2	3.551	0.132	0.34	4.893	0.160	0.71
	lattice	3.674		0.39	5.038		0.38

Table 17: American call option on single asset: r=0.09.

	t_i=105, 120			t_i=105, 110, 115, 120			t_i=105, 108, ..., 120		
	Price	C.I.	CPU	Price	C.I.	CPU	Price	C.I.	CPU
K=90, σ=0.2									
SPSA	13.109	0.136	5.98	13.040	0.138	5.70	13.174	0.137	5.61
PASA	13.118	0.138	1.55	13.213	0.137	1.48	13.258	0.137	1.53
ST	12.943	0.204	2.26	13.215	0.191	21.66	13.170	0.189	344.35
DP1	12.857	0.168	2.11	12.569	0.166	5.80	12.783	0.168	12.26
DP2	12.976	0.136	1.93	13.238	0.138	2.22	13.251	0.138	4.69
K=90, σ=0.3									
SPSA	14.202	0.193	5.97	14.225	0.191	5.81	14.246	0.193	5.78
PASA	14.463	0.194	1.52	14.461	0.194	1.46	14.453	0.195	1.52
ST	14.178	0.284	2.28	14.528	0.269	21.77	14.522	0.266	345.81
DP1	13.825	0.233	2.10	13.836	0.232	5.56	13.970	0.234	11.28
DP2	14.255	0.193	1.95	14.691	0.196	2.33	14.647	0.194	4.67
K=95, σ=0.2									
SPSA	8.977	0.124	6.08	8.968	0.123	5.90	9.047	0.124	5.96
PASA	9.062	0.124	1.53	9.176	0.124	1.52	9.205	0.126	1.58
ST	8.901	0.182	2.26	9.117	0.172	21.59	9.116	0.170	343.32
DP1	8.835	0.152	2.05	8.813	0.149	5.53	8.759	0.149	11.11
DP2	8.930	0.124	2.00	9.076	0.125	2.31	9.167	0.124	4.80
K=95, σ=0.3									
SPSA	10.654	0.174	6.09	10.733	0.176	5.96	10.830	0.177	5.92
PASA	10.778	0.178	1.55	10.844	0.176	1.52	10.683	0.175	1.59
ST	10.624	0.256	2.27	10.916	0.245	21.70	10.958	0.242	344.55
DP1	10.467	0.214	2.06	10.385	0.211	5.37	10.523	0.212	11.14
DP2	10.855	0.178	1.97	10.669	0.176	2.38	10.975	0.178	4.69
K=100, σ=0.2									
SPSA	5.665	0.104	6.21	5.736	0.105	7.53	5.692	0.104	6.35
PASA	5.673	0.104	1.53	5.704	0.104	1.55	5.847	0.106	1.58
ST	5.596	0.150	2.26	5.768	0.145	21.54	5.806	0.143	343.12
DP1	5.532	0.127	2.01	5.541	0.125	5.29	5.512	0.125	10.71
DP2	5.674	0.105	2.10	5.821	0.106	2.40	5.781	0.107	4.77
K=100, σ=0.3									
SPSA	7.676	0.156	6.16	7.925	0.156	6.17	7.805	0.155	6.30
PASA	7.757	0.155	1.56	7.814	0.155	1.51	7.839	0.156	1.56
ST	7.647	0.223	2.26	7.911	0.214	21.65	7.979	0.212	344.69
DP1	7.398	0.184	2.01	7.229	0.182	5.29	7.588	0.187	11.33
DP2	7.813	0.155	2.06	8.006	0.156	2.44	7.924	0.156	4.91

Table 18: American-Asian call option on single asset, r=0.09.

	t_i=105, 120			t_i=105, 110, 115, 120			t_i=105, 108, ..., 120		
	Price	C.I.	CPU	Price	C.I.	CPU	Price	C.I.	CPU
K=105, σ=0.2									
SPSA	3.222	0.080	6.31	3.283	0.083	6.44	3.196	0.081	6.65
PASA	3.263	0.082	1.63	3.375	0.083	1.56	3.395	0.084	1.62
ST	3.186	0.114	2.26	3.328	0.112	21.47	3.370	0.111	341.23
DP1	3.172	0.099	1.96	3.114	0.096	5.11	3.119	0.096	10.53
DP2	3.258	0.082	2.13	3.296	0.082	2.44	3.366	0.083	4.98
K=105, σ=0.3									
SPSA	5.458	0.132	6.27	5.508	0.134	6.36	5.442	0.131	6.54
PASA	5.452	0.134	1.60	5.626	0.137	1.55	5.431	0.132	1.59
ST	5.292	0.186	2.27	5.525	0.182	21.64	5.593	0.181	342.40
DP1	5.234	0.162	1.97	5.257	0.158	5.06	5.220	0.159	10.74
DP2	5.381	0.132	2.10	5.555	0.135	2.52	5.489	0.136	4.75
K=110, σ=0.2									
SPSA	1.644	0.058	6.40	1.660	0.059	6.57	1.664	0.058	6.85
PASA	1.717	0.059	1.62	1.742	0.060	1.58	1.750	0.060	1.63
ST	1.638	0.079	2.25	1.742	0.079	21.40	1.782	0.080	339.23
DP1	1.629	0.072	1.92	1.640	0.071	5.11	1.638	0.071	10.32
DP2	1.676	0.060	2.18	1.765	0.061	2.52	1.694	0.059	5.03
K=110, σ=0.3									
SPSA	3.609	0.109	6.33	3.670	0.110	7.93	3.591	0.111	6.75
PASA	3.712	0.112	1.63	3.810	0.113	1.54	3.700	0.111	1.60
ST	3.521	0.151	2.27	3.716	0.149	21.58	3.785	0.149	341.74
DP1	3.585	0.134	1.96	3.504	0.131	5.09	3.483	0.130	11.22
DP2	3.637	0.109	2.12	3.809	0.113	2.52	3.706	0.112	4.85

Table 19: American-Asian call option on single asset r=0.09 (continued).

	t_i=0, 0.5			t_i=0, 0.25, 0.5		
	Price	C.I.	CPU	Price	C.I.	CPU
SPSA	8.364	0.175	1.62	8.488	0.154	1.07
ST	8.458	0.278	0.05	8.577	0.290	1.55
DP1	*			8.595	0.276	0.40
DP2	8.401	0.175	0.14	8.565	0.166	0.36
	t_i=0, 1/6, 1/3, 1/2			t_i=0, 1/8, 1/4, 3/8, 1/2		
	Price	C.I.	CPU	Price	C.I.	CPU
SPSA	8.623	0.154	1.43	8.298	0.134	1.64
ST	8.667	0.279	70.510	8.827	0.489	88.50
DP1	8.807	0.271	0.98	8.676	0.273	1.78
DP2	8.877	0.163	0.54	8.734	0.159	0.73

Table 20: American put option on jump-diffusion process $S_0 = K$=100; r=0.10.
$*$ indicates method not applicable to this case.

# assets		Price S_0=70	C.I.	CPU	Price S_0=80	C.I.	CPU	Price S_0=90	C.I.	CPU
	SPSA	0.221	0.023	1.76	1.252	0.060	1.76	4.064	0.115	1.76
	ST	0.234	0.019	83.36	1.261	0.061	85.17	4.076	0.158	86.15
2	SM	0.238	0.034	316.54	1.248	0.121	316.55	4.107	0.312	316.55
	DP1	0.214	0.027	2.51	1.244	0.073	2.52	3.916	0.140	2.67
	DP2	0.242	0.026	0.72	1.251	0.059	0.71	4.030	0.107	0.71
	true	0.237			1.259			4.077		
	SPSA	0.521	0.037	3.88	2.656	0.091	3.90	7.674	0.157	3.92
	ST	0.556	0.027	404.05	2.734	0.088	405.88	7.896	0.203	406.20
5	SM	0.561	0.094	626.50	2.791	0.309	626.05	8.135	0.719	624.83
	DP1	0.469	0.037	4.94	2.612	0.100	4.99	7.366	0.166	5.29
	DP2	0.569	0.039	1.82	2.603	0.088	1.82	7.876	0.157	1.86
	BG	0.55	0.003		2.7	0.02		7.8	0.1	
		S_0=100			S_0=110			S_0=120		
	SPSA	9.160	0.174	1.75	16.513	0.228	1.72	25.307	0.267	1.71
	ST	9.354	0.311	86.09	16.908	0.474	85.63	25.935	0.605	85.16
2	SM	9.390	0.335	314.73	16.835	0.518	311.25	25.965	0.764	309.55
	DP1	9.333	0.211	3.13	16.586	0.263	3.45	26.031	0.292	3.59
	DP2	9.341	0.155	0.74	16.909	0.195	0.67	25.959	0.231	0.69
	true	9.361			16.924			25.980		
	SPSA	15.590	0.215	3.91	25.344	0.256	3.92	35.872	0.284	3.90
	ST	16.021	0.336	313.54	25.973	0.435	405.26	36.692	0.503	405.21
5	SM	16.524	1.360	622.93	26.804	1.755	620.32	37.900	2.417	619.26
	DP1	15.512	0.237	5.95	25.508	0.284	6.19	36.154	0.320	6.20
	DP2	15.585	0.208	1.86	25.739	0.255	1.87	35.980	0.278	1.84
	BG	15.9	0.2		25.8	0.2		36.5	0.3	
		S_0=130		
	SPSA	35.166	0.296	1.69
	ST	35.687	0.710	84.77
2	SM	35.805	0.944	307.62
	DP1	35.819	0.329	3.63
	DP2	35.660	0.237	0.63
	true	35.763		
	SPSA	46.712	0.309	3.86
	ST	47.649	0.561	405.30
5	SM	49.366	3.128	617.68
	DP1	47.117	0.337	6.16
	DP2	46.794	0.293	1.81
	BG	47.4	0.3	

Table 21: American max call option on multiple assets: r=0.05; t_i=0, 1/3, 2/3, 1.

References

[1] Barraquand, J., and Martineau, D., "Numerical Valuation of High Dimensional Multivariate American Securities," *Journal of Financial and Quantitative Analysis*, **30**, 383-405, 1995.

[2] Bossaerts, P., "Simulation Estimators of Optimal Early Exercise," working paper, Graduate School of Industrial Administration, Carnegie-Mellon University, 1989.

[3] Boyle, P.P., "Options: A Monte Carlo Approach," *Journal of Financial Economics,* **4,** pp. 323-338, 1977.

[4] Boyle, P.P., Broadie, M., and Glasserman, P., "Simulation Methods for Security Pricing," *Journal of Economic Dynamics and Control*, Vol.21, 1267-1321, 1997.

[5] Broadie, M. and J. Detemple (1996), "American Option Valuation: New Bounds, Approximations, and a Comparison of Existing Methods," *The Review of Financial Studies*, Winter 1996, Vol. 9, No.4, 1211-1250.

[6] Broadie, M. and J. Detemple, "The Valuation of American Options on Multiple Assets," *Mathematical Finance*, **7**, 241-286, 1997a.

[7] Broadie, M. and J. Detemple, "Recent Advances in Numerical Methods for Pricing Derivative Securities," in *Numerical Methods in Finance*, L.C.G. Rogers and D. Talay, eds., Cambridge University Press, 43-66, 1997b.

[8] Broadie, M., and Glasserman, P., "Estimating Security Price Derivatives Using Simulation," *Management Science*, **42**, 269-285, 1996.

[9] Broadie, M., and Glasserman, P., "Pricing American-Style Securities Using Simulation," *Journal of Economic Dynamics and Control*, Vol.21, No.8/9, 1323-1352, 1997a.

[10] Broadie, M., and Glasserman, P., "Monte Carlo Methods for Pricing High-Dimensional American Options: An Overview," *Net Exposure*, Issue 3, 15-37, 1997b.

[11] Broadie, M., and Glasserman, P., "A Stochastic Mesh Method for Pricing High-Dimensional American Options," working paper, 1998.

[12] Broadie, M., Glasserman, P., and Jain, G., "Enhanced Monte Carlo Estimates for American Option Prices," *Journal of Derivatives*, **5**, 25-44, 1997.

[13] Carriere, J.F., "Valuation of the Early-Exercise Price for Derivative Securities using Simulations and Splines," *Insurance: Mathematics and Economics*, **19**, 19-30, 1996.

[14] Clewlow, L. and Strickland, C., "Pricing Interest Rate Exotics by Monte Carlo Simulation," *Monte Carlo: Methodologies and Applications for Pricing and Risk Management*, Dupire, B., Editor, Risk Publications, London, 1998.

[15] Dupire, Bruno, Consultant Editor, *Monte Carlo: Methodologies and Applications for Pricing and Risk Management*, Risk Publications, London, 1998.

[16] Fu, M.C., "Optimization via Simulation: A Review," *Annals of Operations Research*, Vol. 53, pp.199-248, 1994.

[17] Fu, M.C., and Hill, S.D., "Optimization of Discrete Event Systems via Simultaneous Perturbation Stochastic Approximation," *IIE Transactions*, Vol.29, No.3, 233-243, 1997.

[18] Fu, M.C., and Hu, J.Q., "Sensitivity Analysis for Monte Carlo Simulation of Option Pricing," *Probability in the Engineering and Information Sciences*, **9**, 417-446, 1995.

[19] Fu, M.C. and Hu, J.Q., *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, Kluwer Academic Publishers, 1997.

[20] Fu, M.C., Wu, R., Gürkan, G., and Demir, A.Y., "A Note on Perturbation Analysis Estimators for American-Style Options," *Probability in the Engineering and Information Sciences*, **14**, 385-392, 2000.

[21] Geske, R. and H. Johnson (1984), "The American Put Options Valued Analytically," *Journal of Finance*, **39**, 1511-1524.

[22] Glasserman, P., *Gradient Estimation Via Perturbation Analysis*, Kluwer Academic, 1991.

[23] Grant, D., Vora, G., and Weeks, D., "Simulation and the Early-Exercise Option Problem," *Journal of Financial Engineering*, Vol.5, No.3, 211-227, 1996.

[24] Grant, D., Vora, G., and Weeks, D., "Path-Dependent Options: Extending the Monte Carlo Simulation Approach," *Management Science*, **43**, 1589-1602, 1997.

[25] Gürkan, G., Özge, A., and Robinson, S., "Sample-Path Solutions of Stochastic Variational Inequalities, with Applications to Option Pricing," *Proceedings of the Winter Simulation Conference*, 337-344, 1996.

[26] Hull, J.C., *Options, Futures, and Other Derivative Securities,* Prentice Hall, 4th edition, 2000 (2nd edition, 1993).

[27] Ibanez, A. and Zapatero, F., "Monte Carlo Valuation of American Options Through Computation of the Optimal Exercise Frontier", Working Paper, 1999.

[28] H.J. Kushner and G.G. Yin, *Stochastic Approximation Algorithms and Applications*, Springer-Verlag, 1997.

[29] Longstaff, F.A. and Schwartz, E.S., "Valuing American Options by Simulation: A Simple Least-Squares Approach", *The Review of Financial Studies*, Spring 2001, Vol.14, No.1, 113-148.

[30] McDonald, R.L. and Schroder, M.D., "A Parity Result for American Options", *Journal of Computational Finance*, **1**, 5-13, 1998.

[31] Raymar, S., and M. Zwecher, "A Monte Carlo Valuation of American Call Options on the Maximum of Several Stocks," *Journal of Derivatives*, **5**, 7-23, 1997.

[32] Spall, J.C., "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Transactions on Automatic Control*, Vol. 37, No. 3, 332-341, 1992.

[33] Stoll, H.R. and Whaley, R.E., *Futures and Options*, South-Western, 1993.

[34] Su, Y., "Pricing American Options via Simulations: A Stochastic Optimization Approach," PhD Dissertation, University of Maryland, College Park, 2000.

[35] Tilley, J., "Valuing American Options in a Path Simulation Model," *Transactions of the Society of Actuaries,* **45,** pp.83-104, 1993.

[36] Welch, R.L. and Chen, D.M., "Static Optimization of American Contingent Claims," *Advances in Futures and Options Research,* **5,** pp. 175-184, 1991.

[37] Wu, R. and Fu, M.C., "Optimal Exercise Policies and Simulation-based Valuation for American-Asian Options," submitted to *Operations Research*, 2000.