JOIM
www.joim.com

JOURNAL OF INVESTMENT MANAGEMENT, Vol. 9, No. 2, (2011), pp. 88–106

SURVEYS AND CROSSOVERS

"Survey and Crossovers" provides surveys of the literature in investment management or short papers exemplifying advances in finance that arise from the confluence with other fields. This section acknowledges current trends in, and the cross-disciplinary nature of the investment management business, while directing the reader to interesting and important recent work.

RANDOM LATTICES FOR OPTION PRICING PROBLEMS IN FINANCE

Sanjiv R. Das[a]

While the use of Monte Carlo methods is well established for pricing derivatives, this paper focuses on a random-lattice approach, also known in the literature as the stochastic-mesh method. The method is reviewed here. We show that the method may be refined with an ad-hoc bias correction, that suitably adjusts these models for accuracy. The paper presents experimental results, related analysis, and a set of applications, demonstrating easy applicability to popular choices for option pricing stochastic processes. The flexibility and ease of implementation of this approach, as seen from the examples, suggests that this approach has wide practical applicability.

1 Introduction

Financial securities are called "derivatives" if their value is derived from some other primary "underlying" security or economic variable. For example, a "call" (C_0) option on a stock (S_0), where the subscript zero indicates "initial price" at time 0, is a contract where the buyer of the option receives at the maturity of the contract (i.e., at time T), the difference between the stock price (S_T) and a preset "strike" price (K), *if* this amount is positive. A put option is the converse contract and pays off when $K > S_T$. At T, the option buyer obtains a positive payoff with some probability. The payoff at maturity for calls is defined to be:

$$C_T = \max(0, S_T - K). \tag{1}$$

The option buyer pays an upfront premium to the option writer. We present an algorithm to determine as precisely as possible what the fair premium (C_0) should be.

The pricing of options requires assumptions about the stochastic process of the underlying security (a stock, for example), and a computation of the fair value of the option under strict economic assumptions which ensure that no arbitrages (i.e., "free

[a]Santa Clara University, Leavey School of Business, 500 El Camino Real, Santa Clara, CA 95053, USA.

lunches") are permitted. The price of a call option is given by:

$$C_0 = E_0^*[e^{-rT}\max(0, S_T - K)], \qquad (2)$$

where r is the market's risk free rate of interest, and the expectation $E^*(\cdot)$ is taken over the possible outcomes of S_T, and sometimes the paths of r as well. The probability measure under which E^* operates is known as the "risk-neutral" measure, and is derived from no-arbitrage principles. No discussion of this aspect of option pricing is offered here, and the reader is referred to the seminal work of Harrison and Kreps (1979) for a complete exposition. If the probability density of S_T under the risk-neutral measure is denoted as $f(S_T)$, then the pricing model comprises an integral as follows:

$$C_0 = \int_K^\infty e^{-rT}(S_T - K)\, f(S_T) dS_T. \qquad (3)$$

In a few limited cases, such as when S_T is log-normal this integral yields a closed form solution. Most often, numerical integration is required, leading to a search for fast, accurate algorithms.

Merton (1990) (Ch3) provides the groundwork for the theoretical validity of modeling security prices using continuous-time mathematics. Applicable numerical techniques usually consist of building a layered lattice of security prices depicting the evolution of prices in time, and performing the required computations on them. If we use a lattice approach, the continuous-time, continuous-space model is transformed into a discrete-time, discrete-space one, leading to approximation error. The error is usually mitigated by choosing a denser lattice representation for the stochastic process. The trade-off comes from the corresponding increase in computational effort of traversing a denser lattice.

The "lattice" is the generic term for any graph we build for the pricing of financial securities. Each lattice is a layered directed graph, where the nodes at each level represent the possible values of the underlying security in that period. The entire life of the option spans the time T. When time is discretized, we obtain a discrete time step h, indexed by a variable t, i.e., $\sum_t h(t) = T$. When h is uniform throughout the lattice, the number of levels is $d = T/h$. Only edges between successive levels are permitted, and each edge is labeled with the probability ($p_{ij}(t)$) of the corresponding change in the security price (from $S_i(t-1)$ to $S_j(t)$) in time period t.

There are two types of lattices we will consider. The first type of lattice only permits nodes to have in-degree[1] of exactly 1, and this lattice is also known as a "tree". The second type allows the in-degree to be greater than 1 for some or all nodes. Hence, there may be more than one path leading to a node, and this type of lattice is also commonly called a "recombining" tree (a misnomer) or simply referred to by the general term "lattice." The starting security price S_0 comprises the single "root" of the lattice, and the last level contains all the possible final outcomes of the stock price, which are the "leaves" of the lattice. Since the lattice represents a stochastic process, the sum of probabilities assigned to edges (which proxy

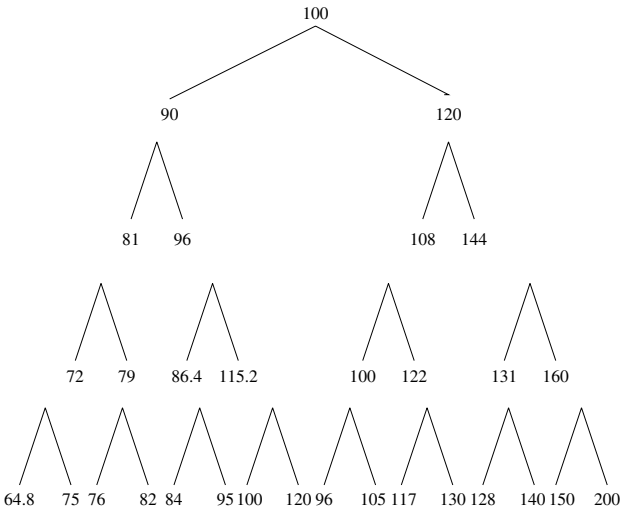

Figure 1 Stock price tree.

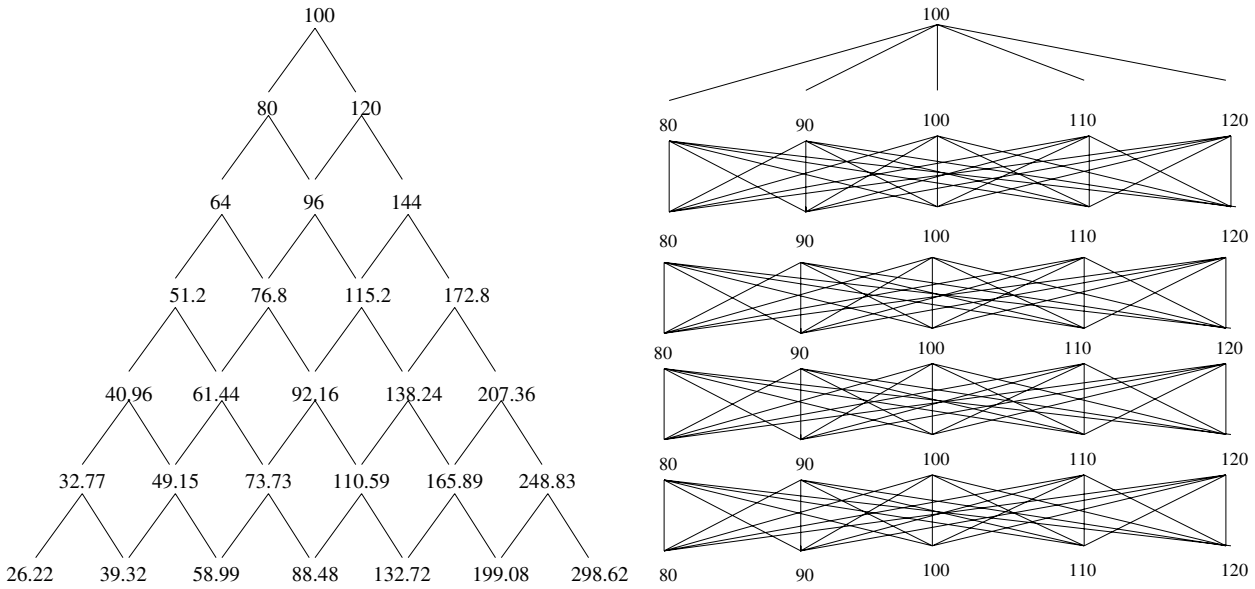

Figure 2 Stock price lattices.

for probabilities) emanating from every node is always 1, i.e., $\sum_j p_{ij}(t) = 1, \forall\, i, t$. Examples of both types of lattice are presented in Figs. 1 and 2. In Fig. 1, we present an example of a symmetric tree,[2] though asymmetric trees are also often used.

Once the graphical representation of the stochastic process on the lattice is ready, we can price the derivative security by computing the expected, discounted value of the payoffs at maturity under the risk-neutral measure. The lattice solution is a discretized version of Eq. (3):

$$C_0 = \sum_{l=1}^{L} e^{-rT} \max[0, S_l(T) - K] Pr(l), \quad (4)$$

where l indexes the leaves, L being the total number of leaves, and $Pr(l)$ is the probability of reaching leaf l on the lattice. Hence each leaf value is given by $e^{-rT}\max[0, S_l(T) - K] \times Pr(l)$, which embeds three components: (i) the discounting factor e^{-rT}, (ii) the terminal payoff $\max[0, S_l(T) - K]$, and (iii) the probability of the leaf $Pr(l)$.[3] If we denote d to be the depth of the tree, then $L = 2^d$ for a binary tree. The probability of each leaf occurrence will be the product of probabilities of all the edges on the path to the leaf, i.e., $Pr(l) = \prod_{(i,j)\in\Omega_l} p_{ij}$, where Ω_l is the set of edges on the path from the root to the leaf l.

The derivative security's value on the lattice can be computed by dynamic programming from the leaves to the root. We first compute the value $C_l(d) = \max[0, S_l(T) - K]$ at each leaf of the tree. To obtain values at nodes at level $(d-1)$, we weight each node at level d by the edge probabilities connecting the node i to the ensuing nodes. Hence, dynamic programming entails computing the value of each node at each level on the lattice as follows:

$$C_i(t-1) = e^{-rh}\left[\sum_j p_{ij}(t) C_j(t)\right], \quad \forall\, i, t \quad (5)$$

This eventually results in obtaining the desired value $C(0)$.

Via dynamic programming, each edge is explored only once, even though on a tree, the edge may

be on the path of more than one leaf. Hence, the computational effort on the tree is a function of the number of edges on the tree. If the width of the tree (i.e., the maximum number of nodes at each level) is m, then the computation time is $O(dm^2)$. In trees (as in Fig. 1), the width m grows exponentially in depth d, making it difficult to exhaustively enumerate the leaf values according to Eq. (4). In recombining lattices (as in Fig. 2), this problem is mitigated as the width m is bounded, or grows only linearly in depth d, and exhaustive enumeration may be feasible. Hence, recombining lattices result in polynomial time (in depth d) algorithms, whereas computing on trees results in exponential effort (in d).

The simplest form of derivative security does not permit the buyer to exercise his option prior to maturity. These contracts are known as "European" options and are to be contrasted with contracts which allow for premature exercise, known as "American" options. American options entail the solution of an *optimal stopping* problem, i.e., stopped random walks, requiring dynamic programming on the lattice. The earliest example of this type of algorithm was developed by Cox *et al.* (1979).

We present an algorithm for American options where the lattice is recombining (i.e., not a tree). Moreover, the lattice itself is generated by Monte Carlo simulation. We call these random lattices. We provide results on using random lattices for (i) *pricing* derivative securities, and (ii) implementing options models with *optimal stopping* (i.e., American option problems).[4] This research builds on earlier work where stochastic meshes have been used to represent stochastic processes to solve American option pricing problems.[11] The results here extend the same idea by providing different bounds for the optimal stopping problem using a bucketing argument, and a new correction for bias.

Say that it is possible to exercise a call option prior to maturity. Then, the value of the option may be higher than one which can be exercised only at maturity, since it affords greater flexibility to the holder of the option. In order to evaluate this option, we start working back along the lattice from the leaves. Each node is computed to be the expected value of the nodes that follow it (i.e., dynamic programming). If we compare this expected value of continuing along the tree with the value of immediate exercise, we are able to decide which strategy to adopt, i.e., to exercise or refrain from doing so. In any case, we choose the more lucrative of the two approaches and hence the value of that node will be the value of the better of the two choices. In this way, we solve the optimal stopping problem on a lattice.

There are two ways in which lattices may be created. The usual approach is to establish a deterministic rule for building the lattice so that it recombines. However, it may not always be possible to preserve the properties of the stochastic process on a recombining tree, and hence, there are many problems which have not yielded to deterministic lattice engineering. An alternate approach is to build the lattice randomly. This gives a very dense, recombining lattice allowing fast, feasible computation (see Figs. 2 and 3). We discuss the existing literature on this next.

1.1 Extant research

Monte Carlo simulation methods allow fast computation of high-dimensional problems. However, direct simulation does not admit the implementation of dynamic programming. By generating lattices randomly, we are able to combine the computational benefits of Monte Carlo simulation with the ability to implement dynamic programming on a lattice.

The method was first suggested in Tilley (1993), who proposed a bundling algorithm whereby the

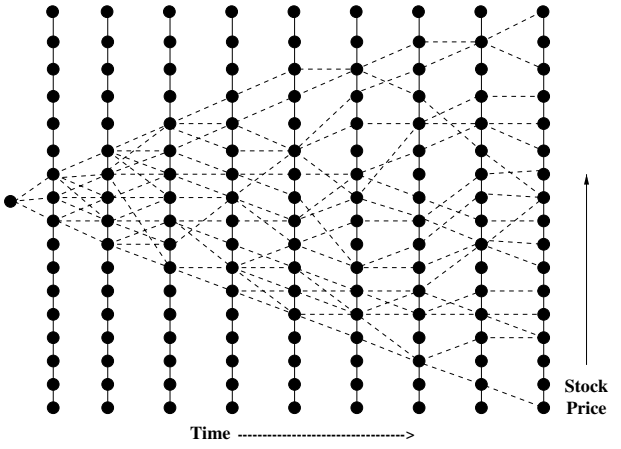

Figure 3 Randomly generated lattice.

results of a Monte Carlo simulation would be summarized by keeping track of the simulated paths in "bundles". This gives rise to the idea of running a Monte Carlo simulation on a restricted, finite state space, and by keeping track of the paths in bundles, we create a lattice which summarizes (with some error) the stochastic process for the underlying assets, so that a lattice can then be used for pricing. A review of this method, and other American option pricing techniques is available in the paper by Boyle *et al.* (1997).

In another enhancement, the work of Barraquand and Martineau (1995) explored the stratification of the state space by payoff and not by the underlying state variables. This injects substantial economy into the random lattice and speeds up computation. There are problems in which the payoff space is Markovian and it may not be necessary to maintain a lattice in the original state variables in order to determine the stopping time for the American option pricing problem. Boyle *et al.* (1997) argue that the stratification algorithm often leads to estimation error. They demonstrate that maintaining the state space in payoff terms may not lead to the correct conditional distribution, generating bias.

There are other biases too. Any lattice scheme for the pricing of American options results in estimators that are biased from "missing the boundary". The optimal stopping rule implicitly defines an "early exercise boundary" for the option. This is just the first passage time boundary for optimal stopping which exists under the optimal solution to the problem. If the trajectory of the stock price crosses the boundary, the option is exercised. In the case of a continuous state space, the option is exercised early when the stock price touches the boundary. However, in a lattice scheme, the stock price never just touches the boundary unless the lattice node resides exactly on the boundary, which occurs in the odd rare instance. Usually, the stock price crosses through the boundary and hence, the exercise value is overstated, leading to an bias in the estimator.

Figure 4 describes early exercise in American put options. Put options pay off when the stock price falls below the exercise price. Early exercise often occurs in the case of put options, which are said to be "in-the-money" when $S_t < K$. At any time, the payoff is given by $\max(0, K - S_t)$. If the stock price drops below K, then the holder of the option may wish to exercise early and cash in on the difference, i.e., $(K - S_T)$, rather than wait for a probabilistic further gain. Dynamic programming identifies the nodes at which this is optimal

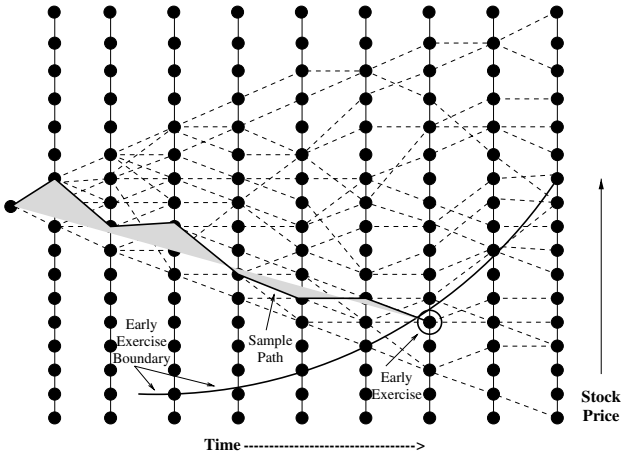

Figure 4 Depiction of early exercise and bias for American put options.

to do, i.e., those nodes below the early exercise boundary. Figure 4 shows the lattice with a single sample path through it which leads to an early exercise when the stock price crosses below the early exercise boundary. Since the node does not lie exactly on the boundary, the payoff $(K - S_T)$ is overstated by the distance from the boundary. This leads to the bias.

In the next few sections, we develop in detail some of the theory of the random lattice method and also provide many numerical experiments to show the efficacy of the method. This work follows on and is complementary to the original prior work of Broadie and Glasserman (1997a,b).

In particular, we highlight some of the new ideas introduced in this paper. First, we increase the accuracy of the scheme by using a pair of random lattices instead of a single random lattice. This so-called "antithetic-lattice" method provides considerable error-reduction. Second, we suggest a computationally facile approach which reduces the lattice bias problem. Third, we exploit the theory of dynamic programming to develop a lower bound based on the "smooth-pasting" conditions.[5] Finally, we show how to adjust the lattice to minimize the expected error from the scheme. We provide numerical examples for many classic models that our approach may be applied to.

Our algorithm is tested on some popular problems. We look at the pricing of Asian options (providing an alternative to the work of Aingworth *et al.* (2000)). We also compute the prices of GARCH options[6] and compare our results to those of Ritchken and Trevor (1999). And finally, we price options on stocks with stochastic volatility, corresponding to the works of Scott (1987), and Heston (1993). Thus, the random lattice algorithm is flexible, and we are able to replicate the results of the distinct approaches in the other papers mentioned above.

2 The random lattice algorithm

This methodology consists of a preprocessing step, in which the random lattice is built to embed the stochastic process, and a post-processing or pricing step.

2.1 Preprocessing

(1) *Bucketing*: the basic idea for the lattice is to restrict the stochastic process to a finite number of discrete values. This defines a set of "buckets" into which every stock price is assigned. We denote the number of buckets to be m, which is the *width* of the tree. The number of levels in the tree is d, its depth. Thus, the random lattice takes on a matrix structure, of size $m \times d$. A simple scheme for assigning buckets is to divide the difference between the minimum and maximum of the stock price by m, i.e., equal buckets. Bucketing leads to approximation error. The approximation error may be further reduced by choosing more complex bucketing schemes.

(2) *Lattice generation*: to generate the lattice, we conduct n root-leaf random walks through it, starting with the same value at the root for each path. If the lattice has depth d, we compute dn transitions. At each node, we generate the next stock price randomly using a discretized version of the stochastic process. For example, if the stock follows a geometric Brownian motion then the next stock price would be generated by drawing a random number $\epsilon \sim N(0, 1)$ and applying the following stochastic process:

$$S_t = S_0 \exp\left[\left(r - \frac{\sigma^2}{2}\right) t + \sigma\epsilon\sqrt{t}\right]. \quad (6)$$

Once we have obtained the next stock price, we round it up or down to the nearest bucket. Then we proceed to generate the next stock value and so on. For any single sample path, the walk makes d moves, i.e., the depth of

the lattice. Given the matrix structure of the lattice, at every level, we have m preceding nodes leading to m nodes in the next period. We index the points in time by t, $t = 0, 1, \ldots, d$, which defines the d levels on the tree. There is one edge for each pair of nodes in adjacent time periods. See Fig. 3 for a representation of the random lattice.

(3) *Lattice probabilities*: for every edge on the lattice from node i at time t to node j at time $t + 1$, we maintain a count of the number of traversals made via the edge, and denote the count as $c_t(i, j)$. We also keep track of the total number of out-traversals from node i at time t, which we denote as $c_t(i)$. Therefore,

$$c_t(i) = \sum_{j=1}^{m} c_t(i, j) \tag{7}$$

From this, we derive the transition probabilities $p_{ij}(t)$ (from the ith node at time t to the jth node at time $t + 1$) on the lattice, i.e.,

$$p_t(i, j) = \frac{c_t(i, j)}{c_t(i)} \tag{8}$$

Once we have the probabilities for the lattice, the stochastic process is in place. The lattice is recombining in the bucket structure. This step consists of making n walks through the lattice, with d steps per walk, resulting in a total processing time of $O(n \times d)$. In addition, the storage required is also polynomial in the depth and width of the tree, i.e., $O(dm^2)$. Note that n determines the accuracy with which the bucket to bucket transition weights are estimated.

2.2 *Postprocessing*

We first compute the final payoffs at the last level of the lattice. For example, if the lattice is used to price call options, the payoff at the dth level would be $\max(0, S_d - K)$, where K is the exercise price. The payoff values at each node define another $m \times d$ matrix, which we denote as $G(i, t)$, where i indexes the bucket and t indexes the time period on the lattice.

The price of the option at the root node is determined via dynamic programming on the lattice. Once the values at time d are computed from the payoff rule, the values at the preceding period $(d - 1)$ are computed to be the expected discounted value of the nodes at time d. In the case of American call options, where exercise of the option is possible before maturity, we check whether the value of immediate exercise would be greater than the value of continuing to hold the option for another period. The scheme is easily modified to accommodate this optimal stopping rule. Therefore, the dynamic programming step implements the following equation:

$$G(i, t) = \max\left[S_t - K, \sum_{j=1}^{m} e^{-rh} p_t(i, j) G(j, t + 1)\right], \quad \forall\, i = 1, \ldots, m. \tag{9}$$

where r is the discount rate, and h is the time interval. From the equation, it is clear that the value $G(i, t)$ is the better of the immediate value of early exercise $(S_t - K)$ and the expected discounted value based on successor nodes $G(j, t + 1), \forall\, j$. This step requires m^2 calculations, and since it is done over d periods, the total effort involved in postprocessing is $O(dm^2)$. (Note that this is usually $\min[O(dm^2), O(dn)]$, but usually $dm^2 < dn$.) The time consumed is usually less than this bound, since in many cases $c_t(i, j) = 0$. Therefore, before applying Eq. (9), we first check whether the out-degree value $c_t(i)$ is non-zero, and indeed, there are many nodes for which the out-degree is zero. In actual experiments, we found the postprocessing time to be a fraction of preprocessing time.

3 Error bounds for the fully polynomial randomized algorithm

Bucketing introduces approximation error. In this section, we characterize the error, and also explore modifications to the scheme to reduce the error.

3.1 Worst-case bounds

Bucketing requires the distribution of the stock price to be supported on m points, $x_1, x_2, \ldots, x_m$, where m is a reasonable number. Denote the probability measure on these points to be $f(x_i), i = 1, \ldots, m$. Because of rounding, bucketing results in a shifting of the support from its true points. The shift will be as much as half the width of the bucket, which we denote to be $\delta = \frac{S^{\max} - S^{\min}}{2m}$. Our payoff function is written as $g(x_i), i = 1, \ldots, m$. The expected value of the true payoff is $\sum_{i=1}^{m} g(x_i) f(x_i)$, $\sum_{i=1}^{m} f(x_i) = 1$ (for continuous probability functions we would write this as $\int g(x) f(x) dx$).

The error induced by bucketing may be as bad as a full $\frac{\delta}{2}$ shift in the support points (since rounding is made to the closest bucket point). If every bucket was at the maximum one-sided error, and $g(x_i)$ is assumed to be monotone, this would result in an expected value of $\sum_{i=1}^{m} g(x_i + \delta/2) f(x_i)$. The difference between this value and the true value is the approximation error:

$$\begin{aligned} &\text{Approximation Error} \\ &= \sum_{i=1}^{m} [g(x_i + \delta/2) - g(x_i)] f(x_i) \\ &\approx \frac{\delta}{2} \sum_{i=1}^{m} \frac{\partial g}{\partial x_i} f(x_i) \end{aligned} \tag{10}$$

where the last step applies a truncated Taylor series expansion for $g(x_i)$. For an option, the payoff function is $g(x) = (x-K)^+ = \max(0, x-K)$. Hence, the maximum value of $\frac{\partial g}{\partial x}$ is 1. Therefore, the worst case approximation error per period is

$$O\left(\frac{\delta}{2} \sum_{i=1}^{m} \frac{\partial g}{\partial x_i} f(x_i)\right) = O(\delta). \tag{11}$$

If there are d periods then the total error bound is $O(d\delta)$.

These error bounds are overly pessimistic. Note that:

(1) In many cases the error is exactly zero, in the regions where $\frac{\partial g}{\partial x} = 0$, because $g(x) = (x - K)^+ = 0$, which occurs when $x \leq K$.
(2) Except for the last level of the lattice the payoff at $t < d$ is always lower than $(x - K)^+$ because the amount has been discounted for $(d - t)$ periods already.

The error may be further reduced by improving the bucketing scheme, which we shall explore in subsequent sections.

3.2 Minimization of expected error

In this section we analyze the expected error from the scheme and show how it needs to be set up so as to reduce bias in the estimator. We begin by examining the stochastic process used for the stock price. The geometric Brownian motion under the risk-neutral measure in the case of the Black–Scholes model (1973) is as follows

$$dS = rSdt + \sigma SdZ \tag{12}$$

where r is the interest rate and dZ is a Wiener increment and so $dZ \sim N(0, 1)$. It is well known that the solution to this SDE, given an initial stock price of S_0, is

$$S_t = S_0 \exp\left[(r - 0.5\sigma^2)t + \sigma Z_t\right] \tag{13}$$

The drift term requires the correction $-\frac{\sigma^2}{2}$ so that the discounted expected stock price is simply $S_0, \forall\, t$. It can easily be shown that $E[S_t e^{-rt}] = S_0$ by working the equation above and taking expectations with respect to the random variable Z_t which is Gaussian.

The maximum error in each bucket is $\frac{\delta}{2}$. We require that the expected error in each bucket be zero. More formally, we want

$$E\left[\delta \sum_{i=1}^{m} \frac{\partial g}{\partial x_i} f(x_i)\right] = 0 \tag{14}$$

where the expectation E_δ is taken over the randomly generated values for the next stock price. Intuitively, embedding an error of mean zero ensures that the scheme is unbiased. In the case of the Black–Scholes model it is easy to develop a scheme of this sort. Since the stock price is distributed lognormally, we know the distribution for the model and can work out the correct dividing point in the bucket so as to make sure that the expected error from rounding up equals that from rounding down.

One approach to doing so is as follows. Assume the bucket has lower and upper limits $[S_\mathrm{l}, S_\mathrm{h}]$ respectively, and recall that $S_\mathrm{h} = S_\mathrm{l} + \delta$. The current stock price is S_0. We want to find the point $S_\mathrm{mid} \in (S_\mathrm{l}, S_\mathrm{h})$ such that

$$\int_{S_\mathrm{l}}^{S_\mathrm{mid}} Sf(S|S_0)dS = \int_{S_\mathrm{mid}}^{S_\mathrm{h}} Sf(S|S_0)dS \tag{15}$$

This equation can be easily solved numerically to yield the value of S_mid. Under this scheme S is distributed equally around S_mid in expectation for each bucket, resulting in a zero expected error from the approximation. The result of this rounding scheme is that the stock price remains a martingale after bucketing, which (i) ensures an arbitrage-free lattice, and (ii) guarantees that the current stock price is equal to the expected value of the future evolution of the stock under the risk-neutral measure.

3.3 Computation of the standard error of the approximation

In the preceding subsection, we provided a modified bucketing scheme which offered a means to eliminating the bias from bucketing, so that the expected error from the approximation would be zero. This approach essentially allocated simulated values to the upper or lower buckets in a manner chosen to ensure that average error from bucketing would be unskewed, i.e., zero. We now compute the variance of the bucketing error conditional on a bucketing scheme that has an expected error of zero.

Recall from the previous section that the bucket of length δ has two end points, S_l, at the lower end and S_h at the upper end. In Eq. (15) we derived the bias-free mid point of the bucket, denoted as S_mid. In most cases in finance, the distribution tends to be Gaussian or some variants thereof. Hence, to be conservative, we examine the uniform distribution, since its variance would be much higher than that of a Gaussian distribution. From the variance of the uniform distribution we find that the variance of the approximation error is given as:

$$\text{Variance of the bucketing scheme} = \frac{1}{12}[S_\mathrm{h} - S_\mathrm{l}]^2 = \frac{1}{12}\delta^2. \tag{16}$$

Hence, the standard error from bucketing is $\frac{\delta}{\sqrt{12}}$. Since the worst-case error bound is $O(\delta)$, it is more than 3.4 times away from the expected error of zero, and is unlikely to occur. Therefore, the worst case standard error indicates that the scheme actually does much better than the worst case approximation error bound. Of course, in the case of the Black–Scholes model where the returns are Gaussian, the variance can be exactly

computed. In general, the models for which we use the random lattice scheme, are likely to be non-Gaussian, and no closed-form solution for the variance is available; then the calculation based on the uniform distribution tends to be a *conservative* bound. We now demonstrate some illustrative results from the algorithm.

4 Applications

We consider many applications for the algorithm. The simplest application is for the Black–Scholes model for which we also have closed form solutions. The next application considers Asian options which is a canonical problem for the instance of path-dependent options.

4.1 Application to the Black–Scholes model

We applied the algorithm to the standard call option pricing problem, under the Black–Scholes assumptions (see Section 3.3.2). Call option prices were computed for various strike prices, and the results are presented in Table 1. Time was divided into 20 time periods, i.e., $d = 20$. The computational time is very short, a few seconds, once the lattice has been set up. The preprocessing step to create the lattice itself takes less than one minute. The results are presented in Table 1, which contains details of the experiment, i.e., the parameter settings. Since we also know the solution to the Black–Scholes model in closed form we are able to present the price ratio between the lattice algorithm and the true continuous time model. It can be seen that the error is negligible. Since the algorithm is accurate across all strikes, the lattice provides highly stable pricing.

Table 1 Comparison with Black–Scholes.

Strike price	Black–Scholes	RandLatt	Price ratio
90	19.9886	19.9759	0.9994
95	16.4386	16.4316	0.9996
100	13.2696	13.2643	0.9996
105	10.5151	10.5105	0.9996
110	8.1809	8.1773	0.9996

This table presents computed values for the Black–Scholes model using the random lattice scheme. The parameters for the call option are a initial stock price of 100, exercise prices of 90–110, option maturity is 1 year. The annualized stock volatility is 20%, and annualized interest rate is 10%. The lattice consists of $d = 20$ levels, and the number of buckets is $m = 300$. The number of sample paths used to generate the random lattice is $n = 100,000$.

4.2 Application to Asian options

This section parallels the work of Aingworth *et al.* (2000) (AMO) and Huan *et al.* (2000). AMO developed a new scheme for pricing a particular class of path-dependent options, namely Asian options. These options pay off based on the *average* stock price along the path taken by the stock from inception of the option until maturity.

Our approach is different from that of AMO in the following ways:

(1) Pricing Asian options is usually exponentially expensive, since the number of paths grows exponentially, and exhaustive enumeration of the lattice is required. Hence, it is usually computed using Monte Carlo methods. The path-dependence makes the average price tree non-recombining even when the stock price tree itself may be recombining. AMO introduce a polynomial algorithm by superimposing on the recombining stock price tree a set of buckets at each node for the average price. The difference between their algorithm and ours is that we use bucketing across the state space at a given time period, not at each node, as AMO do.
(2) We generate the tree randomly, not deterministically as they do.
(3) Our scheme uses a lattice of out-degree m, instead of a tree with out-degree 2 in

AMO, which offers faster rates of convergence to the true continuous time stochastic process.

(4) The lattice here is general, and applies to any stochastic process. The preceding papers were specific to models that used the simple geometric Brownian motion process for stock prices, and exploited features of that tree. In the AMO paper, the stock price tree is recombining, so that the number of stock prices at each level exactly equals the level of the tree (i.e., at the dth level there are d stock prices). This means that the stock price space is very small resulting in an economical lattice, even after expanding the state space to include buckets for the average stock price. Our algorithm is more general, in that we make no assumptions about the availability of a recombining tree for the underlying stock price. We only generate buckets for the stock and average stock price across maturity, making no assumptions about recombination, allowing the structure of the random lattice to impose that feature.

There are refinements possible to the bucketing scheme to reduce the approximation error introduced by forcing the average stock price into a set of fixed buckets. The paper by Huang *et al.* (2000) (HLD) provides one such approach. The HLD paper is a refinement of the AMO algorithm, where the bucketing scheme is improved to reduce the error bound. We ran our algorithm (denoted RL) on the same parameters as those of HLD and the results are reported in Table 2. We show that our results give the same values as those of HLD (Huang *et al.*, 2000).

4.3 Improvement in accuracy using an antithetic lattice

An application of the antithetic variable approach goes a long way in reducing estimation error. We applied the technique to the random lattice in the following way. Since bucketing results in error from forced branching to a nodal value different from that actually simulated, we can reduce error by generating an antithetic random variate for every simulation.

Table 2 Pricing Asian options.

		Exercise price				
Maturity (years)	Algorithm	40	45	50	55	60
0.5	HLD	10.754	6.361	3.007	1.104	0.315
	RL	10.762	6.362	3.001	1.109	0.326
1.0	HLD	11.544	7.613	4.519	2.417	1.174
	RL	11.514	7.571	4.478	2.395	1.171
1.5	HLD	12.283	8.668	5.740	3.583	2.122
	Ds	12.301	8.674	5.732	3.579	2.127
2.0	HLD	12.953	9.580	6.790	4.631	3.055
	RL	12.965	9.577	6.779	4.627	3.060

This table presents computed values for Asian options using the random lattice scheme. The parameters for the call option are a initial stock price of 50, exercise prices of 40–60, option maturity is 0.5–2 years. The annualized stock volatility is 30%, and annualized interest rate is 10%. The lattice consists of $d = 20$ levels, and the number of buckets is $m = 300$. The number of sample paths used to generate the random lattice is $n = 100,000$. HLD stands for the Huang-Lyuu-Dai algorithm and RL stands for the Random-Lattice algorithm.

Recall that the stochastic process for the Black–Scholes model is given by the following geometric Brownian motion:

$$dS = rSdt + \sigma SdZ \tag{17}$$

The solution to this stochastic differential equation is:

$$S_t = S_0 \exp[(r - 0.5\sigma^2)t + \sigma Z_t] \tag{18}$$

$$= S_0 \exp[(r - 0.5\sigma^2)t + \sigma\epsilon\sqrt{t}] \tag{19}$$

where $\epsilon \sim N(0, 1)$. We simulate this stochastic process by generating random variables ϵ from a Gaussian distribution. The antithetic approach we adopt consists of simulating two random lattices in parallel. Denote these lattices by A and B. During the preprocessing step, we construct lattices A and B using random numbers ϵ and $-\epsilon$ respectively. After obtaining the random lattices, we get two option prices. The average of these two prices is an estimate with much lower error because of the negative correlation that exists between lattice A and lattice B. The required caveat is that we achieve this if the payoff function $G(x)$ is monotone, which it certainly is in the case of pricing options, either for the Black–Scholes model or the Asian option model (see the arguments in Boyle *et al.* (1997)). The antithetic pair of random lattices also ensures an unbiased estimator.

We tested this algorithm by pricing options on both lattices. The details of the experiment are summarized in Table 3. The accuracy of the price ratio is very high, and the variance ratio between the antithetic random lattice algorithm and the standard algorithm is 19. The cost is that the run time is doubled.

Table 3 Error reduction using antithetic lattices.

Black–Scholes call option model	
Stock price	100
Exercise price	90
Maturity (yrs)	1.0
Volatility	0.2
Interest rate	0.1
No of estimate samples	50
Black–Scholes price	19.9886
Lattice A price	20.0049
Lattice B price	19.9681
Average price	19.9864
Lattice A std error	0.000261
Lattice B std error	0.000092
Average lattice std error	0.000021
Accuracy ratio	0.9999
Pessimistic variance ratio	19.19

This table presents computed values for the Black–Scholes model using the random lattice scheme. The parameters for the call option are a initial stock price of 100, exercise price of 90, option maturity is 1 year. The annualized stock volatility is 20%, and annualized interest rate is 10%. The lattice consists of $d = 20$ levels, and the number of buckets is $m = 300$. The number of sample paths used to generate the random lattice is $n = 100,000$. We ran three estimators, Lattice A stand-alone, its antithetic lattice, Lattice B stand alone, and the average of the two lattices. For each estimator, we ran the algorithm 50 times (i.e., sample size 50), to get the mean and variance of the three estimators. The accuracy ratio is the ratio of the RL estimator (antithetic average price) to the true closed-form solution price, i.e., 19.9864/19.9886. The variance ratio is the ratio of the variance of the antithetic average price estimator to the variance of Lattice B which is the better of the two stand-alone estimators.

5 Extensions to other popular models

In this section, we extend the random lattice technology to several popular stochastic processes. We also price options on bivariate processes here, and obtain a high degree of accuracy. In the case of bivariate stochastic processes, the random lattice is a very economical representation of the stochastic process when compared to the tree version. If a bivariate stochastic process is depicted on a non-recombining tree, then for each variate, we would have an up and down move, resulting in a tree with out-degree 4. At depth $d = 20$, this tree would have 4^{20} distinct paths, yet the random lattice is set up using only 100,000 sample paths, a

small fraction of the total. In the following examples, we show that the algorithm maintains a high degree of accuracy.

5.1 GARCH models

It is now well-known that the stochastic process for the stock price is not i.i.d lognormal as we assumed it to be in the preceding sections. The stock price distribution is usually fat-tailed and negatively skewed. Modifications to the distribution for the stock price come in many flavors and one of the most popular is the Generalized Autoregressive Conditionally Heteroskedastic model, also known as GARCH. This model was developed originally in papers by Engle (1982), Bollerslev (1986), and a nice survey is provided in Bollerslev *et al.* (1994).

The GARCH model posits the lognormal diffusion for stock prices but volatility is non constant and varies over time in a conditionally heteroskedastic manner. Essentially, the volatility in period t depends on the volatility in preceding periods $(t-1, t-2, \ldots, t-k)$. It also depends on the diffusion from period $(t-1)$. A particular case is the case where volatility depends only on the last lagged volatility, which econometricians denote the GARCH (1,1) model. We may thus write volatility σ as following the discrete time evolution:

$$\sigma_{t+1}^2 = a_0 + a_1\sigma_t^2 + a_2\sigma_t^2\epsilon_t^2. \tag{20}$$

In this equation we have ignored the risk premium, without loss of any generality. The discretized version of the lognormal model for the stock price then becomes:

$$S_{t+1} = S_t \exp\left[\left(r - \frac{1}{2}\sigma_{t+1}^2\right) + \sigma_{t+1}\epsilon_{t+1}\right] \tag{21}$$

where volatility σ_t is not a constant, but varies over time. Note that here, unlike in the preceding sections, the volatility σ is per period and is not annualized. Also, we require that $a_0, a_1, a_2 \geq 0$.

The GARCH system injects additional skewness and kurtosis into the conditional distribution of stock prices. The version presented here is one of the simplest possible specifications. More details can be found in the extensive work of Duan (1996), Duan and Simonato (2000) and Duan and Zhang (2001). In essence the GARCH model postulates that high volatility in one period is more likely to be followed by high volatility in the next.

Pricing American options under GARCH requires the development of a bivariate lattice: in (a) the stock price and (b) the volatility. Extending the univariate lattice is extremely easy to do, and the algorithm fits seamlessly into the preceding framework. Accuracy is managed by choosing the correct buckets for volatility in addition to the buckets for the stock price. We found that the number of buckets required for volatility is much smaller, and there is little degradation in accuracy. One plausible reason for this might be the fact that the volatility is mean reverting and the range in which it lies is quite narrow.

Generating the lattice randomly off the stochastic process has three advantages: (i) there is no need to develop a separate scheme for a recombining lattice for each different stochastic process chosen. (ii) The lattice is much finer and more accurate than other schemes that have some specific recombination mechanics. (iii) Simulation of the risk neutral process automatically ensures that the lattice preserves martingale properties so that the model remains arbitrage-free. In comparison to the Markov chain approach of Duan and Simonato (2000) where the transition function is known in closed-form, the approach in this paper works even when the transition density function is not available.

The paper by Ritchken and Trevor (1999) develops a specific recombination scheme for GARCH models. Their algorithm matches moments in a recombining tree with out-degree m, where m is usually 3. Our model instead uses a random lattice with out-degree $m > 200$. In order to compare our algorithm with theirs we replicated their prices using our algorithm. Our prices are within a small fraction of theirs, even though the random lattice approach is not tailored to the GARCH model, as their's is. Table 4 presents prices of European calls for a range of strike prices and maturities. We also report the prices they obtained in their paper for comparison. Most prices are within a small fraction of the comparison algorithm.

5.2 Special adjustments for American options

In the introduction, we observed that the estimator is biased for American options (specifically American put options, see Fig. 4). To recall, the estimator is biased because whenever early exercise occurs the node is never really on the early exercise boundary of the option, on account of the discreteness injected into the state space whenever a lattice is used. It is usually away from the boundary which means that the early exercise value is an overestimate. Hence, at the point at which the early exercise decision is made we would like to embed a suitable reduction in the early exercise value to correct the bias.

In this section, we consider two adjustments that are possible to reduce the bias of the estimator. We use American put options to illustrate the ideas in the algorithm. The following arguments allow us to impose the right correction.

- The solution to the American option pricing problem comprises determining the optimal

Table 4 Random lattices for the GARCH model.

		Maturity (days)				
Strike	Algorithm	5	10	30	50	100
95.0	RT	5.012	5.086	5.560	6.030	7.028
	RL	5.010	5.085	5.563	5.993	7.018
97.5	RT	2.665	2.915	3.712	4.316	5.468
	RL	2.665	2.911	3.710	4.279	5.458
100.0	RT	0.927	1.309	2.268	2.930	4.148
	RL	0.916	1.297	2.261	2.896	4.141
102.5	RT	0.178	0.439	1.263	1.885	3.069
	RL	0.184	0.441	1.262	1.867	3.070
105.0	RT	0.018	0.108	0.639	1.148	2.214
	RL	0.021	0.111	0.643	1.143	2.221

This table presents computed values for the GARCH model using the random lattice scheme. The parameters for the call option are a initial stock price of 100, exercise prices of 95–105, option maturity varying from 5 to 100 days. The average annualized stock volatility is 20%, which is also equal to the initial volatility σ_0, and annualized interest rate is zero percent. The lattice consists of $d = 20$ levels, and the number of stock price buckets is 250. The number of volatility buckets is 11. The number of sample paths used to generate the random lattice is $n = 100{,}000$. The GARCH equation parameters are: $a_0 = 6.575 \times 10^{-6}$, $a_1 = 0.90$, and $a_2 = 0.04$. RL stands for the Random-Lattice algorithm and RT stands for the Ritchken–Trevor algorithm. The prices from their algorithm were in Table 3 of their paper.

stopping time in the life of an option. This occurs when taking the immediate exercise value is more lucrative than the continuation value from not exercising the option. Hence there is a boundary $B(t)$ which is a function of time, which may be drawn such that if the stock price crosses the boundary then it is optimal to exercise the option early (see Fig. 4). It has been theoretically established that this boundary is unique and continuous (for an excellent exposition of such issues in optimal stopping problems, see Dixit (1991) and refer to Fig. 4). The bias comes from the slippage encountered when the nodes in the pricing lattice do not sit exactly on the boundary.

- Recall that the random lattice contains a discretized state space in stock price values which are δ apart from each other. Hence the maximum possible amount of bias is δ.
- We now exploit the "value-matching" and "smooth-pasting" conditions from the theory of continuous-time dynamic programming which states that under optimality, these two conditions are always satisfied. The value-matching condition states that at the stopping boundary, the value of continuation and early exercise are exactly equal. We define a new quantity, which we call the "exercise distance" or γ. This is computed as follows:

$$\gamma = [S_t(i) - K]^+ - P(S_t(i), t) \qquad (22)$$

where $S_t(i)$ is the stock price at time t and node i on the lattice. K is, as before, the exercise price. The function $P(S_t(i), t)$ stands for the current continuation value of the American put option on the lattice at $S_t(i)$. The value-matching condition simply states that when $S_t(i) \in B(t)$, then $\gamma = 0$.

- Since the boundary is known to be unique, and continuous, it is also true that $|\gamma|$ increases monotonically as we move away from the boundary $B(t)$. This result is based on the well-known "smooth-pasting" condition, which is

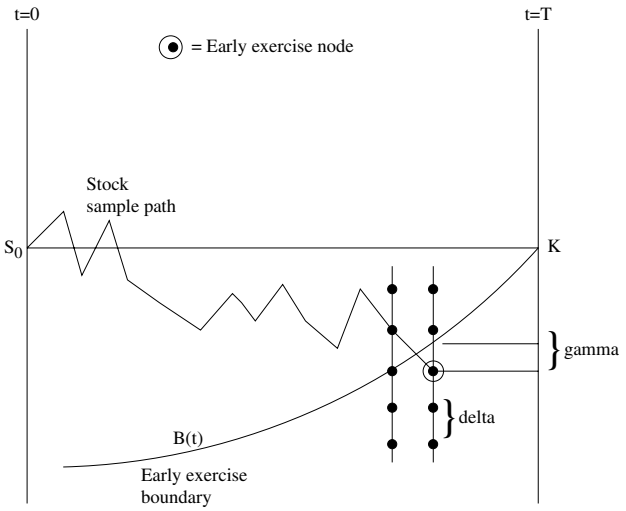

Figure 5 Depiction of bias correction variables for American put options.

essentially a smoothness condition. The condition, which is satisfied under optimal stopping is that (see Samuelson (1965), Merton (1990) Ch 8, Dixit (1991) and Dumas (1991)).

$$\text{[Smooth-pasting]: } \frac{\partial P_t}{\partial B(t)} = 1. \qquad (23)$$

Hence, we may use the value of γ to impose the correction to the bias (see Fig. 5).

- Note that $\gamma \leq \delta$, because at worst, the stock price at which it pays to exercise early will be one bucket away from the boundary $B(t)$. Hence, a useful correction is to take the early exercise value, when exercise occurs, and subtract an amount equal to $\frac{\gamma}{2}$. In order to get a lower bound, subtract the amount $\frac{\delta}{2}$. Hence Eq. (9) becomes a modified early exercise rule is as follows (SP stands for smooth-pasting):

$$\begin{aligned} &\text{If } (\gamma \geq 0) \quad \text{and} \quad (\gamma \leq \delta) \text{ then} \\ &\text{[SP-1]: } G(i,t) \\ &\qquad = \max\left(G(i,t), S_t - K - \frac{\gamma}{2}\right), \\ &\qquad\qquad \forall\, i = 1, \ldots, m. \end{aligned} \qquad (24)$$

which gives an intermediate correction, and for a lower bound we apply the following:

$$\text{[SP-2]: } G(i,t) = \max\left(G(i,t),\, S_t - K - \frac{\delta}{2}\right), \quad \forall\, i = 1, \ldots, m. \tag{25}$$

The IF condition above ensures that the correction is applied only to nodes around the boundary and in the exercise region and nowhere else.

- Hence, the value-matching and smooth-pasting conditions together allow us to apply the stated bias correction.[7]

To illustrate, we priced the American put in the GARCH model. For comparison we report the prices from Ritchken and Trevor (1999). Table 5 shows that the two corrections (RL-SP-1 & RL-SP-2) result in lower values of the estimator than RL. As expected, the bound from RL-SP-2 is lower than that from the average correction in RL-SP-1. Therefore, we can think of RL-SP-2 as a lower bound (low bias), RL-SP-1 as the intermediate price, and RL as an upper bound (high bias).

Here again, the values are very close to those reported by Ritchken and Trevor and the various corrections to the RL method for the bias do give better values.

Table 5 Random lattices for the GARCH model, American puts.

Maturity (days)	Algorithm	American price	European price
10	RT	1.192	1.175
	RL	1.304	1.172
	RL-SP-1	1.288	–
	RL-SP-2	1.189	–
50	RT	2.398	2.281
	RL	2.457	2.270
	RL-SP-1	2.437	–
	RL-SP-2	2.322	–
100	RT	3.143	2.882
	RL	3.184	2.873
	RL-SP-1	3.126	–
	RL-SP-2	2.997	–

This table presents computed values for the GARCH model using the random lattice scheme to price American put options. We do not price calls here since we know that theoretically, in the presence of no dividends, the American call is never exercised early. The parameters for the put option are a initial stock price of 100, exercise price of 100, option maturity varying from 10 to 100 days. The average annualized stock volatility is 20%, which is also equal to the initial volatility σ_0, and annualized interest rate is 0%. The lattice consists of $d = 20$ levels, and the number of stock price buckets is 250. The number of volatility buckets is 11. The number of sample paths used to generate the random lattice is $n = 100{,}000$. The GARCH equation parameters are: $a_0 = 6.575 \times 10^{-6}$, $a_1 = 0.90$, and $a_2 = 0.04$. The interest rate is assumed to be $r = 0.10$. RL stands for the Random-Lattice algorithm and RT stands for the Ritchken–Trevor algorithm. RL-SP stands for the smooth pasting version of the Random-Lattice algorithm, which effectively generates a lower estimate. We implemented two versions of this algorithm. The prices from the RT algorithm were in Table 3 of their paper.

5.3 *Stochastic volatility models*

In the case of GARCH models, both the stock return process and the volatility process are driven by the same random shock, which we denoted as ϵ_t. A more general form of volatility is obtained by making the variance of stock returns follow its own stochastic process. Hence we get a bivariate diffusion system represented by the following stochastic differential equations:

$$dS = rSdt + \sqrt{V}SdZ \tag{26}$$

$$dV = \kappa(\theta - V)dt + \eta\sqrt{V}dW \tag{27}$$

$$dZ.dW = \rho\, dt \tag{28}$$

Here, r is the riskfree interest rate and $\sqrt{V}$ is the stock volatility. The stock price is driven by the Wiener increment dZ, which is a standard Brownian motion. The variance of stock returns

V follows a mean-reverting stochastic process where the mean reversion rate is κ and the long run mean of the process is θ. The variance V itself has its own volatility, governed by parameter η, and Wiener increment dW. The correlation between the diffusions dZ and dW is determined by parameter ρ.

The GARCH model is applied with a minor modification to the volatility process to account for the differences in specification. Therefore, with a small change, the approach applies seamlessly. There is only one known closed form solution for this particular model, developed in the work of Scott (1987) and the more recent work of Heston (1993). For a detailed analytical exploration of the statistical properties of the model see the paper by Das and Sundaram (1999). Heston undertakes a graphical comparison of the prices from a model with stochastic volatility versus one with constant volatility, i.e., the Black–Scholes model. We present illustrative results in Table 6.

Table 6 Random lattices for the stochastic volatility model.

Strike price	SV model ($\theta = 0.01$)	SV model ($\theta = 0.04$)	Black–Scholes
90.0	10.045	10.095	10.073
92.5	7.687	7.748	7.727
95.0	5.520	5.594	5.577
97.5	3.667	3.748	3.739
100.0	2.218	2.302	2.301
102.5	1.223	1.296	1.292
105.0	0.607	0.663	0.656
107.5	0.271	0.307	0.301
110.0	0.109	0.128	0.125

This table presents computed values for the Heston SV model using the random lattice scheme. The parameters for the call option are a initial stock price of 100, exercise price of 80–120, option maturity of 100 days. The average annualized stock volatility is 10%, which is also equal to the initial volatility σ_0, and annualized interest rate is 0%. The lattice consists of $d = 20$ levels, and the number of stock price buckets is 250. The number of volatility buckets is 11. The number of sample paths used to generate the random lattice is $n = 100{,}000$. The SV equation parameters are: $\kappa = 2$, $\theta = 0.01$, and $\eta = 0.1$. The interest rate is assumed to be $r = 0$.

6 Conclusions

In this article, we showed that it is possible to achieve a very fast computation scheme for option pricing using a random lattice approach. The lattice is dense, which minimizes the approximation error. The error may be designed to be zero in expectation by choosing an appropriate bucketing scheme. The use of antithetic variates drastically cuts down the simulation error. A benefit from the random lattice approach comes from the reduced storage required in the algorithm. The lattice often only needs to be maintained in the relevant state variable, not in all variables. Hence, the memory required is reduced. We are also able to prune some processing by ignoring all nodes with outdegree zero, since there is no probability mass on those edges of our random lattice.

Many refinements are possible to the algorithm. Since our algorithm entails numerical integration over the buckets, we can increase accuracy by employing better aggregation of values using Newton–Coates formulae. The lattice may be drawn in return space rather than in stock price space. This makes the lattice symmetric, which usually results in better convergence rates. The algorithm may be applied to a larger gamut of option types.

Random lattices are especially useful for pricing American options. The lattice enables fast computation, which allows for rapid checking of the optimal stopping conditions during dynamic programming. We developed alternatives to earlier bounds on American options by exploiting the theory of continuous-time optimal control, via the use of the value-matching and smooth-pasting conditions. We also developed a correction to the bias that emanates naturally from implementing optimal stopping in discrete state spaces. Our

experiments on GARCH models demonstrated these bounds and corrections.

For example, other common models of option pricing may be implemented, and also the pricing of mortgage-backed securities. Very high dimensional problems such as the average of 15 stocks may be computed on a lattice of much lower dimension (see also Broadie and Glassermann (1997b)). Better randomization schemes for preprocessing may also be used.

Finally, we believe that this algorithmic idea may be extended to the domain of stochastic control problems. Preliminary work has shown that an expansion of the state space with the addition of a control space permits the same approach to be extended to the solution of optimal control problems with a polynomial time algorithm. The flexibility and ease of implementation of this approach, as seen from the examples, suggests that this approach has wide practical applicability.

Acknowledgements

Many thanks to Satish Rao and Alistair Sinclair for numerous comments.

Notes

[1] The term "in-degree" carries the natural connotation, i.e., how many arcs or branches of the tree enter a particular node.

[2] Symmetry here is the situation in which the same number of branches emanates from each node.

[3] In more complex options, the terminal payoff may not be a function of the terminal price $S_l(t)$, but may depend on the *path* leading to leaf l. In such cases, the leaf value contains a summation over root-leaf paths.

[4] American options do not admit closed-form pricing equations. The only American options for which closed-form solutions are available are American call options on stocks that do not pay dividends, since it can be shown that it is never optimal to exercise these options early, making them equal in price to European calls.

[5] These are conditions that the optimal solution must satisfy in continuous-time dynamic programming.

[6] These are options written on stocks where the volatility of the stock follows a Generalized Auto-Regressive Conditionally Heteroskedastic (GARCH) process.

[7] Since we have converted the problem into one pertaining to a discrete state space, we only need the value-matching and smooth-pasting conditions. In continuous state space problems, an additional condition on the second derivative also applies, known as the "super-contact" condition. See Dumas (1991) for details.

References

Acharya, V., Das, S., and Sundaram, R. (2002). "Arbitrage-free Pricing of Credit Derivatives with Rating Transitions." *Financial Analysts Journal*, May–June, 28–44.

Aingworth, D., Motwani, R., and Oldham, J. (2000). "Accurate Approximations for Asian Options." In *Proc. 11th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 891–900.

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

Berger, J.O. (1985). *Statistical Decision Theory and Bayesian Analysis*, 2nd edn. *Springer-Verlag*, New York.

Black, F. and Scholes, M. (1973). "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy* **81**, 637–654.

Bollerslev, T. (1986). "Generalized Autoregressive Conditional Heteroskedasticity." *Journal of Econometrics* **31**, 307–327.

Bollerslev, T., Robert, E., and Nelson, D. (1994). "ARCH Models." *Handbook of Econometrics*, Vol. IV, eds. Engle, R. and McFadden, D. (Amsterdam: North Holland), 2959–3038.

Boyle, P.P. (1977). "Options: A Monte Carlo Approach." *Journal of Financial Economics* **4**, 323–338.

Boyle, P.P., Broadie, M., and Glasserman, P. (1997). "Simulation Methods for Security Pricing." *Journal of Economic Dynamics and Control* **21**, 1267–1321.

Broadie, M. and Glasserman, P. (1997). "Pricing American-Style Securities using Simulation." *Journal of Economic Dynamics and Control* **21**, 1323–1352.

Broadie, M. and Glasserman, P. (1997). "A Stochastic Mesh Method for Pricing High-Dimensional American Options," Working paper, Columbia University.

Cox, J., Ross, S., and Rubinstein, M. (1979). "Option Pricing: A Simplified Approach." *Journal of Financial Economics* **7**, 229–264.

Das, S.R. (1999). Pricing Credit Derivatives, *Handbook of Credit Derivatives* (eds.), Frost, J., Francis, J., and Whittaker, J.G. NY: John Wiley, pp. 101–138.

Das, S. and Sundaram, R. (1999). "Of Smiles and Smirks: A Term-Structure Perspective." *Journal of Financial and Quantitative Analysis* **34**(2), 211–240.

Deliadenis, G. and Geske, R. (1998). "Credit Risk and Risk Neutral Default Probabilities: Information about Rating Migrations and Defaults." Working paper, UCLA.

Dixit, A. (1991). "A Simplified Treatment of the Optimal Regulation of Brownian Motion." *Journal of Economic Dynamics and Control* **15**(4), 657–673.

Duan, J. (1996). "A Unified Theory of Option Pricing under Stochastic Volatility—from GARCH to Diffusion, Working paper.

Duan, J. and Simonato, J.G. (2000). "American Option Pricing Under GARCH by A Markov chain Approximation." *Journal of Economic Dynamics and Control* **25**, 1689–1718.

Duan, J. and Zhang, H. (2001). "Pricing Hang Seng Index Options around the Asian Financial Crisis—A GARCH Approach." *Journal of Banking and Finance* **25**, 1989–2014.

Duffie, D. and Singleton, K. (1996). "Modeling Term Structures of Defaultable Bonds." Working Paper, Stanford University, *Review of Financial Studies* **12**, 687–720.

Dumas, B. (1991). "Super Contact and Optimality Related Conditions." *Journal of Economic Dynamics and Control* **15**(4), 675–685.

Engle, R. (1982). "Autoregressive Conditional Heteroskedasticity with Estimates of the Variance of U.K. Inflation." *Econometrica* **50**, 987–1008.

Figlewski, S. and Bin, G. (1999). "The Adaptive Mesh Model: A New approach to Efficient Option Pricing." *Journal of Financial Economics* **53**, 313–351.

Garcia, D. (1999). "A Monte Carlo Method for Pricing American Options," Working paper, Haas School of Business, UC Berkeley.

Glasserman, P., Heidelberger, P., and Shahabuddin, P. (1999). "Importance Sampling in the Heath-Jarrow-Morton framework." *Journal of Derivatives* **7**(1), 32–50.

Glasserman, P., Heidelberger, P., and Shahabuddin, P. (1999). "Asymptotically Optimal Importance Sampling and Stratification for Pricing Path-Dependent Options." *Mathematical Finance* **9**, 117–152.

Harrison, J. and Kreps, D. (1979). "Martingales and Arbitrage in Multiperiod Securities Markets." *Journal of Economic Theory* **20**, 381–408.

Harrison, J. and Pliska, S. (1981). "Martingales and Stochastic Integrals in the Theory of Continuous Trading." *Stochastic Processes and Their Applications* **11**, 215–260.

Heston, S. (1993). "A Closed-Form Solution for Options with Stochastic Volatility with Application to Bond and Currency Options." *Review of Financial Studies* **6**(2), 327–343.

Huang, G.-S., Lyuu, Y.-D., and Dai, T.-S. (2000). "Very Accurate Approximations for Asian Options, Working Paper, Taipei, Taiwan: Institute of Information Science, Academia Sinica.

Jarrow, R. and Turnbull, S. (1995). Pricing Options on Financial Securities Subject to Default Risk." *Journal of Finance* **50**, 53–86.

Jarrow, R., Lando, D., and Turnbull, S. (1997). A Markov Model for the Term Structure of Credit Spreads." *Review of Financial Studies* **10**, 481–523.

Jerrum, M. and Sinclair, A. (1996). The Markov Chain Monte Carlo Method: An Approach to Approximate Counting and Integration, in "Approximation Algorithms for NP-hard Problems," (ed.) Hochbaum, D.S., PWS Publishing, Boston, pp. 482–520.

Merton, R.C. (1990). *Continuous-Time Finance*, NY: Blackwell.

Moro, B. (1995). The Full Monte." *Risk* **8**, 57–58.

Ritchken, P. and Trevor, R. (1999). Pricing Options under Generalized GARCH and Stochastic Volatility Processes." *Journal of Finance* **54**, 377–402.

Samuelson, P. (1965). "Rational Theory of Warrant Pricing." *Industrial Management Review* **6**, 41–49.

Scott, L. (1987). "Option Pricing When the Variance Changes Randomly: Theory, Estimation and Application." *Journal of Financial and Quantitative Analysis* **22**(4), 419–438.

Sinclair, A. (1997). "Convergence rates for Monte Carlo Experiments." In *Numerical Methods for Polymeric Systems*, S. G. Whittington, ed., IMA Volumes in Mathematics & its Applications, pp. 1–18.

Sinclair, A. and Jerrum, M. (1989). "Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains." *Information and Computation* **82**, 93–133.

Schoenmakers, J.G. and Heemink, A.W. (1997). "Fast Valuation of Financial Derivatives." *Journal of Computational Finance* **1**, 47–62.

Tilley, J. (1993). "Valuing American Options in a Path Simulation Model." *Transactions of the Society of Actuaries* **45**, 83–104.

Keywords: Random lattice; derivatives; bias