Extremely Accurate and Efficient Tree Algorithms for Asian Options with Range Bounds

Tian-Shyr Dai* Guan-Shieng Huang† Yuh-Dauh Lyuu‡

Abstract

Asian options can be priced on the unrecombining binomial tree. Unfortunately, without approximation, the running time is exponential. This paper presents efficient and extremely accurate approximation algorithms for Asian options on the binomial tree. For a European-style Asian option with strike price X on an n-period binomial tree, our algorithm runs in $O(kn^2)$ time with a guaranteed error bound of $O(X\sqrt{n}/k)$ for any positive integer k. Parameter k can be adjusted for any desired trade-off between time and accuracy or to guarantee convergence. This basic algorithm is then modified to give increasingly tighter upper and lower bounds (or range bounds) that bracket the desired option value while maintaining the same computational efficiency. As

*Department of Computer Science & Information Engineering, National Taiwan University, Taipei, Taiwan. The author was supported in part by NSC grant 90-2213-E-002-081.

†Department of Computer Science & Information Engineering, National Taiwan University, Taipei, Taiwan.

‡Corresponding author. Department of Finance and (preferred address) Department of Computer Science & Information Engineering, National Taiwan University, No 1, Sec 4, Roosevelt Rd, Taipei, Taiwan. E-mail: lyuu@csie.ntu.edu.tw. The author was supported in part by NSC grant 90-2213-E-002-081.

the upper and lower bounds are essentially numerically identical in practice, the proposed algorithms can be said to price European-style Asian options exactly without combinatorial explosion. These algorithms are then extended to give extremely accurate and efficient schemes for American-style Asian options. As before, they run in $O(kn^2)$ time and produce range bounds that bracket the desired option value. Because the upper and lower bounds are again essentially numerically identical in practice, the proposed algorithms can be said to price American-style Asian options exactly without combinatorial explosion. Our results also imply for the first time in the literature that the popular Hull-White algorithms are upper-bound algorithms. Extensive computer experiments are conducted to confirm the extreme accuracy of the algorithms and their competitiveness in comparison with alternative schemes. The exercise boundary estimated by our algorithm is of independent interest. In particular, it gives rise to a general two-phase computational framework whereby any upper-bound tree algorithm is guaranteed to remain an upper-bound algorithm if it employs the estimated exercise boundary to reduce the range of price sums at each tree node. Furthermore, the new algorithm should outperform the original version because of the reduced price-sum ranges. Any algorithmic progress in pricing American-style Asian options using trees therefore immediately results in an improved two-phase version in which the first phase calls upon our algorithm to provide an estimated exercise boundary.

1 Introduction

Path-dependent derivatives are derivative securities whose payoff depends nontrivially on the price history of the underlying asset. Some path-dependent derivatives such as barrier options can be efficiently priced as in Lyuu (1998). Others, however, are known to be difficult to price in terms of speed and/or accuracy as surveyed in Lyuu (2002). The (arithmetic) Asian option is perhaps the most representative of the latter

category.

Pricing Asian options has been a long-standing problem when the underlying asset's price is log-normally distributed. Approximate closed-form solutions are suggested in Levy (1992), Milevsky (1998), and Turnbull and Wakeman (1991). Geman and Yor (1993) derive an analytical expression for the Laplace transform of the Asian call. Numerical inversion of this transform is considered in Geman and Eydeland (1995) and Shaw (1998). Some inversion algorithms based on the Euler and Post-Widder methods can be found in Abate and Whitt (1995).

Because no simple closed-form solutions exist yet for the Asian option, the development of efficient numerical algorithms is critical. First, there are the popular Monte Carlo and related quasi-Monte Carlo methods; see Boyle et al. (1997), Broadie and Glasserman (1996), Broadie et al. (1999), and Kemna and Vorst (1990). But both the Monte Carlo approach and the analytical approach suffer from the inability to handle early exercise without bias. Recently, Longstaff and Schwartz (2001) have developed a least-squares Monte Carlo approach to tackle the problem rigorously.

Tree methods and the related discretized partial-differential-equation approach are more general than the above-mentioned schemes because they can handle early exercise. The difficulty with the tree method in the case of Asian options lies in its exponential nature: It seems that 2^n paths have to be individually evaluated for a binomial tree with n periods (see Fig. 1). Many proposed approaches to solve this **combinatorial explosion** augment a state variable to the tree nodes, which is usually the price sum or price average. A very successful paradigm by Hull and White (1993) limits the number of price sums at each node of the tree to some manageable magnitude k. It then resorts to interpolation in backward induction; see Hull and White (1993), Klassen (2001), Ritchken et al. (1993), and Zvan et al. (1999). We will call this methodology the **Hull-White paradigm**. The Hull-White paradigm is efficient, with a running time of $O(kn^2)$. But it lacks convergence guarantees unless k and n are related in a certain manner; see Forsyth et al. (2001). This paper

will prove rigorously that algorithms based on the Hull-White paradigm with linear interpolation are upper-bound algorithms.

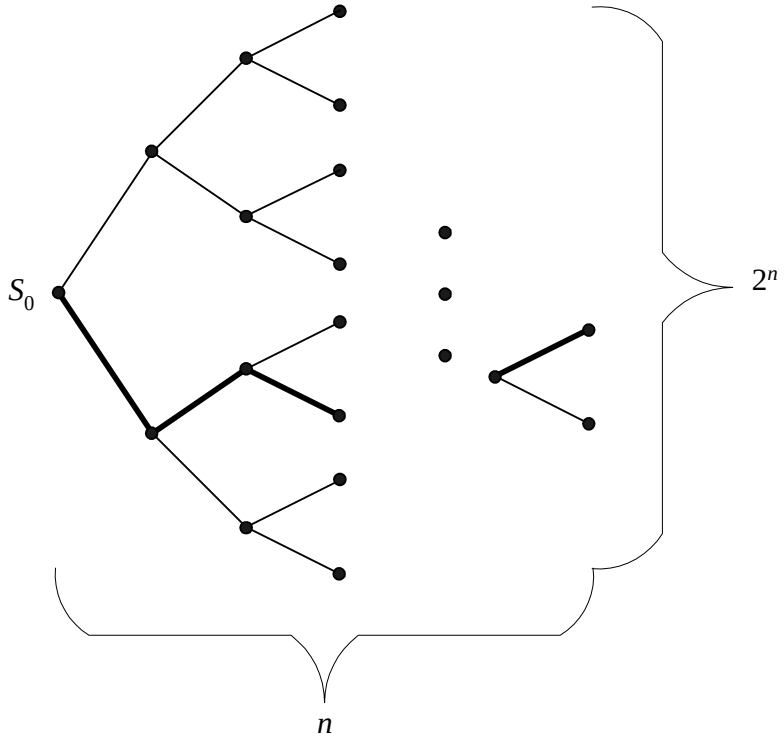

Figure 1: **The unrecombining binomial tree.** There are 2^n paths (one of which follows the darkened edges in the plot).

Another paradigm due to Dai and Lyuu (1999) constructs a trinomial tree with stock prices which are rational numbers of finite precision. The advantage is that the possible price sums at each node are finitely enumerable. This done, backward induction can be carried out without the need of approximation such as interpolation. The algorithm is guaranteed to be convergent. The worst-case running time seems to be $2^{O(\sqrt{n})}$. Although this time bound substantially improves on the naive $O(2^n)$-time algorithm, its superpolynomial nature forbids the use of very large n's. This does not seem to pose a problem for typical parameters, however.

The convergence guarantee of numerical algorithms is very important indeed.

However, rigorous convergence analysis of numerical algorithms for Asian options is surprisingly rare. Barraquand and Pudet (1996) and Forsyth et al. (2001) analyze the Hull-White paradigm for European-style Asian options. For example, under some plausible yet unproven assumptions, Forsyth et al. (2001) prove that the Hull-White paradigm with linear interpolation is convergent with a convergence rate of $O(n^{-1})$ if k is proportional to $n^{1.5}$. This results in a running time of $O(n^{3.5})$. Such guarantees, though reassuring, are concerned with an algorithm's limiting behavior. But that may not say much about its performance for a finite n. A typical heuristic is to run the algorithm for a few n's to gain confidence. Still, little can be said about the magnitude of error with respect to the desired option value. These considerations lead us to the next paradigm.

The last paradigm seeks approximation algorithms that produce provable upper and lower bounds (called **range bounds**) that bracket the option value on the exponential-sized **unrecombining binomial tree**. The hope is that the desired option value becomes practically available when the upper bound and the lower bound are essentially identical. Furthermore, the difference between the upper bound and the lower bound, call it e, gives an upper limit on the uncertainty surrounding the desired option value (see Fig. 2). Chalasani et al. (1999) propose $O(n^4)$-time range-bound algorithms for Asian options. Rogers and Shi (1995) derive range-bound algorithms in the case of European-style Asian options. The $O(kn^2)$-time algorithm of Aingworth et al. (2000) guarantees a theoretical error bound of $O(Xn/k)$, where k can be varied for any desired trade-off between time and accuracy. (Their error analysis of American-style Asian options seems mistaken.) This bound is also interesting in that it is known before the algorithm starts. Akcoglu et al. (2001) derive a complex trade-off by a recursive application of the algorithm of Aingworth et al. in the case of European-style Asian options. See Hochbaum (1997) for more information on approximation algorithms.

This paper follows the last paradigm on binomial trees (the same ideas can be

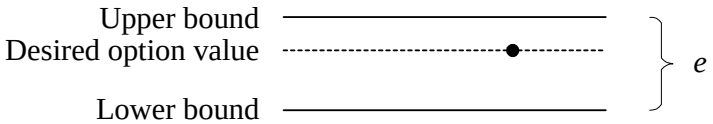

Figure 2: **Range bound and uncertainty e about the desired but unknown option value.**

extended to trinomial trees). Our first algorithm has an error bound of $O(X\sqrt{n}/k)$ for European-style Asian options. This is an improvement over the result of Aingworth et al. (2000) and is independent of the stock price volatility. One can choose n and k to obtain an upper bound on the pricing error one is comfortable with before the algorithm is executed. The resulting algorithm is guaranteed to output a value that does not deviate from the desired option value by more than the predetermined upper bound. Variations on this basic algorithm tighten the bounds further. Extensive computer experiments conclude that the upper and lower bounds after such tightening are essentially identical in practice. Because all the algorithms run in time $O(kn^2)$, the desired option value is obtained without combinatorial explosion. As the magnitude of the pricing error is $O(X\sqrt{n}/k)$, it suffices to pick k to be proportional to $\sqrt{n}$ to satisfy any desired error bound and the running time is $O(n^{2.5})$. To guarantee a convergence rate of $O(n^{-0.5})$, as another example, it suffices to pick k to be proportional to n, making the algorithms' running time $O(n^3)$. To guarantee a convergence rate of $O(n^{-1})$, as the last example, it suffices to pick k to be proportional to $n^{1.5}$, making the algorithms' running time $O(n^{3.5})$.

The convergence result is reached without ad hoc assumptions as in Forsyth et al. (2001). Our experiments show that the theoretical error bound is probably over-pessimistic and a convergence rate of $O(n^{-2})$ is most likely achievable with a running time of $O(n^3)$ by picking k to be proportional to n.

American-style Asian options are harder to price because of the need to estimate the optimal exercise boundary. As a consequence, rigorous analysis of their algo-

rithms is even scarcer than that for European-style Asian options. Our range-bound algorithms for American-style Asian options derive from those for European-style ones. The running times remain $O(kn^2)$. Furthermore, we give formal proof that the proposed algorithms do give range bounds. Computer experiments again demonstrate that the range bounds are so tight in practice that the desired option value is essentially obtained within the time bound. We conclude that our proposed algorithms price the American-style Asian option correctly for all practical purposes. The intricate range-bound proof is of independent interest. It yields an unexpected corollary that the Hull-White paradigm with linear interpolation in Hull (1997) and Hull and White (1993) is an upper-bound algorithm.

A very important feature of our approximation algorithms is the way with which they attempt to limit the range of price sums at each node. The motivation is obvious: A small dynamic range allows finer resolution in the approximation given the same amount of computational efforts. In the case of European-style Asian options, price sums at or over a certain numerical bound will necessarily result in the option being in the money at expiration. Luckily, in this scenario their contribution to the option value is given by a simple exact formula. This implies that the algorithms can direct their computational resources to price sums lower than the said bound.

In the case of American-style Asian options, a similar role is played by the exercise boundary. Price sums above the exercise boundary will force the option to be exercised immediately, whose contribution to the option value is known exactly and trivially computable. We circumvent the difficulty of finding the exact boundary with a way to estimate it while respecting the desired range bounds. Once the boundary is available, the algorithms will again work on more limited price-sum ranges.

The algorithm to estimate the exercise boundary is of independent interest. It gives rise to a general two-phase computational framework in which phase one calculates the estimated exercise boundary and phase two employs any purported upper-bound algorithm. The resulting algorithm is guaranteed to be an upper-bound al-

gorithm. Because the algorithm in phase two works on reduced price-sum ranges, it is expected to offer substantially more accurate results than if phase one is not in place. Any algorithmic progress in pricing the American-style Asian options, therefore, immediately results in an improved two-phase version in which the first phase calls upon our algorithm to provide the estimated exercise boundary.

This paper is organized as follows. Section 2 reviews the familiar CRR binomial model and defines the Asian option. In Section 3 we lay down the framework for the error analysis of European-style Asian options. Section 4 presents and analyzes a range-bound algorithm for European-style Asian options, which is the starting point of all the algorithms to follow. This basic range-bound algorithm achieves the error bound $O(X\sqrt{n}/k)$. We then propose an algorithm with an even tighter range bound in Section 5 and evaluate its numerical performance in Section 6. Range-bound algorithms for American-style Asian options and their performance are presented in Section 7. Their range-bound properties, the general two-phase framework, and the upper-bound result for the Hull-White paradigm are proved in Section 8. Section 9 concludes the paper.

2 Basic Terms

The standard CRR binomial model will be used to approximate the continuous-time stock price dynamics $dS/S = \mu\,dt + \sigma\,dW$, where W is the Wiener process and σ will be referred to as the stock price volatility. Let S_i denote the stock price at time i. The binomial model says that S_{i+1} equals $S_i u$ with probability p and $S_i d$ with probability $1-p$, where $d < u$. Although identity $ud = 1$ holds, this property is not needed for the theoretical results. The binomial model will start at time 0 and end at time n throughout the paper. The stock price at time i that results from j down moves and $i-j$ up moves therefore equals $S_0 u^{i-j} d^j$ with probability $\binom{i}{j} p^{i-j}(1-p)^j$. A 2-period binomial model is illustrated in Fig. 3(a). We shall assume that the stock

does not pay dividends for ease of presentation.

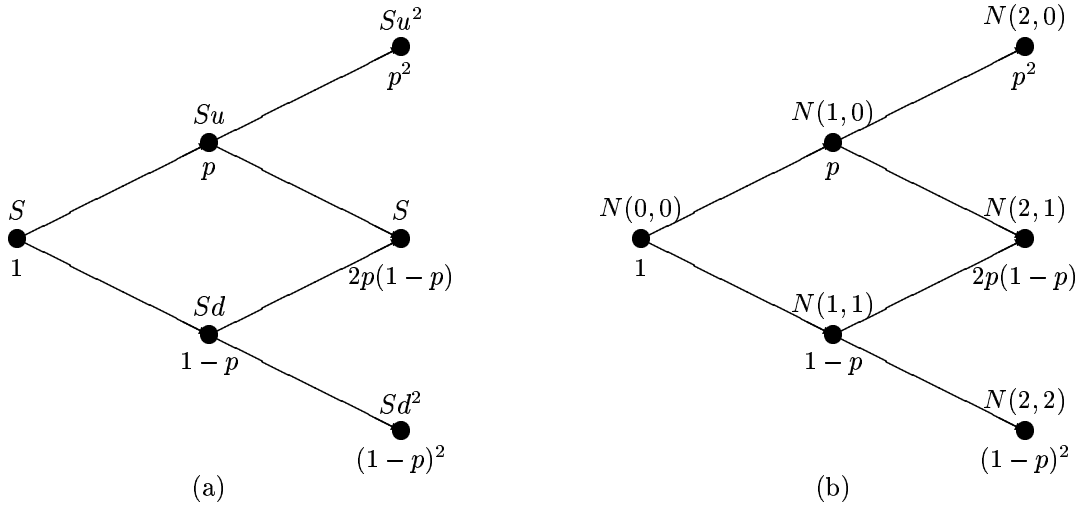

Figure 3: **The binomial model and the binomial tree.** (a) A 2-period binomial model and (b) a 2-period binomial tree. The probability of reaching each node is listed under the node.

We now map the stock prices to nodes on a binomial tree used for pricing. Node $N(i,j)$ stands for the node at time i with j cumulative down moves. Its associated stock price is hence $S_0 u^{i-j} d^j$. The stock price can move from $N(i,j)$ to $N(i+1,j)$ with probability p and to $N(i+1,j+1)$ with probability $1-p$. As a consequence, node $N(i,j)$ can be reached from the root with probability $\binom{i}{j} p^{i-j}(1-p)^j$. See Fig. 3(b) for illustration.

A path from the root to a node at maturity contains $n+1$ prices $S_0, S_1, \ldots, S_n$. For pricing purposes, the probability p for an up move is set to $(e^r - d)/(u-d)$, where r denotes the continuously compounded risk-free interest rate per period, τ is the time to maturity in years, and $u = e^{\sigma\sqrt{\tau/n}}$. That $r \geq 0$ will be assumed throughout the paper. Both $d \leq e^r \leq u$ and $0 \leq p \leq 1$ must hold to avoid arbitrage.

Let $X > 0$ be the strike price. The European-style Asian call has a payoff of

$$\left(\frac{1}{n+1} \sum_{i=0}^{n} S_i - X \right)^+$$

at maturity, where $(x)^+$ means $\max(x, 0)$. Its arbitrage-free price is therefore

$$e^{-rn} E\left[\left(\frac{1}{n+1}\sum_{i=0}^{n} S_i - X\right)^+\right]. \tag{1}$$

The option value can be evaluated by averaging the payoffs of all possible 2^n price paths $(S_0, S_1, \ldots, S_n)$. This value will serve as our **benchmark**, which converges to the true value as n goes to infinity at a rate of n^{-1}; see Duffie (1996). This simple pricing methodology, however, results in the exponential-time algorithm alluded to in the introduction. When deriving explicit error bounds, we shall disregard the e^{-rn} factor in formula (1) for convenience. This omission leads to pessimistic error bounds.

Notation `A:B` will refer to the range-bound algorithm that employs the lower-bound algorithm `A` and the upper-bound algorithm `B` to bracket the benchmark option value (1). Because $\texttt{A} \leq \texttt{benchmark} \leq \texttt{B}$ by definition, the difference of their outputs, written as $\texttt{B} - \texttt{A}$, is an upper bound on the deviate of `A` and `B` from the benchmark value. In numerical experiments $\texttt{B} - \texttt{A}$ will be used as a measure of the pricing error of the range-bound algorithm `A:B`.

American-style options give holders the right to exercise the options before maturity. Define a **running average** as

$$A_i \equiv \frac{S_0 + S_1 + \cdots + S_i}{i+1}.$$

On each node of the binomial tree during backward induction, the option value if not exercised (that is, the continuation value) should be compared to the exercise value $(A_i - X)^+$ with the larger one retained. The existence of early exercise complicates the design and analysis of algorithms. Again, the option value given by the exponential-time binomial tree algorithm will serve as our benchmark. It goes without saying that some of the 2^n paths may be terminated prematurely by early exercise.

3 Preliminaries for European-Style Asian Options

The sum of a **path prefix** $(S_0, S_1, \ldots, S_j)$ is defined by $\sum_{i=0}^{j} S_i$. We call it a **prefix sum**. To speed up the computation, an approximation algorithm replaces the random variable S_i by some other random variable $\hat{S}_i$, which can be thought of as an approximation to S_i with deviate $D_i = S_i - \hat{S}_i$. Note that $D_0 = 0$. The core computational problem can now be rephrased as

$$E\left[\max\left\{\frac{1}{n+1}\sum_{i=0}^{n}(\hat{S}_i + D_i), X\right\}\right].$$

If the algorithm calculates

$$E\left[\max\left\{\frac{1}{n+1}\sum_{i=0}^{n}\hat{S}_i, X\right\}\right]$$

instead, the magnitude of error will be bounded above by

$$E\left[\frac{1}{n+1}\sum_{i=0}^{n}\mid D_i \mid\right]. \tag{2}$$

A path with a prefix sum equal to or exceeding $(n+1)X$ at some node is guaranteed to end with a price average at least X, thus in or at the money. This path's contribution to the option value is given by the following lemma under the risk-neural probability.

Lemma 3.1 (Aingworth et al. (2000)) *Suppose that a path prefix of length j has price sum $(n+1)X+\epsilon$, where $\epsilon \geq 0$, and it ends at a node with stock price S_j. Then the discounted expected European-style Asian option payoff by the extension of that path prefix to paths of length n equals*

- $[\,\epsilon + (n-j)S_j\,]/(n+1)$ *when* $r = 0$, *and*
- $e^{-nr}[\,\epsilon + \frac{1-e^{(n-j)r}}{1-e^r}\,S_j e^r\,]/(n+1)$ *when* $r > 0$.

Proof. Assume that the path prefix is $(S_0, S_1, \ldots, S_j)$. If $r > 0$, then the expected value of the future price sum $S_{j+1} + S_{j+2} + \cdots + S_n$ equals

$$\sum_{i=j+1}^{n} S_j \left[e^r + e^{2r} + \cdots + e^{(n-j)r} \right] = S_j e^r \frac{1 - e^{(n-j)r}}{1 - e^r}$$

under the risk-neutral probability. The expected value of the average $A_n = (S_0 + S_1 + \cdots + S_n)/(n+1)$ therefore equals

$$\left[(n+1)X + \epsilon + S_j e^r \frac{1 - e^{(n-j)r}}{1 - e^r} \right] /(n+1) = X + \left[\epsilon + S_j e^r \frac{1 - e^{(n-j)r}}{1 - e^r} \right] /(n+1).$$

Because each path will end up in or at the money, the expected option payoff equals the above minus X, i.e.,

$$\left[\epsilon + S_j e^r \frac{1 - e^{(n-j)r}}{1 - e^r} \right] /(n+1).$$

The case of $r = 0$ is similar. □

Lemma 3.1 implies that a European-style Asian option pricing algorithm can limit the range of prefix sums at each node to range $[\,0, (n+1)X)$. The contribution of prefix sums equal to or exceeding $(n+1)X$ to the option value can be calculated exactly by the lemma. Without this upper limit of $(n+1)X$, algorithms for European-style Asian options may have difficulty converging for sufficiently large σ (see §7.2). We remark that the lemma can be improved by incorporating information regarding the last price S_j on the path prefix.

4 The Design and Error Analysis of the Basic Range-Bound Algorithm

4.1 Description of the Algorithms

We start by segmenting the range $[\,0, (n+1)X\,]$ at each node $N(i,j)$ by $k_{ij} + 1$ *equally-distanced* **buckets**. The ℓth bucket, $0 \le \ell \le k_{ij}$, is associated with prefix

sum $\ell(n+1)X/k_{ij}$. Two adjacent buckets' associated prefix sums are thus separated by $(n+1)X/k_{ij}$. See Fig. 4 for illustration. In practice, bucket k_{ij} is not really needed in pricing European-style Asian options. The reason is that it is guaranteed to end up in or at the money, making Lemma 3.1 applicable. We maintain it to simplify the presentation.

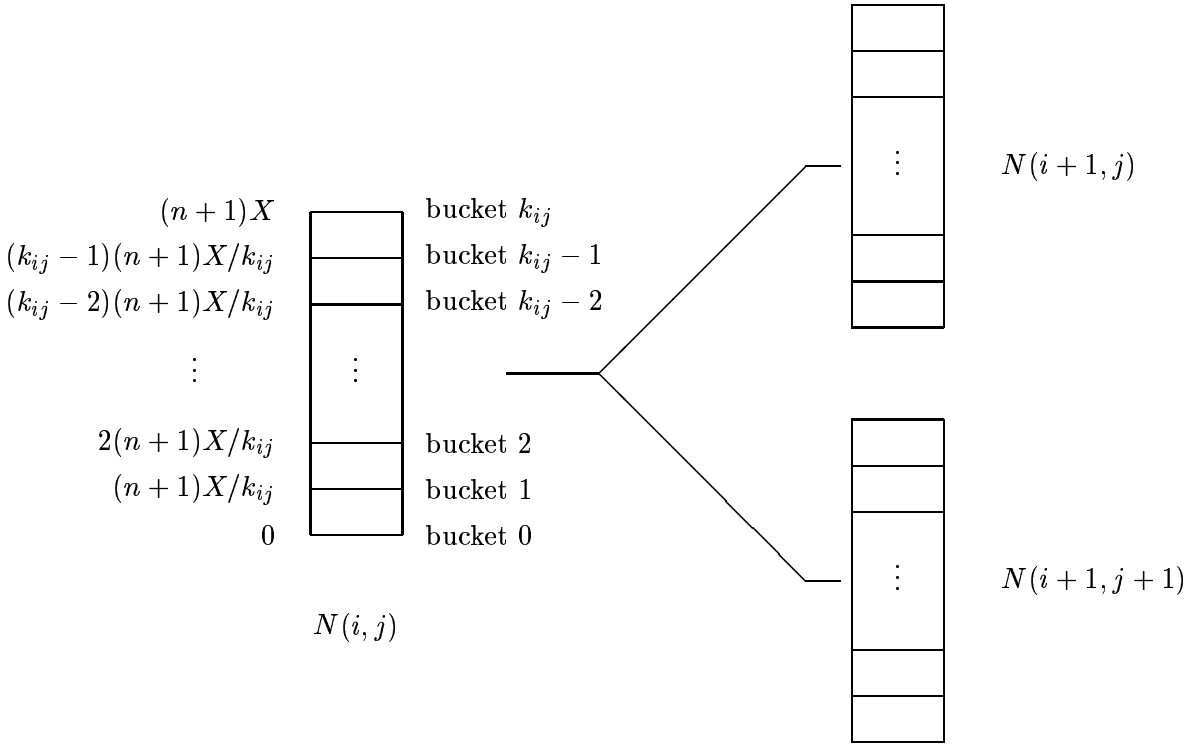

Figure 4: **Bucketing.** Each node $N(i,j)$ has $k_{ij}+1$ buckets, starting from prefix sum 0 and ending at prefix sum $(n+1)X$ with increments of $(n+1)X/k_{ij}$.

Let $b(i,j,\ell)$ denote the ℓth bucket at node $N(i,j)$ and $s(i,j,\ell)$ denote the associated prefix sum. For the current algorithm, the prefix sums are fixed at

$$s(i,j,\ell) = \ell(n+1)X/k_{ij}. \tag{3}$$

The root node $N(0,0)$ is a special case, with $k_{00} = 1$ and $s(0,0,0) = S_0$. The contribution to the option value by prefix sums equal to or exceeding $(n+1)X$ before

maturity is given by Lemma 3.1. The contribution of each prefix sum $s \geq (n+1)X$ at maturity is $e^{-nr}[\, s/(n+1) - X\,]$. Any other bucket at maturity (that is, those $b(n,j,\ell)$ with $\ell < k_{ij}$) contributes nothing to the option value because

$$e^{-nr}\left[\frac{s(n,j,\ell)}{n+1} - X\right]^+ = e^{-nr}\left(\frac{\ell X}{k_{ij}} - X\right)^+ = 0.$$

The algorithm uses forward induction to calculate the probability associated with each bucket. Each bucket's associated prefix sum most likely does not correspond to a valid prefix sum on the binomial tree. Paths are hence rounded to the nearest bucket in the following way. When a prefix sum falls between two adjacent buckets, the sum is rounded *down* to the lower bucket. Specifically, at node $N(i,j)$ with stock price $S_0u^{i-j}d^j$, the paths collected at bucket $b(i,j,\ell)$ are expected to move up to node $N(i+1,j)$ with prefix sum $s(i,j,\ell) + S_0u^{i-j+1}d^j$ and down to node $N(i+1,j+1)$ with prefix sum $s(i,j,\ell) + S_0u^{i-j}d^{j+1}$. These two prefix sums are then rounded down to the nearest buckets, called the **up bucket** and the **down bucket** of $b(i,j,\ell)$, respectively.

The above discussion entails the following inductive procedure for the desired probabilities. The probability $p(i,j,\ell)$ for bucket $b(i,j,\ell)$ is multiplied by p and added to the probability of the up bucket:

$$b\left(i+1, j, \left\lfloor \frac{s(i,j,\ell) + S_0u^{i-j+1}d^j}{(n+1)X/k_{i+1,j}} \right\rfloor\right). \tag{4}$$

Similarly, the same probability $p(i,j,\ell)$ is multiplied by $1-p$ and added to the probability of the down bucket:

$$b\left(i+1, j+1, \left\lfloor \frac{s(i,j,\ell) + S_0u^{i-j}d^{j+1}}{(n+1)X/k_{i+1,j+1}} \right\rfloor\right). \tag{5}$$

See Fig. 5 for illustration. Bucket $b(i,j,\ell)$ therefore **covers** the prefix sums s in range

$$\ell(n+1)X/k_{ij} \leq s < (\ell+1)(n+1)X/k_{ij}. \tag{6}$$

Observe that the range has a width of $(n+1)X/k_{ij}$. The algorithm in effect calculates for each bucket the probability that a path has a prefix sum covered by that bucket.

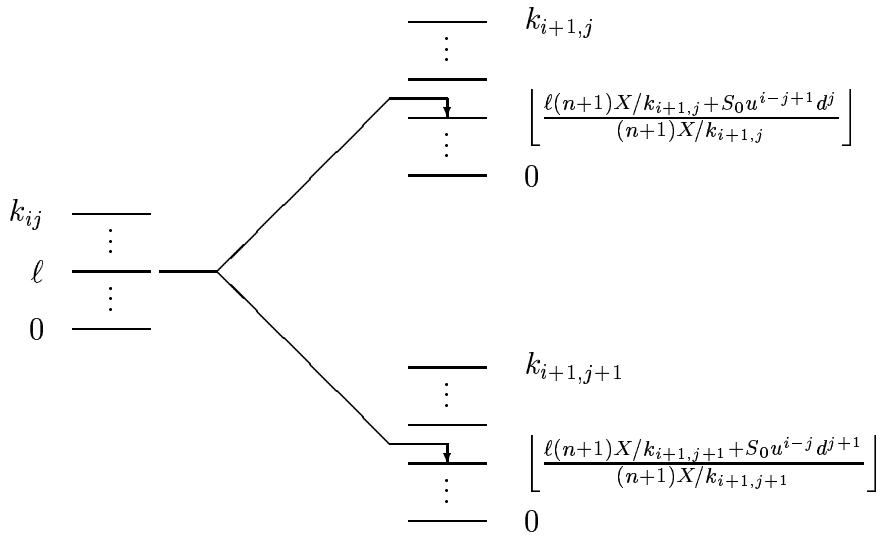

Figure 5: **Rounding down the partial sums.** Each of the $k_{ij}+1$ buckets at $N(i,j)$ moves up to the bucket of node $N(i+1,j)$ and down to the bucket of node $N(i+1,j+1)$ as shown. In the rounding-up version, the floor operations are replaced with the ceiling operations.

We call this algorithm `nUnifDown`. The prefix `nUnif` emphasizes the nonuniformity of bucketing, as the number of buckets, $k_{ij}+1$, may vary from nodes to nodes. The suffix `Down` means that the successor buckets are located by rounding down. If we change the rounding-down operations in formulas (4) and (5) to rounding-up, the resulting algorithm is called, naturally, `nUnifUp` with the suffix `Up`. Evidently, `nUnifDown` and `nUnifUp` bracket the benchmark option value:

$$\texttt{nUnifDown} \leq \texttt{benchmark} \leq \texttt{nUnifUp}. \tag{7}$$

Hence `nUnifDown:nUnifUp` is a range-bound algorithm. In the next subsection, we shall show that, by a judicious choice of k_{ij}, very tight error bounds obtain.

Both `nUnifDown` and `nUnifUp` share the bucketing idea of the Hull-White paradigm. But there are also significant differences. The Hull-White paradigm does not take advantage of the limited range of prefix sums made possible by Lemma 3.1. It also uses interpolation, not rounding, in pricing, making the analysis of error much more difficult. Later in the paper, we will prove rigorously for the first

time in the literature that the algorithms in Hull (1997) and Hull and White (1993) are upper-bound algorithms. The forward shooting grid method of Barraquand and Pudet (1996) finds the successor bucket by rounding to the nearest bucket, which is either the up bucket or the down bucket.

By picking a uniform $k_{ij} = k$ to make the number of buckets the same $k+1$ for every node, the algorithm in Aingworth et al. (2000) is a special case of our `nUnifDown`. We shall call it `UnifDown` to emphasize its uniformity of bucketing. Its rounding-up version shall be labeled `UnifUp`. The range-bound result (7) clearly does not depend on the buckets being equally distanced. A very popular alternative is for the logarithms of the buckets to be equally distanced (see Hull and White (1993)). All our algorithms and all the theoretical results (except the one with a specific error bound, Theorem 4.1) are independent of the ways the buckets are distanced.

4.2 An Optimal Choice of the Number of Buckets

Denote the rounding error at node $N(i,j)$ by D_{ij}. Then the pricing error's upper bound (2) is further bounded above by

$$\frac{1}{n+1}\sum_{i=0}^{n}\sum_{j=0}^{i}\binom{i}{j}p^{i-j}(1-p)^j|D_{ij}|. \tag{8}$$

Because $|D_{ij}|$ are not identically weighted in formula (8), k_{ij} should not be a constant.

Set

$$\texttt{TIME} = \sum_{0\le j\le i\le n} k_{ij}, \tag{9}$$

the total number of buckets allocated by the algorithm. The running time is proportional to `TIME`. Note that the summands are not $k_{ij}+1$ because bucket k_{ij}, as we mentioned earlier, is not allocated in practice.

As any two adjacent buckets at node $N(i,j)$ are $(n+1)X/k_{ij}$ apart, it follows that $|D_{ij}| \le (n+1)X/k_{ij}$. With $B(i,j;p) \equiv \binom{i}{j}p^{i-j}(1-p)^j$, pricing error (8) is bounded

above by

$$\texttt{ERROR} = X \sum_{i=1}^{n} \sum_{j=0}^{i} \frac{B(i,j;p)}{k_{ij}}. \tag{10}$$

Apply the Cauchy-Schwartz inequality to formulas (9) and (10) to obtain

$$\texttt{TIME} \times \texttt{ERROR} \geq X \left[\sum_{0 \leq j \leq i \leq n} \sqrt{B(i,j;p)} \right]^2.$$

Equality holds if and only if $\sqrt{B(i,j;p)}/k_{ij}$ are equal for all i and j. If the budget for `TIME` is fixed, then `ERROR` is minimized by setting

$$k_{ij} = \texttt{TIME} \times \frac{\sqrt{B(i,j;p)}}{\sum_{0 \leq j \leq i \leq n} \sqrt{B(i,j;p)}}.$$

The pricing error's upper bound,

$$\frac{X}{\texttt{TIME}} \times \left[\sum_{0 \leq j \leq i \leq n} \sqrt{B(i,j;p)} \right]^2, \tag{11}$$

is then maximized with $p = 1/2$.

The algorithm can now be completely specified with

$$k_{ij} = \left\lceil \texttt{TIME} \times \frac{\sqrt{B(i,j;p)}}{\sum_{0 \leq j \leq i \leq n} \sqrt{B(i,j;p)}} \right\rceil. \tag{12}$$

Equation (9) and the choice

$$\texttt{TIME} = kn^2/2$$

imply that k measures the average number of buckets per node.

4.3 Error Analysis

We proceed to derive an upper bound on the pricing error (11) of `nUnifDown`. The same proof works for `nUnifUp`. Recall that $p = 1/2$. Bender (1974) shows that

$$\sum_{0 \leq j \leq i} \sqrt{B(i,j;1/2)} \sim (2\pi i)^{1/4}.$$

Substitute the above into formula (11) to yield

$$\texttt{ERROR} \leq \frac{X}{\texttt{TIME}} \times \left[\sum_{0 \leq i \leq n} (2\pi i)^{1/4} \right]^2 .$$

As $\sum_{0 \leq i \leq n} i^{1/4} \sim \int_0^n x^{1/4} dx = \frac{4}{5} n^{5/4}$ and $\texttt{TIME} = kn^2/2$,

$$\texttt{ERROR} \leq \frac{X}{kn^2/2} \sqrt{2\pi}\, (4/5)^2 n^{5/2} < 4X\sqrt{n}/k$$

asymptotically. We have therefore proved the following theorem.

Theorem 4.1 *European-style Asian options can be approximated within $O(X\sqrt{n}/k)$ by the $O(kn^2)$-time range-bound algorithm* `nUnifDown:nUnifUp` *with the numbers of buckets given by formula (12).*

We make a few remarks regarding the error bound in the above theorem. Although k_{ij} are chosen in an optimal way, optimality is relative to the upper bound (10) on the pricing error and not the exact pricing error. Hence, a better choice than formula (12) is conceivable. By the same token, the same error bound may be achievable with much smaller k_{ij} than asked for in the theorem. Finally, the bound is independent of the stock price volatility σ.

5 Tighter Range Bounds

Both bucketing and rounding schemes impact the quality of approximation. There are uniform bucketing and the more general nonuniform bucketing to choose from. There also exist different ways to distance the buckets, a topic not investigated in this paper. As to rounding, there have been two choices: rounding-down and rounding-up. Neither is ideal because every path is either uniformly rounded down or uniformly rounded up at each node, generating a systematic bias. A straightforward answer is to average the results of `UnifDown` and `UnifUp` in the hope of cancelling the rounding errors but at the cost of doubling the running time. Call this method `UnifAvg`.

5.1 A tighter upper-bound algorithm

Our next algorithm applies the averaging idea bucket by bucket. Although the algorithm is similar to `nUnifDown` and `nUnifUp`, it no longer always rounds down a path to the down bucket as in `nUnifDown` or rounds up a path to the up bucket as in `nUnifUp`. Instead, a path is split into both buckets in such a portion that the average prefix sums associated with the two buckets equals the prefix sum of the original path.

More precisely, consider node $N(i+1,j)$ and a path with a prefix sum s at $N(i+1,j)$ after making an up move from bucket $b(i,j,k)$. Recall that scheme `nUnifDown` directs the path to the down bucket $b(i+1,j,\texttt{RoundDown})$, whereas scheme `nUnifUp` directs the path to the up bucket $b(i+1,j,\texttt{RoundUp})$, where

$$\begin{aligned} \texttt{RoundDown} &= \left\lfloor \frac{sk_{i+1,j}}{(n+1)X} \right\rfloor, \\ \texttt{RoundUp} &= \left\lceil \frac{sk_{i+1,j}}{(n+1)X} \right\rceil. \end{aligned}$$

Our new algorithm does a bit of both by solving

$$s = \lambda \cdot s(i+1,j,\texttt{RoundUp}) + (1-\lambda) \cdot s(i+1,j,\texttt{RoundDown}) \tag{13}$$

for λ. It then gives proportions λ and $1-\lambda$ of $p(i,j,k)$—the probabilistic weight of $b(i,j,k)$—to the up bucket and the down bucket, respectively. See Fig. 6 for illustration. In the unlikely event that $\texttt{RoundDown} = \texttt{RoundUp}$, we let $\lambda = 1$. Repeat the same steps for the down move from bucket $b(i,j,k)$. Other than the splitting of paths, the algorithm is identical to `nUnifDown`. Call this algorithm `nUnifSpl` (for splitting).

We make a few remarks here. Every path is conceptually split into 2^n paths in `nUnifSpl`. Some of the paths may be terminated earlier if their prefix sums ever equal or exceed $(n+1)X$, where Lemma 3.1 takes over. Each of the paths is rounded to buckets along the way to prevent combinatorial explosion. Furthermore, for any $0 \le m \le n$, a path's prefix sum of m prices equals the expected length-m prefix sum of all the split paths.

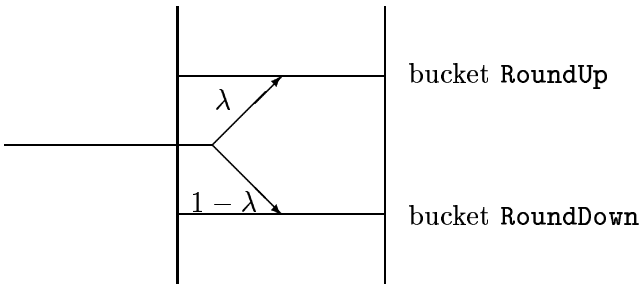

Figure 6: **Splitting a path.** Bucket `RoundUp` receives λ of the probabilistic weight. Bucket `RoundDown` receives $1 - \lambda$ of the probabilistic weight.

5.2 A tighter lower-bound algorithm

The core idea of our next algorithm is to use

$$e^{-rn}\left(E\left[\frac{1}{n+1}\sum_{i=0}^{n} S_i - X\right]\right)^+$$

to approximate the desired option value (1). The approximation underestimates the option value because of Jensen's inequality. Our algorithm adds bucketing to the above idea.

To implement the idea with bucketing, the prefix sums $s(i,j,\ell)$ will no longer be fixed by formula (3). Instead, they need to be calculated explicitly to hold the average prefix sum of all the paths covered by bucket $b(i,j,\ell)$ with range (6). Because any path ending up at bucket $b(i,j,\ell)$ carries the same probability $p^{i-j}(1-p)^j$, $s(i,j,\ell)$ equals the simple arithmetic average of those said prefix sums. Probability $p(i,j,\ell)$ still records the probability of reaching $b(i,j,\ell)$ from the root and is calculated the same way as in `nUnifDown`. But the expected price sum of the up bucket,

$$s\left(i+1, j, \left\lfloor \frac{s(i,j,\ell) + S_0 u^{i-j+1} d^j}{(n+1)X/k_{i+1,j}} \right\rfloor\right),$$

is updated by adding $p \cdot p(i,j,\ell) \cdot [\, s(i,j,\ell) + S_0 u^{i-j+1} d^j \,]$ to it. Similarly, the expected price sum of the down bucket,

$$s\left(i+1, j+1, \left\lfloor \frac{s(i,j,\ell) + S_0 u^{i-j} d^{j+1}}{(n+1)X/k_{i+1,j+1}} \right\rfloor\right),$$

is updated by adding $(1-p)\cdot p(i,j,\ell)\cdot[\, s(i,j,\ell)+S_0u^{i-j}d^{j+1}\,]$ to it. After the above is done for every $s(i,j,\ell)$ at time i and before we move on to time $i+1$, every nonzero $s(i,j,\ell)$ is divided by $p(i,j,\ell)$ to turn it into the average prefix sum. The rest of the algorithm is identical to `nUnifDown`. Specifically, the algorithm applies Lemma 3.1 for price sums at or over $(n+1)X$ before maturity and add up the contributions of the prefix sums over $(n+1)X$ at maturity. We call this algorithm `nUnifCvg` (for convergence).

We illustrate the idea in Fig. 7 without bucketing for simplicity. Take the node with probability $2p(1-p)$ for example. Its prefix sum is calculated by adding up the contribution through the S_0u node and that through the S_0d node. The result is

$$\begin{aligned} & (1-p)p(S_0+S_0u+S_0)+p(1-p)(S_0+S_0d+S_0) \\ = \;& p(1-p)[\,(S_0+S_0u+S_0)+(S_0+S_0d+S_0)\,] \end{aligned}$$

Its associated probability is calculated in the standard way,

$$p(1-p)+(1-p)p=2p(1-p).$$

Finally, we divide the nonzero prefix sum by the above probability to obtain

$$[\,(S_0+S_0u+S_0)+(S_0+S_0d+S_0)\,]/2,$$

the desired average prefix sum. The other average price sums at maturity are given in Fig. 7. The approximation is hence

$$\begin{aligned} e^{-rn} \quad & \Big[p^2\left\{(S_0+S_0u+S_0u^2)-X\right\}^+ \\ & +2p(1-p)\left\{\frac{(S_0+S_0u+S_0)+(S_0+S_0d+S_0)}{2}-X\right\}^+ \\ & +(1-p)^2\left\{(S_0+S_0d+S_0d^2)-X\right\}^+\Big]. \end{aligned}$$

The following key result says that the algorithm `nUnifCvg:nUnifSpl` produces tighter range bounds than `nUnifDown:nUnifUp`.

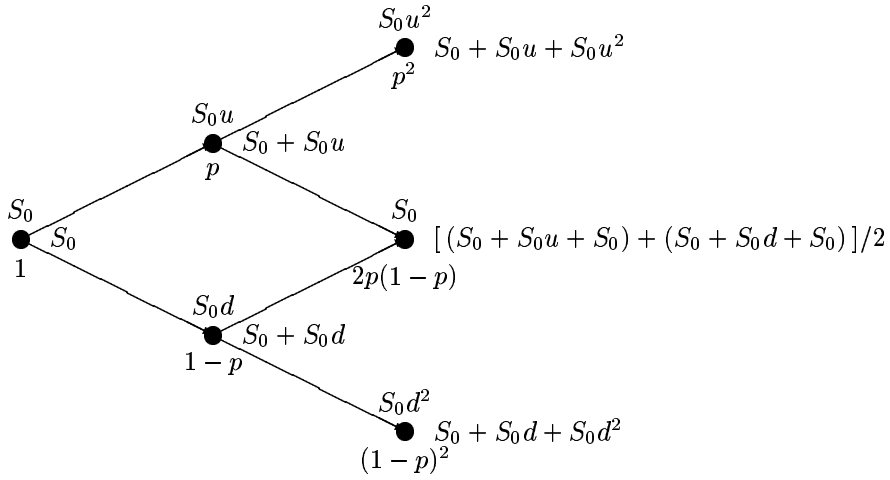

Figure 7: **Price-sum averaging with `nUnifCvg`.** The average prefix sums are listed to the right of the nodes. The probabilities for the average prefix sums are listed under the nodes.

Theorem 5.1 `nUnifDown` $\leq$ `nUnifCvg` $\leq$ `benchmark` $\leq$ `nUnifSpl` $\leq$ `nUnifUp`.

Proof. We knew that `nUnifDown` $\leq$ `benchmark` $\leq$ `nUnifUp`. It should also be clear that `nUnifDown` $\leq$ `nUnifCvg` and `nUnifSpl` $\leq$ `nUnifUp`. That `nUnifCvg` $\leq$ `benchmark` is a consequence of the inequality

$$\left(E\left[\frac{1}{n+1}\sum_{i=0}^{n} S_i - X\right]\right)^+ \leq E\left[\left(\frac{1}{n+1}\sum_{i=0}^{n} S_i - X\right)^+\right],$$

which holds for each bucket at maturity by Jensen's inequality.

It remains to show that `benchmark` $\leq$ `nUnifSpl`. Recall that `nUnifSpl` splits each path $P = (S_0, S_1, \ldots, S_n)$ into 2^n paths with the consequence that the expected value of the average price for any path x, say A_x, equals the average price of P, i.e., $E[\,A_x\,] = \sum_{i=0}^{n} S_i/(n+1)$. By Jensen's inequality, P's payoff, $\{\sum_{i=0}^{n} S_i/(n+1) - X\}^+$, is dominated by `nUnifSpl`'s approximation, $E[\,(A_x - X)^+\,]$. As this inequality holds for every path, the desired result follows. □

If `nUnifCvg:nUnifSpl` chooses the numbers of buckets, k_{ij}, according to formula (12), then Theorem 4.1 implies an error bound of $O(X\sqrt{n}/k)$. Our experiments in

the next section will show that this error bound is overpessimistic.

6 Numerical Results for European-Style Asian Options

All the experimental results reported here are based on a Pentium III 500MHz PC with 256MB DRAM. The programs are written in C++. Our key finding will be that the range-bound algorithm `nUnifCvg:nUnifSpl` is efficient and brackets the benchmark value extremely tightly. It also outperforms all the other range-bound algorithms in the paper for accuracy and efficiency.

We first confirm that our particular nonuniform bucketing scheme (12) improves upon the uniform bucketing scheme. Toward that end, the nonuniform `nUnifUp` is compared against the uniform `UnifUp` and `UnifAvg`, with the Monte Carlo algorithm as the proxy benchmark. To be fair, the total number of buckets is the same for all three algorithms. The plot in Fig. 8 shows that the nonuniform `nUnifUp` converges more smoothly and quickly than either `UnifUp` or `UnifAvg`, which in turn can be seen to outperform `UnifUp`. This conclusion is consistent with the theoretical result.

Having demonstrated the advantage of using our nonuniform bucketing scheme, we next compare the nonuniform lower-bound algorithm `nUnifCvg` against `UnifAvg` and the multiresolution (MR) algorithm in Dai and Lyuu (1999). Again, both `nUnifCvg` and `UnifAvg` use the same total number of buckets. The results are tabulated in Table 1. Observe that `nUnifCvg` is superior to `UnifAvg` in terms of accuracy and speed despite that it takes slightly less time than `UnifAvg`. The MR algorithm, which is based on trinomial trees, is the best performer when n is small. But since it runs in time mildly exponential in n, it is not competitive when n is large. We then add the algorithm in Hull and White (1993) and Levy's analytical approach in Levy (1992) to the comparisons in Table 2. Observe that `nUnifCvg` is competitive in all the scenarios whether the option is in the money or out of the money. Many proposed

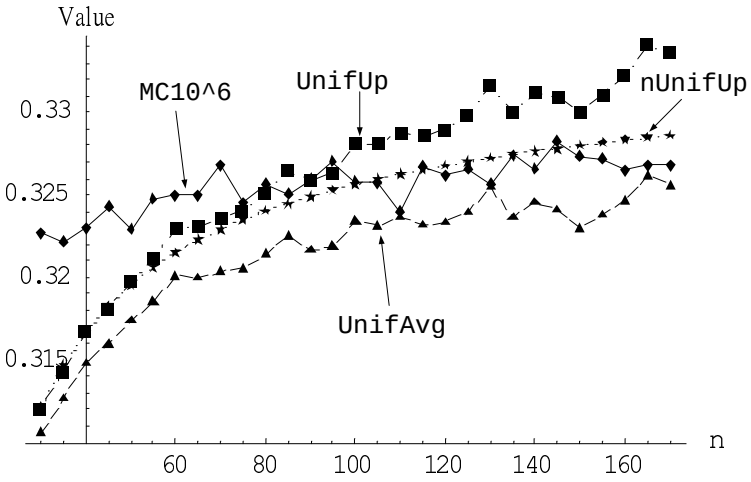

Figure 8: **Comparing `UnifUp`, `UnifAvg`, and `nUnifUp`.** MC10^6 denotes Monte Carlo simulation based on 1,000,000 trials. Both `UnifAvg` and `UnifUp` allocate $k = 50,000$ buckets at each node, and `nUnifUp` allocates an average of $k = 50,000$ buckets per node. The benchmark option values must lie below `nUnifUp`. The data are: $S_0 = 50$, $X = 60$, annual volatility $\sigma = 30\%$, $r = 10\%$ per annum, and maturity $\tau = 0.5$ (year).

algorithms have been found to fail in extreme cases. When the cases of Fu et al. (1998/1999) are tested in Table 3, `nUnifCvg` does not display any discernible biases. These tests strongly suggest that `nUnifCvg` gives extremely tight lower bounds.

Both `nUnifCvg` and `nUnifSpl` are more sophisticated strategies to handle pricing errors than `UnifUp`, `UnifDown`, or `UnifAvg`. We now demonstrate that they are also better strategies. For the purpose of fair comparison between these methods, we reduce the number of buckets for `nUnifCvg` and `nUnifSpl` so that they take slightly less time than `UnifUp`. The results are illustrated in Fig. 9. Clearly both `nUnifSpl` and `nUnifCvg` converge better than `UnifUp`, `UnifDown`, or `UnifAvg`. In fact, the range bounds given by `nUnifCvg:nUnifSpl` are so tight that they form a single curve instead of a band. The same cannot be said of `UnifDown:UnifUp`. As `nUnifCvg:nUnifSpl` brackets the benchmark option value (Theorem 5.1), its pricing error must be negligible.

By how much is `nUnifCvg:nUnifSpl` superior to `nUnifDown:nUnifUp`? Because

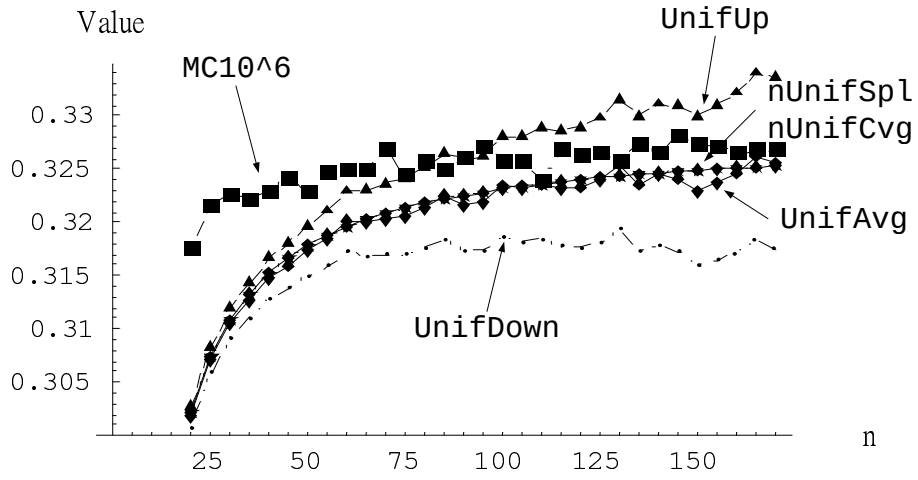

Figure 9: **Comparing `nUnifCvg` and `nUnifSpl` against `UnifUp`, `UnifDown`, and `UnifAvg`.** `UnifUp`, `UnifDown`, and `UnifAvg` each set $k = 50,000$, whereas `nUnifSpl` and `nUnifCvg` set $k = \lfloor 50,000/7 \rfloor$. These choices make `nUnifCvg` and `nUnifSpl` take slightly less time than `UnifUp`. The plots of `nUnifCvg` and `nUnifSpl` essentially coincide. The data are: $S_0 = 50$, $X = 60$, $\sigma = 30\%$, $r = 10\%$ per annum, and $\tau = 0.5$.

both employ the same nonuniform bucketing scheme, this comparison extracts the benefits which accrue to different algorithms, not bucketing schemes. We use the range bounds to measure the pricing error, i.e., `nUnifSpl` $-$ `nUnifCvg` and `nUnifUp` $-$ `nUnifDown`. It is clear from Fig. 10 that `nUnifCvg:nUnifSpl` produces much tighter range bounds, hence smaller errors, than `nUnifDown:nUnifUp`.

It is not paradoxical for the pricing error to rise with n in Fig. 10. The reason is that our k in the experiments are constants independent of n. Suppose we now pick $k = n$ to make `nUnifCvg:nUnifSpl`'s time bound $O(n^3)$. The data in Table 4 show a convergence rate of $O(n^{-2})$. This is much better than the $O(n^{-0.5})$ guaranteed by Theorem 4.1, which overestimates the pricing error. They also demonstrate that the resulting range bounds are extremely tight for σ as high as 100% and τ as long as 5 years. Indeed, our theoretical error bounds are independent of σ. To our knowledge, Aingworth et al. (2000) is the only paper besides ours with numerical results at $\sigma > 50\%$ for European-style Asian options. Not surprisingly, that paper makes use of

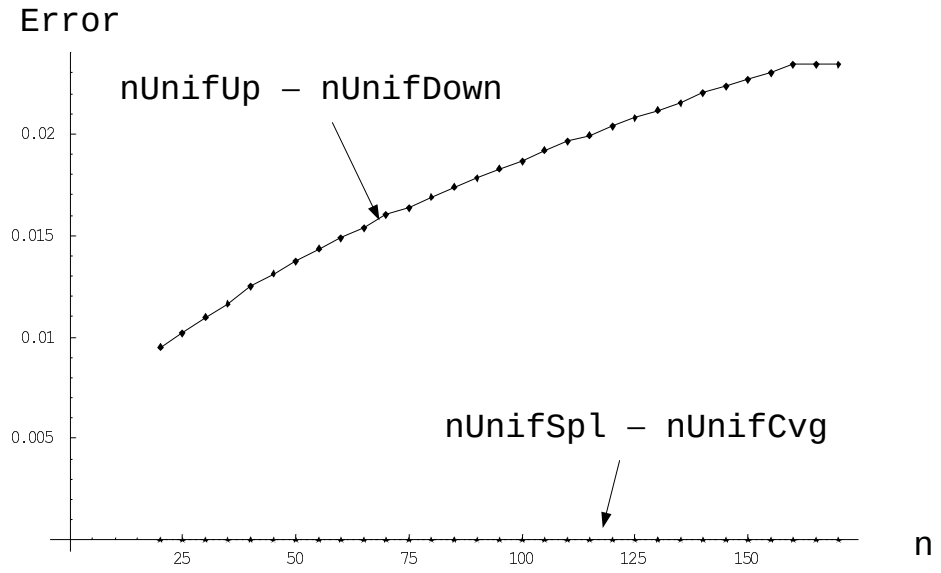

Figure 10: **Pricing errors of `nUnifCvg:nUnifSpl` and `nUnifDown:nUnifUp`.** All four algorithms set $k = \lfloor 50,000/7 \rfloor$. The other parameters are identical to the ones used in Fig. 9.

Lemma 3.1 too. Data in §7.2 strongly suggest that an upper limit on the price sum like the one given by Lemma 3.1 is extremely helpful for convergence when σ is large. Stocks with a large volatility are quite common; technology stocks such as Cisco, Corning, Lucent, Nokia, Oracle, and Sun, for example, have an implied volatility ranging from 51% to 87% as of February 2002.

Finally, we are interested in knowing how k—the average number of buckets per node—impacts the accuracy of the algorithms. Figure 11 shows the effects of k on the theoretical error upper bound,

$$\texttt{nUnifUp} - \texttt{nUnifDown} \leq 8X\sqrt{n}/k,$$

and our range-bound algorithms' error bounds as defined above. We can see that `nUnifCvg:nUnifSpl` is several orders better than `nUnifDown:nUnifUp` for any k with other conditions being equal.

We conclude from the above experiments that `nUnifCvg:nUnifSpl` is an extremely

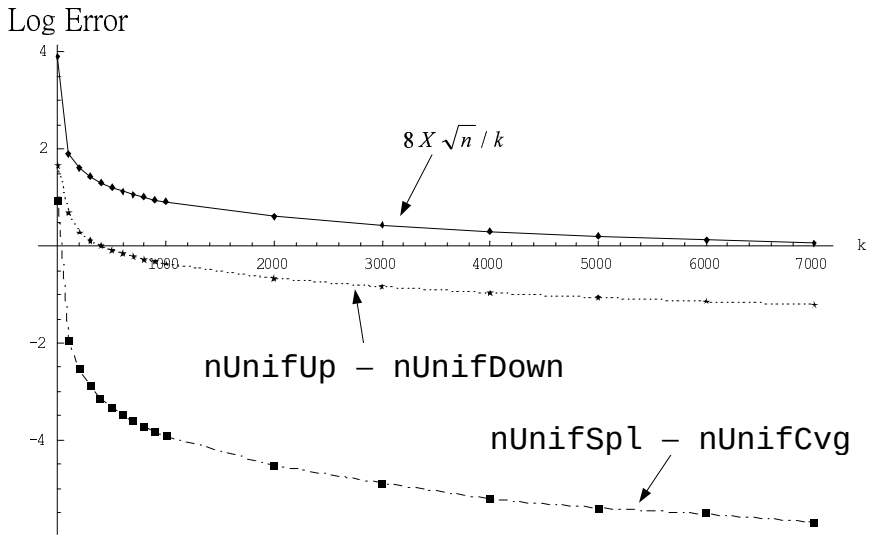

Figure 11: **The number of buckets and accuracy.** This plot shows the log-plot (base 10) of the various error bounds vs. k. We use $n = 285$ here. The other parameters are identical to the ones used in Fig. 9.

efficient and accurate pricing algorithm for European-style Asian options.

7 Algorithms for American-Style Asian Options

The early-exercise feature makes American-style Asian options harder to price than their European-style counterparts. The feature, for instance, invalidates Lemma 3.1, which has been instrumental in the pricing of European-style Asian options. To find the optimal exercise strategy for American-style Asian options, backward induction must somehow be used. The algorithms to follow focus on calls. Their extension to puts is straightforward.

7.1 Some useful terminology

We first introduce a few terms for ease of later discussions. Let $\mathcal{R}_{\max}(i,j)$ and $\mathcal{R}_{\min}(i,j)$ denote the largest and the smallest prefix sums among all the paths that

end at node $N(i,j)$. Obviously, $\mathcal{R}_{\max}(i,j)$ is achieved by the path that makes $i-j$ up moves followed by j down moves, whereas $\mathcal{R}_{\min}(i,j)$ is achieved by the path that makes j down moves followed by $i-j$ up moves. Both are straightforward to calculate (see Lyuu (2002)). The prefix-sum range at node $N(i,j)$ is $[\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$.

We now define the **ideal tree**, a critical concept that allows us to derive range bounds. The ideal tree has an uncountably infinite number of buckets. A bucket exists at node $N(i,j)$ for *each* real number $s \in [\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$. Any prefix sum encountered by our approximation algorithms must correspond to some bucket at the same node in the ideal tree. **Practical trees** refer to the necessarily finite-sized, bucket-based trees used by our approximation algorithms.

For any bucket b, we use P_b instead of the earlier $s(\cdot,\cdot,\cdot)$ to denote its associated prefix sum for brevity. As before, the prefix sum P_b includes the stock price at b. Let E_b denote the option value for bucket b. Because the option value in the ideal tree may differ from that in the practical tree, we use the superscripts I and P to distinguish them. The option value at bucket b in the ideal tree and that in the practical tree, if b exists, become E_b^I and E_b^P, respectively. The loose notion of E_b^P applies to any backward-induction algorithm which keeps the maximum of the early-exercise value and the continuation value as the option value for each bucket. Because the unrecombining binomial tree is "embedded" in the ideal tree, so to speak, the option value at the single bucket allocated for the root node in the ideal tree equals the benchmark option value.

7.2 Algorithms for European-style Asian options: some needed adjustments

Before modifying our earlier algorithms to handle American-style Asian options, we first adopt $[\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$ instead of $[\,0, (n+1)X\,]$ as their prefix-sum ranges. This change is needed because a price sum exceeding $(n+1)X$ no longer results in

any easily calculated contribution to the payoff for American-style options.

Because the price-sum ranges have changed, the bucket-number distribution must vary with them. Define

$$R_{ij} \equiv \frac{\mathcal{R}_{\max}(i,j) - \mathcal{R}_{\min}(i,j)}{n+1}.$$

The steps in §4.2 remain valid after $B(i,j;p)$ is replaced with $B(i,j;p)R_{ij}$. In particular, the optimal choice (12) for k_{ij} becomes

$$k_{ij} = \left\lceil \texttt{TIME} \times \frac{\sqrt{B(i,j;p)R_{ij}}}{\sum_{0\le j\le i\le n}\sqrt{B(i,j;p)R_{ij}}} \right\rceil. \tag{14}$$

Algorithms `nUnifSpl` and `nUnifCvg` for European-style Asian options will refer to the modified versions henceforth. These modified algorithms will be called the **full-range** versions if clarity is at stake.

The price-sum ranges depend on σ and τ in an exponential manner because $u = e^{\sigma\sqrt{\tau/n}}$. Recall that Lemma 3.1 helps place a limit $(n+1)X$ on the price-sum ranges for earlier algorithms but the full-range versions cannot employ this lemma. Table 5 documents the performance of the full-range `nUnifCvg:nUnifSpl` under the same conditions as Table 4. Whereas the related `nUnifCvg:nUnifSpl`'s fast convergence is not affected by large σ in Table 4, the full-range version runs into some difficulty for large σ or large τ despite that much more buckets are allocated. The importance of the limit is thus confirmed.

7.3 The first upper-bound algorithm

The first algorithm is called `nUnifSplA` with the suffix `A` indicating American style. It is based on `nUnifSpl` with three straightforward modifications. First, backward induction is used instead of forward induction. But a new issue arises that needs to be addressed. Backward induction requires two option values from the following time. In the algorithm, each of the two option values will be linearly interpolated from the option values at buckets `RoundUp` and `RoundDown` using the proportions λ and $1-\lambda$

in Eq. (13). See Fig. 12 for illustration. Second, early exercise is considered at each bucket. Exercise generates value $P_b/(i+1) - X$ for a bucket b at time i. Third, k_{ij} buckets are allocated at node $N(i,j)$ for the prefix-sum range $[\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$ instead of the earlier range $[\,0, (n+1)X\,]$.

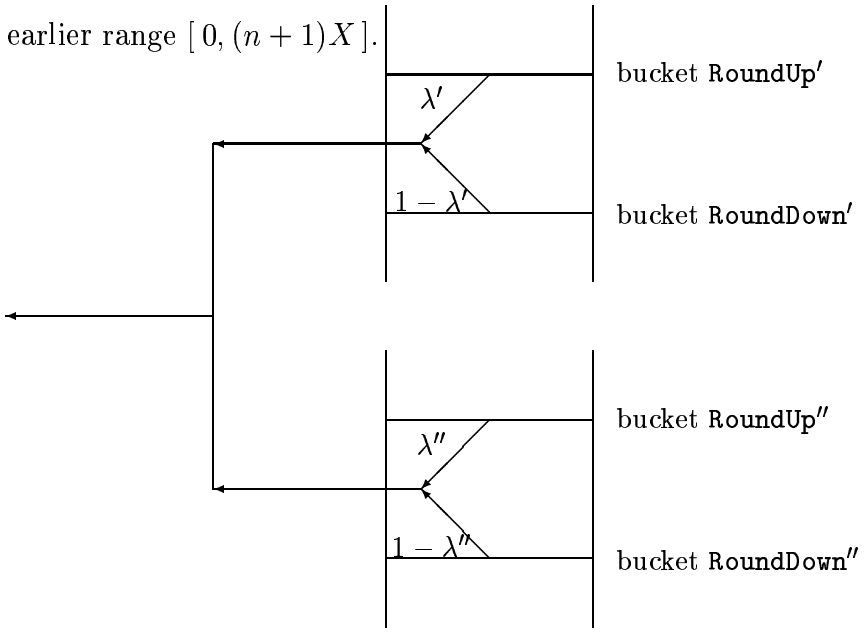

Figure 12: **Interpolation of option values in `nUnifSplA` in backward induction.** Linear interpolation is carried out at the successors.

7.4 The second upper-bound algorithm

As a rule, the smaller the prefix-sum range, the better the approximation. Because the payoffs of early-exercise buckets are clear, such nodes can be removed from the prefix-sum ranges, thus limiting the prefix-sum ranges further. But how are the early-exercise buckets distributed within a node? Before answering this key question, we state a useful lemma below.

Lemma 7.1 (Contraction lemma) *Suppose that $P_{b_1} > P_{b_2}$ at buckets b_1 and b_2 of the same node in the ideal tree at time m. Then*

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} \geq E^I_{b_1} - E^I_{b_2}.$$

Proof. See Appendix A. □

Fortunately, there exists a prefix sum at each node in the ideal tree that separates the early-exercise buckets from the non-early-exercise buckets. We next prove the key theorem establishing this fact.

Theorem 7.2 *Suppose that $P_{b_1} > P_{b_2}$ at buckets b_1 and b_2 of the same node in the ideal tree at time m. Assume that it is optimal to exercise the option at bucket b_2. Then it is optimal to exercise the option at b_1.*

Proof. We claim that

$$0 \geq \frac{P_{b_1}}{m+1} - X - E^I_{b_1} \geq \frac{P_{b_2}}{m+1} - X - E^I_{b_2} = 0.$$

That the option value is at least the exercise value proves the first inequality. The second inequality is by Lemma 7.1. The equality holds because b_2 is an early-exercise bucket. As $E^I_{b_1} = P_{b_1}/(m+1) - X$, bucket b_1 is an early-exercise bucket. □

What we are looking for at each node is a prefix sum that separates the early-exercise buckets from the non-early-exercise ones. This prefix sum is an **exercise boundary**. Early-exercise buckets can be pruned, which tightens the prefix-sum range at each node $N(i,j)$ by lowering $\mathcal{R}_{\max}(i,j)$ to the exercise boundary. The payoff of a pruned bucket is known anyway; it is the running average minus the strike price X.

Of course, it may happen that, in the practical tree, the early-exercise buckets are scattered within a node without a clear boundary separating them from the non-early-exercise ones. We will settle with an approximate boundary at and above which all early-exercise buckets lie. See Fig. 13 for illustration. Theoretically, some non-early-exercise buckets may still lie above that boundary.

We now present a two-phase algorithm, called `nUnifSplA2`, that incorporates the idea of exercise boundary. Phase one uses `nUnifSplA` to estimate the exercise

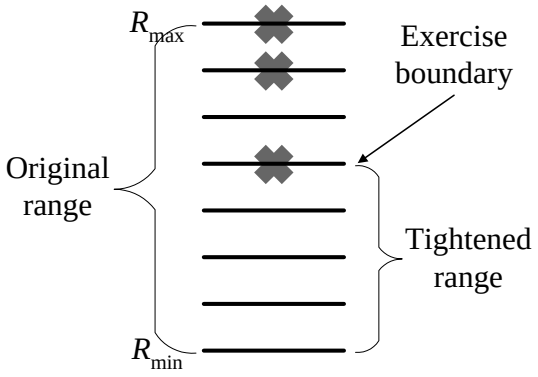

Figure 13: **Determination of the exercise boundary.** Buckets marked with a "×" are early-exercise buckets under `nUnifSplA`.

boundary at each node. This is done by inspecting each node for early-exercise buckets. The prefix-sum range is then tightened by lowering the maximum price sum to the lowest prefix sum whose corresponding bucket is exercised early. Phase two runs `nUnifSplA` on the reduced prefix-sum ranges. The k_{ij} need to be recalculated with formula (14) because ranges R_{ij} have been reduced in phase one. A bucket with a prefix sum on or above the exercise boundary will be exercised in phase two.

Although `nUnifSplA2` allocates the same number of buckets as `nUnifSplA`, its buckets cover more limited prefix-sum ranges. This has the effects of raising the "resolution" of the prefix-sum range at each bucket and thus the pricing accuracy. To be sure, `nUnifSplA2`'s running time doubles `nUnifSplA`'s. As mentioned in the introduction, the exercise boundary given by our algorithm is not useful only to the algorithms in the paper. It in fact gives rise to a general two-phase computational framework, in which any upper-bound algorithm can be substituted in phase two to give a more accurate two-phase upper-bound algorithm (see Section 8).

7.5 A lower-bound algorithm

We next present a lower-bound algorithm called `nUnifCvgA`. It is based on `nUnifCvg` and contains two phases. Phase one is identical to `nUnifSplA2`'s phase one. In other

words, it calls upon the upper-bound algorithm `nUnifSplA` to yield an exercise boundary, and the prefix-sum ranges are subsequently tightened. Phase two is identical to `nUnifSplA2`'s phase two except that it runs `nUnifCvg` instead of `nUnifSplA` over the tightened prefix-sum ranges. A bucket with a prefix sum on or above the exercise boundary will be exercised. We remark that the lower-bound result for `nUnifCvg` holds for any exercise boundary. It just turns out that the exercise boundary given by `nUnifSplA` does an excellent job.

7.6 Numerical results for American-style Asian options

Experimental results for `nUnifSplA`, `nUnifSplA2`, and `nUnifCvgA` will be presented below. All the experimental results reported here are based on an Athlon Thunderbird 1.33GHz PC with 1GB DRAM. Recall that `nUnifSplA` and `nUnifSplA2` provide upper bounds, whereas `nUnifCvgA` gives lower bounds (the claims will be proved in the next section). Figure 14 plots the pricing errors of `nUnifCvgA:nUnifSplA` and `nUnifCvgA:nUnifSplA2` as functions of n. The pricing errors are measured by `nUnifSplA` $-$ `nUnifCvgA` and `nUnifSplA2` $-$ `nUnifCvgA`. Not surprisingly, both increase with n. But `nUnifSplA2` has a much smaller pricing error than `nUnifSplA`. Because the pricing errors of `nUnifCvgA:nUnifSplA2` are extremely small, the algorithm practically gives the benchmark option value.

To further investigate the performance of `nUnifCvgA:nUnifSplA2` vs. `nUnifCvgA:nUnifSplA`, we perform a comprehensive test in Table 6. Although both algorithms perform well with small pricing errors, `nUnifCvgA:nUnifSplA2` is more competitive. Perhaps the most important lesson to draw from that table is that the pricing error rises as σ increases. This phenomenon is most apparent in the case of `nUnifCvgA:nUnifSplA`. For the algorithm with a tighter range bound, `nUnifCvgA:nUnifSplA2`, the absolute pricing errors never exceed 0.000454. The advantage of the two-phase `nUnifCvgA:nUnifSplA2` again demonstrates the benefit of taking advantage of limited price-sum ranges. Indeed, the data in Table 6 suggest

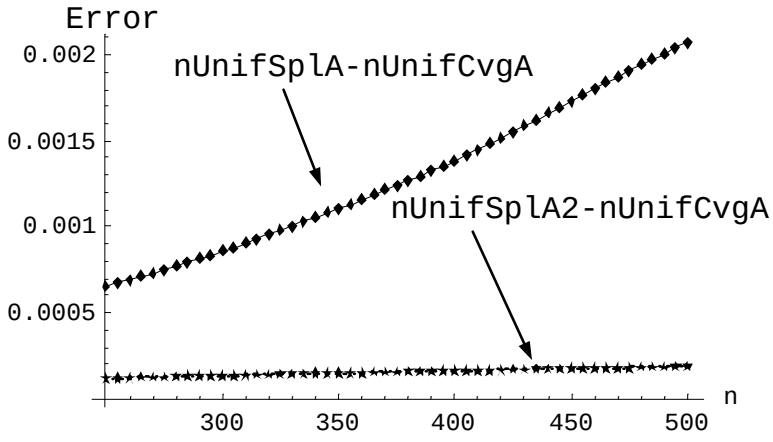

Figure 14: **Pricing errors of `nUnifCvgA:nUnifSplA` and `nUnifCvgA:nUnifSplA2`.** The data are: $S_0 = X = 100$, $\sigma = 30\%$, $r = 20\%$ per annum, and $\tau = 1$. All use an average of $k = 500$ buckets per node.

that a k of only 300 is sufficient in most cases to price the option accurately.

We next investigate the convergence behavior of `nUnifCvgA:nUnifSplA2`. The results tabulated in Table 7 are based on $k = 8n$; hence the running time is $O(n^3)$. To our knowledge, no papers in the literature on American-style Asian options mention numerical results for volatilities larger than 50%. (As mentioned earlier, it is not uncommon to find stocks with a large σ.) Interestingly, Table 7 shows our $O(n^3)$ algorithm produces very tight range bounds for σ as high as 100%. Even where the algorithm runs into some difficulty when $\sigma = 100\%$ and $\tau = 5$ year, the relative errors are less than 0.14% up to $n = 400$.

8 The Range-Bound Proofs

Proofs will now be given to show that the proposed algorithms provide the claimed lower or upper bounds for the benchmark option value. As the lower-bound result for `nUnifCvgA` is independent of how the exercise boundary is determined, it will be given first.

Theorem 8.1 `nUnifCvgA` $\leq$ `benchmark`.

Proof. Define a **terminal bucket** to be a bucket reachable from the root and which is either an early-exercise bucket or an in-the-money bucket at maturity. Only terminal buckets contribute to the option value. The benchmark option value equals the discounted expected payoff of the terminal buckets in the unrecombining tree *when buckets are exercised optimally.* It therefore suffices to prove that `nUnifCvgA` produces an option value that does not exceed the one given by the unrecombining tree with some exercise strategy which is not necessarily optimal.

When a bucket is terminal, all the paths that pass through it terminate there. Let $\wp_b$ be the set of paths terminated at terminal bucket b at time t_b under `nUnifCvgA`. Every path in $\wp_b$ has length t_b. We now use those $\wp_b$ to define an exercise strategy on the unrecombining tree: Each path in $\wp_b$ on the unrecombining tree is terminated at the same node where bucket b resides. In other words, the unrecombining tree uses the same early-exercise strategy as `nUnifCvgA`. This exercise strategy produces an option value A that cannot exceed the benchmark option value because it may not be optimal.

We complete the proof by showing that `nUnifCvgA` generates an option value that cannot exceed A. Fix any terminal bucket b. The contribution of $\wp_b$ to the option value A is

$$e^{-rt_b} \sum_{\rho=(S_0,S_1,\ldots,S_{t_b})\in\wp_b} \text{prob}[\,\rho\,] \times \left\{ \frac{1}{t_b+1} \sum_{i=0}^{t_b} S_i - X \right\}^+ . \tag{15}$$

Recall that each bucket in `nUnifCvgA` stores the average prefix sum of all the paths covered by it. Hence the contribution of $\wp_b$ to `nUnifCvgA`'s option value is

$$e^{-rt_b} \left\{ \sum_{\rho\in\wp_b} \text{prob}[\,\rho\,] \right\} \left\{ \frac{\sum_{(S_0,S_1,\ldots,S_{t_b})\in\wp_b} \frac{1}{t_b+1} \sum_{i=0}^{t_b} S_i}{|\,\wp_b\,|} - X \right\}^+ ,$$

which is smaller than (15) by Jensen's inequality. By summing the contributions over all terminal buckets b, we conclude that `nUnifCvgA` gives a lower bound on A.

□

The next lemma says that the Asian option value is convex with respect to the prefix sum in the ideal tree.

Lemma 8.2 (Convexity lemma) *Let b_1, b_2, and b_3 be buckets at node $N(i,j)$ in the ideal tree with $P_{b_1} > P_{b_2} > P_{b_3}$. If λ satisfies $P_{b_2} = \lambda P_{b_1} + (1-\lambda)P_{b_3}$, then*

$$E^I_{b_2} \leq \lambda E^I_{b_1} + (1-\lambda)E^I_{b_3}.$$

Proof. We define the American-style **bonus Asian option** $A(P,m)$ to facilitate the proof. It pays

$$\left(\frac{P+\Sigma}{m+s+1} - X\right)^+$$

if exercised at time s from its initiation date, where Σ equals the prefix sum from the option's initiation date up to the exercise point. Option $A(P,m)$, if initiated at time m, is identical to the Asian option at time m which was initiated at time 0 and at time m has accumulated a prefix sum of P (which excludes the price at time m). They thus must have the same option value.

Consider three bonus Asian options $A(P_{b_1},i)$, $A(P_{b_2},i)$, and $A(P_{b_3},i)$ initiated at node $N(i,j)$ and maturing at time n. By the above discussions, the value of option $A(P_{b_k},i)$ equals $E^I_{b_k}$, $k=1,2,3$. Assume that $E^I_{b_2} > \lambda E^I_{b_1} + (1-\lambda)E^I_{b_3}$ instead and proceed to show that arbitrage opportunities exist. Assemble a portfolio of long λ unit of $A(P_{b_1},i)$, long $1-\lambda$ unit of $A(P_{b_3},i)$, and short 1 unit of $A(P_{b_2},i)$. The initial income $E^I_{b_2} - \lambda E^I_{b_1} - (1-\lambda)E^I_{b_3}$ is positive. From that point on, whenever $A(P_{b_2},i)$ is exercised, we exercise $A(P_{b_1},i)$ and $A(P_{b_3},i)$, generating zero net cash flow. □

We next establish that `nUnifSplA` is an upper-bound algorithm.

Theorem 8.3 `Benchmark` $\leq$ `nUnifSplA`.

Proof. We prove the theorem by induction. We will prove that $E_b^I \leq E_b^P$, where the practical tree refers to the tree constructed by `nUnifSplA`. The theorem holds at maturity as the option value equals $\{P_b/(n+1) - X\}^+$ at any bucket b. So $E_b^I = E_b^P$. The induction hypothesis is that $E_b^I \leq E_b^P$ for any bucket b in the practical model at time t. We next show that this remains true at time $t-1$ for $t \geq 1$.

Consider any bucket b in the ideal tree at time $t-1$. Let the upward and downward movements from bucket b lead to buckets $u(b)$ and $d(b)$, respectively. But buckets $u(b)$ and $d(b)$ may not exist in the practical tree. Let bucket $u(b)$ be sandwiched between buckets $u(b_1)$ and $u(b_2)$ in the practical tree. With λ satisfying

$$P_{u(b)} = \lambda P_{u(b_1)} + (1-\lambda) P_{u(b_2)}$$

by Eq. (13), we have

$$E_{u(b)}^I \leq \lambda E_{u(b_1)}^I + (1-\lambda) E_{u(b_2)}^I \leq \lambda E_{u(b_1)}^P + (1-\lambda) E_{u(b_2)}^P = E_{u(b)}^P.$$

The first inequality is by Lemma 8.2, and the second inequality is by the induction hypothesis. The equality holds because `nUnifSplA` computes $E_{u(b)}^P$ as the linear interpolation of $E_{u(b_1)}^P$ and $E_{u(b_2)}^P$ with the said weights. By the same argument, $E_{d(b)}^I \leq E_{d(b)}^P$. We next consider three cases.

Case 1: Suppose that b is not an early-exercise bucket in both trees. Then

$$E_b^I = \left[pE_{u(b)}^I + (1-p)E_{d(b)}^I \right] e^{-r} \leq \left[pE_{u(b)}^P + (1-p)E_{d(b)}^P \right] e^{-r} = E_b^P.$$

Case 2: Suppose that b is an early-exercise bucket in the practical tree. Then

$$\left[pE_{u(b)}^I + (1-p)E_{d(b)}^I \right] e^{-r} \leq \left[pE_{u(b)}^P + (1-p)E_{d(b)}^P \right] e^{-r} \leq (P_b/t) - X.$$

So it is also optimal to exercise b in the ideal tree. The option values at b are identical in both trees.

Case 3: Suppose that b is an early-exercise bucket in the ideal tree but not an early-exercise bucket in the practical tree. Then, trivially, $E_b^I = (P_b/t) - X < E_b^P$.

Hence, $E_b^I \leq E_b^P$ in all cases, and the induction step is complete. □

Theorem 8.3 holds for a large class of algorithms, not just `nUnifSplA`. This is because the proof only requires that the option value at a non-existing bucket be linearly interpolated from the option values of its two bracketing buckets. Neither the number of buckets allocated per node nor the way the buckets are distanced in the prefix-sum range matters. It follows that the popular approximation algorithms in Hull (1997) and Hull and White (1993) are upper-bound algorithms.

Corollary 8.4 *The Hull-White paradigm with linear interpolation is an upper-bound algorithm.*

The next result states a general property that the exercise boundary determined by `nUnifSplA` satisfies.

Corollary 8.5 *A bucket with a prefix sum equal to or larger than the exercised boundary determined by* `nUnifSplA` *must be an early-exercise bucket in the ideal tree.*

Proof. By case 2 in the proof of Theorem 8.3, an early-exercise bucket under `nUnifSplA` must also be an early-exercise bucket in the ideal tree. Theorem 7.2 completes the proof. □

The above corollary implies that the exercise boundary determined by `nUnifSplA` cannot be lower than the exercise boundary of the ideal tree. In fact, any upper-bound algorithm can be substituted in phase two to produce a new two-phase upper-bound algorithm. Because this two-phase algorithm takes advantage of the limited price-sum ranges made possible by the exercise boundary estimated in phase one, it should outperform the original one-phase algorithm.

Theorem 8.6 *An upper-bound algorithm that uses the estimated exercise boundary given by* `nUnifSplA` *remains an upper-bound algorithm as long as it works on the same tree.*

Proof. If bucket b lies above the tightened prefix-sum range, then $E_b^P = E_b^I$ because b must be an early-exercise bucket in both the practical and the ideal trees by Corollary 8.5. If bucket b lies within the tightened prefix-sum range, then $E_b^I \leq E_b^P$ because of the algorithm being an upper-bound one and induction. □

For the upper-bound Hull-White algorithms in Corollary 8.4, for instance, the above theorem gives rise to two-phase versions. We finally prove that `nUnifSplA2` is an upper-bound algorithm.

Corollary 8.7 `Benchmark` $\leq$ `nUnifSplA2`.

Proof. It is immediate from Theorem 8.3 and Theorem 8.6. □

9 Conclusions

We have proposed several efficient algorithms to price European- and American-style Asian options. These algorithms bracket the benchmark option value, whose brute-force computation is prohibitive. The range-bound results hold regardless of the average number of buckets per node, k. But k can be varied to strike a desired balance between accuracy and efficiency. The pricing error can be measured by the difference of the upper bound and the lower bound. In the case of European-style options, the pricing error can even be bounded before the algorithms begin. Computer experiments indicate that the range-bound algorithms give numerical results which are so tight that they practically give the correct option value. These results together show that Asian options can generally be priced correctly and efficiently. Surprisingly, our techniques also imply that some of the popular Hull-White approximation algorithms are upper-bound algorithms. A two-phase computational framework is set up as a general paradigm to price Asian options.

Acknowledgements

We thank Profs. Ed Bender, Ira Gessel, and Ming-Yang Kao for discussions.

References

[1] Abate, J., and Whitt, W. (1995). Numerical Inversion of Laplace Transforms of Probability Distributions. *ORSA Journal of Computing*, 7, 36–43.

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

[3] Akcoglu, K., Kao, M.-Y., and Raghavan, S.V. (2001). "Fast Pricing of European Asian Options with Provable Accuracy: Single-Stock and Basket Options." In *Lecture Notes in Computer Science*, 2161. Berlin: Springer-Verlag, 2001, pp. 404–415.

[4] Barraquand, J., and Pudet, T. (1996). Pricing of American Path-Dependent Contingent Claims. *Mathematical Finance*, 6(1), 17–51.

[5] Bender, E.A. (1974). Asymptotic Methods in Enumeration. *SIAM Review*, 16(4), 485–515.

[6] Boyle, P., Broadie, M., and Glasserman, P. (1997). Monte Carlo Methods for Security Pricing. *Journal of Economic Dynamics & Control*, 21, 1267–1321.

[7] Broadie, M., and Glasserman, P. (1996). Estimating Security Price Derivatives Using Simulation. *Management Science*, 42(2), 269–285.

[8] Broadie, M., Glasserman, P., and Kou, S. (1999). Connecting Discrete and Continuous Path-Dependent Options. *Finance and Stochastics*, 3, 55–82.

[9] Chalasani, P., Jha, S., Egriboyun, F., and Varikooty, A. (1999). A Refined Binomial Lattice for Pricing American Asian Options. *Review of Derivatives Research*, 3, 85–105.

[10] Dai, T.-S., and Lyuu, Y.-D. (1999). Efficient Algorithms for Average-Rate Option Pricing. In *Proc. 1999 National Computer Symposium* (NCS'99), Tamkang University, Taiwan, A-359–A-366.

[11] Duffie, D. (1996). *Dynamic Asset Pricing Theory.* 2nd ed. Princeton: Princeton University Press.

[12] Forsyth, P.A., Vetzal, K.R., and Zvan, R. (2001). The Use of Interpolation in Pricing Path-Dependent Options: A Detailed Examination of Convergence for the Case of Asian Options. Working Paper, University of Waterloo.

[13] Fu, M.C., Dilip, D.B., and Wang, T. (1998/1999). Pricing Continuous Asian Options: a Comparison of Monte Carlo and Laplace Transform Inversion Methods. *Journal of Computational Finance*, 49–74.

[14] Geman, H., and Eydeland, A. (1995). Domino Effect. *Risk* 8(4), 65–67.

[15] Geman, H., and Yor, M. (1993). Bessel Processes, Asian Options, and Perpetuities. *Mathematical Finance*, 3, 349–375.

[16] Hochbaum, D.S., ed. (1997). *Approximation Algorithms for NP-Hard Problems.* Boston: PWS.

[17] Hull, J. (1997). *Options, Futures, and Other Derivatives.* 3rd edition. Englewood Cliffs, N.J.: Prentice Hall.

[18] Hull, J., and White, A. (1993). Efficient Procedures for Valuing European and American Path-Dependent Options. *Journal of Derivatives*, 21–31.

[19] Kemna, A.G.Z., and Vorst, A.C.F. (1990). A Pricing Method Based on Average Asset Values. *Journal of Banking & Finance*, 14(1), 113–129.

[20] Klassen, T.R. (2001). Simple, Fast and Flexible Pricing of Asian Options. *Journal of Computational Finance*, 4(3), 89–124.

[21] Levy, E. (1992). Pricing European Average Rate Currency Options. *Journal of International Money and Finance*, 11, 474–491.

[22] Longstaff, F.A., and Schwartz, E.S. (2001). Valuing American Options by Simulation: a Simple Least-Squares Approach. *Review of Financial Studies*, 14(1), 113–147.

[23] Lyuu, Y.-D. (1998). Very Fast Algorithms for Barrier Option Pricing and the Ballot Problem. *Journal of Derivatives*, 5(3), 68–79.

[24] Lyuu, Y.-D. (2002). *Financial Engineering and Computation: Principles, Mathematics, Algorithms*. Cambridge, U.K.: Cambridge University Press.

[25] Milevsky, M.A., and Posner, S.E. (1998). Asian Options, the Sum of Lognormals, and the Reciprocal Gamma Distribution. *Journal of Financial and Quantitative Analysis*, 33(3), 409–422.

[26] Ritchken, P., Sankarasubramanian, L., and Vijh, A.M. (1993). The Valuation of Path Dependent Contracts on the Average. *Management Science*, 39(10), 1202–1213.

[27] Rogers, L.C.G., and Shi, Z. (1995). The Value of an Asian Option. *Journal of Applied Probability*, 32(4), 1077–1088.

[28] Shaw, W.T. (1998). *Modeling Financial Derivatives with* Mathematica. Cambridge, U.K.: Cambridge University Press.

[29] Turnbull, S.M., and Wakeman, L.M. (1991). A Quick Algorithm for Pricing European Average Options. *Financial and Quantitative Analysis*, 26(3), 377–389.

[30] Zvan, R., Vetzal, K., and Forsyth, P. (1999). Discrete Asian Barrier Options." *Journal of Computational Finance*, 3(1), 41–67.

A Proof of the Contraction Lemma

Lemma A.1 *Under the assumptions given in Lemma 7.1,*

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} \geq E^I_{b_1} - E^I_{b_2}. \tag{16}$$

Proof. Assume that bucket b_i moves up to bucket $u(b_i)$ and down to bucket $d(b_i)$, $i = 1, 2$, in the ideal tree. The lemma will be proved by induction on the level of the tree. The base case involves the buckets allocated at maturity. There $E^I_{b_i} = \max(P_{b_i}/(m+1), X) - X$. Thus

$$E^I_{b_1} - E^I_{b_2} = \max\left(\frac{P_{b_1}}{m+1}, X\right) - \max\left(\frac{P_{b_2}}{m+1}, X\right) \leq \frac{P_{b_1} - P_{b_2}}{m+1}$$

because $P_{b_1} > P_{b_2}$. The induction step is divided into four cases.

Case 1: Neither b_1 nor b_2 is exercised immediately. Then

$$\begin{aligned}
\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} &\geq \frac{P_{b_1} - P_{b_2}}{m+2} \\
&= p\,\frac{P_{u(b_1)} - P_{u(b_2)}}{m+2} + (1-p)\,\frac{P_{d(b_1)} - P_{d(b_2)}}{m+2} \\
&\geq \left\{ p[\, E^I_{u(b_1)} - E^I_{u(b_2)}\,] + (1-p)[\, E^I_{d(b_1)} - E^I_{d(b_2)}\,] \right\} e^{-r} \\
&= \left\{ \left[\, pE^I_{u(b_1)} + (1-p)E^I_{d(b_1)}\,\right] - \left[\, pE^I_{u(b_2)} + (1-p)E^I_{d(b_2)}\,\right] \right\} e^{-r} \\
&= E^I_{b_1} - E^I_{b_2},
\end{aligned}$$

where the second inequality is by the induction hypothesis.

Case 2: b_1 is exercised immediately, but b_2 is not. Then

$$\begin{aligned} E^I_{b_1} &= \frac{P_{b_1}}{m+1} - X, \\ E^I_{b_2} &> \frac{P_{b_2}}{m+1} - X. \end{aligned}$$

Subtract the inequality from the equality to obtain inequality (16).

Case 3: Both b_1 and b_2 are exercised immediately. In this case, $E^I_{b_i} = P_{b_i}/(m+1) - X$ for $i = 1, 2$, and inequality (16) holds as an equality.

Case 4: b_1 is not exercised, but b_2 is. We will show that this is impossible. Assume otherwise. Then

$$\frac{P_{b_1}}{m+1} - X < \left[\, pE^I_{u(b_1)} + (1-p)E^I_{d(b_1)} \,\right] e^{-r}, \tag{17}$$

$$\frac{P_{b_2}}{m+1} - X \geq \left[\, pE^I_{u(b_2)} + (1-p)E^I_{d(b_2)} \,\right] e^{-r}. \tag{18}$$

Subtracting inequality (18) from inequality (17) results in

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} < \left\{ p[\, E^I_{u(b_1)} - E^I_{u(b_2)} \,] + (1-p)[\, E^I_{d(b_1)} - E^I_{d(b_2)} \,] \right\} e^{-r}. \tag{19}$$

But

$$\begin{aligned} \frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} &\geq \frac{P_{b_1} - P_{b_2}}{m+2} \\ &= p\,\frac{P_{u(b_1)} - P_{u(b_2)}}{m+2} + (1-p)\,\frac{P_{d(b_1)} - P_{d(b_2)}}{m+2} \\ &\geq \left\{ p[\, E^I_{u(b_1)} - E^I_{u(b_2)} \,] + (1-p)[\, E^I_{d(b_1)} - E^I_{d(b_2)} \,] \right\} e^{-r}, \end{aligned}$$

contradicting inequality (19). □

	Monte Carlo		nUnifCvg		UnifAvg		MR	
n	Lower	Upper	Value	Time	Value	Time	Value	Time
86	0.324	0.328	0.322*	78	0.322*	118	0.324*	13
97	0.325	0.329	0.323*	100	0.323*	154	0.325	23
108	0.324	0.328	0.324	124	0.323*	196	0.327	39
119	0.326	0.330	0.324*	152	0.323*	245	0.328	61
130	0.324	0.328	0.324	181	0.325	298	0.327	96
141	0.325	0.329	0.325	213	0.324*	358	0.326	145
152	0.326	0.330	0.325*	249	0.323*	422	0.325*	210
163	0.324	0.328	0.325	286	0.326	492	0.325	302
174	0.326	0.329	0.325*	327	0.325*	562		
185	0.324	0.328	0.326	370	0.326	634		
196	0.327	0.330	0.326*	414	0.326*	711		
207	0.326	0.330	0.326	463	0.326	792		
218	0.326	0.329	0.326	513	0.326	879		
229	0.326	0.329	0.326	564	0.326	972		
240	0.327	0.330	0.326*	618	0.326*	1066		
251	0.326	0.330	0.326	675	0.326	1166		
262	0.326	0.329	0.326	738	0.326	1269		
273	0.326	0.329	0.326	799	0.327	1379		
284	0.326	0.329	0.327	865	0.327	1493		

Table 1: **Monte Carlo simulation, `nUnifCvg`, `UnifAvg`, and MR.** Monte Carlo simulations are based on 2,000,000 trials. The "Lower" and "Upper" columns represent the 95% confidence interval obtained from Monte Carlo simulations. Both `nUnifCvg` and `UnifAvg` set $k = 50,000$. At $n \geq 174$, MR's demand for computer memory is beyond what the system can offer. The computation times are measured in seconds. Asterisks mark those answers which are out of the 95% confidence interval. The data are: $S_0 = 50$, $X = 60$, $r = 10\%$ per annum, $\sigma = 30\%$, and $\tau = 0.5$.

Maturity (years)	Algorithm	Strike price X				
		40	45	50	55	60
0.5	HW	10.755	6.363	3.012	1.108	0.317
	MC	10.759	6.359	2.998	1.112	0.324
		(0.003)	(0.005)	(0.007)	(0.005)	(0.003)
	MR	10.754	6.356	2.997	1.104	0.317
	L	10.765	6.386	3.024	1.105	0.313
	`nUnifCvg`	10.754	6.361	3.007	1.104	0.315
1.0	HW	11.545	7.616	4.522	2.420	1.176
	MC	11.544	7.606	4.515	2.401	1.185
		(0.006)	(0.008)	(0.010)	(0.009)	(0.007)
	MR	11.547	7.616	4.517	2.412	1.170
	L	11.576	7.662	4.557	2.431	1.172
	`nUnifCvg`	11.544	7.613	4.519	2.417	1.174
1.5	HW	12.285	8.670	5.743	3.585	2.124
	MC	12.289	8.671	5.734	3.577	2.135
		(0.008)	(0.010)	(0.012)	(0.012)	(0.010)
	MR	12.284	8.674	5.750	3.585	2.118
	L	12.337	8.738	5.801	3.619	2.133
	`nUnifCvg`	12.283	8.668	5.740	3.583	2.122
2.0	HW	12.953	9.582	6.792	4.633	3.057
	MC	12.943	9.569	6.786	4.639	3.055
		(0.010)	(0.013)	(0.014)	(0.015)	(0.013)
	MR	12.944	9.577	6.786	4.625	3.045
	L	13.024	9.671	6.874	4.691	3.087
	`nUnifCvg`	12.953	9.580	6.790	4.631	3.055

Table 2: **Comparing `nUnifCvg` against various algorithms.** "HW" denotes the algorithm in Hull White (1993) based on $n = 40$ and $h = 0.005$. "MC" denotes Monte Carlo simulation based on $n = 40$ and 100,000 trials (the sample standard deviations are in parentheses). MR uses $n = 30$. "L" denotes the analytical approach described in Levy (1992). Algorithm `nUnifCvg` sets $k = \lfloor 50,000/7 \rfloor$. The data are: $S_0 = 50$, $r = 10\%$ per annum, and $\sigma = 30\%$. Some of the data are taken from Dai and Lyuu (1999).

r	σ	T	S_0	GE	Shaw	Euler	PW	TW	MC10	MC100	S.D.	`Cvg`
0.050	0.50	1	1.9	.195	.193	.194	.194	.195	.192	.196	.004	.193
0.050	0.50	1	2.0	.248	.246	.247	.247	.250	.245	.249	.004	.246
0.050	0.50	1	2.1	.308	.306	.307	.307	.311	.305	.309	.005	.306
0.020	0.10	1	2.0	.058	.520	.056	.0624	.0568	.0559	.0565	.0008	.0559
0.180	0.30	1	2.0	.227	.217	.219	.219	.220	.219	.220	.003	.218
0.125	0.25	2	2.0	.172	.172	.172	.172	.173	.173	.172	.003	.172
0.050	0.50	2	2.0	.351	.350	.352	.352	.359	.351	.348	.007	.349

Table 3: **Comparing `nUnifCvg` against the analytical approaches.** The strike price X is 2.0, and `nUnifCvg` uses $n = 40$. The alternative methods are: Geman-Eydeland (GE), Shaw, Euler, Post-Widder (PW), and Turnbull-Wakeman (TW). MC10 uses 10 periods per day, whereas MC100 uses 100. Both are based on 100,000 trials. S.D. stands for sample standard deviation. `Cvg` stands for `nUnifCvg`. Some of the data are from Fu et al. (1998/1999). As before, `nUnifCvg` sets $k = \lfloor 50,000/7 \rfloor$.

σ	τ	n	nUnifCvg	nUnifSpl	nUnifSpl − nUnifCvg
10%	0.25	50	1.800870	2.175705	0.374835
		100	1.839875	1.932832	0.092957
		200	1.847834	1.870414	0.022580
		400	1.850455	1.855982	0.005527
50%	1.00	50	13.179130	13.210789	0.031659
		100	13.193776	13.202119	0.008343
		200	13.200312	13.202382	0.002070
		400	13.203293	13.203823	0.000530
50%	5.00	50	28.386460	28.395814	0.009354
		100	28.395902	28.398327	0.002425
		200	28.400568	28.401189	0.000620
		400	28.402879	28.403038	0.000159
100%	1.00	50	23.407397	23.422099	0.014702
		100	23.434382	23.438502	0.004120
		200	23.447782	23.448835	0.001053
		400	23.454417	23.454680	0.000263
100%	5.00	50	42.769952	42.774652	0.004700
		100	42.823800	42.825049	0.001249
		200	42.851203	42.851529	0.000326
		400	42.865018	42.865102	0.000084

Table 4: **Convergence of `nUnifCvg:nUnifSpl`.** Both algorithms use $k = n$. The data are: $S_0 = X = 100$ and $r = 10\%$ per annum. Algorithm `nUnifCvg:nUnifSpl` takes fewer than 35 seconds for the $n = 400$ case.

σ	τ	n	nUnifCvg	nUnifSpl	nUnifSpl − nUnifCvg
10%	0.25	50	1.848515	1.848533	0.000018
		100	1.850035	1.850044	0.000009
		200	1.850809	1.850813	0.000004
		400	1.851199	1.851201	0.000002
50%	1.00	50	13.185396	13.185639	0.000243
		100	13.195530	13.195701	0.000171
		200	13.200738	13.200898	0.000160
		400	13.203354	13.203612	0.000258
50%	5.00	50	28.387935	28.389159	0.001224
		100	28.395811	28.398385	0.002574
		200	28.397866	28.413588	0.015722
		400	28.370135	28.920558	0.550423
100%	1.00	50	23.410075	23.411095	0.001020
		100	23.434776	23.436654	0.001878
		200	23.446473	23.453710	0.007237
		400	23.442168	23.561833	0.119665
100%	5.00	50	42.747360	42.835029	0.087669
		100	42.135964	44.997394	2.861430
		200	38.257653	69.258656	31.001003
		400	38.224317	184.271619	146.047302

Table 5: **Convergence of the full-range** `nUnifCvg:nUnifSpl`**.** The setup is identical to Table 4 except that $k = 8n$. Note that more buckets are allocated than in Table 4

σ	X	r	nUnifCvgA	nUnifSplA2	nUnifSplA	nUnifSplA −nUnifCvgA	nUnifSplA2 −nUnifCvgA
0.1	95	0.05	8.088364	8.088422	8.088522	0.000158	0.000058
0.1	95	0.15	11.267781	11.267846	11.267954	0.000173	0.000065
0.1	105	0.05	1.344226	1.344292	1.344403	0.000177	0.000066
0.1	105	0.15	3.623832	3.623887	3.623980	0.000148	0.000055
0.3	95	0.05	12.358376	12.358517	12.359182	0.000806	0.000141
0.3	95	0.15	14.428086	14.428229	14.428934	0.000848	0.000143
0.3	105	0.05	6.311839	6.311984	6.312741	0.000902	0.000145
0.3	105	0.15	8.208416	8.208553	8.209280	0.000864	0.000137
0.5	95	0.05	17.341037	17.341237	17.344196	0.003159	0.000200
0.5	95	0.15	18.922948	18.923150	18.926233	0.003285	0.000202
0.5	105	0.05	11.623434	11.623636	11.627077	0.003643	0.000202
0.5	105	0.15	13.214077	13.214273	13.217725	0.003648	0.000196
0.7	95	0.05	22.536275	22.536540	22.552333	0.016058	0.000265
0.7	95	0.15	23.775811	23.776080	23.792101	0.016290	0.000269
0.7	105	0.05	17.065704	17.065979	17.084335	0.018631	0.000275
0.7	105	0.15	18.382506	18.382779	18.401274	0.018768	0.000273
0.9	95	0.05	27.841546	27.841955	27.952798	0.111252	0.000409
0.9	95	0.15	28.797383	28.797804	28.908081	0.110698	0.000421
0.9	105	0.05	22.587415	22.587869	22.719667	0.132252	0.000454
0.9	105	0.15	23.650191	23.650639	23.779582	0.129391	0.000448

Table 6: **Comprehensive tests for `nUnifSplA`, `nUnifSplA2`, and `nUnifCvgA`.** The data are: $S_0 = 100$, $n = 300$, $k = 500$ and $\tau = 1$. Algorithms `nUnifCvgA`, `nUnifSplA2`, and `nUnifSplA` take an average of 4.60 seconds in generating their respective results.

σ	τ	n	nUnifCvgA	nUnifSplA2	nUnifSplA2 − nUnifCvgA
10%	0.25	50	1.937256	1.937271	0.000015
		100	1.947621	1.947626	0.000005
		200	1.953399	1.953401	0.000002
		400	1.956484	1.956485	0.000001
50%	1.00	50	14.763087	14.763184	0.000097
		100	14.912143	14.912180	0.000037
		200	14.996588	14.996602	0.000014
		400	15.042595	15.042600	0.000005
50%	5.00	50	33.444456	33.444608	0.000152
		100	33.837743	33.837809	0.000066
		200	34.062623	34.062648	0.000025
		400	34.184574	34.184584	0.000010
100%	1.00	50	27.595989	27.596134	0.000145
		100	27.963737	27.963799	0.000062
		200	28.175147	28.175170	0.000023
		400	28.290796	28.290804	0.000008
100%	5.00	50	58.262845	58.262854	0.000009
		100	59.448244	59.448330	0.000086
		200	60.130631	60.130817	0.000186
		400	60.501092	60.582166	0.081074

Table 7: **Convergence of nUnifCvgA:nUnifSplA2.** Both algorithms use $k = 8n$. The data are: $S_0 = X = 100$ and $r = 10\%$ per annum.