Extremely Accurate and Efficient Algorithms for European-Style 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 European-style 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. 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 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. 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.

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

*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.

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. No simple closed-form solutions exist yet. 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 are suggested in Abate and Whitt (1995). Monte Carlo simulation is another popular methodology in which variance reduction techniques are almost always applied; see Boyle *et al.* (1997), Broadie and Glasserman (1996), Broadie *et al.* (1999), and Kemna and Vorst (1990).

Tree methods and their related discretized partial-differential-equation method are another common approach. 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. Many approaches have been proposed to tackle this **combinatorial explosion**. 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 or even extrapolation in backward induction; see Hull and White (1993), Klassen (2001), and Zvan *et al.* (1999). This paradigm is efficient, with a running time of $O(kn^2)$. But it lacks convergence guarantees. In fact, this paper will prove rigorously that some versions of Hull-White algorithms are upper-bound algorithms. Rogers and Shi (1995) propose approximations for European-style Asian options based on analytical insights.

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 at all. The algorithm is also guaranteed to converge to the true value. The worst-case running time seems to be proportional to $O(2^{\sqrt{n}})$. 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 last paradigm seeks approximation algorithms that produce 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. 1). Chalasani, *et al.* (1999) propose $O(n^4)$-time range-bound algorithms for

Asian options. The $O(kn^2)$-time algorithm of Aingworth, *et al.* (2000) guarantees a theoretical error bound of $O(Xn/k)$ in the case of European-style Asian options. The parameter k can be varied for any desired trade-off between time and accuracy. This bound is also interesting in that it is known before the algorithm starts.

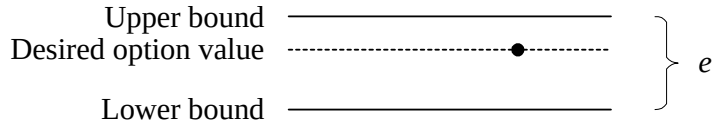

Figure 1: **Range bound and uncertainty e about the desired but unknown option value.**

This paper follows the last paradigm. Our first algorithm has an error bound of $O(X\sqrt{n}/k)$ for European-style Asian options. This is an improvement over the results of Aingworth, *et al.* 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 $\sqrt{n}$. So the algorithms' practical running time is at most $O(n^{2.5})$. Besides, the theoretical proofs of range bounds, which are of some independent interest, yield an unexpected corollary that says the popular approximation algorithms in Hull (1997) and Hull and White (1993) are in fact upper-bound algorithms.

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. Section 7 concludes the paper.

2 Basic Terms

The standard CRR binomial model will be used to model the stock price dynamics. Let S_i denote the stock price at time i. The binomial model says that S_{i+1} equals S_iu with probability p and S_id with probability $1-p$, where $d < u$ and $ud = 1$. 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 $S_0u^{i-j}d^j$ with probability $\binom{i}{j}p^{i-j}(1-p)^j$. A 2-period binomial model is illustrated in Fig. 2(a). We shall assume that the stock does not pay dividends for ease of presentation.

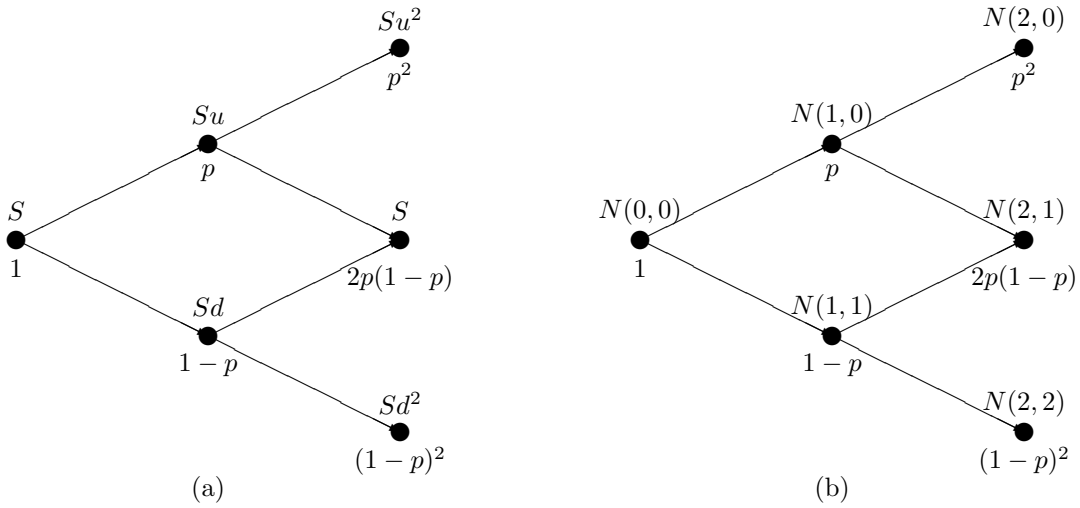

Figure 2: **The binomial model and the binomial tree.** (a) A 2-period binomial model and (b) a 2-period binomial tree. The probability of reaching each node is listed under the node.

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 $S_0u^{i-j}d^j$. 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 with probability $\binom{i}{j}p^{i-j}(1-p)^j$. See Fig. 2(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. Both $d \le e^r \le u$ and $0 \le p \le 1$ 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; 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, we will use $\texttt{B} - \texttt{A}$ to measure the pricing error of the range-bound algorithm `A:B`.

3 Preliminaries

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

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

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; see Aingworth *et al.* (2000).

Lemma 3.1 *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* $r > 0$.

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.

4 The Basic Range-Bound Algorithm and Its Pricing Error

4.1 Description of the Algorithms

We start by segmenting the range $[\,0, (n+1)X\,]$ 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. 3 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.

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 our 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 $s \ge (n+1)X$ 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

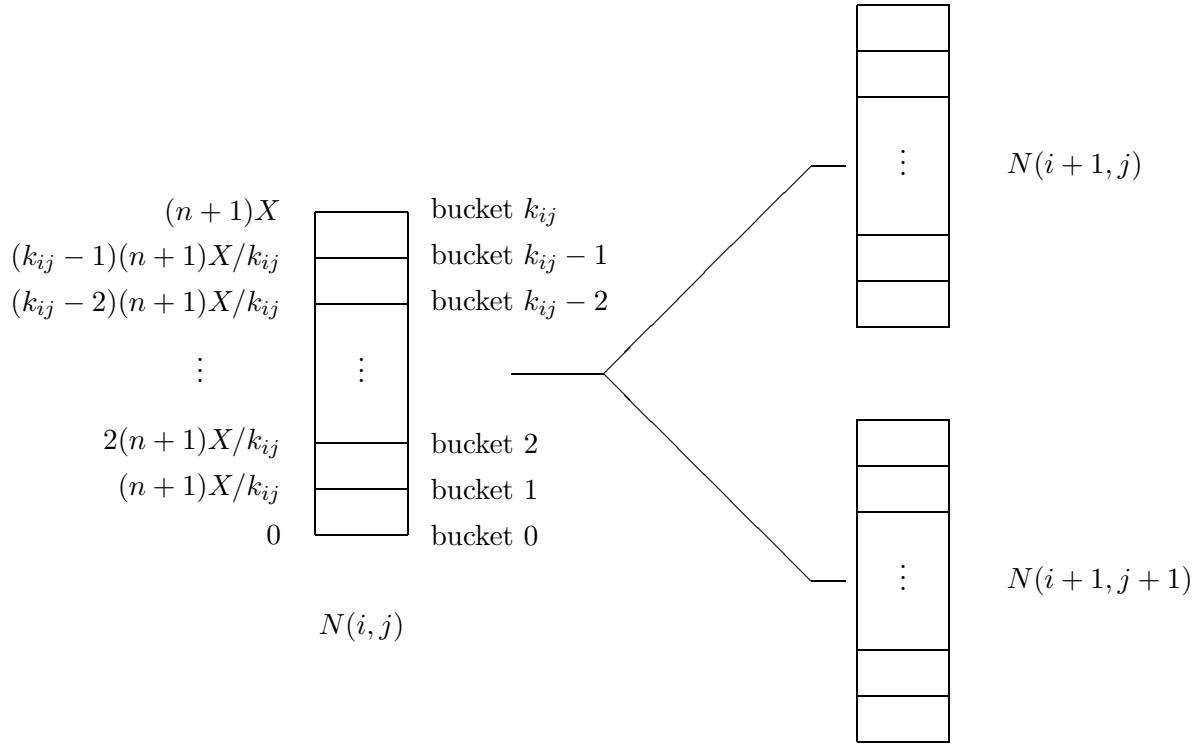

Figure 3: **Bucketing.** Each node $N(i,j)$ has $k_{ij}+1$ buckets, starting from prefix sum 0 and ending at prefix sum $(n+1)X$ with increments of $(n+1)X/k_{ij}$.

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:

$$b\left(i+1, j, \left\lfloor \frac{s(i,j,\ell) + S_0 u^{i-j+1} d^j}{(n+1)X/k_{i+1,j}} \right\rfloor\right). \tag{4}$$

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

$$b\left(i+1, j+1, \left\lfloor \frac{s(i,j,\ell) + S_0 u^{i-j} d^{j+1}}{(n+1)X/k_{i+1,j+1}} \right\rfloor\right). \tag{5}$$

See Fig. 4 for illustration. Bucket $b(i,j,\ell)$ therefore **covers** the prefix sums s in range

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

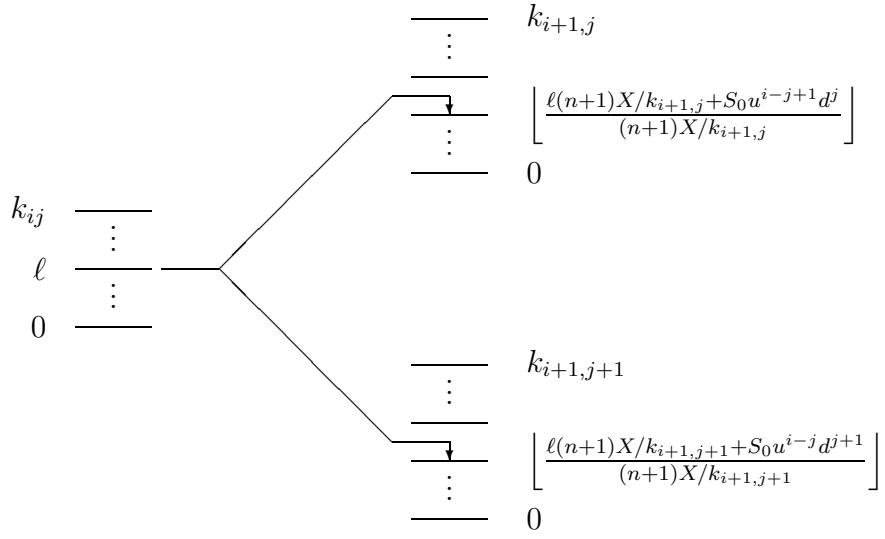

Figure 4: **Rounding down the partial sums.** Each of the $k_{ij}+1$ buckets at $N(i,j)$ moves up to the bucket of node $N(i+1,j)$ and down to the bucket of node $N(i+1,j+1)$ as shown. In the rounding-up version, the floor operations are replaced with the ceiling operations.

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.

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:

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

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.

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`.

4.2 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

$$\frac{1}{n+1}\sum_{i=0}^{n}\sum_{j=0}^{i}\binom{i}{j}p^{i-j}(1-p)^{j}|D_{ij}|. \tag{7}$$

Because $|D_{ij}|$ are not identically weighted in formula (7), k_{ij} should not be a constant.

Set

$$\texttt{TIME} = \sum_{0\le j\le i\le n} k_{ij}, \tag{8}$$

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 $b(i,j;p) \equiv \binom{i}{j}p^{i-j}(1-p)^{j}$ and $|D_{ij}|$ set to $(n+1)X/k_{ij}$, pricing error (7) is bounded above by

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

If the budget for TIME is fixed, then ERROR is minimized by setting

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

The pricing error's upper bound,

$$\frac{X}{\texttt{TIME}}\times\left[\sum_{0\le j\le i\le n}\sqrt{b(i,j;p)}\right]^{2}, \tag{9}$$

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

The algorithm can now be completely specified with

$$k_{ij} = \left\lceil \texttt{TIME}\times\frac{\sqrt{b(i,j;p)}}{\sum_{0\le j\le i\le n}\sqrt{b(i,j;p)}}\right\rceil. \tag{10}$$

Equation (8) 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 (9) of `nUnifDown`. The same proof works for `nUnifUp`. Recall that $p = 1/2$. Bender (1974) shows that

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

Substitute the above into formula (9) to yield

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

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

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

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 (10).*

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. Rounding also offers 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

$$s = \lambda \cdot s(i+1, j, \texttt{RoundUp}) + (1-\lambda) \cdot s(i+1, j, \texttt{RoundDown}) \tag{11}$$

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. 5 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).

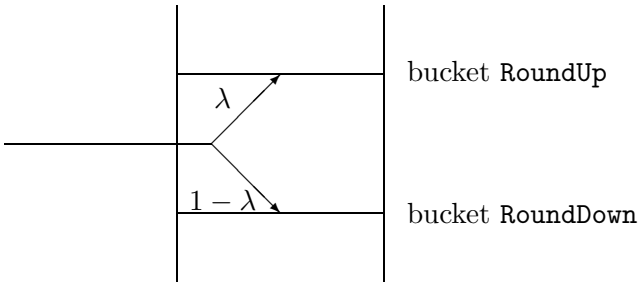

Figure 5: **Splitting a path.** Bucket `RoundUp` receives λ of the probabilistic weight. Bucket `RoundDown` receives $1-\lambda$ of the probabilistic weight.

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.

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 simply adds bucketing to the above idea. This algorithm will be called `nUnifCvg` (for convergence). We illustrate the idea in Fig. 6 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. 6. 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`.

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, $\{\sum_{i=0}^{n} S_i/(n+1) - X\}^+$,

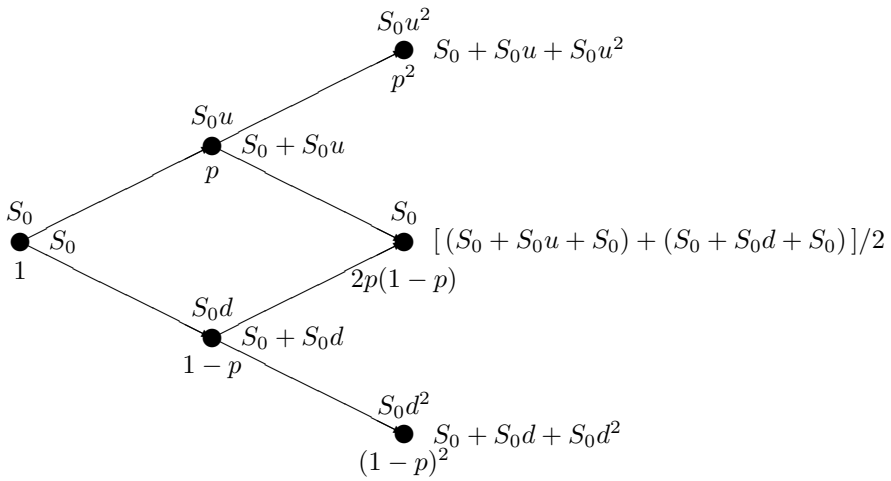

Figure 6: **Price-sum averaging with `nUnifCvg`.** The average prefix sum is listed to the right of the node. The probability for the average prefix sum is listed under the node.

is dominated by `nUnifSpl`'s approximation, $E[\,(A_x - X)^+\,]$. As this inequality holds for every path, the desired result follows. □

The popular approximation algorithms in Hull (1997) and Hull and White (1993) are special cases of our `Spl` algorithm. It follows that these algorithms compute upper bounds.

Corollary 5.2 *The approximation algorithms in Hull (1997) and Hull and White (1993) are upper-bound algorithms for European-style Asian options.*

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 (10) 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. 7 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.

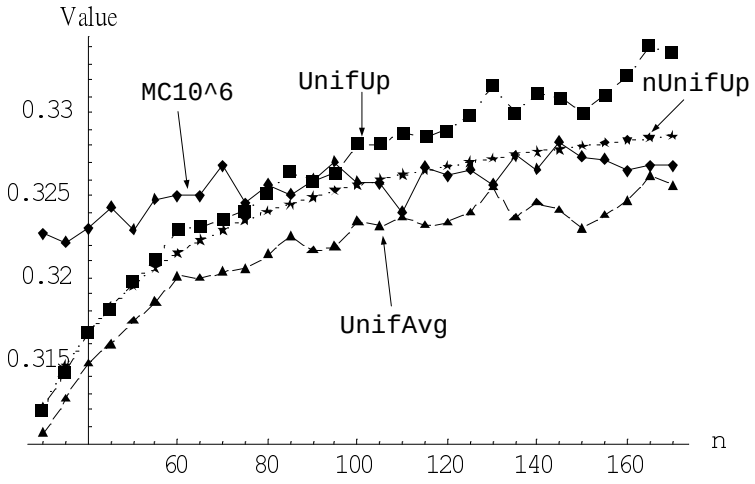

Figure 7: **Comparing `UnifUp`, `UnifAvg`, and `nUnifUp`.** MC10^6 denotes Monte Carlo simulation based on 1,000,000 trials. Both `UnifAvg` and `UnifUp` allocate $k = 50,000$ buckets at each node, and `nUnifUp` allocates an average of $k = 50,000$ buckets per node. The benchmark option values must lie below `nUnifUp`. The data are: $S_0 = 50$, $X = 60$, annual volatility $\sigma = 30\%$, $r = 10\%$ per annum, and maturity $\tau = 0.5$ (year).

Having demonstrated the advantage of using our nonuniform bucketing scheme, we next compare the nonuniform lower-bound algorithm `nUnifCvg` against `UnifAvg` and the multiresolution algorithm in Dai and Lyuu (1999). Again, both `nUnifCvg` and `UnifAvg` use the same total number of buckets. The results are tabulated in Fig. 8. Observe that `nUnifCvg` is superior to `UnifAvg` in terms of accuracy and speed despite that it takes slightly less time than `UnifAvg`. The multiresolution 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 for large n. We then add the algorithm in Hull and White (1993) and Levy's analytical approach in Levy (1992) to the comparisons in Fig. 9. Observe that `nUnifCvg` is competitive in all the scenarios whether the option is in the money or out of the money. Many proposed algorithms have been found to fail in extreme cases. When the cases of Fu *et al.* (1998/1999) are tested in Fig. 10, `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 use slightly less time than `UnifUp`. The results are illustrated in Fig. 11. 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

	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		

Figure 8: **Monte Carlo simulation, `nUnifCvg`, `UnifAvg`, and multiresolution.** 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 $k = 50,000$. "MR" denotes the multiresolution algorithm in Dai and Lyuu (1999). At $n \geq 174$, MR's demand for computer memory is beyond what the system can offer. The computational 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 $\tau = 0.5$.

`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 both employ the same nonuniform bucketing scheme, this comparison extracts the benefits which are due 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. 12 that `nUnifCvg:nUnifSpl` produces much tighter range bounds, hence smaller errors, than `nUnifDown:nUnifUp`.

Finally, we are interested in knowing how k—the average number of buckets per node—impacts the accuracy of the algorithms. Figure 13 shows the effects of k on

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

Figure 9: **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" is the multiresolution algorithm with $n = 30$. "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).

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

7 Conclusions

We have proposed several efficient algorithms to price European-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

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

Figure 10: **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 $k = \lfloor 50,000/7 \rfloor$.

of buckets per node, k. But k can be varied to strike any desired balance between accuracy and efficiency. The pricing error can be measured by the difference of the upper bound and the lower bound. It 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 be priced correctly and efficiently. Surprisingly, our techniques also imply that some of the Hull-White approximation algorithms are upper-bound algorithms.

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.*

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

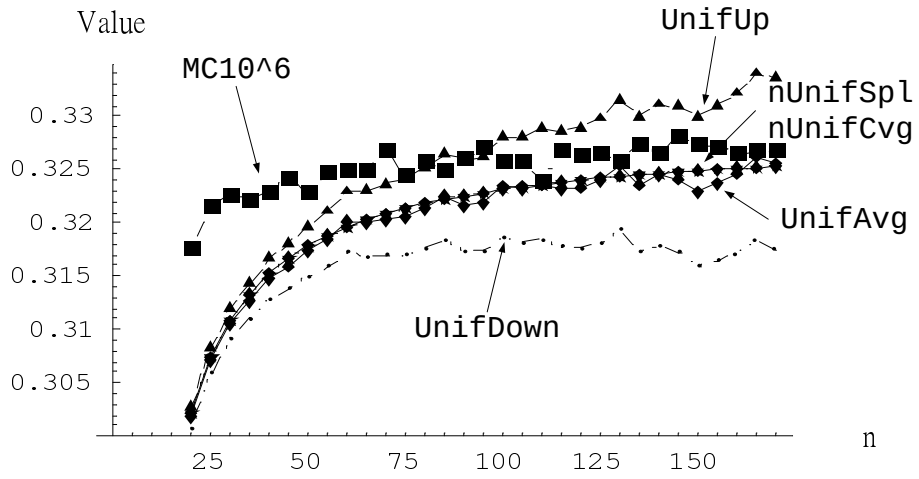

Figure 11: **Comparing `nUnifCvg` and `nUnifSpl` against `UnifUp`, `UnifDown`, and `UnifAvg`.** `UnifUp`, `UnifDown`, and `UnifAvg` each set $k = 50,000$, whereas `nUnifSpl` and `nUnifCvg` set $k = \lfloor 50,000/7 \rfloor$. These choices make `nUnifCvg` and `nUnifSpl` take slightly less time than `UnifUp`. The plots of `nUnifCvg` and `nUnifSpl` essentially coincide. The data are: $S_0 = 50$, $X = 60$, $\sigma = 30\%$, $r = 10\%$ per annum, and $\tau = 0.5$.

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

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

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

[7] 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.

[8] 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.

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

[10] 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.

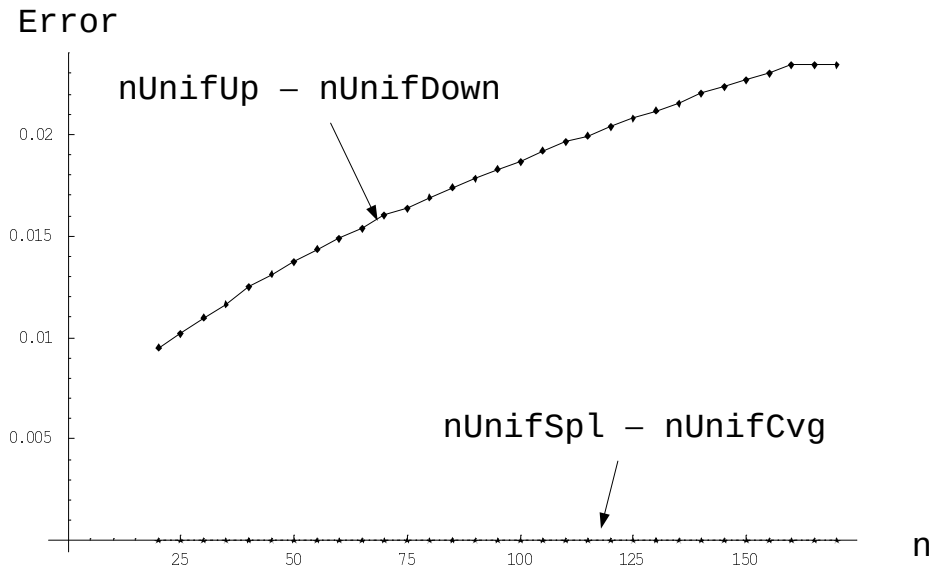

Figure 12: **Pricing errors of `nUnifCvg:nUnifSpl` and `nUnifDown:nUnifUp`.** All four algoritms set $k = \lfloor 50,000/7 \rfloor$. The other parameters are identical to the ones used in Fig. 11.

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

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

[13] Hull, J. (1997). *Options, Futures, and Other Derivatives.* 3rd edition. Prentice Hall, New Jersey.

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

[15] 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.

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

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

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

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

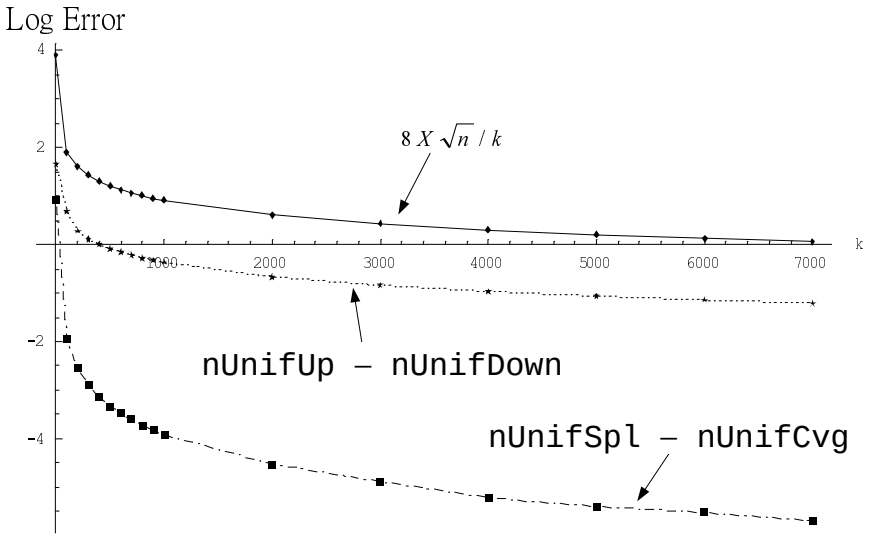

Figure 13: **The number of buckets and accuracy.** This plot shows the log-plot (base 10) of the various error bounds vs. k. We use $n = 285$ here. The other parameters are identical to the ones used in Fig. 11.

[20] 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.

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

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

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

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