Accurate and Efficient Lattice Algorithms for American-Style Asian Options with Range Bounds

Abstract

Asian options are popular path-dependent options and it has been a long-standing problem to price them efficiently and accurately. Since there is no known exact pricing formula for Asian options, numerical pricing formulas like lattice models must be employed. A lattice divides a certain time interval into n time steps and the pricing results generated by the lattice (called desired option values for convenience) converge to the theoretical option value as $n \to \infty$. Since a brute-force lattice pricing algorithm runs in subexponential time in n, some heuristics, like interpolation method, are used to strike the balance between the efficiency and the accuracy. But the pricing results might not converge due to the accumulation of interpolation errors. For pricing European-style Asian options, the evaluation on the major part of the lattice can be done by a simple formula, and the interpolation method is only required on the minor part of the lattice. Thus polynomial time algorithms with convergence guarantee for European-style Asian options can be derived. However, such a simple formula does not exist for American-style Asian options. This paper suggests an efficient range-bound algorithm that bracket the desired option value. By taking advantages of the early exercise property of American-style options, We show that part of the lattice can be evaluated by a simple formula. The interpolation method is required on the remaining part of the lattice and the upper and the lower bounds option values produced by the proposed algorithm are essentially numerically identical. Thus the theoretical option value is said to be obtained practically when the range bound algorithm runs on a lattice with large number of time steps.

Running title: Range-Bound Algorithm for American Asian Options

Keywords: Asian option, option pricing, lattice, path-dependent derivative, range-bound algorithm,

1 Introduction

Options are financial derivatives that give their buyers the right but not obligation to buy or sell the underlying asset for a contractual price (called the exercise price) at a certain date (called the maturity date). The underlying asset is assumed to be stock for convenience. Take a European-style call option for example. An option holder will exercise the option (i.e. the right to buy a stock) at the maturity date if the stock price S exceeds the exercise price X. Thus he can realize a payoff of $S - X$. On the other hand, if S is below X at the maturity date, he can simply junk the call option. To meet different requirements of financial markets, various options are developed. For example, an American-style option allows an option holder to exercise the option prior to maturity date. A path-dependent option is the option whose payoff depends nontrivially on the stock price history. Asian options are path-dependent options since their payoffs depend on the average stock prices. They play important rules in financial markets because their prices are less subject to price manipulation. However, it has been a long-standing problem to price them efficiently and accurately [1].

There is no exact analytical pricing formula for Asian options. Approximate closed-form solutions are suggested in [2, 3, 4, 5, 6, 7, 8]. But most approximate analytical formulas lack the accuracy guarantees and even produce large pricing errors under certain cases [9, 10, 11]. Some papers use Monte Carlo and related quasi-Monte Carlo methods [12, 13, 14, 15]. But they produce probabilistic results and are usually inefficient. Besides, the aforementioned approaches are hard to handle the American-style Asian options.

Lattice methods and the related discretized partial-differential-equation approaches are more general than the aforementioned schemes because they can handle American-style options more easily. A lattice divides the time horizon of the option into n discrete time steps and specifies the stock prices discretely at each time step. Take a 2-time-step CRR lattice [16] in Fig. 1 as an example. (The details of the CRR lattice will be described later.) The time interval is evenly divided into 2 time steps. The stock price at time step 0 is S_0 (at node $N(0,0)$). The stock price can either move up to S_0u (at node $N(1,0)$) with probability p or down to S_0d (at node $N(1,1)$) with probability $1-p$ at the first time step. Similarly, each stock price can either move up or move down in subsequent time steps. The pricing results generated by the lattice (called the desired option values) converge to the theoretical option value as $n \to \infty$ [17].

The difficulty to price Asian options with the lattice lies in its exponential nature: Consider the binomial random walk of the stock price illustrated in Fig. 1. After n time steps, the history contains 2^n possible price paths, each with its own average stock price. As the payoff of the Asian option depends on the average stock price, there are 2^n possible payoffs at time step n. Up to now, the best algorithm to price an Asian option on a lattice exactly runs in subexponential time [18]. Its superpolynomial nature forbids the use of very large n to approximate the theoretical

option value.

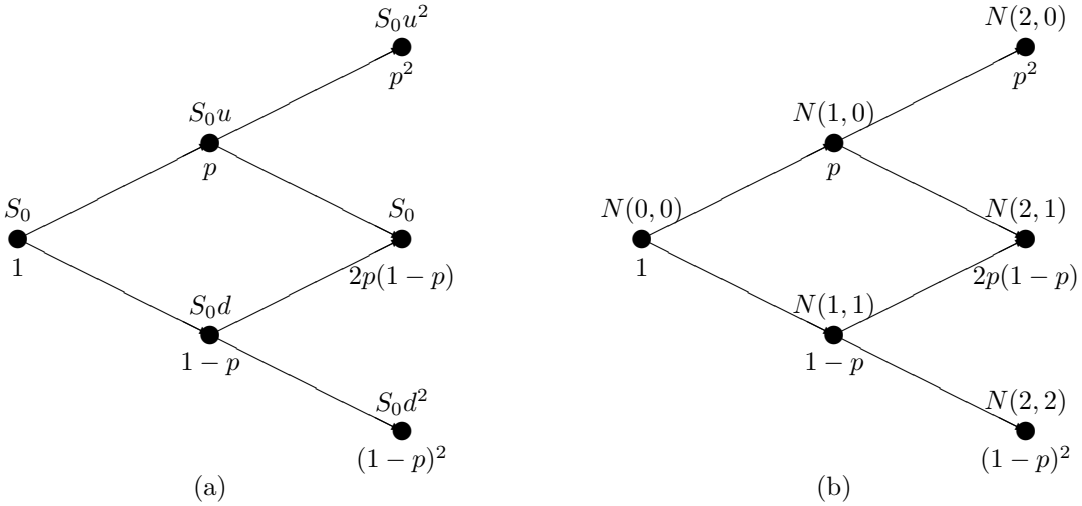

Figure 1: **The Binomial Lattice Model** (a) The stock price is placed above the node, where u and d denote the upward and the downward multiplication factors. (b) The node name is above the node. The probability of reaching each node from the root is listed under the node.

To strike a balance between efficiency and accuracy, Hull and White limit the number of possible average stock prices at each node of the lattice to be some manageable magnitude k [19]. These average stock prices are called representative averages for convenience. The option value for a missing average stock price at an arbitrary node is then estimated by interpolation. This popular method is widely accepted [20, 21, 22]. However, Forsyth *et. al.* show that the pricing results might not converge due to accumulations of interpolation errors [23]. Roughly speaking, this is because the range of average stock prices, defined as the range between the maximum and the minimum average stock prices, grows exponentially in n. To keep the pricing algorithm runs in polynomial time, the number of representative averages at each node must be a polynomial function in n. Therefore, the distance between two representative averages and the interpolation errors grows explosively. Aingworth *et. al.* prove that a simple, exact pricing formula for European-style Asian options exists at a node of the lattice if the average stock price exceeds a certain numerical bound [24]. (Their analysis of American-style Asian options seems mistaken.) There formula is useful in deriving polynomial-time algorithms with rigorous convergent proofs since the interpolation method is now required in a small portion (that grows only polynomially in n) of the range of average stock prices. Thus the interpolation error introduced at each node can be decreased with n by putting polynomially many representative averages at that node. Aingworth *et. al.* derive a $O(kn^2)$ algorithm with a error bound of $O(n/k)$, where k now denotes the average number of representative averages allocated at a node. Note that k can be varied for any desired trade-off between time and

accuracy. Dai *et. al.* determine the number of representative average stock prices allocated at each node by Lagrange multipliers so the accumulated interpolation error is minimized [27]. Their algorithm runs in $O(kn^2)$ with an error bound of $O(1/k^2)$. Thus they claim their algorithm runs in $O(n^{2.5})$ with a convergence rate $O(n^{-1})$.

Unfortunately, Aingworth's formula does not work for pricing American-style Asian options because of the need to estimate the exercise boundary — a threshold of average stock price that determines whether the option is exercised immediately— at each node. Up to now, rigorous convergence analysis for pricing American-style Asian options is even scarcer than that for European-style Asian options. To address this problem, this paper proposes approximation algorithms that produce provable upper and lower bounds (called **range bounds**) that bracket the desired option value. The range bound idea is previously adopted by [24, 25, 26] for pricing European-style Asian options. 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). This paper modifies Dai's algorithm in [27] to serve the upper bound algorithm and proposes a lower bound algorithm by taking advantages of Jensen's inequality. The error e of the proposed range bound algorithm converges to 0 for European-style Asian options if Aingworth's formula is applied. To obtain accurate pricing algorithm for American-style Asian options, the error e is reduced by using the exercise boundary (estimated by the upper bound pricing algorithm) to play a similar role as Aingworth's formula. Average stock prices above the estimated 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 exercise boundary with a way to estimate it while respecting the desired range bounds. It gives rise to a two-phase computational framework in which phase one calculates the estimated exercise boundary and phase two apply the range-bound algorithm to a portion of the range of average stock price. Thus phase two is expected to offer substantially more accurate results than if phase one is not in place. The numerical results suggest that this two-phase range bound algorithm provides tight upper and lower bounds that the desired option value is practically available.

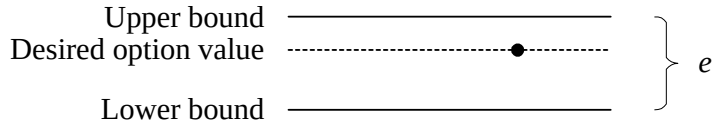

Figure 2: **Range bound and uncertainty e about the desired but unknown option value.**

This paper is organized as follows. Section 2 reviews required background knowledge for pricing Asian options on the lattice. Section 3 proposes accurate and efficient

two-phase range-bound algorithms for pricing both European-style and American-style Asian options. Rigorous proofs for the proposed range-bound algorithms are given in Section 4. Numerical results in section 5 verify the accuracy of our algorithms. Section 6 concludes the paper.

2 Basic Terms

Assume that an Asian option initiates at time 0 (in year) and matures at time T (in years). Define $S(t)$ as the stock price at year t. $S(t)$ follows the continuous-time stock price dynamics

$$S(t+dt) = S(t)\exp[(r-0.5\sigma^2)\,dt + \sigma\,dW_t], \tag{1}$$

where r, σ, and W_t denote the risk-free interest rate, the volatility of the stock price, and the standard Wiener process, respectively.

The payoff to exercise an Asian option at time τ ($\tau \leq T$) depends on the average stock price from time 0 to time τ defined as $A_\tau \equiv \frac{\int_0^\tau S(t)dt}{\tau}$. Let X be the exercise price. The payoff to exercise an Asian call option at time τ is $(A_\tau - X)^+$, where $(a)^+$ denotes $\max(a, 0)$. The theoretical option value is equal to expected discounted payoffs [28]. Thus the value of a European-style Asian call option is

$$\mathrm{E}[e^{-rT}(A_T - X)^+] \tag{2}$$

since a European-style option can only be exercised at the maturity date. On the other hand, the value of an American-style Asian call option is

$$\max_{\tau \leq T} \mathrm{E}[e^{-r\tau}(A_\tau - X)^+], \tag{3}$$

where τ is a random stopping time for exercising the option. This paper focuses on Asian call options; the extension to Asian put options is straightforward.

The above problem can be numerically solved by discrete-time models like lattice models. A lattice partitions the time between time 0 and time T into n equal-length time steps. The length of a time step Δt is therefore T/n. Let S_i denote the stock price at time step i, which corresponds to $S(i\Delta t)$ in the continuous-time model. Define a prefix sum of a price path from time step 0 to time step j as $\sum_{i=0}^{j} S_i$. The average stock price is then $A(j) \equiv \frac{\sum_{i=0}^{j} S_i}{j+1}$. The desired option values of European-style and American-style Asian options evaluated by lattice are

$$\mathrm{E}\left[\, e^{-rT}(A(n) - X)^+ \right] \tag{4}$$

and

$$\max_{j \leq n} \mathrm{E}\left[\, e^{-jr\Delta t}(A(j) - X)^+ \right], \tag{5}$$

respectively. Note that Eqs. (4) and (5) converge to Eqs. (2) and (3), respectively, as $n \to \infty$ [17].

This paper adopts the CRR lattice model suggested in [16]. A 2-time-step CRR lattice is depicted in Fig. 1(a). In the CRR lattice model, S_{i+1} equals $S_i u$ with probability p and $S_i d$ with probability $1-p$, where $d = 1/u$. The probability p for an up move is set to $(e^{r\Delta t} - d)/(u - d)$, where $u = e^{\sigma\sqrt{\Delta t}}$. The stock price at time step i that results from j down moves and $i - j$ up moves therefore equals $S_0 u^{i-j} d^j$. For convenience, the lattice nodes are labelled in Fig. 1(b). Let node $N(i, j)$ stand for the node at time step i reachable from the root with j cumulative down moves. Its associated stock price is $S_0 u^{i-j} d^j$. 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 $1 - p$. Node $N(i, j)$ can therefore be reached from the root node with probability $\binom{i}{j} p^{i-j}(1 - p)^j$.

Evaluating Eqs. (2) and (3) can be done by backward induction as follows. The option value at the maturity date is $(A_T - X)^+$. Let (i, j, A) denote the bucket with an average stock price A (from time step 0 to time step i) at node $N(i, j)$ and $\mathrm{v}(i, j, A)$ denote the corresponding option value. If this stock price moves up to node $N(i+1, j)$ at time step $i+1$, the average stock price becomes $A' \equiv \frac{(i+1)A + Su^{i+1-j}d^j}{i+2}$. If the stock price moves down to node $N(i + 1, j + 1)$, the average stock price becomes $A'' \equiv \frac{(i+1)A + Su^{i-j}d^{j+1}}{i+2}$. For pricing European-style options, the desired option value $\mathrm{v}(i, j, A)$ then equals

$$\mathrm{v}(i, j, A) = e^{-r\Delta t} \left[p \times \mathrm{v}(i + 1, j, A') + (1 - p) \times \mathrm{v}(i + 1, j + 1, A'') \right]. \tag{6}$$

On the other hand, an American-style option holder maximizes his profit by choosing whether to hold the option (with the value in Eq. (6)) or to exercise the option (with the payoff $(A - X)^+$). Thus the desired option value $\mathrm{v}(i, j, A)$ for American-style options at (i, j, A) is

$$\mathrm{v}(i, j, A) = e^{-r\Delta t} \max\left\{ (A - X)^+, \left[p \times \mathrm{v}(i + 1, j, A') + (1 - p) \times \mathrm{v}(i + 1, j + 1, A'') \right] \right\}. \tag{7}$$

The above formula can be applied inductively from time step $n - 1$ back to time step 0 with $\mathrm{v}(0, 0, S_0)$ at the root node giving the desired price under the lattice model.

The aforementioned algorithm is computationally intractable since there are $\binom{i}{j}$ price paths that reach node $N(i, j)$ and each such path gives rise to a distinct average price (bucket). The sum of number of buckets of the nodes at time i is $\sum_{j=0}^{i} \binom{i}{j} = 2^i$. Thus Eq. (6) (or Eq. (7)) should be evaluated $\sum_{i=0}^{n} 2^i$ times, which leads the pricing algorithm runs in exponential time. To address this problem, Hull and White limits the number of buckets at each node to a manageable magnitude [19]. When bucket (i, j, A) is missing, its corresponding option value is estimated by linear interpolation from its two nearest allocated buckets (i, j, A^-) and (i, j, A^+) via:

$$\mathrm{v}(i, j, A) = \frac{A - A^-}{A^+ - A^-} \mathrm{v}(i, j, A^+) + \frac{A^+ - A}{A^+ - A^-} \mathrm{v}(i, j, A^-), \tag{8}$$

where $A^- < A < A^+$. The aforementioned method may not converge due to accumulations of interpolation error [23]. Note that the maximum of the average stock price at time step n (i.e. $\frac{S+Su+...+Su^n}{n} = \frac{S\left(u^{n+1}-1\right)}{n(u-1)}$) grows exponentially in n. Indeed, the range between the maximum and the minimum average stock price also grows exponentially in n. However, the number of representative averages at each node must grow polynomially in n to make the pricing algorithm run in polynomial time. Thus the distance between two representative averages (like A^+ and A^- in Eq. (8)) and the interpolation error grows explosively with n.

Aingworth *et. al.* [24] derive a simple pricing formula for European-style Asian options for a node $N(i,j)$ at the lattice if the average stock price exceeds a certain amount as follows:

Theorem 2.1 *Note that the corresponding prefix sum for bucket (i,j,A) is $(i+1)A$. If the prefix sum is larger than $(n+1)X$ by ϵ. Then the option value* $\mathrm{v}(i,j,A)$ *equals*

- $[\,\epsilon + (n-i)S_0u^{i-j}d^j\,]/(n+1)$ *when* $r=0$, *and*
- $e^{-rT}\left[\,\epsilon + S_0u^{i-j}d^je^{r\Delta t}\frac{1-e^{(n-\ell)r\Delta t}}{1-e^{r\Delta t}}\,\right]/(n+1)$ *when* $r>0$.

This formula limit the use of the interpolation (i.e. Eq. (8)) on a smaller range of the average stock price: $[0, \frac{(n+1)X}{i+1}]$ since $\mathrm{v}(i,j,A)$ can be exactly solved by aforementioned formulas if $A > \frac{(n+1)X}{i+1}$. They then derive an $O(kn^2)$ algorithm with an error bound of $O(n/k)$, where k now denotes the average number of representative averages allocated at a node.

Dai *et. al.* determine the number of representative average stock prices allocated at each node by Lagrange multipliers so the accumulated interpolation error is minimized [27]. Let $k_{i,j}$ stand for the number of buckets allocated at node $N(i,j)$. The total number of buckets is equal to $\sum_{0\le j\le i\le n} k_{i,j} \approx k(n^2/2)$ as there are approximately $n^2/2$ nodes. These $k_{i,j}$ buckets (the average stock prices) shall divide the range $[0,(n+1)X/(i+1)]$ evenly. For example, let $b(i,j,\ell)$ denote the ℓth bucket at node $N(i,j)$. Then the corresponding average stock price and the prefix sum for this bucket are $\frac{\ell(n+1)X}{(i+1)k_{ij}}$ and $\frac{\ell(n+1)X}{k_{ij}}$, respectively.[1] Note that the difference between two average stock prices of two adjacent buckets is $\frac{(n+1)X}{(i+1)k_{i,j}} \le \frac{2nX}{(i+1)k_{i,j}}$. The linear interpolation error to estimate an arbitrary bucket at node $N(i,j)$ by Eq. (8) is bounded above by $M(2nX/ik_{i,j})^2$, where the constant M denotes the upper bound of $\left|\frac{\partial^2 v(i,j,A)}{\partial A^2}\right|$ for $0 \le j \le i \le n$ [23]. Thus the accumulated interpolation error is bounded above by

$$\sum_{0\le j\le i\le n} \binom{i}{j} p^{i-j}(1-p)^j M(2nX/ik_{i,j})^2. \tag{9}$$

[1] The root node $N(0,0)$ is a special case with k_{00}=1. The average stock price and the prefix sum for this bucket is S_0.

The aforementioned formula can be minimized by Lagrange multipliers by setting

$$k_{i,j} = \frac{n^2 k}{2} \times \frac{[B(i,j;p)/i^2]^{1/3}}{\sum_{0 \le m \le l \le n} [B(l,m;p)/l^2]^{1/3}}, \tag{10}$$

where $B(i,j,p) \equiv \binom{i}{j} p^{i-j}(1-p)^j$. They find that the error converges at a rate of $O(k^{-2})$.

3 The Range Bound Algorithm

The aforementioned algorithms do not work well for American-style Asian options since Theorem 2.1 is not valid. This section will propose approximation algorithms that produce provable upper and lower bounds (called **range bounds**) that bracket the desired option value. We will first modify Dai's algorithm [27] to price American-style Asian options and then prove that the resulting algorithm `Up1` overestimates the desired option values. To further reduce the pricing error, the early exercise property for American-style options is used to tighten the range of average stock price (or equivalently, the range of prefix sum for ease of later discussion) that requires interpolation. We use `Up1` to tighten the range of prefix sum that requires interpolation and apply the same algorithm to the remaining portion of prefix sum. The resulting algorithm, called `Up2`, provides a tighter upper bound for the desired option value. On the other hand, the lower bound algorithm `DownE` for European-style Asian option is constructed by taking advantages of Jensen's inequality. The lower bound algorithm for American-style Asian options `DownA` is then constructed by modifying `DownE`.

3.1 Some useful terminologies

Some terms are introduced here 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 [1]. The prefix-sum range (or the range of average stock price) at node $N(i,j)$ is $[\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$ (or $\left[\,\frac{\mathcal{R}_{\min}(i,j)}{i+1}, \frac{\mathcal{R}_{\max}(i,j)}{i+1}\,\right]$).

Then we define the **ideal lattice**, a critical concept for deriving range bounds. The ideal lattice 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 the approximation pricing algorithms must correspond to some bucket at the same node in the ideal lattice. **Practical lattices** refer to the necessarily finite-sized, bucket-based lattices used by approximation pricing algorithms.

For any bucket b, we use P_b and E_b to denote its associated prefix sum and the option value for brevity. Because the option value for bucket b evaluated by the ideal

lattice may differ from that in the practical lattice, the superscripts I and P are added to distinguish them. The option value at bucket b in the ideal lattice and that in the practical lattice, if b exists, become E_b^I and E_b^P, respectively. Because the ideal lattice contains any possible average stock prices, no interpolation is required in the ideal lattice. Thus the value computed by the ideal lattice equals the desired option value, which is denoted by `DesiredValue` for convenience.

3.2 The first upper-bound algorithm: `Up1`

The algorithm for pricing European-style Asian options proposed in [27] allocate buckets at the prefix sum range $[\,0, (n+1)X\,]$. My algorithm for American-style Asian options adopt $[\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$ instead of $[\,0, (n+1)X\,]$ for the prefix-sum range of any arbitrary node $N(i,j)$. This change is needed because a prefix sum exceeding $(n+1)X$ no longer results in any easily calculated option evaluation formula as in Theorem 2.1 for American-style options.

Because the prefix sum range for each node of the lattice have changed, the number of buckets allocate to each node must vary to minimize the accumulated interpolation error. Define the range of prefix sum of node $N(i,j)$ as

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

Thus the interpolation error to estimate an arbitrary bucket at node $N(i,j)$ is bounded above by $M(R_{ij}/ik_{i,j})^2$. The accumulated interpolation error is then derived by modifying Eq. (9) as follows:

$$\sum_{0\le j\le i\le n} \binom{i}{j} p^{i-j}(1-p)^j M(R_{ij}/ik_{i,j})^2.$$

The accumulated interpolation errors can be minimized by Lagrange multipliers (like Eq. (10)) by setting

$$k_{i,j} = \frac{n^2 k}{2} \times \frac{[B(i,j;p)R_{ij}/i^2]^{1/3}}{\sum_{0\le m\le l\le n}[B(l,m;p)R_{lm}/l^2]^{1/3}}, \tag{11}$$

My `Up1` algorithm bases on the Dai's algorithm [27] with two straightforward modifications. First, the range of prefix sums and the bucket allocation scheme are changed as mentioned above. Second, early exercise is considered at each bucket. That is, the backward induction formula for American-style options Eq. (7) is used instead of Eq. (6). Note that rigorous convergence analysis is not available for American-style Asian options. This is because the prefix sum range node $N(i,j)$ grows exponentially in i. When the number of time steps n (of the lattice) increases, the prefix-sum ranges of newly added nodes grows exponentially in n. The interpolation errors introduced by newly added nodes do not decrease with n unless the number of buckets allocated at

these nodes also grows exponentially in n. Numerical experiments in Section 5 also suggests that pricing European-style Asian options without reducing the prefix-sum ranges by Theorem 2.1 results in convergence problem.

3.3 The second upper-bound algorithm: `Up2`

As a rule, the smaller the prefix-sum ranges, the better the approximation. Note that the payoff of an early-exercise bucket b at time step m is simply $P_b/(m+1) - X$, such buckets can be removed from the prefix-sum ranges, thus limiting the prefix-sum ranges further. But how are the early-exercise buckets distributed within the prefix-sum range? Before answering this key question, we state a useful lemma below.

Lemma 3.1 (Contraction lemma) *Suppose that $P_{b_1} > P_{b_2}$ at buckets b_1 and b_2 of the same node in the ideal lattice at time step 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. □

The following theorem states that there exists a prefix sum at each node in the ideal lattice that separates the early-exercise buckets from the non-early-exercise ones.

Theorem 3.2 *Suppose that $P_{b_1} > P_{b_2}$ at buckets b_1 and b_2 of the same node in the ideal lattice at time step 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. It can be claimed 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.[2] The second inequality is by Lemma 3.1. The last 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. □

The prefix sum at each node of the ideal lattice that separates the early-exercise buckets from the non-early-exercise ones is called the **optimal exercise boundary**. The optimal exercise boundary can be estimated by `Up1`. 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 estimated exercise boundary. The option value of an early-exercise bucket is simply the average stock price minus the strike price X.

[2] This can be observed in Eq. (7). The option value $v(i,j,A)$ is larger than or equal to the exercise value $A - X$.

We now present a two-phase algorithm, called `Up2`, which incorporates the idea of tightening prefix-sum ranges with the estimated exercise boundary. Phase one uses `Up1` to estimate the exercise 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 prefix sum to the lowest prefix sum whose corresponding bucket is exercised early. Phase two runs `Up1` on the tightened prefix-sum ranges. Note that the number of buckets allocated in each node (i.e. k_{ij}) must be recalculated with Eq. (11) because ranges R_{ij} have been reduced in phase one. A bucket b at time step i with a prefix sum P_b on or above the exercise boundary will be exercised in phase two (with the option value $P_b/(i+1) - X$).

Although `Up2` allocates the same number of buckets as `Up1`, 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. In fact, the exercise boundary estimated by `Up1` is not useful only to `Up2` 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 4).

3.4 The lower-bound algorithm

We will first develop a lower-bound algorithm for pricing European-style Asian option `DownE` by taking advantages of Jensen's inequality. The algorithm for American-style Asian options (`DownA`) can be viewed as a two-phase algorithm: The first phase tightens the prefix-sum ranges by the estimated exercise boundary computed by `UpA` and the second phase runs `DownE` on the tightened prefix-sum ranges.

The lower-bound algorithm for European-style Options: `DownE`

The core idea of our lower bound algorithm is to use

$$e^{-rT} \left(E \left[A(n) - X \right] \right)^+$$

to approximate the desired option value (see Eq. (4)). The approximation underestimates the option value because of Jensen's inequality. To improve accuracy, our algorithm adds bucketing to the above idea.

To implement the idea with bucketing, the prefix sum for bucket $b(i,j,\ell)$ will no longer be a fixed value $\frac{\ell(n+1)X}{k_{ij}}$. Instead, it is calculated explicitly to hold the average prefix sum of all the paths covered by bucket $b(i,j,\ell)$ with range $\left[\frac{\ell(n+1)X}{k_{ij}}, \frac{(\ell+1)(n+1)X}{k_{ij}}\right)$. Instead of the backward induction method used in upper bound algorithms like Eq. (6) and (7), the lower bound algorithm uses forward induction method described as follows. For convenience, let $s(i,j,\ell)$ be the average prefix sum of bucket $b(i,j,\ell)$, $p(i,j,\ell)$ be the probability of reaching $b(i,j,\ell)$ from the root node, and $t_s(i,j,\ell)$ and

$t_p(i,j,\ell)$ be temporal storages for calculating the average prefix sum and the probability for $b(i,j,\ell)$. The computation begins at the bucket $b(0,0,0)$ with $p(0,0,0)=1$ and $s(0,0,0)=S_0$. The average prefix sums and the probabilities of the buckets at time step $i+1$ are calculated by taking advantages of those information of the buckets at time step i as follows. First, all temporal storages for the buckets at time step $i+1$ are set to zero. Next, adding the "prefix sum and the probability" contribution for each bucket $b(i,j,\ell)$ with prefix sum smaller than $(n+1)X$ to the following buckets at time step $i+1$. 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 $N(i+1,j)$ with prefix sum $s(i,j,\ell)+S_0u^{i-j+1}d^j$ and down to $N(i+1,j+1)$ with prefix sum $s(i,j,\ell)+S_0u^{i-j}d^{j+1}$. The up movement from $b(i,j,\ell)$ goes to the bucket

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

Thus the temporal storages t_s and t_p for this bucket are updated by adding $p\cdot p(i,j,\ell)\cdot[\,s(i,j,\ell)+S_0u^{i-j+1}d^j\,]$ and $p\cdot p(i,j,\ell)$ to them, respectively. Similarly, the down movement from $b(i,j,\ell)$ goes to the bucket

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

Thus the temporal storages t_s and t_p for this bucket are updated by adding $(1-p)\cdot p(i,j,\ell)\cdot[\,s(i,j,\ell)+S_0u^{i-j}d^{j+1}\,]$ and $(1-p)\cdot p(i,j,\ell)$ to them, respectively. After the above is done for every bucket at time step i, the probability and the average prefix sum for every bucket $b(i+1,j,\ell)$ can then be calculated by $p(i+1,j,\ell)\equiv t_p(i+1,j,\ell)$ and $s(i+1,j,\ell)\equiv t_s(i+1,j,\ell)/p(i+1,j,\ell)$, respectively. The aforementioned procedure can be applied forwardly from time step 1 to time step n. The option value for the buckets with prefix sums at or above $(n+1)X$ is computed by Theorem 2.1. The option value estimated by `DownE` is therefore

$$\sum_{b\in B}p(b)v(b),$$

where B denotes the set of buckets with prefix sum larger than $(n+1)X$ in `DownE`, $p(b)$ denotes the probability of bucket b, and $v(b)$ denotes the option value of b computed by Theorem 2.1.

To describe the core idea of our lower bound algorithm, a brief example is illustrated in Fig. 3 without the help of Theorem 2.1 and bucketing (i.e. each node has only one bucket) for simplicity. Take node E for example. Its average prefix sum is calculated by adding up the contributions of the price path A-B-E and the A-C-E. The result (stored in $t_s(2,1,0)$) is

$$\begin{aligned}&(1-p)p(S_0+S_0u+S_0)+p(1-p)(S_0+S_0d+S_0)\\ =\;&p(1-p)[\,(S_0+S_0u+S_0)+(S_0+S_0d+S_0)\,].\end{aligned}\tag{12}$$

Its associated probability $p(2,1,0)$ is $p(1-p)+(1-p)p = 2p(1-p)$. Finally, the average prefix sum is computed by dividing Eq. (12) by the associated probability (i.e. $\frac{t_s(2,1,0)}{p(2,1,0)}$) to obtain

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

The average prefix sum and the probability for each node of the lattice are given in Fig. 3. Finally, the option value computed by `DownE` (without the help of Theorem 2.1) is hence

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

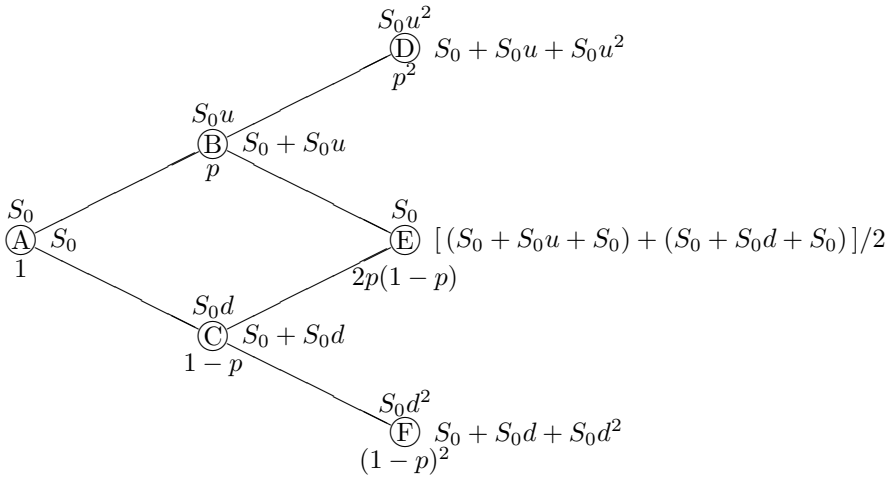

Figure 3: `DownE` **without bucketing.** The stock prices are listed above the nodes. The average prefix sums are listed to the right of the nodes. The probabilities for the average prefix sums are listed under the nodes.

The lower-bound algorithm for American-style options: `DownA`

`DownA` is based on `DownE` and contains two phases. Phase one is identical to `Up2`'s phase one. In other words, it calls upon the upper-bound algorithm `Up1` to yield an exercise boundary, and the prefix-sum ranges are subsequently tightened. Phase two runs `DownE` over the tightened prefix-sum ranges. A bucket b with a prefix sum on or above the exercise boundary will be exercised early with value $A_b - X$, where A_b denotes the average stock price of the bucket b. The option value estimated by `DownA`

is simply the sum of the values contributed by the early-exercise buckets and by the buckets with average stock price larger than X at maturity date. These buckets are called **terminal buckets** for simplicity.

4 The Range-Bound Proofs

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

Theorem 4.1 `DownA` $\leq$ `DesiredValue`.

Proof. The desired option value equals the discounted expected payoff of the terminal buckets in the ideal lattice *when buckets are exercised optimally* (see Eq. (5)). It therefore suffices to prove that `DownA` produces an option value that does not exceed the one given by the ideal lattice 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. Every path in $\wp_b$ has length t_b. Now we use those $\wp_b$ to define an exercise strategy on the ideal lattice: Each path in $\wp_b$ on the ideal lattice is terminated at the same node where bucket b resides. In other words, the ideal lattice uses the same early-exercise strategy as `DownA`. This exercise strategy produces an option value A that cannot exceed the desired option value because it may not be optimal.

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

$$e^{-rt_b} \sum_{\rho=(S_0,S_1,\ldots,S_{t_b})\in\wp_b} \text{prob}[\,\rho\,] \times \left\{ \frac{1}{t_b+1} \sum_{i=0}^{t_b} S_i - X \right\}^+ . \tag{13}$$

Recall that each bucket in `DownA` stores the average prefix sum of all the paths covered by it. Hence the contribution of $\wp_b$ to `DownA`'s option value is[3]

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

which is smaller than (13) by Jensen's inequality. By summing the contributions over all terminal buckets b, we can conclude that `DownA` gives a lower bound on A. □

[3]Note that any path ending up at the same bucket $b(i,j,\ell)$ carries the same probability $p^{i-j}(1-p)^j$. Thus the average prefix sum is simple the arithmetic average of those said prefix sums.

The next lemma says that the Asian option value is convex with respect to the prefix sum in the ideal lattice. Before introducing this lemma, we will discuss an importance concept in finance: arbitrage. Arbitrage denotes the situation that one can earn extra profit without suffering any risk. Financial experts argue that the arbitrage opportunity can not exist for long due to self-beneficial nature of human. The following lemma will show that under certain assumption, one can construct an arbitrage portfolio to earn money (extra profit) at the option initial date but not to suffer any loss (or risk) in the future. Thus, this assumption must not hold due to no arbitrage nature of financial markets. Similar arguments can be widely found in financial text books like [29].

Lemma 4.2 (Convexity lemma) *Let b_1, b_2, and b_3 be buckets at node $N(i,j)$ in the ideal lattice 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** $A(P,i)$ that initiates at time step i to facilitate the proof. It pays

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

if it is exercised at time step $i+\ell$, where Σ equals the prefix sum from time step $i+1$ to time step $i+\ell$. The option $A(P,i)$ is identical to an Asian option at time step i if P is set to the prefix sum of the stock price from time step 0 to time step i. Thus the values of these two options are equal.

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 matured at time step 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 we proceed to construct an arbitrage portfolio. 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 at time step $i+\ell$, we exercise $A(P_{b_1},i)$ and $A(P_{b_3},i)$ to generate nonnegative cash flow as follows:

$$-\left(\frac{P_{b_2}+\Sigma}{i+\ell+1} - X\right)^+ + \lambda\left(\frac{P_{b_1}+\Sigma}{i+\ell+1} - X\right)^+ + (1-\lambda)\left(\frac{P_{b_3}+\Sigma}{i+\ell+1} - X\right)^+ \geq 0.$$

If $A(P_{b_2},i)$ is never exercised, we junk $A(P_{b_1},i)$ and $A(P_{b_3},i)$ at time step n to generate zero cash flow. □

We next establish that `Up1` is an upper-bound algorithm.

Theorem 4.3 `DesiredValue` $\leq$ `Up1`.

Proof. For every bucket b, we prove that $E_b^I \leq E_b^P$ by induction, where the practical lattice here refers to the lattice constructed by `Up1`. The theorem holds at time step n as the option value equals $\{P_b/(n+1) - X\}^+$ at any bucket b. So $E_b^I \leq E_b^P$ at time step n. The induction hypothesis is that $E_b^I \leq E_b^P$ for any bucket b in the practical model at time step t. We next show that this remains true at time step $t-1$ for $t \geq 1$.

Consider any bucket b in the ideal lattice at time step $t-1$. 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 lattice. Let bucket $u(b)$ be sandwiched between buckets $u(b_1)$ and $u(b_2)$ in the practical lattice. With λ satisfying

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

we have

$$E^I_{u(b)} \leq \lambda E^I_{u(b_1)} + (1-\lambda) E^I_{u(b_2)} \leq \lambda E^P_{u(b_1)} + (1-\lambda) E^P_{u(b_2)} = E^P_{u(b)}.$$

The first inequality is by Lemma 4.2, and the second inequality is by the induction hypothesis. The equality holds because `Up1` computes $E^P_{u(b)}$ as the linear interpolation of $E^P_{u(b_1)}$ and $E^P_{u(b_2)}$ with the said weights (see Eq. (8)). By the same argument, $E^I_{d(b)} \leq E^P_{d(b)}$. We next consider three cases.

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

$$E_b^I = \left[\, pE^I_{u(b)} + (1-p)E^I_{d(b)} \,\right] e^{-r\Delta t} \leq \left[\, pE^P_{u(b)} + (1-p)E^P_{d(b)} \,\right] e^{-r\Delta t} = E_b^P.$$

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

$$\left[\, pE^I_{u(b)} + (1-p)E^I_{d(b)} \,\right] e^{-r\Delta t} \leq \left[\, pE^P_{u(b)} + (1-p)E^P_{d(b)} \,\right] e^{-r\Delta t} \leq (P_b/t) - X.$$

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

Case 3: Suppose that b is an early-exercise bucket in the ideal lattice but not an early-exercise bucket in the practical lattice. 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 4.3 holds for a large class of algorithms, not just `Up1`. 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 all popular approximation algorithms that follow Hull and White's algorithm [19] with linear interpolation (see Eq. (8)) are upper-bound algorithms.

Corollary 4.4 *The approximation algorithm that follows Hull and White's algorithm [19] with linear interpolation is an upper-bound algorithm.*

The next result states a general property that the exercise boundary determined by `Up1` satisfies.

Corollary 4.5 *A bucket with a prefix sum equal to or larger than the exercised boundary determined by* `Up1` *must be an early-exercise bucket in the ideal lattice.*

Proof. By case 2 in the proof of Theorem 4.3, an early-exercise bucket under `Up1` must also be an early-exercise bucket in the ideal lattice. Theorem 3.2 completes the proof. □

The above corollary implies that the exercise boundary determined by `Up1` cannot be lower than the exercise boundary of the ideal lattice. 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 tightened prefix-sum ranges made possible by the exercise boundary estimated in phase one, it should outperform the original one-phase algorithm.

Theorem 4.6 *An upper-bound algorithm that uses the estimated exercise boundary given by* `Up1` *remains an upper-bound algorithm as long as it works on the same lattice.*

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 lattices by Corollary 4.5. 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. □

We finally prove that `Up2` is an upper-bound algorithm.

Corollary 4.7 `DesiredValue` $\leq$ `Up2`.

Proof. It is immediate from Theorem 4.3 and Theorem 4.6. □

5 Numerical results

Tightening the prefix-sum ranges that require interpolation is a key factor to obtain a convergent lattice algorithm for pricing Asian options as illustrated in Table 1 and Table 2. Table 1 illustrates convergent pricing results for pricing European-style Asian options by tightening the prefix-sum ranges by Theorem 2.1. Dai's algorithm [27] serves as the upper bound algorithm and `DownE` serves as the lower bound algorithm. The pricing error denotes the difference between the upper bound and the lower

bound. Dai *et. al.* claim that the pricing errors converge at a rate of $O(k^{-2})$ by adopting their bucket allocation scheme (see Eq. (10)). Their claim is verified by observing that the pricing errors converge at a rate of $O(n^{-2})$.[4] On the other hand, Table 2 illustrates the pricing results of Dai's algorithm and `DownE` without applying Theorem 2.1. Note that the average number of buckets used in this case is eight times larger than that in Table 1. But the pricing errors do not converge especially when the volatility of the stock price and the time to maturity are high.

For pricing American-style options, the prefix-sum ranges can be tightened by taking advantages of the early exercise property. The advantages for tightening the prefix-sum ranges are illustrated in Table 3. The pricing errors under tightened prefix-sum ranges are bounded above by the differences between the pricing results of `Up2` and `DownA`, while the pricing errors for original prefix sum ranges are bounded above by the differences between the pricing results of `Up1` and `DownA`. Obviously, the range bound algorithm with tightened prefix-sum ranges is more competitive.

Finally, we investigate the convergence behavior of the proposed two-phase range-bound algorithm. The pricing results tabulated in Table 4 suggested that the proposed algorithm converges well even when the volatility of the stock price and the time to maturity are high.

6 Conclusions

How to price Asian options efficiently and accurately is a long-standing problem. While polynomial time lattice algorithms with convergence guarantee are available for pricing European-style Asian options, rigorous convergence analysis for pricing American-style Asian options is even scarcer. This paper suggests an efficient two-phase range-bound algorithm that bracket the desired option value, whose brute-force computation is prohibitive. By taking advantages of the early exercise property of the American-style Asian options, the pricing error can be significantly reduced. Numerical experiments show that the upper and the lower bounds option values produced by the proposed range-bound algorithm are essentially numerically identical. Thus the desired option values for American-style Asian options can be obtained practically and efficiently by the proposed range-bound algorithm.

References

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

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

[4] Note that the average number of buckets for each node k is set to n.

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

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

[5] E. Levy, Pricing European Average Rate Currency Options, J. Internat. Money and Finance 11 (1992) 474–491.

[6] M.A. Milevsky, and S.E. Posner, Asian Options, the Sum of Lognormals, and the Reciprocal Gamma Distribution, J. Financial and Quant. Anal. 33 (1998) 409–422.

[7] W. T. Shaw, Modeling Financial Derivatives with Mathematica, Cambridge University Press, Cambridge, U.K., 1998.

[8] S.M. Turnbull, and L.M. Wakeman, A Quick Algorithm for Pricing European Average Options, Financial and Quant. Anal. 26 (1991) 377–389.

[9] M.C. Fu, D.B. Dilip, and T. Wang, Pricing Continuous Asian Options: A Comparison of Monte Carlo and Laplace Transform Inversion Methods, J. of Comput. Finance 2 (1998/9) 49–74.

[10] J.E. Zhang, A Semi-Analytical Method for Pricing and Hedging Continuously Sampled Arithmetic Average Rate Options, J. Comput. Finance 5 (2001) 59–79.

[11] J.E. Zhang, Pricing Continuously Sampled Asian Options with Perturbation Method, J. Futures Markets 23 (2003) 535–560.

[12] P.P. Boyle, M. Broadie, and P. Glasserman, Monte Carlo Methods for Security Pricing, J. Econom. Dynam. Control 21 (1997) 1267–1321.

[13] M. Broadie, and P. Glasserman, Estimating Security Price Derivatives Using Simulation, Management Sci. 42 (1996) 269–285.

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

[15] A.G.Z. Kemna, and A.C.F. Vorst, A Pricing Method Based on Average Asset Values, J. Banking and Finance 14 (1990) 113–129.

[16] J. Cox, S. Ross, and M. Rubinstein, Option Pricing: A Simplified Approach, J. of Financial Econom. 7 (1979) 229–264.

[17] D. Duffie, Dynamic Asset Pricing Theory, 2nd ed. Princeton University Press, Princeton, 1996.

[18] T.-S. Dai, and Y.-D. Lyuu, An Exact Subexponential-Time Lattice Algorithm for Asian Options, Acta Inform. 44 (2007), 23–39.

[19] J. Hull, and A. White, Efficient Procedures for Valuing European and American Path-Dependent Options, J. Derivatives 1 (1993) 21–31.

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

[21] T.R. Klassen, Simple, Fast and Flexible Pricing of Asian Options, J. of Comput. Finance 4 (2001) 89–124.

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

[23] P.A. Forsyth, K.R. Vetzal, and R. Zvan, Convergence of Numerical Methods for Valuing Path-Dependent Options Using Interpolation, Rev. of Derivatives Rech. 5 (2002) 273–314.

[24] D. Aingworth, R. Motwani, and J.D. Oldham, Accurate Approximations for Asian Options, In Proceedings of 11th Annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia: Society for Industrial and Applied Mathematics (2000) 891–900.

[25] K. Akcoglu, M.-Y. Kao, and S.V. Raghavan, (2001). Fast Pricing of European Asian Options with Provable Accuracy: Single-Stock and Basket Options, Lecture Notes in Comput. Sci. 2161 (2001) 404–415.

[26] L.C.G. Rogers, and Z. Shi, The Value of an Asian Option, J. Appl. Probab. 32 (1995) 1077–1088.

[27] Dai, T.-S., G.-S. Huang, and Y.-D. Lyuu. An Efficient Convergent Lattice Algorithm for European Asian Options, Appl. Math. and Comput. 169 (2005) 1458-1471.

[28] J.M. Harrison, S.R. Pliska, Martingales and Dtochastic Integrals in the Theory of Continuous Trading, Stochastic Process. Appl. 11, (1981) 215–260.

[29] J. Hull, Options, Futures, and Other Derivatives, 3rd ed. Prentice Hall Press, Englewood Cliffs, N.J., 1997.

A Proof of the Contraction Lemma

Lemma A.1 *Under the assumptions given in Lemma 3.1,*

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} \geq E^I_{b_1} - E^I_{b_2}. \tag{14}$$

Proof. The lemma will be proved by induction backwardly from time step n to time step 0. The base case involves the buckets allocated at maturity (i.e. time step n). Note that $E^I_{b_i} = \left(\frac{P_{b_i}}{n+1} - X\right)^+ = \max\left(\frac{P_{b_i}}{n+1}, X\right) - X$, $i = 1, 2$. Thus

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

because $P_{b_1} > P_{b_2}$. Assume that Eq. (14) is valid at time steps $m+1$. The followings will show that Eq. (14) is valid at time step m. Assume that buckets b_1 and b_2 are located at node $N(m, j)$ at time step m. Bucket b_i moves up to bucket $u(b_i)$ (at node $N(m+1, j)$) and down to bucket $d(b_i)$ (at node $N(m+1, j+1)$) in the ideal lattice. The induction step is divided into the following four cases.

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

$$\begin{aligned}
\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} &\geq \frac{P_{b_1} - P_{b_2}}{m+2} \\
&= p\,\frac{P_{u(b_1)} - P_{u(b_2)}}{m+2} + (1-p)\,\frac{P_{d(b_1)} - P_{d(b_2)}}{m+2} && (15) \\
&\geq \left\{ p[\, E^I_{u(b_1)} - E^I_{u(b_2)}\,] + (1-p)[\, E^I_{d(b_1)} - E^I_{d(b_2)}\,] \right\} e^{-r\Delta t} && (16) \\
&= \left\{ \left[\, pE^I_{u(b_1)} + (1-p)E^I_{d(b_1)}\,\right] - \left[\, pE^I_{u(b_2)} + (1-p)E^I_{d(b_2)}\,\right] \right\} e^{-r\Delta t} \\
&= E^I_{b_1} - E^I_{b_2}. && (17)
\end{aligned}$$

Note that Eq. (15) holds since

$$P_{u(b_1)} - P_{u(b_2)} = \left(P_{b_1} + S_0 u^{m+1-j} d^j\right) - \left(P_{b_2} + S_0 u^{m+1-j} d^j\right) = P_{b_1} - P_{b_2}$$

and

$$P_{d(b_1)} - P_{d(b_2)} = \left(P_{b_1} + S_0 u^{m-j} d^{j+1}\right) - \left(P_{b_2} + S_0 u^{m-j} d^{j+1}\right) = P_{b_1} - P_{b_2}.$$

Equation (16) is by the induction hypothesis Eq. (14). Equation (17) is by the backward induction formula Eq. (7). Note that neither b_1 nor b_2 is exercised immediately. Thus the value to exercise the option early is less than the value to hold the option.

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 (14).

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 (14) 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

$$\frac{P_{b_1}}{m+1} - X < \left[\, pE^I_{u(b_1)} + (1-p)E^I_{d(b_1)} \,\right] e^{-r\Delta t}, \tag{18}$$

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

Subtracting inequality (19) from inequality (18) results in

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} < \left\{ p[\, E^I_{u(b_1)} - E^I_{u(b_2)} \,] + (1-p)[\, E^I_{d(b_1)} - E^I_{d(b_2)} \,] \right\} e^{-r\Delta t}. \tag{20}$$

But

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

contradicts the inequality (20). □

σ	τ	n	DownE	Dai	Pricing Error
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

Table 1: **Pricing European-Style Asian Options with Tightened Prefix-Sum Ranges.** The data are: $S_0 = X = 100$ and $r = 10\%$ per annum. σ, τ, and n denote the volatility of the stock price, time to maturity, and the number of time steps, respectively. The average number of buckets k allocated at each node is set to n. `Dai` denotes Dai's algorithm in [27]. The pricing errors converge at a rate of $O(n^{-2})$.

σ	τ	n	DownE	Dai	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

Table 2: **Pricing European-Style Asian Options without Tightening the Prefix-Sum Ranges.** The setup is identical to Table 1 except that the average number of buckets k allocated at each node is set to $8n$.

σ	X	r	DownA	Up2	Up1	Up1 −DownA	Up2 −DownA
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 3: **Comprehensive tests for Applying the Range-Bound Algorithm on the Original or Tightened Prefix-Sum Ranges.** The data are: $S_0 = 100$, $n = 300$, $k = 500$ and $\tau = 1$. σ, X, and r listed in the first three columns denote the volatility, the strike price, and the risk-free interest rate, respectively.

σ	τ	n	DownA	Up2	Up2 − DownA
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

Table 4: **Convergence of the Two-Phase Range-Bound Algorithm for Pricing American-Style Asian Options.** The data are: $S_0 = X = 100$ and $r = 10\%$ per annum. σ, τ, and n denote the volatility of the stock price, time to maturity, and the number of time steps, respectively. The average number of buckets k allocated at each node is set to $8n$.