# Pricing Asian Options on Lattices

Tian-Shyr Dai
Department of Computer Science and Information Engineering
National Taiwan University

# Contents

# List of Figures

# List of Tables

## Abstract

Path-dependent options are options whose payoff depends nontrivially on the price history of an asset. They play an important role in financial markets. Unfortunately, pricing path-dependent options could be difficult in terms of speed and/or accuracy. The Asian option is one of the most prominent examples. The Asian option is an option whose payoff depends on the arithmetic average price of the asset. How to price such a derivative efficiently and accurately has been a long-standing research and practical problem. Up to now, there is still no simple exact closed form for pricing Asian options. Numerous approximation methods are suggested in the academic literature. However, most of the existing methods are either inefficient or inaccurate or both.

Asian options can be priced on the lattice. A lattice divides the time interval between the option initial date and the maturity date into $n$ equal time steps. The pricing results converge to the true option value as $n \to \infty$. Unfortunately, only exponential-time algorithms are currently available if such options are to be priced on a lattice without approximations. Although efficient approximation methods are available, most of them lack convergence guarantees or error controls. A pricing algorithm is said to be *exact* if no approximations are used in backward induction.

This dissertation addresses the Asian option pricing problem with the lattice approach. Two different methods are proposed to meet the efficiency and accuracy requirements. First, a new trinomial lattice for pricing Asian options is suggested. This lattice is designed so the computational time can be dramatically reduced. The resulting exact pricing algorithm is proven to be the first exact lattice algorithm to break the exponential-time barrier. Second, a polynomial time approximation algorithm is developed. This algorithm computes the upper and the lower bounds of the option value of the exact pricing algorithm. When the number of time steps of the lattice becomes larger, this approximation algorithm is proven to converge to the true option value for pricing European-style Asian options. Extensive experiments also reveal that the algorithm works well for American-style Asian options.

# Chapter 1

# Introduction

## 1.1 Setting the Ground

In recent decades, financial derivatives have played an increasingly important role in the world of finance. A derivative is a financial instrument whose payoff depends on the value of other more basic underlying economical variables, like stock indexes and interest rates. Basically, derivatives can be categorized into four groups: futures, forwards, options and swaps. Standardized futures and options are traded actively in the exchanges. Forwards, swaps, and many nonstandardized derivatives are traded in the rapidly growing over-the-counter market. These complex contracts are usually hard to manage, so they give rise to new problems in designing new contracts, pricing these contracts and hedging them. To deal with this problem, a new principle, named *financial engineering*, is founded. This new principle involves the design, management, and implementation of financial instruments through which we can meet the requirements of risk managements. Knowledge from the finance, mathematics, and computer science are combined to face the new challenges from financial engineering. Recently, financial engineering has become the hottest topic in both finance and applied mathematics.

## 1.2 Options

An option is a kind of financial derivatives that gives the owner the right to buy or sell another financial asset. There are two basic types of options: the call options and and the put options. A *call* option allows the holder to buy the underlying asset with a predetermined price at or before a certain date. On the other hand, a *put* option gives the holder the right to sell the underlying asset. The predetermined price mentioned above is called the *strike price*. The certain date that a option owner is allowed to exercise the option at or before it is known as the *maturity date*. The financial asset that the option holder can buy or sell with the exercise price is called the *underlying asset*. Exercising an option denotes that the holder exercises the right to buy or sell the underlying asset. An American-style option can be exercised any time before maturity while a European-style option can only be exercised at maturity. More details about options are introduced in Section 2.2.

With the rapid growth and the deregulation of financial markets, nonstandardized options are created by financial institutions to fit their clients needs. These complex options are usually traded in the rapidly growing over-the-counter markets. Most of these options' values depend nontrivially on the price history of other financial assets. We call these options path-dependent. Path-dependent options are now playing important roles in financial markets. Some path-dependent derivatives such as barrier options can be efficiently priced [34]. Others, however, are known to be difficult to price in terms of speed and/or accuracy [35]. Pricing these options accurately and efficiently is an important problem in financial field.

## 1.3 Asian Options

An Asian option is an option whose payoff depends on the arithmetic average price of the underlying asset. Take a European-style Asian call as an example. Assume that the average price of the underlying asset between the option initial date and the maturity date is $A$. Then the option holder has the right to buy (or sell) the underlying asset with price $A$. This contract is useful for hedging transactions whose cost is related to the average price of the underlying asset (such as crude oil). Its price is also less subject to price manipulation; hence variants of Asian options are popular especially in thinly-traded markets. How to price an Asian option accurately and efficiently is important in both financial and academic fields.

The Asian option is one of the most representative example of the options that are hard to be priced in terms of speed and/or accuracy. Up to now, there is still no simple closed form for pricing Asian options. Numerous approximation methods are suggested in academic literatures. However, most of the existing methods are either inefficient or inaccurate or both [20, 21]. Generally speaking, these approximation methods can be grouped into three different categories: approximation analytical formulae, (quasi-) Monte Carlo simulations, and the lattice (and the related PDE) approach.

The analytical formulae approach denotes that the option is priced by (semi-) closed form formulae. Usually these formulae are derived by solving (stochastic) partial differential equations or by applying some probability tools. The major problem of this approach is that the partial differential equations for pricing the Asian option cannot be solved. Some ad-hoc approximation methods are suggested in the literature, but most of them fail in extreme cases [21]. Besides, the American-style Asian option cannot be priced by this approach easily. Related works can be found in [1, 6, 22, 23, 32, 37, 41].

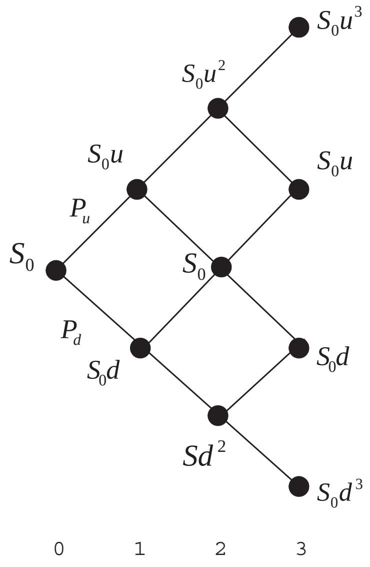


Figure 1.1: A 3-Time-Step CRR Binomial Lattice.
The initial underlying asset's price is $S_0$. Let $u$ and $d$ denote the upward and the downward multiplication factors, respectively. Let $P_u$ and $P_d$ denote the upward and the downward branching probabilities, respectively.

The Monte Carlo approach divides the time interval between the option initial date and the maturity date into several time steps. Then it simulates some (discrete-time) price paths of the underlying asset. A price path consists of a series of prices that corresponds to the underlying asset's price at each time step. The option value for each price path can be computed separately. The pricing result is obtained by averaging the option value of the simulated price paths. The major drawback of this approach is inefficiency – huge amounts of price paths should be simulated to obtain a satisfying answer. The pricing result is only probabilistic. In addition, the American-style Asian option cannot be priced by this approach easily. Related work can be found in [9, 10, 11, 28, 31, 33].

The lattice (and the related PDE) approach is more general than the first two since most methods from the first two categories suffer from the inability to price American-style Asian options without bias. Under this consideration, two proposed pricing methods in this dissertation follow the lattice approach. In the following sections, the lattice model will be introduced first. After exploring the problems of the existing lattice pricing methods, a simple intuition is given to show how the two proposed methods alleviate these problems.

## 1.4 The Lattice Approach

A lattice consists of nodes and edges connecting them. It simulates the (discrete-time) price process of the underlying asset from the option's initial date to the maturity date. Assume that an option initiates at year 0 and matures at year $T$. A lattice divides the time interval $[0, T]$ into $n$ equal time steps. Then the length of each time step is $T/n$. The price of the underlying asset at discrete time step $l$ (corresponding to year $lT/n$) can be observed on the lattice. Take the well-known Cox-Ross-Rubinstein (CRR) binomial lattice [15] as an example. A 3-time-step CRR binomial lattice is illustrated in Fig. 1.1. At each time step, the underlying asset's price $S$ can either become $Su$—the up move—with probability $P_u$ or $Sd$—the down move—with probability $P_d \equiv 1 - P_u$. Note that the price process of an asset simulated by the lattice should be guaranteed to converge to the continuous-time underlying asset's price process as $n \to \infty$. Thus the option value priced on the lattice also converges to continuous-time option value (call it the true option value) [19]. More detailed knowledge about the lattice construction is surveyed in Section 2.5.2.

## 1.5 Pricing Asian Options with the Lattice Approach and Its Problems

The difficulty with the lattice method in the case of Asian options lies in its exponential nature: Since the price of the underlying asset at each time step influences the option's payoff, it seems that $2^n$ paths have to be individually evaluated for an $n$-time-step binomial lattice (see Fig. 1.2). This makes the pricing problem intractable even with a small $n$. Many proposed approaches solve this **combinatorial explosion** by employing approximation [4, 13, 20, 26, 29, 43]. The resulting algorithms become more efficient, but most of them lack error controls [20]. Thus an efficient and accurate pricing algorithm is needed.

Some shorthand that will be used frequently later are introduced below. A pricing algorithm is said to be *exact* if no approximations are used in backward induction. Besides, a polynomial-time algorithm means that the running time is polynomial in $n$. Similar convention is adopted for exponential-time and subexponential-time algorithms.

## 1.6 The Contributions of this Dissertation

This dissertation provides two different lattice methods to address the pricing problem. First, a new trinomial lattice for pricing Asian options is proposed. The resulting exact pricing algorithm is proved to break the exponential time barrier. This algorithm is hence more efficient than any existing exact pricing algorithm. Second, an

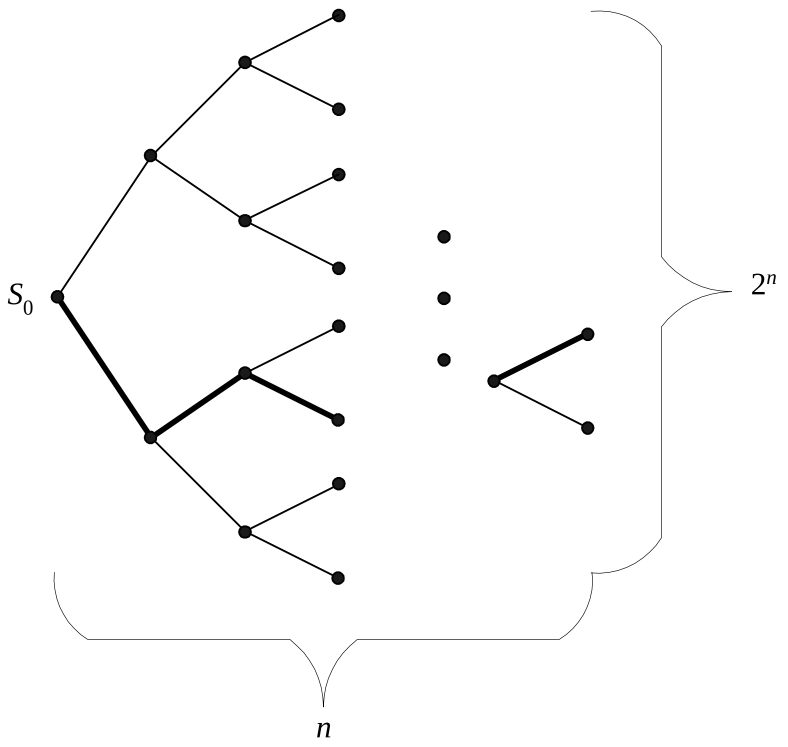


Figure 1.2: The Combinatorial Explosion.
There are $2^n$ paths (one of which follows the darkened edges in the plot) in this binomial lattice.

approximation algorithm based on the CRR lattice model is derived. This approximation algorithm computes the upper and the lower bounds (call it **range bound**) that bracket the option value of the exact pricing algorithm. When pricing European-style Asian options, this range-bound algorithm is guaranteed to converge to the true option value. Extensive experiments also reveal that the algorithm works well for American-style Asian options.

### 1.6.1 The Subexponential-Time Lattice Algorithm

We next give the intuition about how the lattice looks like and why the computational time of the resulting exact algorithm is dramatically reduced. Imagine a new lattice composed of integral asset prices. The asset price sum of any arbitrary price path on this lattice must be an integer. Thus all possible asset price sums must be integers between the maximum and the minimum asset price sums, which can be easily calculated. Take a hypothetical 3-time-step lattice in Fig. 1.3 as an example. The asset's prices are printed on the nodes. Consider the price paths that reach the shaded node with an asset price of 4. The paths with the maximum price sum and the minimum price sum are $(8, 12, 8, 4)$ and $(8, 4, 2, 4)$, respectively. The maximum and the minimum price sums are thus $8+12+8+4 = 32$ and $8+4+2+4 = 18$, respectively. The possible asset price sums at that node must be some of the 15 integers between 18 and 32, inclusively. Recall that the value of an Asian option depends on the average price of the asset. The number of possible asset's price sums at an arbitrary node $N$ equals the number of possible option values there. An exact pricing algorithm suffers from computational intractability since the number of possible asset's price sums is exponential in $n$. If it can be shown that the total number of possible asset's price sums on our new lattice is dramatically reduced, then an efficient exact algorithm can be successfully constructed. Indeed, the proposed new exact algorithm is much more efficient as it is subexponential in $n$. A simple numerical example on a 160-time step lattice helps us to realize the drastic difference: While the total number of possible asset's price sums to the middle node at the maturity date is the astronomical $8.429 \times 10^{74}$, the total number of price sums at the same node in the proposed new lattice is only 57887.

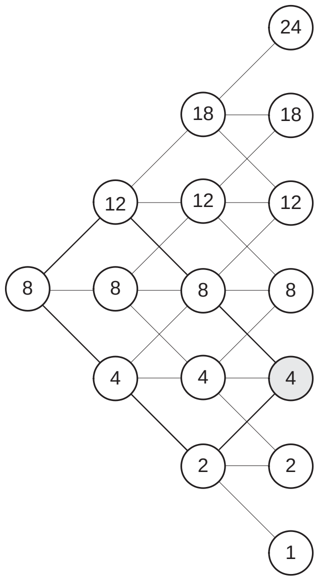


Figure 1.3: A 3-Time-Step Trinomial Lattice with Integral Asset Prices.
All paths reaching the shaded node have integral price sums. The maximum price sum at the shaded node is achieved by the upper path in thickened lines, whereas the minimum price sum at the shaded node is achieved by the lower path in thickened lines.

Although the performance of the proposed exact pricing algorithm is improved significantly, it is still not a polynomial-time algorithm. The efficiency problem remains open. The computational performance can be further improved by employing some approximations, but the accuracy problem should be considered simultaneously. The next approach is proposed to meet these requirements.

### 1.6.2 The Range-Bound Algorithms

Efficient approximation pricing algorithms that provide pricing error control by producing provable range bounds are introduced. On an $n$-time-step lattice model, a range-bound approximation algorithm can produce upper and lower bounds that bracket the option value of the exact pricing algorithm (call it the desired option value) as shown in Fig. 1.4. The desired option value becomes practically available if the upper bound and the lower bound are essentially identical. Note that the difference between the upper bound and the lower bound, call it $e$, gives an upper limit of the pricing error between exact and approximation pricing algorithms. The desired option value is known to converge to the true option value as $n \to \infty$. Thus the approximation algorithm is guaranteed to converge to the true option value if $e \to 0$ as $n \to \infty$. This relationship is illustrated in Fig. 1.5. In this dissertation, a range-bound algorithm for pricing European-style Asian options is developed, and this algorithm is proved to converge to the true option value. Extensive experiments also reveal that the algorithm works well for American-style Asian options.

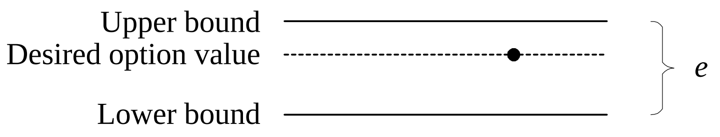


Figure 1.4: The Range-bound Algorithm.
The difference between the upper bound and the lower bound pricing results $e$ denotes the upper limit of the pricing error between exact and approximation pricing algorithms.

## 1.7 Structures of this Dissertation

This dissertation is organized as follows. Some background knowledge, including required financial knowledge, mathematical tools, and the survey on related literatures, is reviewed in Chapter 2. The first subexponential exact pricing algorithm and the lattice it based on is introduced in Chapter 3. Chapter 4 introduces the convergence approximation algorithm for pricing European-style Asian options. The improvement of the accuracy for the approximation algorithm for pricing American-style Asian options is also introduced in this chapter. Finally, Chapter 5 concludes the paper.

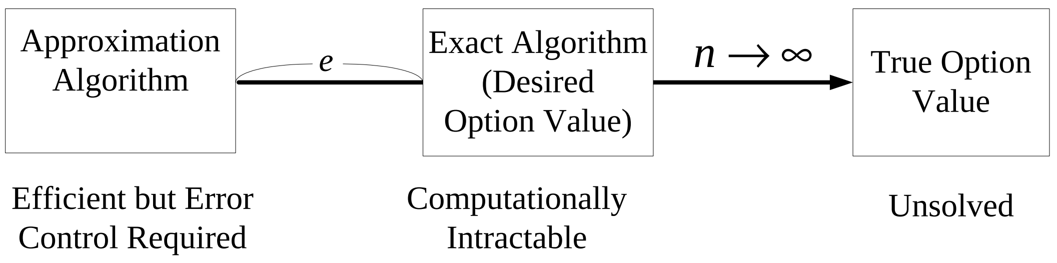

Approximation Algorithm
$e$
Exact Algorithm (Desired Option Value)
$n \to \infty$
True Option Value
Efficient but Error Control Required
Computationally Intractable
Unsolved

Figure 1.5: Convergence of the Range-Bound Algorithm.
The true option value is unsolved. An exact lattice pricing algorithm is computationally intractable. Thus an approximation pricing algorithm with good error control is attractive. The pricing error between the approximation pricing algorithm and the exact pricing algorithm is bounded by $e$. Thus the approximation algorithm is guaranteed to converge to the true option value if $e \to 0$ as $n \to \infty$.

# Chapter 2

# Preliminaries

Some required background knowledge is introduced in this chapter. Basic assumptions on financial markets and the related mathematical concepts are introduced first. A simple survey on options and their advantages is given next. The modern arbitrage-based pricing theory is then introduced to describe how the Asian option can be priced. Academic literature related to the pricing of Asian options is then introduced. These academic works are grouped into three categories: approximation analytical formulae, (quasi-) Monte Carlo simulations, and the lattice (and the related PDE) approach. The drawbacks of these academic works and the improvements on them are also discussed.

## 2.1 Basic Assumptions

Some economic assumptions that are adopted in this dissertation are illustrated as follows.

1. The mean and the volatility of the underlying asset's price, and the risk-free interest rate are assumed to be fixed constants.
2. The short selling of financial assets with full use of proceeds is permitted.
3. All assets are perfectly divisible.
4. There are no transactions costs or taxes.
5. No stocks pay dividends during the life of the option.
6. There are no risk-less arbitrage opportunities.
7. Asset trading is continuous.
8. These is no liquidity problem. That is, you can always trade at the market price.

A stochastic process is a variable that changes over time in an uncertain way. Economical variables, like the asset's price and the exchange rates, are usually modelled as stochastic processes in academic models. The randomness of these processes are usually governed by some fundamental stochastic processes like the Brownian motion. Define $\{B_t\}$ as a Brownian motion where $B_s$ denotes the process value at time $s$. Then we have the following properties [30]:

1. Normal increments: $B_t - B_s$ has normal distribution with mean 0 and variance $t - s$.

2. Independent of increments: $B_t - B_s$ is independent of the past, that is, of $B_u$, where $0 \leq u \leq s$.

3. Continuity of paths: $B_t$, $t \geq 0$ are continuous functions of $t$.

In this dissertation, a financial asset's price is assumed to follow a log-normal stochastic process. To be more specific, consider a time interval starting at year 0 and ending at year $T$. Define $S(t)$ as the price of a financial asset at year $t$. The price process follows the continuous-time diffusion process as follows:

$$S(t + dt) = S(t)\text{exp}[(r - 0.5\sigma^2)dt + \sigma dB_t], \tag{2.1}$$

where $r$ is the risk-free interest rate per annum, and $\sigma$ is the annual volatility.

The continuous log-normal stochastic price process can be approximated by a discrete-time model like the lattice model. A lattice model partitions the time between year 0 and year $T$ into $n$ equal time steps. The length of each time step $\Delta t$ is equal to $T/n$. For convenience, all the time notations are expressed in terms of the number of time steps unless stated otherwise. Let $S_i$ denote the asset's price at (discrete) time $i$. Then $S_i$ corresponds to $S(i\Delta t)$ in the continuous-time model. For the pricing purpose, the price process simulated by a lattice model is required to converge to continuous log-normal stochastic process as $n \to \infty$. The sufficient conditions to achieve this is done by calibrating the drift term $(r - 0.5\sigma^2)$ and the volatility term $(\sigma)$ of the underlying asset's price process (in Eq. 2.1) [19]. More details about the lattice constructions are introduced in Subsection 2.5.2.

## 2.2 Option Basics

Fundamental knowledge about options is introduced here. This includes the types of options, the payoff of each type of option, and how to price them.

### 2.2.1 Definitions of Options

An option is a right to buy or sell the a specific asset at (or within) a certain date with a predetermined price. This specific asset is called the underlying asset. Generally speaking, options can be classified into two groups: call options, and put options. A call option gives the holder the right to buy a financial asset with a specific price at (or within) some certain time, while a put option gives the holder the right to sell it. The price for the holder to buy or sell the asset is called the strike price (denoted as $X$). The date the option is expired is called the maturity date (denoted as $T$).

The options can also be classified based on the time in which they can be exercised. An *American-style option* can be exercised at any time up to the maturity date; while a *European-style option* can only be exercised at the maturity date. Since an American option gives all the advantages that a European option possesses plus the extra advantage of early exercise, the value of an American option is at least as great as that of a European one, other conditions being equal.

There are two sides to every option contract. On the one side is the investor who take the long position (i.e., he buys the option), while on the other side is the investor who takes the short position (i.e., he sells the option). An option holder is given the right of gaining benefit without any obligation. An option will be exercised only when it is the best choice for the holder to gain maximum benefit. Take a European-style option as an example. Recall that a European option can only be exercised at the maturity date. The payoff for the long position at the maturity date is $\max(0, S(T) - X)$ for call options; $\max(0, X - S(T))$ for put options. On the other hand, the loss for a short position in call options can be expressed as

$$-\max(0, S(T) - X) = \min(0, X - S(T)),$$

while the profit for a short position in put options is

$$-\max(0, X - S(T)) = \min(0, S(T) - X).$$

Fig. 2.1 illustrates profit/loss of a European option graphically.

The payoff for an American-style option is more complex since the option holder can exercise the option early before the maturity date. Define $\tau$ as the time the option is exercised, then the payoff to exercise an option at year $\tau$ is $\max(S(\tau) - X, 0)$ and $\max(X - S(\tau), 0)$ for call options and put options, respectively. Similarly, the loss for a short position in call options is expressed as

$$-\max(S(\tau) - X, 0) = \min(X - S(\tau), 0),$$

while the loss for a short position in put options is

$$-\max(X - S(\tau), 0) = \min(S(\tau) - X, 0).$$

An option can be priced in a discrete time model. The payoff for a European-style option at the maturity (time $n$) is

$$\text{Payoff} = \begin{cases} \max(S_n - X, 0), & \text{for a call option} \\ \max(X - S_n, 0), & \text{for a put option} \end{cases}. \tag{2.2}$$

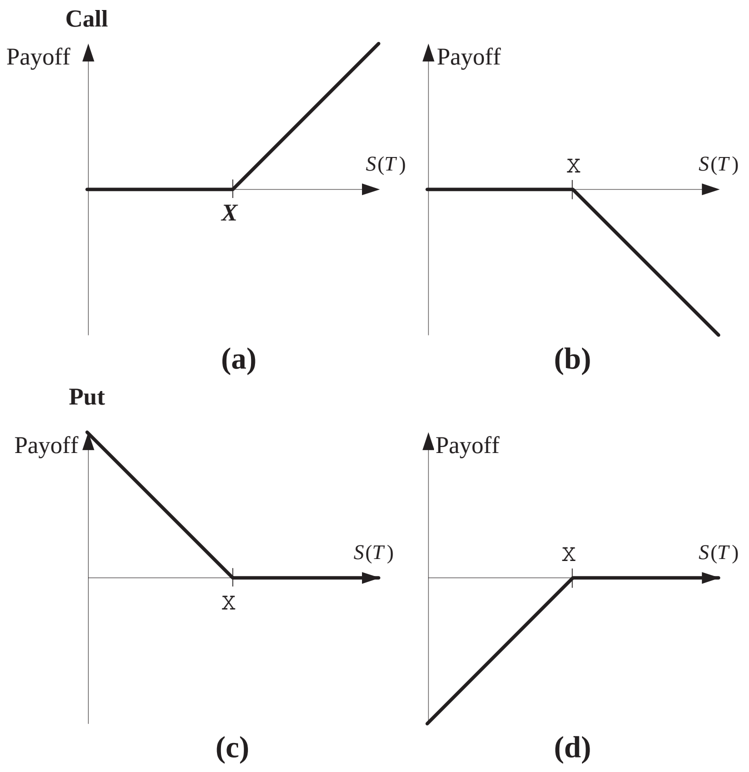


Figure 2.1: Profit/loss of Options.
(a) Long a call. (b) Short a call. (c) Long a put. (d) Short a put.

The payoff to exercise an American-style option at time $i$ $(i \leq n)$is

$$\text{Payoff} = \begin{cases} \max(S_i - X, 0), & \text{for a call option} \\ \max(X - S_i, 0), & \text{for a put option} \end{cases}, \tag{2.3}$$

where $i$ denotes the time step when the option is exercised.

### 2.2.2 Who Needs Options

Basically speaking, options attract three different types of traders: speculators, hedgers, and arbitragers. A speculator tries to take a position to gain more benefits in the market by forecasting the future. He longs (or shorts) an option if he believes it is beneficial. A hedger is the one who tries to avoid risk by buying or selling the options. A simple example is given in the next section to show how a hedger hedges the risk. An arbitrager is the one who can gain risk-less profit if the option value is not "fair." A trading strategy used to gain risk-less profits by taking advantages of mispriced options is called *arbitrage*. Theoretically, there is a fair price for each option in the market. The market price of an option should be equal to its fair price; otherwise the arbitragers can take advantage of it. How to price the option by arbitrage-free based pricing theory is described in Subsection 2.2.4.

## 2.2.3 Hedging and Hedgers

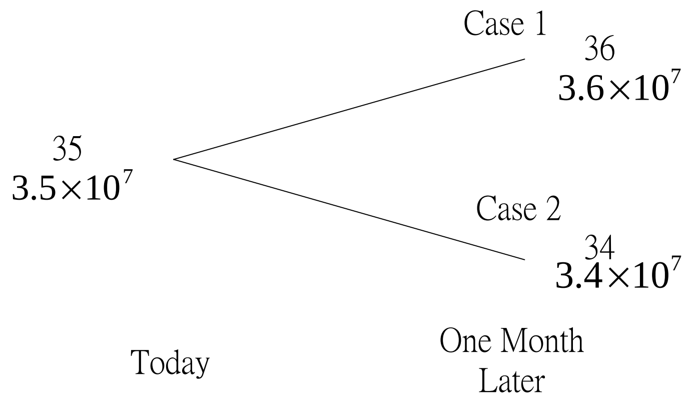


Figure 2.2: Exchange Rate Risk.
A simple one-step model illustrates the evolution of the exchange rate. The exchange rate (TWD/USD) is 35 today. The exchange rate may go up to 36 (in case 1) or go down to 34 (in case 2) one month later. The first column for each node denotes the spot exchange rate, and the second column denotes the cost to buy 1 million USDs.

A simple example on hedging the foreign exchange rate risk is given as follows. The risk-free rate is set to 0 for simplicity. Assume that there is a foreign trading company $XYZ$ in Taiwan. It is required to pay 1 million USDs next month.[1] Assume that the exchange rate (TWD/USD) today is 35. Obviously, XYZ can buy 1 million USDs with 35 million TWDs today, keep these USDs for a month, and then pay back the debt next month.

Unfortunately, XYZ may not have 35 million TWDs today. It may decide to buy 1 million USDs next month. But this will introduce foreign exchange rate risk. That is, the cost to buy 1 million USDs next month is not determined today as the exchange rate is floating. A picture to describe this risk is illustrated in Fig. 2.2. Assume that the USD may appreciate to 36 TWDs (case 1) or devalue to 34 TWDs (case 2) next month. Note that XYZ has to pay one more million TWDs ($= 3.6 \times 10^7 - 3.5 \times 10^7$) if the USD appreciates to 36 TWDs. Company XYZ might not put up with such a huge risk.

A currency call option can help XYZ to avoid such uncertainty. A currency option is an option whose underlying asset is a foreign currency. XYZ can buy a currency call on 1 million USDs. To hedge the exchange rate risk, the contract for this call is designed as follows: the maturity date for this option is one month from now, and

[1]USD here denotes the United States Dollars. TWD denotes the Taiwan Dollars.

the strike price is 35 TWDs for each USD. Now the exchange rate risk is completely hedged by this call option. If the exchange rate moves up to 36 a month later, XYZ can simply exercise the option to buy a million USDs with 35 million TWDs. On the other hand, if the exchange rate moves down to 34, XYZ can simply junk the option and buy the spot from the foreign exchange market.

Obviously, this option gives XYZ the right to buy the USDs but no obligation. XYZ should be charged by the option seller for this right. It is important to find out the fair option's price that XYZ needs to pay.

### 2.2.4 Pricing an Option with Arbitrage-Free Base Pricing Theory

In financial markets, an option buyer pays so called *option premium* to a seller at the option initial date. This premium can be viewed as the fair price of an option. The fair option's price can be priced by the arbitrage-free base pricing theory. In this subsection, I will show how the fair option price is obtained by replication first. Next, I will show that why the market price must be equal to the fair price by considering the behavior of arbitragers. Finally, a simple sketch is given to show that option can be priced by taking the expectation of future discounted payoff under the so-called risk-neutral probability.

#### Replicate an Option

An option is said to be replicated by a portfolio $A$ if $A$ can be constructed so that the future payoff of $A$ is always equal to the payoff of the option. It is intuitive that the fair price of the option should be equal to the cost of constructing portfolio $A$ since the future payoffs of $A$ and the option are the same. Take the currency call option mentioned above as an example. The replication portfolio $A$ is assumed to be consisted of $x$ units of USD and $y$ units of TWD. Assume that the USD appreciates to 36 TWDs (case 1). Since the value of the option is worth 1 million TWDs ($= \max(36-35,0)\times 10^6$) in this case, the value of $A$ ($= 36x+y$) should be also equal to 1 million TWDs. On the other hand, assume that the USD devalues to 34 TWDs (case 2). The value of the option is 0 ($= \max(34-35,0)\times 10^6$). Therefore, the value of $A$ ($= 34x+y$) is also equal to 0. By solving the two equations,

$$\begin{aligned} 36x + y &= 10^6, \\ 34x + y &= 0, \end{aligned}$$

we obtain $x = 5\times 10^5$ and $y = -1.7\times 10^7$. Thus the initial cost to construct the portfolio $A$ is equal to $5\times 10^5 \times 35 - 1.7\times 10^7 = 5\times 10^5$. The fair price of the call option is $5\times 10^5$.

### Arbitrage and Arbitragers

We have argued that the fair price of the option is equal to the cost of the replication portfolio. Moreover, we can also claim that the market price of the option should be equal to the fair price. Otherwise, the arbitragers can gain riskless profit by taking advantage of the mispriced options. Take the currency call option mentioned above as an example. Assume that the currency call's market price $V$ is larger than $5 \times 10^5$. An arbitrager can short a call and buy the portfolio $A$. He can earn $V - 5 \times 10^5 > 0$ today. At maturity (one month later), he will neither win nor lose anything since the final payoffs of the call and $A$ are equal. In other words, he can gain $V - 5 \times 10^5 > 0$ without paying anything or suffering any risk! Economists argue that such arbitrage opportunities should not exist for long since every market participant will try to gain from this "free lunch." Thus the market price of this currency call will quickly go back to the fair price $5 \times 10^5$. On the other hand, if the option value $V$ is lower than $5 \times 10^5$, an arbitrager can construct an arbitrage strategy by longing a currency call and shorting the portfolio $A$. By the same argument, the market price of the option will finally go back to the fair price $5 \times 10^5$. So we conclude that the market price of the option is equal to the cost of the replication portfolio under arbitrage-free considerations.

### Risk Neutral Valuation

Option pricing problem can be reduced to the expectation evaluation problem under the so-called *risk-neutral* probability [24]. The expected return of any security is risk-free rate under risk-neutral probability. Take the currency option mentioned in Fig. 2.2 as an example. Assume that the USD may appreciate to 36 TWDs with probability $P$ (in case 1) and devalue to 34 TWDs with probability $1 - P$ (see Fig. 2.3). Since the risk-free rate ($r$) is set to 0, the expected return of the USD should also be 0. Thus we have

$$36 \times P + 34 \times (1 - P) = 35 \times e^{r \times 1/12},$$

where $1/12$ denotes the time span of one month (in years). We have $P = 0.5$ by solving the above equation. The payoff of the option in case 1 (marked by *) is

$$\max(36 - 35, 0) \times 10^6 = 10^6,$$

and the payoff of the option in case 2 (marked by **) is

$$\max(34 - 35, 0) \times 10^6 = 0.$$

The option value is evaluated by taking expectation of the discounted future payoff under risk neutral probability as follows:

$$\frac{10^6 \times P + 0 \times (1 - P)}{e^{r \times 1/12}} = 10^5.$$

Note that the option value evaluated by using replication is the same as the value evaluated by taking expectation, but the latter method is simpler.

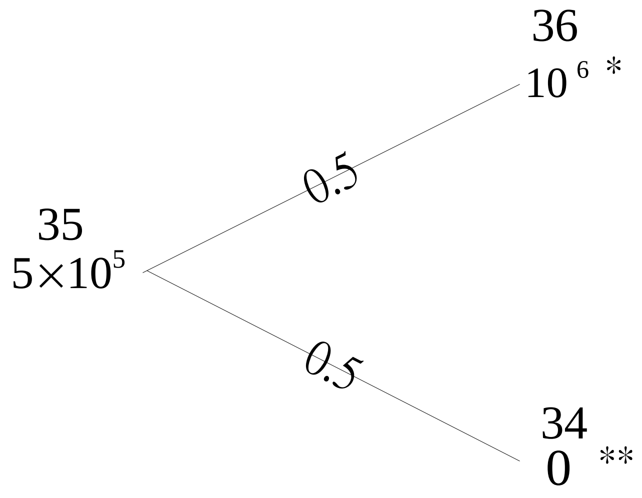


Figure 2.3: Pricing the Currency Option by Taking Expectation.
The (risk-neutral) probability for each case is marked directly on the branch. The first column of each node denotes the spot exchange rate and the second column denotes the option value at each node.

The above result can be formalized as follows: In a continuous time model, the value of a European-style option can be expressed as

$$\text{Option Value} = \begin{cases} e^{-rT}E[\max(S(T) - X, 0)], & \text{for a call option} \\ e^{-rT}E[\max(X - S(T), 0)], & \text{for a put option} \end{cases}. \tag{2.4}$$

The value of an American-style option can be expressed as

$$\text{Option Value} = \begin{cases} E[e^{-r\tau}\max(S(\tau) - X, 0)], & \text{for a call option} \\ E[e^{-r\tau}\max(X - S(\tau), 0)], & \text{for a put option} \end{cases},$$

where $\tau$ denotes the time when the option is optimally exercised. In a discrete time model, the European-style option's value is obtained by changing $S(T)$ in Eq. (2.4) into $S_n$. The American-style option's value is

$$\text{Option Value} = \begin{cases} E[e^{-ri\Delta t}\max(S_i - X, 0)], & \text{for a call option} \\ E[e^{-ri\Delta t}\max(X - S_i, 0)], & \text{for a put option} \end{cases},$$

where $i$ denotes the time step when the option is exercised.

## 2.3 Asian Options

### 2.3.1 Definitions

An Asian option is an option whose payoff depends on the average price of the underlying asset during a specific period. Define the average price of the underlying asset from year 0 to year $t$ as

$$A(t) \equiv \frac{\int_0^t S(u)\,du}{t}. \tag{2.5}$$

Then the payoff for a European-style Asian option at the maturity date is

$$\text{Payoff} = \begin{cases} \max(A(T) - X, 0), & \text{for a call option} \\ \max(X - A(T), 0), & \text{for a put option} \end{cases}.$$

The payoff for exercising an American-style Asian option at year $\tau$ is

$$\text{Payoff} = \begin{cases} \max(A(\tau) - X, 0), & \text{for a call option} \\ \max(X - A(\tau), 0), & \text{for a put option} \end{cases}.$$

In a discrete time model, the average price of the underlying asset is redefined as

$$A_j \equiv \frac{\sum_{i=0}^{j} S_i}{j+1}. \tag{2.6}$$

Thus the payoff for a European-style Asian option is

$$\text{Payoff} = \begin{cases} \max(A_n - X, 0), & \text{for a call option} \\ \max(X - A_n, 0), & \text{for a put option} \end{cases}. \tag{2.7}$$

The payoff for exercising an American-style Asian option at time $i$ is

$$\text{Payoff} = \begin{cases} \max(A_i - X, 0), & \text{for a call option} \\ \max(X - A_i, 0), & \text{for a put option} \end{cases}. \tag{2.8}$$

### 2.3.2 Pricing Asian Option by Applying Risk Neutral Variation

The value of an Asian option can be evaluated by taking expectation of the future discounted payoff as we do in Eq. (2.4). For a European-style Asian option, the option value is

$$\text{Option Value} = \begin{cases} e^{-rT}E[\max(A(T) - X, 0)], & \text{for a call option} \\ e^{-rT}E[\max(X - A(T), 0)], & \text{for a put option} \end{cases}.$$

The option value for an American-style Asian option is

$$\text{Option Value} = \begin{cases} E[e^{-r\tau}\max(A(\tau) - X, 0)], & \text{for a call option} \\ E[r^{-r\tau}\max(X - A(\tau), 0)], & \text{for a put option} \end{cases}.$$

In a discrete time model, the value of a European-style Asian option is evaluated as

$$\text{Option Value} = \begin{cases} e^{-rT}E[\max(A_n - X, 0)], & \text{for a call option} \\ e^{-rT}E[\max(X - A_n, 0)], & \text{for a put option} \end{cases}, \tag{2.9}$$

while the value for an American-style Asian option is

$$\text{Option Value} = \begin{cases} E[e^{-ri\Delta t}\max(A_i - X, 0)], & \text{for a call option} \\ E[r^{-ri\Delta t}\max(X - A_i, 0)], & \text{for a put option} \end{cases}. \tag{2.10}$$

### 2.3.3 Advantages of Asian Options

Since the payoff of an Asian option depends on the average price of the underlying asset, it is useful for hedging transactions whose cost is related to the average price of the underlying asset. Take the foreign exchange rate risk case discussed above as an example. Assume that the company XYZ needs to pay $10^6$ USDs per month for the next six months. XYZ may buy six currency call options maturing at the next month, two months later, . . ., and six months later, respectively. On the other hand, XYZ may hedge the exchange rate risk by buying six Asian call options that matures six month later. It can be observed in the markets that an Asian option is usually much cheaper than an otherwise identical ordinary option. Thus XYZ can reduce the hedge cost significantly by buying six Asian call options instead of six ordinary call options.

Besides, the price of an Asian option is also less subject to price manipulation due to the following reasons: The payoff of a European-style ordinary option is determined by the underlying asset's value at the maturity date, while the payoff of a European-style Asian option is determined by the average price of the underlying asset between the option initial date and the maturity date. And it is easier to control an asset's price at a specific time point than to manipulate the whole price path. Preventing price manipulation is an attractive property of an option especially in thinly-traded markets.

In practice, as varieties of asian option are widely traded in today's financial markets, how to price them accurately and efficiently is important in both financial and academic fields.

## 2.4 Review of Literature

The major problem in pricing Asian option is that we do not know much about the distribution of the underlying asset's average price $A(T)$ (see Eq. (2.1) and (2.5)). $A(T)$ can be viewed as the sum of log-normal random variables; and the density function of a sum of log normal random variables is currently unavailable. That is why there is no simple and exact closed-form solutions for pricing Asian options. Approximation methods suggested in the academic literature can be grouped into

| $r$ | $\sigma$ | $T$ | $S_0$ | GE | Shaw | Euler | PW | TW | MC10 | MC100 | SE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5.0% | 50% | 1 | 1.9 | .195 | .193 | .194 | .194 | .195 | .192 | .196 | .004 |
| 5.0% | 50% | 1 | 2.0 | .248 | .246 | .247 | .247 | .250 | .245 | .249 | .004 |
| 5.0% | 50% | 1 | 2.1 | .308 | .306 | .307 | .307 | .311 | .305 | .309 | .005 |
| 2.0% | 10% | 1 | 2.0 | .058 | .520 | .056 | .0624 | .0568 | .0559 | .0565 | .0008 |
| 18.0% | 30% | 1 | 2.0 | .227 | .217 | .219 | .219 | .220 | .219 | .220 | .003 |
| 12.5% | 25% | 2 | 2.0 | .172 | .172 | .172 | .172 | .173 | .173 | .172 | .003 |
| 5.0% | 50% | 2 | 2.0 | .351 | .350 | .352 | .352 | .359 | .351 | .348 | .007 |

Table 2.1: Stress Tests.
The exercise price $X$ is 2.0. The approximation methods for comparison are from Geman and Eydeland [22] (GE), Shaw [39], Euler, Post-Widder method (PW) [1], and Turnbull-Wakeman [41] (TW). The benchmark values (MC10 and MC100) and the approximation values are from [21]. MC10 uses 10 time steps per day, whereas MC100 uses 100. Both are based on 100,000 trials. SE stands for standard error, also from [21].

three different categories: approximation analytical formulae, (quasi-)Monte Carlo simulations, and the lattice (and the related PDE) approach.

### 2.4.1 Approximation Analytical Formulae

This approach denotes that the value of the option is approximated by (semi-)closed form formulae. Some related academic works in this category try to approximate the probability density function of $A$. Turnbull and Wakeman [41] and Levy [32] try to approximate the density function of $A$ by Edgeworth series expansion. Milevsky and Posner [37] approximate it by the reciprocal gamma distribution. Carverhill and Clewlow [12] and Benhamou [6] use Fourier transform to approximate the payoff function at maturity. Geman and Yor [23] derive an analytical expression for the Laplace transform of the continuous Asian calls, and numerical inversion of this transform is considered by Geman and Eydeland [22] and Shaw [39]. Some inversion algorithms based on the Euler and Post-Widder methods are suggested in Abate and Whitt [1]. The forward starting Asian options are approximated with Taylor's series expansion as in [8, 40]. Zhang [42] approximates the option value by combining an analytical closed form with a numerical adjustment (computed by finite difference method). The major problem of the approximation analytical formulae approach is that most suggested methods from this approach lack error control [21]. Table 2.1 illustrates that some well-known approximation methods fail in extreme cases. Besides, the American-style Asian option's value cannot be approximated by this approach easily.

### 2.4.2 Monte Carlo Simulation

The Monte Carlo simulation approach denotes a pricing procedure that values a derivative by randomly sampling changes in economic variables. To value a European-style Asian option, a typical Monte Carlo simulation can divide the time interval between the option initial date and the maturity date into $n$ time steps. Then it simulate the price path of the underlying asset by the following formula:

$$S_i = S_{i-1}exp[(r - 1/2\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\omega],$$

where $S_i$ denotes the price of the underlying asset at time $i$, $\Delta t$ is equal to $T/n$, and $\omega$ denotes the a standard normal random variable. Note that the distribution of this $n+1$-dimensional random price vector $(S_0, S_1, S_2, \cdots, S_n)$ is the same as the distribution of random vector $(S(0), S(\Delta t), S(2\Delta t), \cdots, S(n\Delta t))$ which is governed by Eq. (2.1). The average price of each price path is computed by Eq. (2.6). Thus the payoff for each price path for the European-style Asian option can be computed by Eq. (2.7). The output option value is obtained by averaging the discounted payoffs.

The Monte Carlo simulation approach suffers from the following problems:

1. The pricing result is only probabilistic.
2. The number of simulated price paths should be large enough to obtain satisfactory pricing results. Thus the algorithm is not efficient enough.
3. The pricing results are significantly influenced by the random sources used to obtain the random variable $\omega$. Biased results are produced if the random sources are unreliable.
4. The American-style option can not be handled by this approach easily.

Some related works that address the first three problems are listed below: Boyle, Broadie, and Glasserman [9], Broadie and Glasserman [10], Broadie, Glasserman, and Kou [11], Kemna and Vorst [28], and Lapeyre and Temam [31]. Briefly speaking, they try to reduce the variance of the pricing results and refine the quality of the random variable ($\omega$) they use. Four simple methods used by them are

- Antithetic variates:
  Assume that the normal random variables $\hat{\omega}_1, \hat{\omega}_2, \cdots, \hat{\omega}_n$ are sampled. A simulated price path $(\hat{S}_0, \hat{S}_1, \ldots, \hat{S}_n)$ is constructed by defining $\hat{S}_i$ as

  $$\hat{S}_i = \hat{S}_0 e^{(r-\sigma^2/2)i\Delta t + \sigma\sqrt{\Delta t}(\hat{\omega}_1+\hat{\omega}_2+\ldots\hat{\omega}_i)}.$$

  A dual price path $(\hat{S'}_0, \hat{S'}_0, \ldots, \hat{S'}_n)$ can be constructed by defining $\hat{S'}(i)$ as

  $$\hat{S'}(i) = \hat{S}_0 e^{(r-\sigma^2/2)i\Delta t + \sigma\sqrt{\Delta t}(-\hat{\omega}_1-\hat{\omega}_2-\ldots-\hat{\omega}_i)}$$

  .

- Moment matching:
  This is done by tuning the sampled random variables so that the first few moments of the tuned sampled random variables match the moments of the distribution where the random variables are sampled from. For example, if we sample $l$ random variables $(\omega_1, \omega_2, \cdots, \omega_l)$ from a normal distribution with mean $\mu$ and standard deviation $\sigma$. The sample mean and standard error of these sampled random variables are $\mu_l$ and $\sigma_l$, respectively. Then the tuned variable $\omega_i'$ is
  $$\omega_i' \equiv (\omega_i - \mu_l)\frac{\sigma}{\sigma_l} + \mu.$$

- Latin hypercube sampling:
  This is a stratified sampling method that forces the cumulative probabilities of the empirical distribution (determined by the sampled random variables) to match the cumulative probabilities of the theoretical distribution (where the variables are sampled from). For example, assume that $\omega_1, \omega_2, \ldots, \omega_{100}$ are sampled from a normal distribution, then each observation $\omega_i$ is forced to lie between $(i-1)$th and $i$th percentile.

- Control variates:
  This method is widely applicable to reduce the variance of the output results. This is done by replacing the evaluation of an unknown expectation with the evaluation of the difference between the unknown quantity and another expectation whose value is known [9]. This is used by Kemna and Vorst [28] for pricing Asian options.

For pricing American-style Asian options, Longstaff and Schwartz (2001) develop a least-squares Monte Carlo approach to tackle the problem.

## 2.5 Lattice and Related PDE Approach

A lattice is a discrete time representation of the evolution of the underlying asset's price. It divides a certain time interval, like the interval between the option initial date (year 0) and the maturity date (year $T$), into $n$ equal time steps. The length of each time step $\Delta t$ is equal to $T/n$. It approximates the distribution of the underlying asset's price at each time step. A lattice consists of nodes and branches connect them. Each node at time $i$ can be viewed as a possible asset's price at time $i$. Each branch that connects two nodes located at adjoint time steps denotes a possible evolution of the underlying asset's price. The well-known CRR binomial lattice will be introduced next as an example to show what a lattice looks like, how it is constructed, and how an option is priced with the lattice model [15].

### 2.5.1 The Structure of CRR Binomial Lattice

A 2-time-step CRR binomial lattice is illustrated in Fig. 2.4(a). Recall that $S_i$ denotes the value of the underlying asset at time $i$. $S_{i+1}$ equals $S_i u$ with probability $P_u$ and $S_i d$ with probability $P_d (\equiv 1 - P_u)$, where $d < u$. $u$ is equal to $e^{\sigma\sqrt{\Delta t}}$, where $\sigma$ denotes the annual volatility of the underlying asset's price (see Eq. (2.1)). The identity

$$ud = 1 \tag{2.11}$$

holds in this lattice model. The probability $P_u$ for an up move is set to $(e^{r\Delta t} - d)/(u - d)$. Both $d \leq e^{r\Delta t} \leq u$ and $0 < P_u < 1$ must hold to avoid arbitrage. The asset's price at time $i$ that results from $j$ down moves and $i - j$ up moves therefore equals $S_0 u^{i-j} d^j$ with probability $\binom{i}{j} P_u^{i-j} P_d^j$.

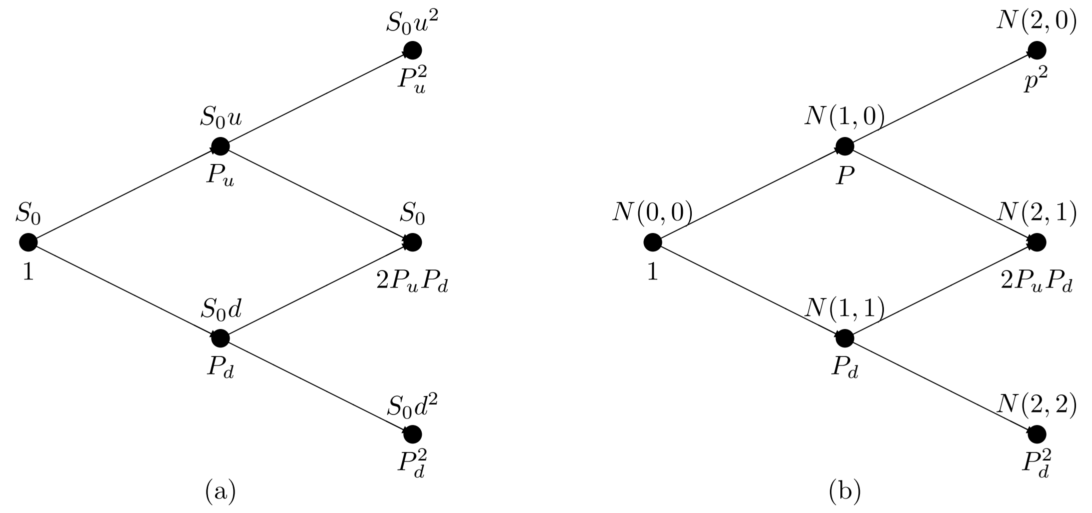


Figure 2.4: The CRR Binomial Lattice.
A 2-time-step CRR binomial lattice model are illustrated in (a) and (b). The price of the underlying asset for each node is illustrated in (a) and the alias ($N(*,*)$) for each node is illustrated in (b). $N(i,j)$ stands for the node at time $i$ with $j$ cumulative down moves. The probability of reaching each node is listed under the node.

We now map the asset's prices to nodes on the CRR binomial lattice used for pricing. Node $N(i,j)$ stands for the node at time $i$ with $j$ cumulative down moves. Its associated asset's price is hence $S_0 u^{i-j} d^j$. The asset's price can move from $N(i,j)$ to $N(i+1,j)$ with probability $P_u$ and to $N(i+1,j+1)$ with probability $P_d$. As a consequence, node $N(i,j)$ can be reached from the root with probability $\binom{i}{j} P_u^{i-j} P_d^j$. See Fig. 2.4(b) for illustration.

## 2.5.2 How to Construct a Lattice

We use the well-known Cox-Ross-Rubinstein (CRR) binomial lattice to illustrate how a lattice is constructed in principle. The logarithmic asset's price mean ($\mu$) and variance (Var) one time step from now are derived from Eq. (2.1) as

$$\begin{aligned} \mu &\equiv (r - 0.5\sigma^2)\Delta t, && (2.12)\\ \text{Var} &\equiv \sigma^2 \Delta t. && (2.13) \end{aligned}$$

To make sure that the lattice converges to the continuous-time asset's price process, the mean and the variance of the logarithmic price process should be calibrated by matching those of the lattice and those of the continuous-time model:

$$\begin{aligned} P_u \ln u + P_d \ln d &= \mu, && (2.14)\\ P_u(\ln u - \mu)^2 + P_d(\ln d - \mu)^2 &= \text{Var}. && (2.15) \end{aligned}$$

Note that

$$P_u + P_d = 1. \tag{2.16}$$

The 4 parameters ($P_u$, $P_d$, $u$, and $d$) are uniquely obtained by solving Eqs. (2.11), (2.14)–(2.16). The branching probabilities $P_u$ and $P_d$ should be between 0 and 1 to meet the no-arbitrage requirements. In the CRR lattice, this demand can always be met by suitably increasing $n$ [35].

### Construction of a Multinomial Lattice

If each node in a lattice can branch to $\ell$ nodes at the next time step, we call it an $\ell$-nomial lattice. The above idea can be applied to construct an $\ell$-nomial lattice. Note that $2\ell$ degrees of freedoms are provided by an $\ell$-nomial lattice. They include $\ell$ price multiplicative factors (like $u$ and $d$ in the CRR binomial lattice) and $\ell$ branching probabilities (like $P_u$ and $P_d$ in the CRR binomial lattice). These branching probabilities must be between 0 and 1 to meet the no-arbitrage requirements. We need $2\ell$ independent equations to determine these $2\ell$ variables uniquely. The calibration of mean and variance gives 2 equations. The branching probabilities sum to 1, giving another one. Additional $2\ell - 3$ equations must be added. For example, Eq. (2.11) is the additional equation used in the CRR binomial model.

Generally speaking, most of these extra equations are enforced to meet specific requirements. The number of these extra equations determines the lattice model that will be constructed. In Chapter 3, a trinomial lattice ($\ell = 3$) will be constructed to price an Asian option.

## 2.5.3 Pricing an Ordinary Option with the Lattice Approach

Options can be priced by the backward induction method on a lattice. Let us focus on pricing a European-style call on the CRR binomial lattice first. Recall that option value can be obtained by taking expectation of the future discounted payoffs as in Eq. (2.9), which can be decomposed into a combination of numerous formulae as follows:

$$\mathrm{C} = e^{-r\Delta t}(P_u C_u + P_d C_d), \tag{2.17}$$

where $C$ denotes the option value of an arbitrary node $N$, $C_u$ denotes the option value of the node that can be reached from $N$ by a upward movement, and $C_d$ denotes the option value of the node that can be reached from $N$ by a downward movement. The option value for each node at the maturity date (at time $n$) is defined by Eq. (2.2). The option value for each node at time $i$ $(0 \leq i \leq n-1)$ is evaluated by substituting the option values of the nodes at time $i+1$ into the right hand side of Eq. (2.17). The pricing result is the option value at the root node.

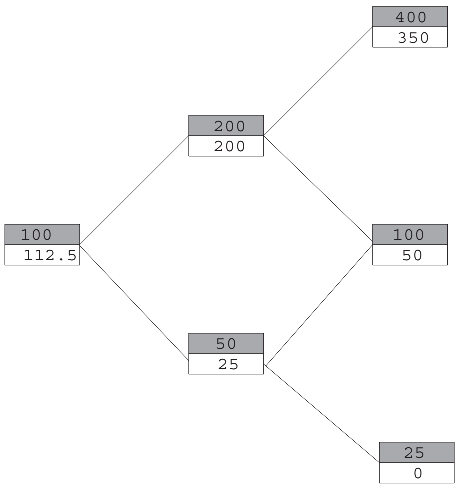


Figure 2.5: Pricing an Ordinary Option on a CRR Lattice.
The upper cell of each node (colored of gray) denotes the price of the underlying asset, and the lower cell of each node denotes the option value at that node.

Take a simple 2-time-step CRR binomial lattice model illustrated in Fig. 2.5 as an example. The upward factor $u$ and the downward factor $d$ are 2 and 0.5, respectively. The upward branching probability $P_u$ and the downward branching probability $P_d$ are both 0.5. The risk-free rate is set as 0, and the strike price is 50. The upper cell of each node (colored of gray) denotes the price of the underlying asset, and the lower cell of each node denotes the option value at that node. The option value of a node at time 2 (the maturity date) is computed by Eq. (2.2). For example, the option value for the uppermost node at time 2 is

$$\max(400 - 50, 0) = 350.$$

For each node at time 0 or time 1 can be evaluated by applying Eq. (2.17). For example, the option value for the upper node at time 1 is

$$e^0(0.5 \times 350 + 0.5 \times 50) = 200.$$

The final pricing result is 112.5.

An American-style option can be exercised before the maturity date. An option will be exercised early if the option holder finds that it is more beneficial to exercise the option than to hold the option. The lattice approach can handle this by modifying Eq. (2.17) as follows:

$$\mathrm{C} = \max(\text{Exercise value}, e^{-r\Delta t}(P_u C_u + P_d C_d)), \tag{2.18}$$

where the "Exercise value" denotes the payoff to exercise the option immediately (see Eq. (2.3)).

In the above method, the option value of each node is evaluated once by applying Eq. (2.2) (for the nodes at time $n$), Eq. (2.17) (for pricing European-style options), or Eq. (2.18) (for pricing American-style option). Since the above mentioned equations can be evaluated in constant time and there are $O(n^2)$ nodes in the lattice, the computational time complexity is $O(n^2)$.

### 2.5.4 The PDE Approach

The value of the option may be represented as the solution of a PDE. When the PDE cannot be solved analytically, the solution can be approximated by the finite difference method numerically. A finite difference method places a grid of points on the space over which the desired function takes value and then approximates the function value at each of these points. The approximation method is similar to the backward induction used in the lattice approach.

## 2.6 Pricing the Asian option with Lattice Approach and Its Problems

#### 2.6.0.1 The Lattice Approach and and the Inefficiency Problem

The Asian option can be priced by the lattice approach. However, the pricing algorithm is much more complex as the value of the Asian option is influenced by the historical average price of the underlying asset (see Eq. (2.7)). For most nodes, there is more than one possible option value at a node since there is more than one price paths reaching this node and most of these price paths carry distinct historical average prices. For convenience, some terms are introduced as follows: A *path prefix* is defined as a partial price path $(S_0, S_1, \ldots, S_j)$ that starts at time 0 and ends at time $j$. The sum of this *path prefix* is defined by $\sum_{i=0}^{j} S_i$. We call it a *prefix sum.* For an arbitrary node $N$ illustrated in Fig. 2.6, we can find a path prefix that has the maximum prefix sum among the path prefixs that end at $N$. This maximum path prefix is denoted by the upper path (in thick lines) that ends at $N$. Similarly, the path prefix that has the minimum prefix sum is denoted by the lower path (in thick

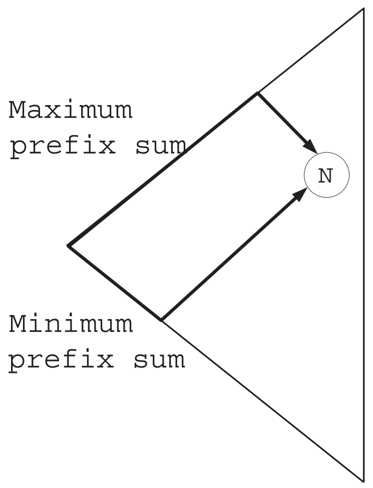


Figure 2.6: Prefix Sums and the Prefix-Sum Range.

lines) that ends at $N$. The prefix sum range for $N$ is defined as the range between the maximum and the minimum prefix sums for $N$.

A 3-time-step CRR lattice is illustrated in Fig. 2.7 to show how an Asian option is priced on a lattice. We focus on the European-style call option first. The settings of the multiplication factors and the branching probabilities are the same as in Fig. 2.5 except that there may be more than one option value at each node. The uppermost cell for each node (colored of gray) denotes the underlying asset's price for that node. The prefix sum for each path prefix is marked directly on the branch. The option value for a path prefix is denoted in the cell connected by the branch that is marked by the prefix sum of the path prefix. For example, the option value for a path prefix $(100, 200, 100)$ is 81.25.

The option value for each path prefix that reaches the maturity date (time 3) can be computed by Eq. (2.7). For example, the payoff for the path prefix $(100, 50, 25, 12.5)$ is $\max(\frac{100+50+25+12.5}{4}, 0) = 0$. Define the option value of a path prefix $(S_0, S_1, \cdots, S_i)$ as $V(i, S_i, M)$, where $M = \sum_{j=0}^{i} S_j$. Assume that the underlying asset's price can either move up to $S_{i+1}^+$ or move down to $S_{i+1}^-$. Then the value of $V(i, S_i, M)$ can be expressed as

$$V(i, S_i, M) = e^{-r\Delta t}\left(P_u V(i+1, S_{i+1}^+, M + S_{i+1}^+) + P_d V(i+1, S_{i+1}^-, M + S_{i+1}^-)\right). \tag{2.19}$$

For example, the option value of the path prefix $(100, 200, 100)$ ($\equiv V(2, 100, 400)$) is

$$V(2, 100, 400) = e^0(0.5 \times V(3, 200, 600) + 0.5 \times V(3, 50, 450)) = 100 + 62.5 = 81.25.$$

When pricing American-style option, the backward induction formula is

$$V(i, S_i, M) = \max(\mathrm{E}, e^{-r\Delta t}\left(P_u V(i+1, S_{i+1}^+, M + S_{i+1}^+) + P_d V(i+1, S_{i+1}^-, M + S_{i+1}^-)\right)), \tag{2.20}$$

where "E" denotes the value to exercise the option immediately (see Eq. (2.8)).

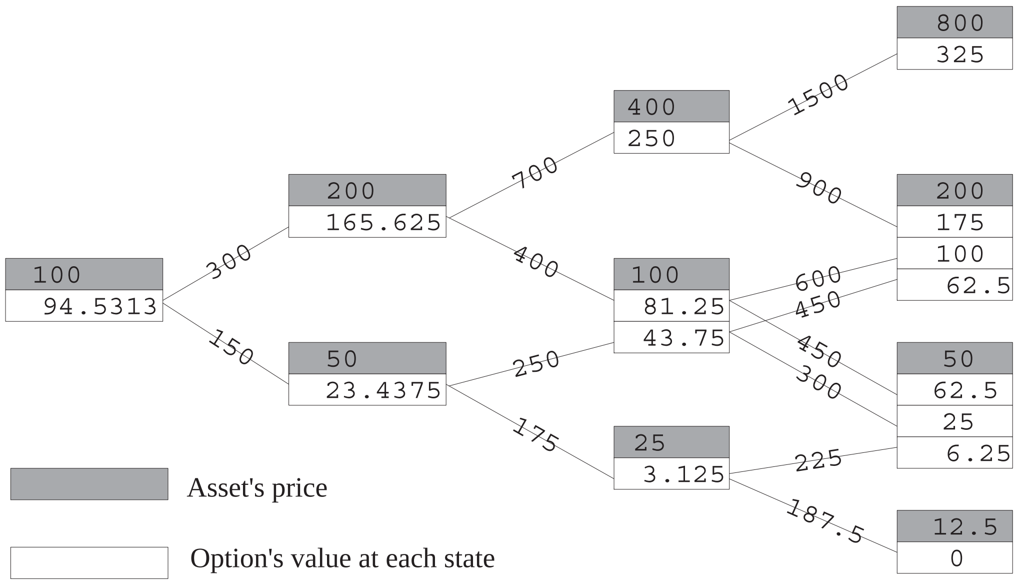


Figure 2.7: Pricing an Asian Call Option on a CRR Lattice.
The uppermost cell for each node (colored of gray) denotes the underlying asset's price at that node. The other cell(s) denote(s) possible option value(s) at that node.

The option value for each path prefix can be evaluated in constant time. Unfortunately, the number of path prefix in a $n$-time step lattice is

$$2^0 + 2^1 + 2^2 + \cdots + 2^n = 2^{n+1} - 1.$$

The time complexity is exponential in $n$. Although some of the path prefixes may have the same option value, no one has reduced the time complexity significantly without approximation.

### 2.6.1 Approximation Algorithms and Its Problems

A successful approximation paradigm is suggested by Hull and White [26], and this paradigm is widely incorporated by other approximation pricing algorithms. We call this paradigm the **Hull-White paradigm**. This paradigm limits the number of possible prefix sums at each node to a manageable magnitude $k$. The option values for the missing prefix sums are estimated by interpolation. Take Fig. 2.8 as an example. There are three nodes in the figure. The leftmost node is at time $i$, and the two other nodes are at time $i + 1$. Max and Min denote the maximum and the minimum prefix sums for each node, respectively. Each line segment at node $N$ denotes the one of the possible option value at $N$. The corresponding prefix sum for each possible option value is marked in the right (or left) of the line segment. The prices of the underlying asset of the upper-right and the lower-right nodes are 4 and 2, respectively.

To solve the option value whose corresponding prefix sum is 20 at the leftmost node, two different option values from the next time step— one with corresponding prefix sum 24 (= 20 + 4) from the upper-right node ($V(i+1, 4, 24)$) and the other one with corresponding prefix sum 22 (= 20 + 2) from the lower-right node ($V(i+1, 2, 22)$)— are required. However, these two required prefix sums are missing since the Hull-White paradigm limits the number of possible prefix sums. These two option values can be estimated by using interpolation. For example, $V(i+1, 4, 24)$ can be estimated by $V(i+1, 4, 25)$ and $V(i+1, 4, 23)$. $V(i+1, 2, 22)$ can be estimated by $V(i+1, 2, 23)$ and $V(i+1, 4, 21)$. The time complexity of this approximation algorithm is $O(n^2k)$, which is much more efficient than the original exact pricing algorithm.

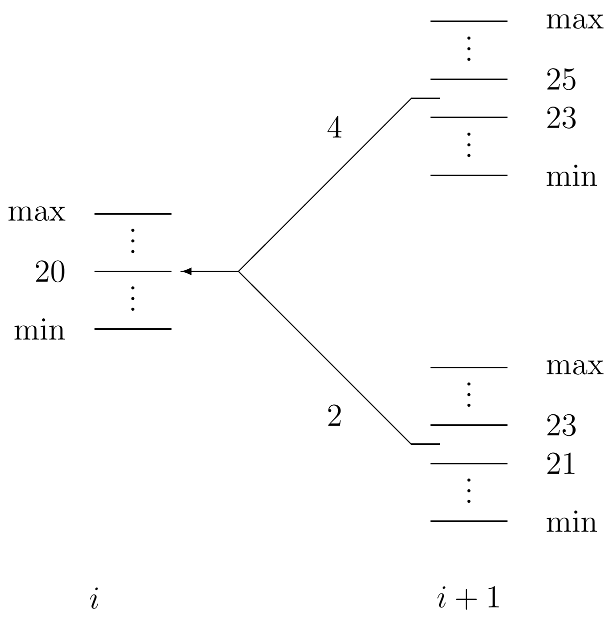


Figure 2.8: The Approximation in Hull-White Paradigm.

A problem of the Hull-White paradigm is that it lacks error control. Since the interpolation errors are introduced and accumulated at each time step, we are no longer sure whether the pricing results converge to the desired option value computed by the exact pricing algorithm as $n \to \infty$. The relationship is listed in Fig. 1.4, where $e$ denotes the maximum error between the approximation algorithm and the exact pricing algorithm. Since more interpolation errors are introduced as $n$ increases, the error $e$ does not necessarily converge to 0. Thus the approximation pricing algorithm does not necessarily converge to the true option value (see Fig. 1.5.)

The Hull-White paradigm has been analyzed and extended [4, 20, 29]. Barraquand and Pudet apply similar idea to the PDE approach [4]. Klassen assumes that the pricing errors of the approximation algorithm converge in $O(1/n)$ and apply extrapolation to eliminate the error [29]. Forsyth et al. argue that an improper interpolation scheme results in divergence or incorrect convergence [20]. They also prove that the modified Hull-White approximation algorithm converges under some ad hoc assumptions when $k$ and $n$ are related in a certain manner. The pricing error control problem is not completely solved in above mentioned papers.

# Chapter 3

# The Integral Trinomial Lattice

The difficulty for pricing the Asian options with the lattice approach lies in its computational intractability: The time complexity of any existing exact pricing algorithm is exponential in the number ($n$) of time steps of the lattice. This chapter suggests a new lattice model to tackle this problem. The resulting exact pricing algorithm is proved to break the exponential-time barrier [18].

## 3.1 A Simple Intuition

Existing exact pricing algorithms for Asian options are computationally intractable since the total number of prefix sums for any existing lattice grows exponentially in $n$. To solve this problem without approximations, a new lattice with fewer prefix sums is proposed. Imagine a new lattice composed of integral asset prices. A hypothetical example is illustrated in Fig. 1.3. In such a integral lattice, all possible prefix sums are restricted to be integers. A simple calculation shows us that there might be fewer prefix sums in an integral lattice.

A rough analysis is done by comparing the number of prefix sums at the last time step (time $n$) between a integral lattice and a usual binomial lattice. There are $2^n$ possible prefix sums at time $n$ in a usual binomial lattice. Estimating the number of prefix sums at time $n$ in an integral lattice is heuristic. Assume that the maximum prefix sum at time $n$ of an integral lattice is $\alpha$. Thus any possible prefix sum must be some integer between 1 and $\alpha$. There must be $O(n\alpha)$ possible prefix sums at time $n$ as there are $O(n)$ nodes at time $n$. $\alpha$ can be roughly estimated by the maximum prefix sum of the CRR binomial lattice as follows:

$$\begin{aligned}
\alpha &\approx S_0 + S_0 u + S_0 u^2 + \ldots + S_0 u^n \\
&= S_0[1 + u + u^2 + \ldots + u^n] = S_0 \frac{u^{n+1} - 1}{u - 1} \\
&\approx S_0 \frac{e^{\sigma\sqrt{Tn}} - 1}{e^{\sigma\sqrt{T/n}} - 1}
\end{aligned}$$

The growth rate (in $n$) of $\ln n\alpha$ and $\ln 2^n$ are compared in Fig. 3.1. It is obvious that $n\alpha$ grows much slower than $2^n$. This simple estimation points out that an integral lattice might have fewer prefix sums than a usual binomial lattice.

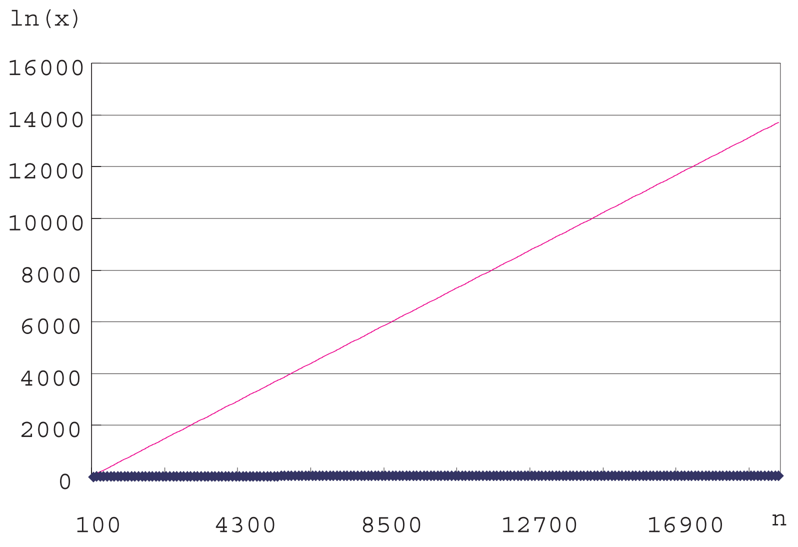
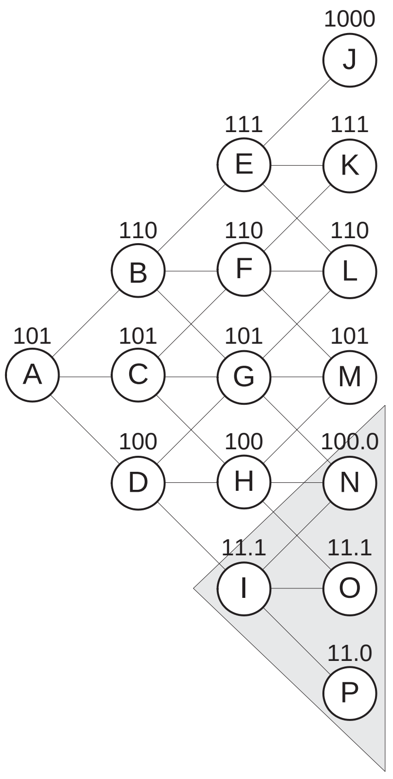


Figure 3.1: Growth Rate Comparison.
The growth rate of $\ln n\alpha$ (in thick line) and $\ln 2^n$ (in thin line) are compared in this figure.

## 3.2 The Multiresolution Lattice Model

Constructing an lattice of integral asset prices is first studied by Dai and Lyuu [16]. A new lattice model, the multiresolution lattice, is thus constructed. In the multiresolution lattice, the asset's price for each node is restricted to be a rational number of finite precision. Thus each possible prefix sum in this lattice is also a rational number of finite precision. The precision of the asset's price varies as necessary so the lattice can be successfully constructed. A 3-time-step multiresolution lattice is illustrated in Fig. 3.2. Part (a) denotes the structure of the multiresolution lattice. The underlying asset's price for each node is marked above the node (in binary form). Part (b) denotes the branching probabilities for each node in the lattice. Note that the underlying asset's price for nodes $I$, $O$, $P$ are rational numbers with one bit precision. Thus the possible prefix sums of the nodes covered by the shaded triangle ($I$, $N$, $O$, $P$) are also rational number of one bit precision while the prefix sums of other nodes are integers. It can be viewed that the resolution (in terms of bits precision) varies in the lattice. That is why this lattice is called multiresolution lattice.



(a)

| | $P_u$ | $P_m$ | $P_d$ |
|---|---|---|---|
| E | 0.363 | 0.451 | 0.186 |
| B F | 0.278 | 0.597 | 0.126 |
| A C G | 0.203 | 0.720 | 0.077 |
| D H | 0.153 | 0.743 | 0.104 |
| I | 0.363 | 0.451 | 0.186 |

(b)

Figure 3.2: A Sample Multiresolution Lattice.
(a) The shaded area covers nodes with an additional bit of precision after the decimal point for their prefix sums. All asset prices marked above the nodes are in binary form. The parameters are $S_0 = 5$, $r = 10\%$, $\sigma = 20\%$, $T = 0.75$, and $n = 3$. (b) In the table, $P_u$, $P_m$ and $P_d$ denote the up, flat, and down branching probabilities for each node, respectively.

Empirical results suggest that this approach dramatically reduce the number of prefix sums in the lattice [16]. One of the results is illustrated in Table 3.1. The convergence of the multiresolution lattice is verified in Fig. 3.3. The multiresolution approach is compared against other analytical approaches in Table 3.2, where Fu et al. claim that most analytical approaches fail under these extreme cases [21]. Fig. 3.4 shows further that the numerical delta as determined by the multiresolution approach varies smoothly with asset price $S$. We therefore do not expect problems in constructing hedge portfolios. A grand comparison among different pricing methods is illustrated in Table 4.2.

All the experimental results up to now are for European-style Asian options. Table 3.3 tabulates American-style Asian option values generated by various algorithms: the Hull-White algorithm [26, 27], the upper- and lower-bound algorithms of Chalasani et al. [13], and the multiresolution approach. Both the Hull-White

| $n$ | 100 | 135 | 150 | 160 |
|---|---|---|---|---|
| Number of prefix sums | 2,969,062 | 9,065,895 | 14,030,903 | 18,280,584 |

Table 3.1: Reduction of the Number of Prefix Sums.
The table records the total number of prefix sums in the multiresolution lattice. The input parameters are: $S_0 = 100$, $X = 100$, $\sigma = 20\%$, $r = 10\%$, $T = 1$. The first column denotes the number of time steps in the lattice while the second column denotes the number of prefix sums in the lattice.

and multiresolution approach generate results that exceed the upper bounds of Chalasani et al. Since the bounds of Chalasani et al. are valid for the CRR binomial lattice only, that the Hull-White and multiresolution approach's results lie outside the bounds does not prevent them from being closer to the true option value. In fact, judging from the multiresolution approach's excellent convergence in the European-style case, we suggest that the bounds of Chalasani et al. may actually underestimate the true option value for any finite $n$. Besides, the fact that the Hull-White algorithm overestimates the desired option value (see Section 4.8) is also verified.

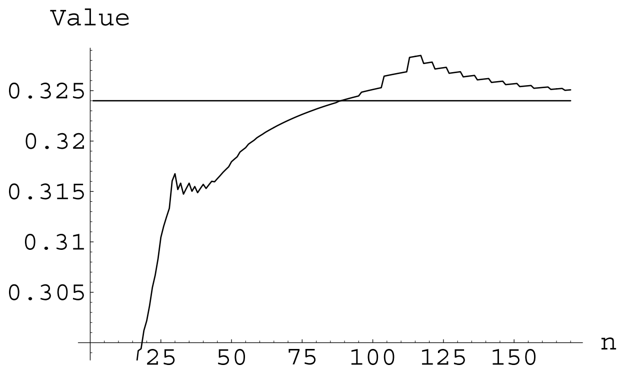


Figure 3.3: Convergence Behavior of the Multiresolution Lattice
The Monte Carlo simulation value from Choa and Lee [14] is plotted for reference. The parameters are $S_0 = 50$, $X = 60$, $r = 10\%$, $\sigma = 30\%$, and $T = 0.5$.

The exact pricing algorithm can be applied on the multiresolution lattice successfully as the total number of prefix sums reduces dramatically. This exact pricing algorithm is guaranteed to converge to the true option value since no interpolation error is introduced in backward induction. This relationship is illustrated in Fig. 3.5. Obviously, the exact pricing algorithm on the multiresolution lattice is much superior to the exact algorithms on other lattices. However, there are two theoretical problems of the multiresolution lattice. First, no proof is offered to show that the resulting exact pricing algorithm runs in subexponential time. Second, the multiresolution lattice is constructed by an ad hoc local search, and there is no proof to guarantee that

| $r$ | $\sigma$ | $T$ | $S_0$ | GE | Shaw | Euler | PW | TW | MC10 | MC100 | SE | MR |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5.0% | 50% | 1 | 1.9 | .195 | .193 | .194 | .194 | .195 | .192 | .196 | .004 | .193 |
| 5.0% | 50% | 1 | 2.0 | .248 | .246 | .247 | .247 | .250 | .245 | .249 | .004 | .246 |
| 5.0% | 50% | 1 | 2.1 | .308 | .306 | .307 | .307 | .311 | .305 | .309 | .005 | .306 |
| 2.0% | 10% | 1 | 2.0 | .058 | .520 | .056 | .0624 | .0568 | .0559 | .0565 | .0008 | .0558 |
| 18.0% | 30% | 1 | 2.0 | .227 | .217 | .219 | .219 | .220 | .219 | .220 | .003 | .219 |
| 12.5% | 25% | 2 | 2.0 | .172 | .172 | .172 | .172 | .173 | .173 | .172 | .003 | .172 |
| 5.0% | 50% | 2 | 2.0 | .351 | .350 | .352 | .352 | .359 | .351 | .348 | .007 | .351 |

Table 3.2: Comparing Multiresolution Approach against Other Analytical Approaches.
All the notations are the same as the ones in Table 2.1 except that "MR" denotes the multiresolution approach.

the lattice can always be constructed successfully. In the following sections, a new integral lattice is constructed and rigorous proofs are also given to tackle above two problems.

## 3.3 An Overview of the Newly Proposed Lattice

The multiresolution lattice allows the asset's price of each node to be a rational number of finite precision. It divides the lattice into different parts and the resolution (the precision of prefix sums) of each part varies as necessary. Varying the resolution of the lattice reduces the number of prefix sums drastically. However, this idea also prevents some important properties – like the time complexity of the algorithm– from being analyzed rigorously. The homogeneous property of the option value is used to construct a new integral lattice [18]. The two theoretical problems mentioned in the last section can now be analyzed rigorously in this new lattice model. A simple overview of this newly proposed lattice is given as follows. Suppose that the prices of the underlying asset are multiplied by a constant $K$ before pricing the option, then the homogeneous property says that this option value divided by $K$ gives the originally desired option value [36]. In this chapter, a $K$ is found to ensure that an integral lattice can be constructed. To ensure that the pricing results based on the purposed lattice converge to the true option value, we have to make sure that the underlying asset's price process simulated by this integral lattice converges to the continuous-time underlying asset's price process described in Eq. (2.1). The conditions are that the lattice should match the mean and the variance of the underlying asset's price process at each time step [19]. It will be proved that at least one integral price that satisfies the above conditions exists for every node in this proposed lattice.

| $\tau$ | $X$ | HW | LB | UB | MR |
|---|---|---|---|---|---|
| 0.5 | 40 | 12.115 | 12.111 | 12.112 | 12.132 |
| | 45 | 7.261 | 7.255 | 7.255 | 7.275 |
| | 50 | 3.275 | 3.269 | 3.269 | 3.272 |
| | 55 | 1.152 | 1.148 | 1.148 | 1.147 |
| | 60 | 0.322 | 0.320 | 0.320 | 0.322 |
| 1.0 | 40 | 13.153 | 13.150 | 13.151 | 13.194 |
| | 45 | 8.551 | 8.546 | 8.547 | 8.576 |
| | 50 | 4.892 | 4.888 | 4.889 | 4.901 |
| | 55 | 2.536 | 2.532 | 2.534 | 2.541 |
| | 60 | 1.208 | 1.204 | 1.206 | 1.210 |
| 1.5 | 40 | 13.988 | 13.984 | 13.985 | 14.013 |
| | 45 | 9.652 | 9.648 | 9.650 | 9.669 |
| | 50 | 6.199 | 6.195 | 6.197 | 6.206 |
| | 55 | 3.771 | 3.767 | 3.770 | 3.786 |
| | 60 | 2.194 | 2.190 | 2.193 | 2.209 |
| 2.0 | 40 | 14.713 | 14.709 | 14.712 | 14.756 |
| | 45 | 10.623 | 10.620 | 10.623 | 10.659 |
| | 50 | 7.326 | 7.322 | 7.325 | 7.358 |
| | 55 | 4.886 | 4.882 | 4.885 | 4.912 |
| | 60 | 3.171 | 3.167 | 3.170 | 3.195 |

Table 3.3: American-Style Asian Options.
HW denotes the Hull-White algorithm [26, 27], while UB and LB denote the upper and lower bounds on the option values given by Chalasani et al. [13] for the CRR model. "MR" denotes the multiresolution approach. All algorithms use $n = 40$. The other parameters are $S_0 = 50$, $r = 10\%$, and $\sigma = 30\%$.

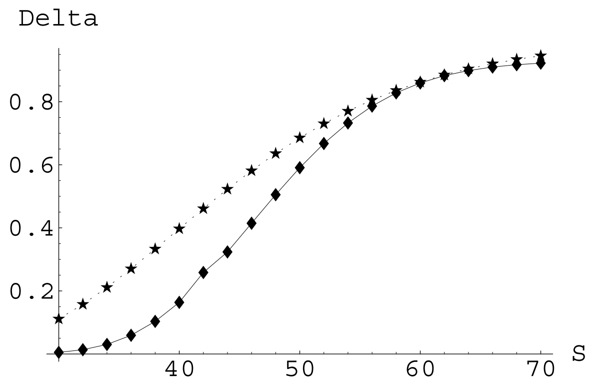


Figure 3.4: Numerical Delta.
The numerical delta of the Asian call with respect to the underlying asset's price as determined by the multiresolution approach is plotted above. The parameters are $30 \leq S_0 \leq 70$, $X = 50$, $r = 10\%$, $\sigma = 30\%$, and $\tau = 1$. For comparison, the higher dotted curve plots the delta of the vanilla call as computed by the Black-Scholes formula with the same parameters.

### 3.3.1 The Structure of a Trinomial lattice

The proposed integral lattice is a trinomial lattice. In a trinomial lattice, each node can branch to three succeeding nodes in the next time step. Let $S_{i,j}$ denotes $(j+1)$th largest asset's price of the nodes at time $i$. A 2-time-step trinomial lattice is illustrated in Fig. 3.6. Take the root node as an example. Its price is $S_{0,0}$. The underlying asset's price can move upward to $S_{1,0}$ with probability $P_u$, move flatly to $S_{1,1}$ with probability $P_m$, and move downward to $S_{1,2}$ with probability $P_d$. The branching probabilities vary for different nodes. There are $2\ell + 1$ nodes at time $\ell$.

The option value at time $n$ can be calculated by Eq. (2.7). Let $P_u$, $P_m$, and $P_d$ denote the branching probabilities of the node with the $(j+1)$th largest asset's price at time $i$. The backward induction formulae for the European-style and American-style Asian options are modified from Eq. (2.19) and (2.20) as follows:

$$\begin{aligned} \mathrm{V}(i, S_{i,j}, M) &= e^{-r\Delta t}\left[P_u \mathrm{V}(i+1, S_{i+1,j}, M + S_{i+1,j}) + \right. \\ &\quad P_m \mathrm{V}(i+1, S_{i+1,j+1}, M + S_{i+1,j+1}) + \\ &\quad \left. P_d \mathrm{V}(i+1, S_{i+1,j+2}, M + S_{i+1,j+2})\right] \end{aligned}$$

and

$$\begin{aligned} \mathrm{V}(i, S_{i,j}, M) &= \max\left(e^{-r\Delta t}\left[P_u \mathrm{V}(i+1, S_{i+1,j}, M + S_{i+1,j}) + \right.\right. \\ &\quad P_m \mathrm{V}(i+1, S_{i+1,j+1}, M + S_{i+1,j+1}) + \\ &\quad \left.\left. P_d \mathrm{V}(i+1, S_{i+1,j+2}, M + S_{i+1,j+2})\right], \mathrm{E}\right). \end{aligned}$$

The "E" in the latter formula denotes the exercise value defined in Eq. (2.8). The

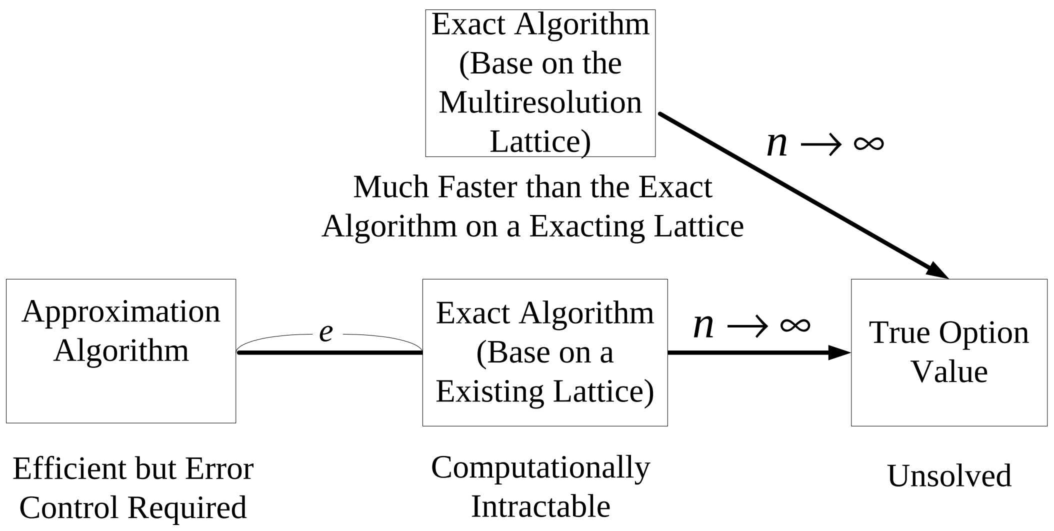

Exact Algorithm (Base on the Multiresolution Lattice)
$n \rightarrow \infty$
Much Faster than the Exact Algorithm on a Exacting Lattice
Approximation Algorithm
$e$
Exact Algorithm (Base on a Existing Lattice)
$n \rightarrow \infty$
True Option Value
Efficient but Error Control Required
Computationally Intractable
Unsolved

Figure 3.5: Convergence of Different Lattice Models.
The exact algorithm on the multiresolution lattice can be applied successfully. But running an exact pricing algorithm on any other lattice $L$ is computationally intractable as the number of prefix sums of $L$ is much larger than that of the multiresolution lattice.

above formulae can be applied in a backward fashion from time $n-1$ to time 0. The root node thus gives the pricing result.

## 3.4 Lattice Construction

The proposed lattice should meet two requirements: (1): The asset's price of each node is restricted to be an integer. (2): The asset's price process simulated by the propsed lattice converges to the underlying asset's price process mentioned in Eq. (2.1). In this section, we will first show how the lattice is constructed step by step to meet the above two requirements. Proof given in the next section shows that our lattice implies a subexponential-time exact pricing algorithm for Asian options.

The homogeneous property says [36]:

$$E[\max(A-X,0)] = \frac{1}{K} E\left[\max\left(KA - KX, 0\right)\right].$$

Thus we can multiply the initial underlying asset's price ($S_0$) and the strike price $X$ by a constant $K$ and price this hypothetical option. The pricing result is then obtained by dividing this hypothetical option price by $K$. To ensure that a proper integral price can be assigned to each node (except the root node), $K$ is defined as follows:

$$K \equiv (0.25 S_0 \sigma)^{-1} \sqrt{n/T} \exp\left[\left(0.5\sigma^2 - r\right) T + 2\sigma\sqrt{Tn}\right]. \tag{3.1}$$

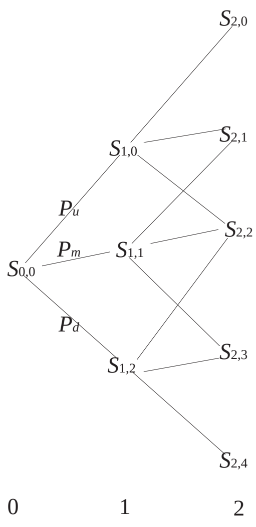


Figure 3.6: A 2-Time-Step Trinomial Lattice.
The initial asset's price is $S_{0,0}$. $S_{i,j}$ denotes the $(j+1)$th largest asset's price at time $i$. $P_u$, $P_m$, and $P_d$ denote the branching probabilities, respectively.

Note that $K \in e^{O(\sqrt{n})}$. We will show later that this $K$ works.

Next, a trinomial lattice is constructed to price this hypothetical option. Note that the underlying asset's price of the root node ($S_{0,0}$) is equal to $KS_0$.[1] Our goal is to find integral underlying asset's prices $S_{i,j}$ for $0 < i \leq n$, $0 \leq j \leq 2i$. Define the $V$-log-price of the underlying asset's price $V'$ as $\ln(V'/V)$ and $c_{i,j} \equiv (r-0.5\sigma^2)\, i\Delta t + 2(i-j)\, \sigma\sqrt{\Delta t}$. $S_{i,j}$ will be some integer whose $KS_0$-log-price belongs to the following interval centered around $c_{i,j}$: $(c_{i,j} - 0.25\sigma\sqrt{\Delta t}, c_{i,j} + 0.25\sigma\sqrt{\Delta t})$. We call $c_{i,j}$ the *log-price center* for $S_{i,j}$. Take a 2-time-step trinomial lattice in Fig. 3.7 as an example. The $x$-axis marks the time step in the lattice, and the $y$-axis denotes $KS_0$-log-prices. Each log-price center is depicted as a hollow circle. Each dotted line segment begins at $c_{i,j}$ and ends at $c_{i+1,j+1}$. The slopes of these dotted lines represent the expected growth rate of the logarithmic underlying asset's price, $(r - 0.5\sigma^2)\Delta t$. The integral asset's price for each node is depicted as a solid circle. Take $S_{2,0}$ as an example. Because $c_{2,0} = (r-0.5\sigma^2)\, 2\Delta t + 4\sigma\sqrt{\Delta t}$, the $KS_0$-log-price of $S_{2,0}$ should fall within the interval $(c_{2,0} - 0.25\sigma\sqrt{\Delta t}, c_{2,0} + 0.25\sigma\sqrt{\Delta t})$. The proof in Subsection 3.5.1.1 shows that there is always at least one integer whose $KS_0$-log-price falls within the above interval.

The branching probabilities for each node are computed as follows. Take a node with price $S_{i,j}$. Recall that the probabilities for the underlying asset's price moving to $S_{i+1,j}$, $S_{i+1,j+1}$, and $S_{i+1,j+2}$ are $P_u$, $P_m$, and $P_d$, respectively. Define $\alpha$, $\beta$, and $\gamma$

[1]Note that $KS_0$ is not necessary an integer since both $K$ and $S_0$ are not necessary integers, too.

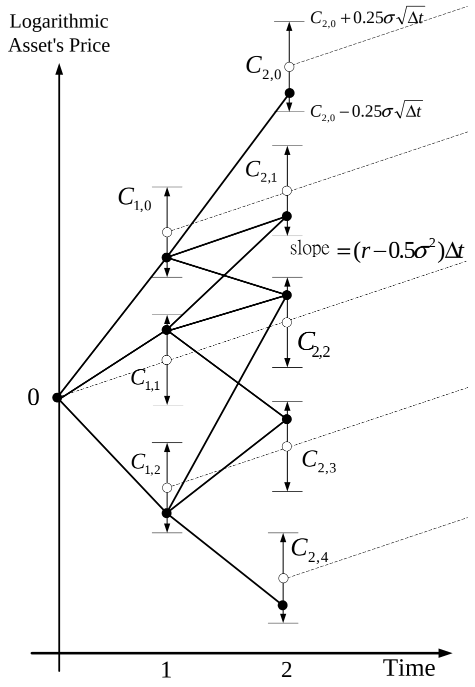

Logarithmic Asset's Price
$C_{2,0}+0.25\sigma\sqrt{\Delta t}$
$C_{2,0}$
$C_{2,0}-0.25\sigma\sqrt{\Delta t}$
$C_{2,1}$
$C_{1,0}$
slope $=(r-0.5\sigma^2)\Delta t$
$C_{2,2}$
$C_{1,1}$
0
$C_{1,2}$
$C_{2,3}$
$C_{2,4}$
1
2
Time

Figure 3.7: A 2-Time-Step Trinomial Lattice over $KS_0$-log-prices.

as follows:

$$\alpha \equiv \ln(S_{i+1,j}/S_{i,j}) - \mu, \tag{3.2}$$
$$\beta \equiv \ln(S_{i+1,j+1}/S_{i,j}) - \mu, \tag{3.3}$$
$$\gamma \equiv \ln(S_{i+1,j+2}/S_{i,j}) - \mu, \tag{3.4}$$

where $\mu$ is defined in Eq. (2.12). The branching probabilities satisfy

$$P_u\alpha + P_m\beta + P_d\gamma = 0, \tag{3.5}$$
$$P_u\alpha^2 + P_m\beta^2 + P_d\gamma^2 = \text{Var}, \tag{3.6}$$
$$P_u + P_m + P_d = 1, \tag{3.7}$$

where Var is defined in Eq. (2.13). Eqs. (3.5) and (3.6) match the mean and the variance of the logarithmic asset's price, respectively. Hence our lattice converges weakly to the lognormal asset's price process. That Eqs. (3.5)–(3.7) give valid branching probabilities will be proved in Subsection 3.5.1.2.

## 3.5 Proof

This section proves that our approach provides a subexponential-time algorithm for pricing Asian options. The proof has two parts. First, we show that a valid lattice is constructed. Next we show that the exact pricing algorithm based on this lattice runs in subexponential time.

### 3.5.1 Validity of the Lattice

#### 3.5.1.1 Existence of Integral Underlying Asset's Prices

When constructing the trinomial lattice, an integral underlying asset's price is assigned to each node. More specifically, $S_{i,j}$ should be an integer whose $KS_0$-log-price falls in $(c_{i,j} - 0.25\sigma\sqrt{\Delta t}, c_{i,j} + 0.25\sigma\sqrt{\Delta t})$. To ensure that such an integer exists, it suffices to show that

$$KS_0\left[\exp\left(c_{i,j} + 0.25\sigma\sqrt{\Delta t}\right) - \exp\left(c_{i,j} - 0.25\sigma\sqrt{\Delta t}\right)\right] > 1.$$

Our goal now is to show that our choice of $K$ in Eq. (3.1) satisfies the above inequality. Without loss of generality, only the case $S_{n,2n}$ is considered as $\exp(c_{i,j} + 0.25\sigma\sqrt{\Delta t}) - \exp(c_{i,j} - 0.25\sigma\sqrt{\Delta t})$ is minimized when $i = n$ and $j = 2n$. Thus it suffices to show that our $K$ satisfies

$$K > S_0^{-1}e^{-c_{n,2n}}\left(e^{0.25\sigma\sqrt{\Delta t}} - e^{-0.25\sigma\sqrt{\Delta t}}\right)^{-1}.$$

Indeed,

$$\begin{aligned}
& S_0^{-1}e^{-c_{n,2n}}\left(e^{0.25\sigma\sqrt{\Delta t}} - e^{-0.25\sigma\sqrt{\Delta t}}\right)^{-1} \\
= \; & S_0^{-1}\exp\left[(0.5\sigma^2 - r)\,T + 2\sigma\sqrt{Tn}\right] \\
& \times\left(e^{0.25\sigma\sqrt{\Delta t}} - e^{-0.25\sigma\sqrt{\Delta t}}\right)^{-1} \\
< \; & S_0^{-1}\exp\left[(0.5\sigma^2 - r)\,T + 2\sigma\sqrt{Tn}\right]/(0.25\sigma\sqrt{T/n}) \qquad (3.8) \\
\leq \; & (0.25S_0\sigma)^{-1}\sqrt{n/T}\exp\left[(0.5\sigma^2 - r)\,T + 2\sigma\sqrt{Tn}\right] \\
= \; & e^{O(\sqrt{n})},
\end{aligned}$$

where Eq. (3.8) holds because

$$e^{0.25\sigma\sqrt{\Delta t}} - e^{-0.25\sigma\sqrt{\Delta t}} > \left(1 + 0.25\sigma\sqrt{\Delta t}\right) - 1 = 0.25\sigma\sqrt{\Delta t}\,.$$

#### 3.5.1.2 Validity of Branching Probabilities

The Eqs. (3.5)–(3.7) are proved to result in valid branching probabilities for the asset's price $S_{i,j}$. To be more specific, the branching probabilities $P_u$, $P_m$, and $P_d$, are proved to be strictly larger than 0 to meet the no-arbitrage requirement (note that this suffices to ensure that they are all less than 1). First we will derive constraints on $\alpha$, $\beta$, $\gamma$ defined in Eqs. (3.2)–(3.4). Then we will show that valid branching probabilities are obtained by solving Eqs. (3.5)–(3.7) given the constraints on $\alpha$, $\beta$, and $\gamma$.

Fig. 3.8 is used to aid the proof. Note that a $KS_0$-log-price of $x$ becomes a $S_{i,j}$-log-price of $x + \frac{\ln KS_0}{\ln S_{i,j}}$. The following computations are in $S_{i,j}$-log-prices unless sated otherwise. The log-price center of $S_{i,j}$, $c'$, equals $c_{i,j} + \frac{\ln KS_0}{\ln S_{i,j}}$, whereas $c$, the log-price center of $S_{i+1,j+1}$, equals $c' + (r - 0.5\sigma^2)\Delta t$. The log-price centers of $S_{i+1,j}$ and $S_{i+1,j+2}$ are $c + 2\sigma\sqrt{\Delta t}$ and $c - 2\sigma\sqrt{\Delta t}$, respectively. The $S_{i,j}$-log-price of $S_{i,j}$ is 0. The conditional mean of the asset's price one time step after it reaches $S_{i,j}$ is $\mu = (r - 0.5\sigma^2)\Delta t$. By construction, the distance between the $S_{i,j}$-log-price of $S_{i,j}$, which equals 0, and its log-price center $c'$ is smaller than $0.25\sigma\sqrt{\Delta t}$. This implies that $|c'| < 0.25\sigma\sqrt{\Delta t}$. Hence,

$$\begin{aligned} |\mu - c| &= \left|(r - 0.5\sigma^2)\Delta t - \left[c' + (r - 0.5\sigma^2)\Delta t\right]\right| \\ &= |\,c'\,| < 0.25\sigma\sqrt{\Delta t}\,. \end{aligned}$$

Thus $\mu$ falls within interval $(c - 0.25\sigma\sqrt{\Delta t}, c + 0.25\sigma\sqrt{\Delta t})$. Now

$$\beta = \ln(S_{i+1,j+1}/S_{i,j}) - \mu$$

falls within interval $(-0.5\sigma\sqrt{\Delta t}, 0.5\sigma\sqrt{\Delta t})$ as the $S_{i,j}$-log-price of $S_{i+1,j+1}$, $\ln(S_{i+1,j+1}/S_{i,j})$, falls within interval $(c - 0.25\sigma\sqrt{\Delta t}, c + 0.25\sigma\sqrt{\Delta t})$. Fig. 3.8 illustrates a case where $\beta < 0$.

We next represent $\alpha$ and $\gamma$ in terms of $\beta$. Define $d_1$ as the difference between $S_{i+1,j}$'s and $S_{i+1,j+1}$'s $S_{i,j}$-log-prices and $d_2$ as the difference between $S_{i+1,j+1}$'s and $S_{i+1,j+2}$'s $S_{i,j}$-log-prices. Thus $\alpha$ and $\gamma$ can be represented as

$$\begin{aligned} \alpha &= \ln(S_{i+1,j}/S_{i,j}) - \ln(S_{i+1,j+1}/S_{i,j}) \\ &\quad + \ln(S_{i+1,j+1}/S_{i,j}) - \mu \\ &= d_1 + \beta, \\ \gamma &= \ln(S_{i+1,j+2}/S_{i,j}) - \ln(S_{i+1,j+1}/S_{i,j}) \\ &\quad + \ln(S_{i+1,j+1}/S_{i,j}) - \mu \\ &= -d_2 + \beta. \end{aligned}$$

Note that $1.5\sigma\sqrt{\Delta t} < d_1 < 2.5\sigma\sqrt{\Delta t}$ because

$$\begin{aligned} c + 1.75\sigma\sqrt{\Delta t} &< \ln(S_{i+1,j}/S_{i,j}) < c + 2.25\sigma\sqrt{\Delta t}, \\ c - 0.25\sigma\sqrt{\Delta t} &< \ln(S_{i+1,j+1}/S_{i,j}) < c + 0.25\sigma\sqrt{\Delta t}, \\ d_1 &= \ln(S_{i+1,j}/S_{i,j}) - \ln(S_{i+1,j+1}/S_{i,j}). \end{aligned}$$

Similarly, $1.5\sigma\sqrt{\Delta t} < d_2 < 2.5\sigma\sqrt{\Delta t}$. Note also that $\alpha = d_1 + \beta > \sigma\sqrt{\Delta t} > 0$ as $\beta \in (-0.5\sigma\sqrt{\Delta t}, 0.5\sigma\sqrt{\Delta t})$ and $d_1 \in (1.5\sigma\sqrt{\Delta t}, 2.5\sigma\sqrt{\Delta t})$. Similarly, $\gamma = -d_2 + \beta <$

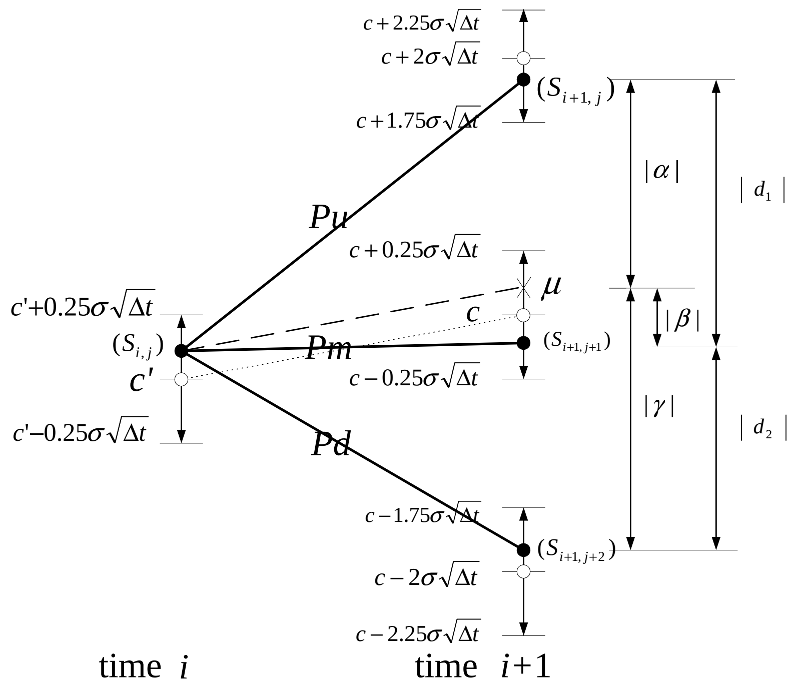

$c+2.25\sigma\sqrt{\Delta t}$
$c+2\sigma\sqrt{\Delta t}$
$(S_{i+1,j})$
$c+1.75\sigma\sqrt{\Delta t}$
$|\alpha|$
$|d_1|$
$Pu$
$c+0.25\sigma\sqrt{\Delta t}$
$\mu$
$c'+0.25\sigma\sqrt{\Delta t}$
$c$
$|\beta|$
$(S_{i,j})$
$Pm$
$(S_{i+1,j+1})$
$c'$
$c-0.25\sigma\sqrt{\Delta t}$
$|\gamma|$
$|d_2|$
$c'-0.25\sigma\sqrt{\Delta t}$
$Pd$
$c-1.75\sigma\sqrt{\Delta t}$
$(S_{i+1,j+2})$
$c-2\sigma\sqrt{\Delta t}$
$c-2.25\sigma\sqrt{\Delta t}$
time $i$
time $i+1$

Figure 3.8: Branching Probabilities for the Node with Price $S_{i,j}$.
All the values in this figure are $S_{i,j}$-log-prices except the ones that are parenthesized. The nodes with asset's prices $S_{i,j}$, $S_{i+1,j}$, $S_{i+1,j+1}$, and $S_{i+1,j+2}$ are represented by solid circles. The branches that connect these nodes are represented by thick lines. The log-price centers for these nodes are represented by hollow circles. The log-price centers of $S_{i,j}$ and $S_{i+1,j+1}$ are $c'$ and $c$, respectively. $\mu$ represents the expected asset's price one step after $S_{i,j}$. $P_u$, $P_m$, and $P_d$ denote the branching probabilities for the upper, middle, and lower branches from the node with price $S_{i,j}$. Values $\alpha$, $\beta$, and $\gamma$ are defined in Eqs. (3.2)–(3.4). Finally, $|\,d_1\,|$ denotes the distance between $S_{i+1,j}$'s and $S_{i+1,j+1}$'s $S_{i,j}$-log-prices, and $|\,d_2\,|$ denotes the distance between $S_{i+1,j+1}$'s and $S_{i+1,j+2}$'s $S_{i,j}$-log-prices.

$-\sigma\sqrt{\Delta t} < 0$ as $d_2 \in (1.5\sigma\sqrt{\Delta t}, 2.5\sigma\sqrt{\Delta t})$. It is also obvious that $\alpha > \beta > \gamma$ as $d_1, d_2 > 0$.

We now show that positive branching probabilities are obtained given the constraints on $\alpha$, $\beta$, and $\gamma$ derived above and summarized below:

$$\begin{aligned}\beta &\in (-0.5\sigma\sqrt{\Delta t}, 0.5\sigma\sqrt{\Delta t}).\\ \alpha &= d_1+\beta, \text{ where } d_1 \in (1.5\sigma\sqrt{\Delta t}, 2.5\sigma\sqrt{\Delta t}).\\ \gamma &= -d_2+\beta, \text{ where } d_2 \in (1.5\sigma\sqrt{\Delta t}, 2.5\sigma\sqrt{\Delta t}).\end{aligned}$$

The branching probabilities are solved by applying Cramer's rule to Eqs. (3.5)– (3.7).

Define

$$\det = \begin{bmatrix} \alpha & \beta & \gamma \\ \alpha^2 & \beta^2 & \gamma^2 \\ 1 & 1 & 1 \end{bmatrix} = (\beta - \alpha)(\gamma - \alpha)(\gamma - \beta) < 0,$$

$$\det_u = \begin{bmatrix} 0 & \beta & \gamma \\ \mathrm{Var} & \beta^2 & \gamma^2 \\ 1 & 1 & 1 \end{bmatrix} = (\beta\gamma + \mathrm{Var})(\gamma - \beta),$$

$$\det_m = \begin{bmatrix} \alpha & 0 & \gamma \\ \alpha^2 & \mathrm{Var} & \gamma^2 \\ 1 & 1 & 1 \end{bmatrix} = (\alpha\gamma + \mathrm{Var})(\alpha - \gamma),$$

and

$$\det_d = \begin{bmatrix} \alpha & \beta & 0 \\ \alpha^2 & \beta^2 & \mathrm{Var} \\ 1 & 1 & 1 \end{bmatrix} = (\alpha\beta + \mathrm{Var})(\beta - \alpha).$$

Then $P_u = \det_u/\det$, $P_m = \det_m/\det$, and $P_d = \det_d/\det$. Note that $\det < 0$ as $\alpha > \beta > \gamma$. To show that the branching probabilities are valid, we have to show that $P_u, P_m, P_d > 0$. As $\det < 0$, it is sufficient to show $\det_u, \det_m, \det_d < 0$. Moreover, as $\gamma - \beta < 0$, $\alpha - \gamma > 0$, and $\beta - \alpha < 0$, we only need to show that $\beta\gamma + \mathrm{Var} > 0$, $\alpha\gamma + \mathrm{Var} < 0$, and $\alpha\beta + \mathrm{Var} > 0$ instead.

**1.** $\beta\gamma + \mathrm{Var} > 0$: Note that

$$\begin{aligned} \beta\gamma + \mathrm{Var} &= \beta(\beta - d_2) + \sigma^2\Delta t \\ &= (\beta - 0.5d_2)^2 - 0.25d_2^2 + \sigma^2\Delta t. \end{aligned}$$

For a given $d_2$, $\beta\gamma + \mathrm{Var}$ reaches its minimum when $\beta = 0.5d_2$. Recall the constraints that $0.5d_2 \in (0.75\sigma\sqrt{\Delta t}, 1.25\sigma\sqrt{\Delta t})$ and $\beta < 0.5\sigma\sqrt{\Delta t}$. Hence $\beta\gamma + \mathrm{Var}$ reaches its minimum for a given $d_2 \in (1.5\sigma\sqrt{\Delta t}, 2.5\sigma\sqrt{\Delta t})$ when $\beta \to 0.5\sigma\sqrt{\Delta t}$. Thus, we have

$$\begin{aligned} \beta\gamma + \mathrm{Var} &= \beta(\beta - d_2) + \sigma^2\Delta t \\ &> 0.5\sigma\sqrt{\Delta t}\,(0.5\sigma\sqrt{\Delta t} - d_2) + \sigma^2\Delta t \\ &= -0.5\sigma\sqrt{\Delta t}\,d_2 + 1.25\sigma^2\Delta t \\ &> -1.25\sigma^2\Delta t + 1.25\sigma^2\Delta t = 0. \end{aligned}$$

**2.** $\alpha\gamma + \mathrm{Var} < 0$: Note that

$$\alpha\gamma + \mathrm{Var} = (\beta + d_1)(\beta - d_2) + \sigma^2\Delta t.$$

As $\beta + d_1 > \sigma\sqrt{\Delta t}$ and $\beta - d_2 < -\sigma\sqrt{\Delta t}$, we have

$$(\beta + d_1)(\beta - d_2) + \sigma^2\Delta t < (\sigma\sqrt{\Delta t})(-\sigma\sqrt{\Delta t}) + \sigma^2\Delta t = 0.$$

**3.** $\alpha\beta + \text{Var} > 0$: This is proved similarly as we prove $\beta\gamma + \text{Var} > 0$. Note that

$$\begin{aligned}\alpha\beta + \text{Var} &= (\beta + d_1)\beta + \sigma^2\Delta t \\ &= (\beta + 0.5d_1)^2 - 0.25d_1^2 + \sigma^2\Delta t.\end{aligned}$$

For a given $d_1$, $\alpha\beta+\text{Var}$ reaches its minimum when $\beta = -0.5d_1$. Recall the constraints that $-0.5d_1 \in (-1.25\sigma\sqrt{\Delta t}, -0.75\sigma\sqrt{\Delta t})$ and $\beta > -0.5\sigma\sqrt{\Delta t}$. Hence $\alpha\beta + \text{Var}$ reaches its minimum for a given $d_1 \in (1.5\sigma\sqrt{\Delta t}, 2.5\sigma\sqrt{\Delta t})$ when $\beta \to -0.5\sigma\sqrt{\Delta t}$. Thus we have

$$\begin{aligned}\alpha\beta + \text{Var} &= (\beta + d_1)\beta + \sigma^2\Delta t \\ &> (-0.5\sigma\sqrt{\Delta t} + d_1)(-0.5\sigma\sqrt{\Delta t}) + \sigma^2\Delta t \\ &= -0.5\sigma\sqrt{\Delta t}\, d_1 + 1.25\sigma^2\Delta t \\ &> -1.25\sigma^2\Delta t + 1.25\sigma^2\Delta t = 0.\end{aligned}$$

### 3.5.2 Running-Time Analysis

Note that the time complexity of an exact pricing algorithm is proportional to the total number of prefix sums on the lattice. Thus the pricing algorithm is subexponential in $n$ if the total number of prefix sums is bounded by a subexponential function. We will first show that the maximum prefix sum in our lattice is bounded by a subexponential function. Then we will show that the total number of prefix sums is, too.

The maximum asset's price $S_{n,0}$ is bounded by $KS_0e^{c_{n,0}+0.25\sigma\sqrt{\Delta t}}$, where

$$\begin{aligned}c_{n,0} &= (r - 0.5\sigma^2)n\Delta t + 2(n-0)\sigma\sqrt{\Delta t} \\ &= (r - 0.5\sigma^2)\,T + 2\sigma\sqrt{Tn}\,.\end{aligned}$$

Thus $S_{n,0}$ is bounded by a subexponential function as both $K$ and $e^{c_{n,0}+0.25\sigma\sqrt{\Delta t}}$ are subexponential in $n$. The maximum prefix sum in the lattice is equal to $\sum_{i=0}^{n} S_{i,0} \leq (n+1)S_{n,0}$. Note that $(n+1)S_{n,0}$ is also a subexponential function. We define $F \equiv (n+1)S_{n,0}$ for simplicity.

The next goal is to show that the total number of prefix sums is bounded by a subexponential function. Recall that the underlying asset's price for each node except the root node in our lattice must be an integer. The underlying asset's price at the root node ($KS_0$) can be represented as $S'+a$ for some integer $S' \geq 0$ and some rational number $a$ where $0 \leq a < 1$. Thus all the possible prefix sums must belong to the set $\{X : X \leq F,\ X = I + a,\ I \text{ is a nonnegative integer.}\}$. The number of elements in this set is at most $\lceil F \rceil$. Thus the maximum number of prefix sums for each node is bounded by $\lceil F \rceil$. Since there are $(n+1)^2$ nodes in an $n$-time-step trinomial lattice, the total number of prefix sums is bounded above by the subexponential function $(n+1)^2\lceil F \rceil$. The time complexity of our algorithm is thus subexponential in $n$.

# Chapter 4

# The Range Bound Algorithms

Although the proposed exact pricing algorithm breaks the exponential-time barrier successfully, it is still not a polynomial-time algorithm. In this chapter, the range bound algorithms are proposed to tackle the efficiency problem. The computational performance of these algorithms can be further improved by employing some approximations while the pricing error is analyzed rigorously.

## 4.1 The Range Bound Paradigm

The exact pricing algorithm proposed in last chapter is still computationally intractable although it significantly improves the efficiency. Another approach to tackle this pricing problem is to construct efficient approximation algorithms that produce provable upper and lower bounds (called **range bounds**) of the desired option value (produced by the exact pricing algorithm). 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. 1.4.) Since the desired option values are known to converge to the true option value as the number of time steps in the lattice ($n$) approaches to infinity, the approximation algorithm is guaranteed to converge to the true option value if $e \rightarrow 0$ as $n \rightarrow \infty$. This relationship is illustrated in Fig. 1.5.

Rogers and Shi derive analytical formulae to estimate the upper and the lower bounds of European-style Asian options [38]. But we can not trade off the computational efficiency against the accuracy level via their formulae. Chalasani et al. propose an $O(n^4)$-time range-bound algorithm for Asian options [13]. However, the pricing error between the approximation and the exact pricing algorithms $e$ is not rigorously analyzed there.

The convergence of the upper bound of pricing error $e$ is first considered in Aingworth et al. [2]. An $O(kn^2)$-time algorithm is proposed to guarantee a theoretical error bound of $O(Xn/k)$, where $k$ can be varied for any desired trade-off between time and accuracy. However, their error analysis of American-style Asian options is mistaken. Akcoglu et al. [3] derive a complex trade-off by a recursive application of the algorithm of [2] in the case of European-style Asian options. See Hochbaum [25] for more information on approximation algorithms.

## 4.2 An Overview of the Proposed Range Bound Algorithms

A series of the range bound algorithms are proposed in this chapter. The first algorithm has an error bound of $O(X\sqrt{n}/k)$ for European-style Asian options. This is an improvement over the result of Aingworth et al. [2]. 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 mentioned in Fig. 1.2. As the magnitude of the pricing error is $O(X\sqrt{n}/k)$, it suffices to pick $k$ to be proportional to $\sqrt{n}$ to satisfy any desired error bound and the running time is $O(n^{2.5})$. To guarantee a convergence rate of $O(n^{-0.5})$, as another example, it suffices to pick $k$ to be proportional to $n$, making the algorithms' running time $O(n^3)$. To guarantee a convergence rate of $O(n^{-1})$, as the last example, it suffices to pick $k$ to be proportional to $n^{1.5}$, making the algorithms' running time $O(n^{3.5})$. This convergence result is reached without ad hoc assumptions as in Forsyth et al. [20]. Our experiments show that the theoretical error bound is probably overpessimistic and a convergence rate of $O(n^{-1})$ is most likely achievable with a running time of $O(n^3)$ by picking $k$ to be proportional to $n$.

American-style Asian options are harder to price because of the need to determine whether it is optimal to exercise early at a path prefix (see Eq. (2.20).) We will show later that the path prefixs that are optimal to be exercised early and the path prefixs that are not optimal to be exercised are divided by the so-called "exercise boundary". As a consequence, rigorous analysis of the algorithms for American-style Asian options is even scarcer than that for European-style Asian options. Our range-bound algorithms for American-style Asian options derive from those for European-style ones. The running times remain $O(kn^2)$. Furthermore, we give formal proof that the proposed algorithms do give range bounds. Computer experiments again demonstrate that the range bounds are so tight in practice that the desired option value is essentially obtained within the time bound. This claim may fail for large $n$ if both the underlying asset's price volatility and the maturity date are very large numbers. (To our knowledge, no papers in the literature on American-style Asian options mention numerical results for volatilities larger than 50%.) We conclude that our proposed algorithms price the American-style Asian option correctly except when both the asset's price volatility and the maturity date are high.

The intricate range-bound proof is of independent interest. It yields an unexpected corollary that the Hull-White paradigm with linear interpolation in [26, 27] is an upper-bound algorithm.

A very important feature of our range bound algorithms is the way with which they attempt to limit the range of prefix sums at each node. The motivation is obvious: A small dynamic range allows finer resolution in the approximation given the same amount of computational efforts. In the case of European-style Asian options, prefix sums at or over a certain numerical bound will necessarily result in the option being in the money at the maturity date. Luckily, in this scenario their contribution to the option value is given by a simple exact formula. This implies that the algorithms can direct their computational resources to the prefix sums lower than the said bound. In fact, without incorporating this numerical bound, any pricing algorithm may fail to converge if volatility or/(and) the maturity date is high.

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

The algorithm to estimate the exercise boundary is of independent interest. It gives rise to a general two-phase computational framework in which phase one calculates the estimated exercise boundary and phase two employs any purported upper-bound algorithm. The resulting algorithm is guaranteed to be an upper-bound algorithm. Because the algorithm in phase two works on reduced prefix sum ranges, it is expected to offer substantially more accurate results than if phase one is not in place. Any algorithmic progress in pricing the American-style Asian options, therefore, immediately results in an improved two-phase version in which the first phase calls upon our algorithm to provide the estimated exercise boundary.

All the experimental results in this chapter are all based on the CRR binomial lattice model. The details of this model can be found in Subsection 2.5.1. Note that the range-bound algorithms and the proofs in this chapter can also be "ported" to other lattice models.

This chapter is structured as follows. A range bound algorithm for European-style Asian options is given first. A rigorous proof is then given in Subsection 4.4.3 to prove that the theoretical convergence rate of the proposed algorithm. After an improvement is introduced to improve the proposed range bound algorithm in Section 4.5, numerical results given in Section 4.6 verify the accuracy of the proposed algorithms. The range bound algorithm for American-style Asian options is extended from the improved range bound algorithm for the European-style options in Section 4.7. The optimal exercise boundary problem is considered rigorously in Section 4.8 and the range bound property still holds in our proposed algorithms. Experimental results are given in Section 4.9.

## 4.3 Preliminaries for European-Style Asian Options

The Hull-While paradigm is adopted to speed up the computation. The approximation algorithm can be viewed to replace the asset's price $S_i$ 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 from Eq. (2.9) as

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

If the algorithm calculates

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

instead, the magnitude of error will be bounded [1] above by

$$E\left[\frac{1}{n+1}\sum_{i=0}^{n}|\, D_i\,|\right]. \tag{4.1}$$

A path prefix 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 prefix's contribution to the option value is given by the following lemma.

**Lemma 4.3.1 (Aingworth et al. [2])** *Suppose that a path prefix of length $j$ has prefix sum $(n+1)X + \epsilon$, where $\epsilon \geq 0$, and it ends at a node with asset's price $S_j$. Then the discounted option value of that path prefix equals*

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

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

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

[1]The discount factor $e^{-rT}$ is ignored.

The expected value of the average $A_n = (S_0 + S_1 + \cdots + S_n)/(n+1)$ therefore equals

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

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

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

The case of $r = 0$ is similar. Q.E.D.

Lemma 4.3.1 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 option value of a prefix sum equals to or exceeding $(n+1)X$ can be calculated exactly by the lemma. We remark that the lemma can be improved by incorporating information regarding the last price $S_j$ on the path prefix. Without this upper limit of $(n+1)X$, algorithms for European-style Asian options may fail to converge for sufficiently large $\sigma$ (See Table 4.7 for supporting data.)

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

### 4.4.1 Description of the Algorithms

We start by segmenting the range $[\,0, (n+1)X\,]$ at each node $N(i,j)^2$ by $k_{ij} + 1$ *equally-distanced* **buckets**. The $\ell$th bucket, $0 \leq \ell \leq k_{ij}$, is associated with prefix sum $\ell(n+1)X/k_{ij}$. Two adjacent buckets' associated prefix sums are thus separated by $(n+1)X/k_{ij}$. See Fig. 4.1 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 4.3.1 applicable. We maintain it to simplify the presentation.

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

$$s(i,j,\ell) = \ell(n+1)X/k_{ij}. \tag{4.2}$$

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 4.3.1. The contribution of each prefix sum $s \geq (n+1)X$

[2] See the definitions in Subsection 2.5.1.

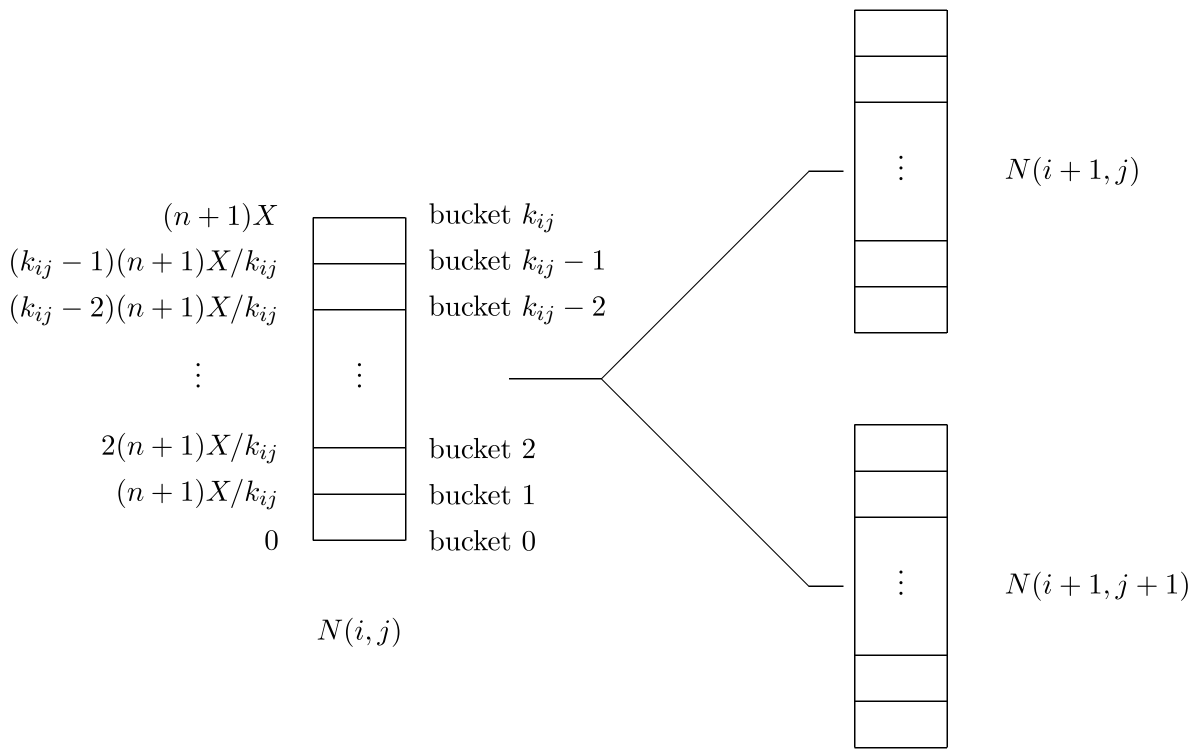


Figure 4.1: Bucketing.
Each node $N(i,j)$ has $k_{ij}+1$ buckets, starting from prefix sum 0 and ending at prefix sum $(n+1)X$ with increments of $(n+1)X/k_{ij}$.

at maturity is $e^{-nr\Delta t}[\,s/(n+1)-X\,]$. Any other bucket at maturity (that is, those $b(n,j,\ell)$ with $\ell < k_{ij}$) contributes nothing to the option value because

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

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 lattice. 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 asset's price $S_0u^{i-j}d^j$, the paths collected at bucket $b(i,j,\ell)$ are expected to move up to node $N(i+1,j)$ with prefix sum $s(i,j,\ell)+S_0u^{i-j+1}d^j$ and down to node $N(i+1,j+1)$ with prefix sum $s(i,j,\ell)+S_0u^{i-j}d^{j+1}$. These two prefix sums are then rounded down to the nearest buckets, called the **up bucket** and the **down bucket** of $b(i,j,\ell)$, respectively.

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

$$b\left(i+1, j, \left\lfloor \frac{s(i,j,\ell)+S_0 u^{i-j+1} d^j}{(n+1)X/k_{i+1,j}} \right\rfloor\right). \tag{4.3}$$

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

$$b\left(i+1, j+1, \left\lfloor \frac{s(i,j,\ell)+S_0 u^{i-j} d^{j+1}}{(n+1)X/k_{i+1,j+1}} \right\rfloor\right). \tag{4.4}$$

See Fig. 4.2 for illustration. Bucket $b(i,j,\ell)$ therefore **covers** the prefix sums $s$ in range

$$\ell(n+1)X/k_{ij} \leq s < (\ell+1)(n+1)X/k_{ij}. \tag{4.5}$$

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.

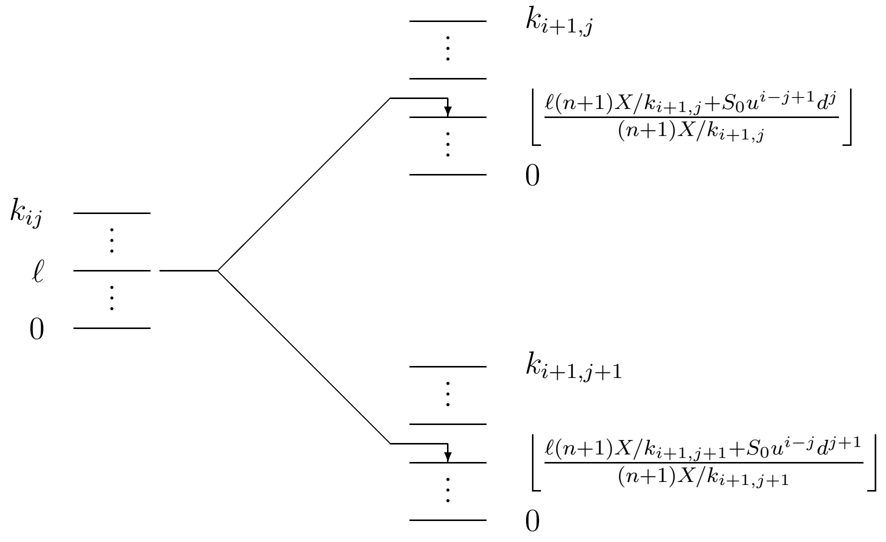

$k_{i+1,j}$
$\left\lfloor \frac{\ell(n+1)X/k_{i+1,j}+S_0u^{i-j+1}d^j}{(n+1)X/k_{i+1,j}} \right\rfloor$
0
$k_{ij}$
$\ell$
0
$k_{i+1,j+1}$
$\left\lfloor \frac{\ell(n+1)X/k_{i+1,j+1}+S_0u^{i-j}d^{j+1}}{(n+1)X/k_{i+1,j+1}} \right\rfloor$
0

Figure 4.2: Rounding Down the Prefix Sums.
Each of the $k_{ij}+1$ buckets at $N(i,j)$ moves up to the bucket of node $N(i+1,j)$ and down to the bucket of node $N(i+1,j+1)$ as shown. In the rounding-up version, the floor operations are replaced with the ceiling operations.

We call this algorithm `nUnifDown`. The prefix `nUnif` emphasizes the nonuniformity of bucketing, as the number of buckets, $k_{ij}+1$, may vary from nodes to nodes. The suffix `Down` means that the successor buckets are located by rounding down. If we change the rounding-down operations in Eq. (4.3) and (4.4) to rounding-up, the resulting algorithm is called, naturally, `nUnifUp` with the suffix `Up`. Evidently, `nUnifDown` and `nUnifUp` bracket the desired option value (Desired):

$$\texttt{nUnifDown} \leq \texttt{Desired} \leq \texttt{nUnifUp}. \tag{4.6}$$

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 4.3.1. 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 which follow Hull-White paradigm with linear interpolation are upper-bound algorithms. The forward shooting grid method in [4] finds the successor bucket by rounding to the nearest bucket, which is either the up bucket or the down bucket.

By picking a uniform $k_{ij} = k$ to make the number of buckets the same $k+1$ for every node, the algorithm in Aingworth et al. [2] is a special case of our `nUnifDown`. We shall call it `UnifDown` to emphasize its uniformity of bucketing. Its rounding-up version shall be labeled `UnifUp`. The range-bound result in Eq. (4.6) clearly does not depend on bucket-distribution scheme. A very popular alternative is for the logarithms of the buckets to be equally distanced [26]. All our algorithms and all the theoretical results (except the one with a specific error bound, Theorem 4.4.1) are independent of the ways the buckets are distributed.

### 4.4.2 An Optimal Choice of the Number of Buckets

Denote the rounding error at node $N(i,j)$ by $D_{ij}$. Then the pricing error's upper bound Eq. (4.1) is further bounded above by

$$\frac{1}{n+1}\sum_{i=0}^{n}\sum_{j=0}^{i}\binom{i}{j}P^{u}_{i-j}P^{j}_{d}|D_{ij}|. \tag{4.7}$$

Because $|D_{ij}|$ are not identically weighted in Eq. (4.7), a thoughtful choice of $k_{ij}$ can reduce the pricing error.

Set

$$\texttt{TIME} = \sum_{0\le j\le i\le n} k_{ij}, \tag{4.8}$$

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 $B(i,j;p) \equiv \binom{i}{j}P_u^{i-j}P_d^j$, pricing error illustrated in Eq. (4.7) is bounded above by

$$\texttt{ERROR} = X\sum_{i=1}^{n}\sum_{j=0}^{i}\frac{B(i,j;p)}{k_{ij}}. \tag{4.9}$$

Apply the Cauchy-Schwartz inequality to Eq. (4.8) and (4.9) to obtain

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

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

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

The pricing error's upper bound,

$$\frac{X}{\texttt{TIME}} \times \left[ \sum_{0 \leq j \leq i \leq n} \sqrt{B(i,j;p)} \right]^2 , \tag{4.10}$$

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

The algorithm can now be completely specified with

$$k_{ij} = \left\lceil \texttt{TIME} \times \frac{\sqrt{B(i,j;p)}}{\sum_{0 \leq j \leq i \leq n} \sqrt{B(i,j;p)}} \right\rceil . \tag{4.11}$$

Eq. (4.8) and the choice

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

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

### 4.4.3 Error Analysis

We proceed to derive an upper bound on the pricing error in Eq. (4.10) of `nUnifDown`. The same proof works for `nUnifUp`. Recall that $p = 1/2$. Bender (1974) shows that [5]

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

Substitute the above into Eq. (4.10) to yield

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

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

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

asymptotically. We have therefore proved the following theorem.

**Theorem 4.4.1** *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 Eq. (4.11).*

We make a few remarks regarding the error bound in the above theorem. Although $k_{ij}$ are chosen in an optimal way, optimality is relative to the upper bound in Eq. (4.9) on the pricing error and not the exact pricing error. Hence, a better choice than Eq. (4.11) is conceivable. By the same token, the same error bound may be achievable with much smaller $k_{ij}$ than asked for in the theorem. Finally, the bound is independent of the asset's price volatility $\sigma$.

## 4.5 Tighter Range Bounds

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

### 4.5.1 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 prefix as in `nUnifDown` or rounds up a path prefix as in `nUnifUp`. Instead, a path is split into two 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 round-down bucket $b(i+1,j,\texttt{RoundDown})$, whereas scheme `nUnifUp` directs the path to the round-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

$$s = \lambda \cdot s(i+1,j,\texttt{RoundUp}) + (1-\lambda)\cdot s(i+1,j,\texttt{RoundDown}) \tag{4.12}$$

for $\lambda$. It then gives proportions $\lambda$ and $1 - \lambda$ of $p(i, j, k)$—the probabilistic weight of $b(i, j, k)$—to the round-up bucket and the round-down bucket, respectively. See Fig. 4.3 for illustration. In the unlikely event that `RoundDown` = `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).

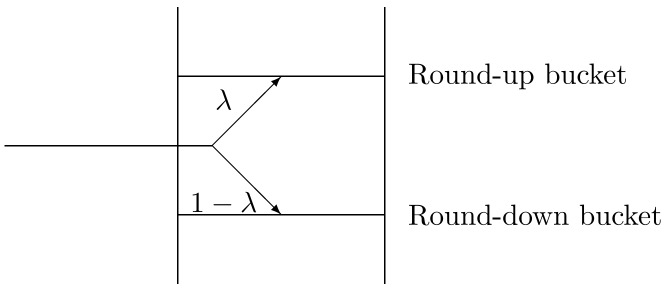


Figure 4.3: Splitting a Path.
The round-up bucket receives $\lambda$ of the probabilistic weight. The round-down bucket receives $1 - \lambda$ of the probabilistic weight.

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 4.3.1 takes over. Each of the paths is rounded to buckets along the way to prevent combinatorial explosion. Furthermore, for any $0 \leq m \leq n$, a path's prefix sum of $m$ prices equals the expected length-$m$ prefix sum of all the split paths.

### 4.5.2 A Tighter Lower-Bound Algorithm

The core idea of our next algorithm is to use

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

to approximate the desired option value in Eq. (2.9). The approximation underestimates the option value because of Jensen's inequality. Our algorithm adds bucketing to the above idea.

To implement the idea with bucketing, the prefix sums $s(i, j, \ell)$ will no longer be fixed by Eq. (4.2). Instead, they need to be calculated explicitly to hold the average prefix sum of all the paths covered by bucket $b(i, j, \ell)$ with range Eq. (4.5). Because any path prefix ending up at bucket $b(i, j, \ell)$ carries the same probability $P_u^{i-j} P_d^{j}$, $s(i, j, \ell)$ equals the simple arithmetic average of those said prefix sums. Probability $p(i, j, \ell)$ still records the probability of reaching $b(i, j, \ell)$ from the root and is calculated the same way as in `nUnifDown`. But the expected prefix sum of the round-up bucket,

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

is updated by adding $P_u \cdot p(i,j,\ell) \cdot [\, s(i,j,\ell)+S_0u^{i-j+1}d^j\,]$ to it. Similarly, the expected prefix sum of the round-down bucket,

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

is updated by adding $P_d \cdot p(i,j,\ell) \cdot [\, s(i,j,\ell)+S_0u^{i-j}d^{j+1}\,]$ to it. After the above is done for every $s(i,j,\ell)$ at time $i$ and before we move on to time $i+1$, every nonzero $s(i,j,\ell)$ is divided by $p(i,j,\ell)$ to turn it into the average prefix sum. The rest of the algorithm is identical to `nUnifDown`.

We illustrate the idea in Fig. 4.4 without using bucketing and applying Lemma 4.3.1 for simplicity. Take the node with probability $2P_uP_d$ for example. Its average prefix sum is calculated by adding up the path prefix through the $S_0u$ node $(S_0, S_0u, S_0)$ and that through the $S_0d$ node $(S_0, S_0d, S_0)$. The result is

$$\begin{aligned} & P_dP_u(S_0+S_0u+S_0)+P_uP_d(S_0+S_0d+S_0) \\ = \;& P_uP_d[\,(S_0+S_0u+S_0)+(S_0+S_0d+S_0)\,] \end{aligned}$$

Its associated probability is calculated in the standard way,

$$P_uP_d+P_dP_u=2P_uP_d.$$

The average prefix sum is obtained by dividing the nonzero prefix sum by the above probability:

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

The other average prefix sums at maturity are given in Fig. 4.4. The approximation is hence

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

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

**Theorem 4.5.1** `nUnifDown` $\le$ `nUnifCvg` $\le$ `Desired` $\le$ `nUnifSpl` $\le$ `nUnifUp`.

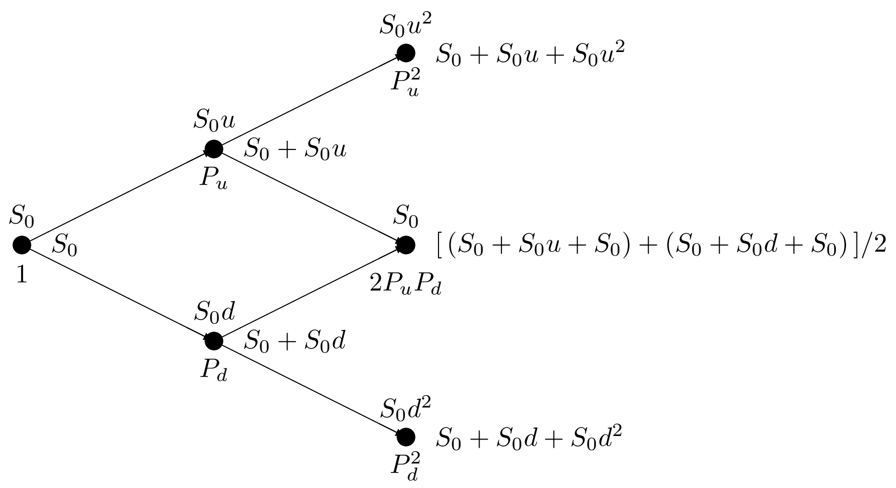


Figure 4.4: Prefix Sum Averaging with `nUnifCvg`.
The average prefix sums are listed to the right of the nodes. The probabilities for the average prefix sums are listed under the nodes. The asset's price for each node is listed above the node.

***Proof.*** We knew that `nUnifDown` $\leq$ `Desired` $\leq$ `nUnifUp`. It should also be clear that `nUnifDown` $\leq$ `nUnifCvg` and `nUnifSpl` $\leq$ `nUnifUp`. That `nUnifCvg` $\leq$ `Desired` 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 by Jensen's inequality.

It remains to show that `Desired` $\leq$ `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 split path $x$, say $A_x$, equals the average price of $P$, i.e., $E[\,A_x\,] = \sum_{i=0}^{n} S_i/(n{+}1)$. By Jensen's inequality, $P$'s payoff, $[\sum_{i=0}^{n} S_i/(n+1) - X]^+$, is dominated by `nUnifSpl`'s approximation, $E[\,(A_x - X)^+\,]$. As this inequality holds for every path, the desired result follows. Q.E.D.

If `nUnifCvg:nUnifSpl` chooses the numbers of buckets, $k_{ij}$, according to Eq. (4.11), then Theorem 4.4.1 implies an error bound of $O(X\sqrt{n}/k)$. Our experiments in the next section will show that this error bound seems overpessimistic.

## 4.6 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 desired option 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 Eq. (4.11) 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. 4.5 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.

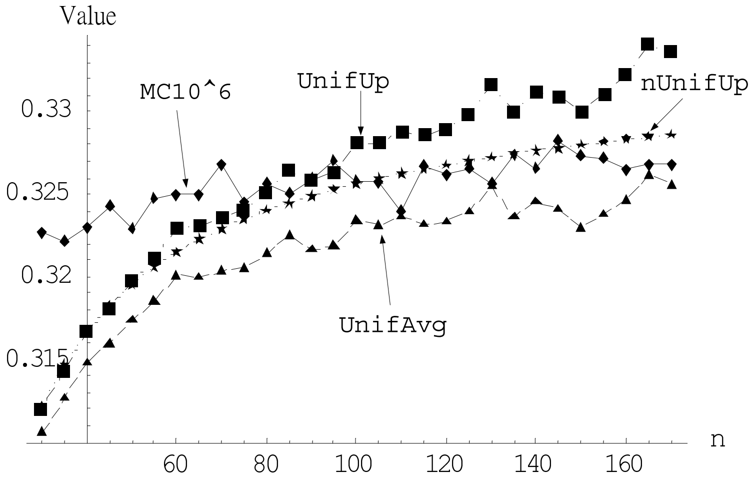


Figure 4.5: Comparing `UnifUp`, `UnifAvg`, and `nUnifUp`.
MC10ˆ6 denotes Monte Carlo simulation based on 1,000,000 trials. Both `UnifAvg` and `UnifUp` allocate $k = 50,000$ buckets at each node, and `nUnifUp` allocates an average of $k = 50,000$ buckets per node. The desired option option values must lie below `nUnifUp`. The data are: $S_0 = 50$, $X = 60$, annual volatility $\sigma = 30\%$, $r = 10\%$ per annum, and maturity $T = 0.5$ (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 (MR) algorithm in Dai and Lyuu [16]. Again, both `nUnifCvg` and `UnifAvg` use the same total number of buckets. The results are tabulated in Table 4.1. 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 the trinomial lattice, is the best when $n$ is small. But since its running time is sub-exponential in $n$, it is not competitive when $n$ is large. We then add the algorithm suggested by Hull and White [26] and Levy's analytical approach [32] to the comparisons in Table 4.2. Observe that `nUnifCvg` is competitive in all the

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

Table 4.1: Monte Carlo Simulation, `nUnifCvg`, `UnifAvg`, and Multiresolution Lattice. 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 [16]. At $n \geq 174$, the demand of the multiresolution lattice for computer memory is beyond what the system can offer. The computation times are measured in seconds. Asterisks mark those answers which are out of the 95% confidence interval. The data are: $S_0 = 50$, $X = 60$, $r = 10\%$ per annum, $\sigma = 30\%$, and $T = 0.5$ year.

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. [21] are tested in Table 4.3, `nUnifCvg` does not display any discernible biases. These tests strongly suggest that `nUnifCvg` gives extremely tight lower bounds.

Both `nUnifCvg` and `nUnifSpl` are more sophisticated strategies to handle pricing errors than `UnifUp`, `UnifDown`, or `UnifAvg`. We now demonstrate that they are also better strategies. For the purpose of fair comparison between these methods, we reduce the number of buckets for `nUnifCvg` and `nUnifSpl` so that they take slightly less time than `UnifUp`. The results are illustrated in Fig. 4.6. 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

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

Table 4.2: Comparing `nUnifCvg` against Various Algorithms.
"HW" denotes the Hull-White algorithm [26] 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 denotes the multiresolution approach based on $n = 30$. "L" denotes the analytical approach described in Levy [32]. Algorithm `nUnifCvg` sets $k = \lfloor 50,000/7 \rfloor$. The data are: $S_0 = 50$, $r = 10\%$ per annum, and $\sigma = 30\%$.

of a band. The same cannot be said of `UnifDown:UnifUp`. As `nUnifCvg:nUnifSpl` brackets the desired option value (Theorem 4.5.1), 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 accrue to different algorithms, not bucketing schemes. We use the range bounds to measure the pricing error, i.e., `nUnifSpl` − `nUnifCvg` and `nUnifUp` − `nUnifDown`. It is clear from Fig. 4.7 that `nUnifCvg:nUnifSpl` produces much tighter range bounds, hence smaller errors, than `nUnifDown:nUnifUp`.

It is not paradoxical for the pricing error to rise with $n$ in Fig. 4.7. The reason is that our $k$ in the experiments are constants independent of $n$. Suppose we now pick $k = n$ to make `nUnifCvg:nUnifSpl`'s time bound $O(n^3)$. The data in Table 4.4 show

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

Table 4.3: Comparing `nUnifCvg` against the Analytical Approaches.
All the notations are the same as the ones in Table 2.1 except that `Cvg` stands for `nUnifCvg`. As before, `nUnifCvg` sets $k = \lfloor 50,000/7 \rfloor$.

a convergence rate of $O(n^{-2})$. This is much better than the $O(n^{-0.5})$ guaranteed by Theorem 4.4.1, which overestimates the pricing error. They also demonstrate that the resulting range bounds are extremely tight for $\sigma$ as high as 100% and maturity as long as 5 years. Indeed, our theoretical error bounds are independent of $\sigma$. To our knowledge, Aingworth et al. (2000) is the only paper besides ours with numerical results at $\sigma > 50\%$ for European-style Asian options. Not surprisingly, that paper makes use of Lemma 4.3.1 too. Data in Section 4.9 strongly suggest that reducing the prefix sum range like the idea given by Lemma 4.3.1 is essential for any deterministic efficient convergent algorithms especially when $\sigma$ is large.

Finally, we are interested in knowing how $k$—the average number of buckets per node—impacts the accuracy of the algorithms. Fig. 4.8 shows the effects of $k$ on the theoretical error upper bound,

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

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

We conclude from the above experiments that `nUnifCvg:nUnifSpl` is an extremely efficient and accurate range-bound algorithm for European-style Asian options.

## 4.7 Algorithms for American-Style Asian Options

The early-exercise feature makes American-style Asian options harder to price than their European-style counterparts. The feature, for instance, invalidates Lemma 4.3.1, which has been instrumental in the pricing of European-style Asian options. To find the optimal exercise strategy for American-style Asian options, backward induction must somehow be used. The algorithms to follow focus on calls. Their extension to puts is straightforward.

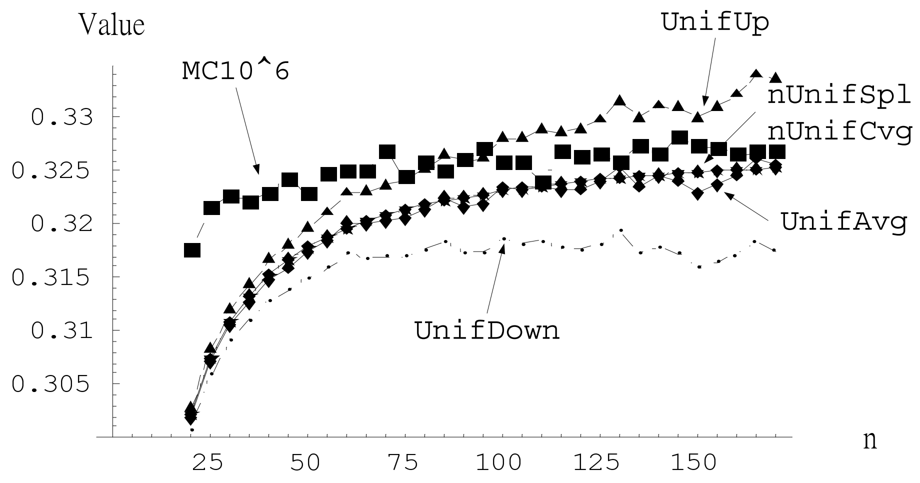


Figure 4.6: Comparing `nUnifCvg` and `nUnifSpl` against `UnifUp`, `UnifDown`, and `UnifAvg`.
`UnifUp`, `UnifDown`, and `UnifAvg` each set $k = 50,000$, whereas `nUnifSpl` and `nUnifCvg` set $k = \lfloor 50,000/7 \rfloor$. These choices make `nUnifCvg` and `nUnifSpl` take slightly less time than `UnifUp`. The plots of `nUnifCvg` and `nUnifSpl` essentially coincide. The data are: $S_0 = 50$, $X = 60$, $\sigma = 30\%$, $r = 10\%$ per annum, and $T = 0.5$ year.

### 4.7.1 Useful Terminology

We first introduce a few terms for ease of later discussions. Let $\mathcal{R}_{\max}(i,j)$ and $\mathcal{R}_{\min}(i,j)$ denote the maximum and the minimum prefix sums[3] at node $N(i,j)$.
Obviously, $\mathcal{R}_{\max}(i,j)$ is achieved by the path that makes $i-j$ up moves followed by $j$ down moves, whereas $\mathcal{R}_{\min}(i,j)$ is achieved by the path that makes $j$ down moves followed by $i-j$ up moves. Both are straightforward to calculate (see [35]). The prefix-sum range at node $N(i,j)$ is $[\,\mathcal{R}_{\min}(i,j), \mathcal{R}_{\max}(i,j)\,]$.

We now define the **ideal lattice**, a critical concept that allows us to derive the range bound proof. 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 our approximation 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 our approximation algorithms.

For any bucket $b$, we use $P_b$ instead of the earlier $s(\cdot,\cdot,\cdot)$ to denote its associated prefix sum for brevity. As before, the prefix sum $P_b$ includes the asset's price at $b$. Let $E_b$ denote the option value for bucket $b$. Because the option value in the ideal lattice may differ from that in the practical lattice, we use the superscripts $I$ and $P$ 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. The loose notion

[3]See the definitions in Section 2.6.

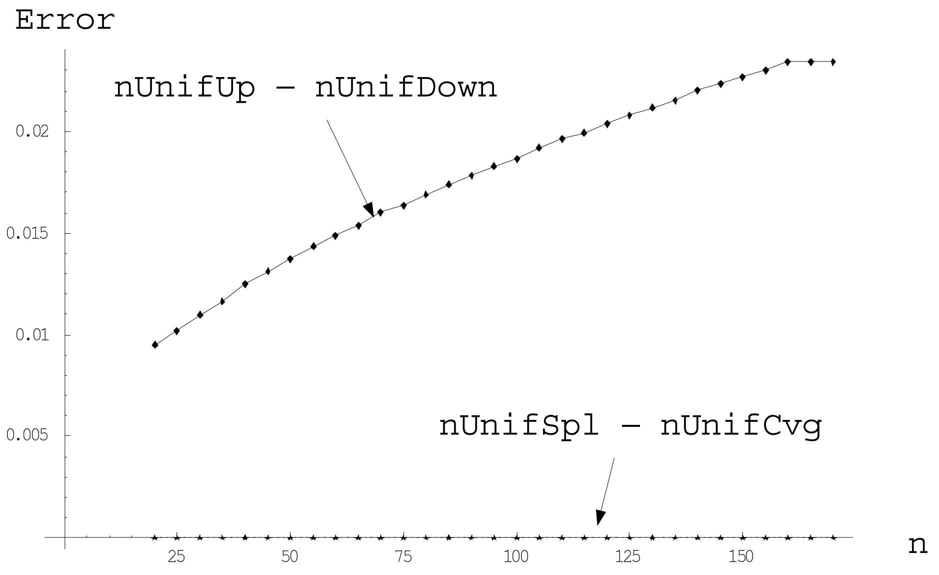


Figure 4.7: Pricing Errors of `nUnifCvg:nUnifSpl` and `nUnifDown:nUnifUp`. All four algorithms set $k = \lfloor 50,000/7 \rfloor$. The other parameters are identical to the ones used in Fig. 4.6.

of $E_b^P$ applies to any backward-induction algorithm which keeps the maximum of the early-exercise value and the continuation value as the option value for each bucket. Because the buckets used in the exact pricing algorithm is "embedded" in the ideal lattice, so to speak, the option value for the single bucket allocated at the root node in the ideal lattice equals the desired option value.

### 4.7.2 Full-Range Algorithms

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

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

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

The steps in Subsection 4.4.2 remain valid after $B(i,j;p)$ is replaced with $B(i,j;p)R_{ij}$. In particular, the optimal choice Eq. (4.11) for $k_{ij}$ becomes

$$k_{ij} = \left\lceil \texttt{TIME} \times \frac{\sqrt{B(i,j;p)R_{ij}}}{\sum_{0\le j\le i\le n} \sqrt{B(i,j;p)R_{ij}}} \right\rceil. \tag{4.13}$$

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

Table 4.4: Convergence of `nUnifCvg:nUnifSpl`.
Both algorithms use $k = n$. The data are: $S_0 = X = 100$ and $r = 10\%$ per annum. Algorithm `nUnifCvg:nUnifSpl` takes fewer than 35 seconds for the $n = 400$ case.

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

### 4.7.3 The First Upper-Bound Algorithm

The first algorithm is called `nUnifSplA` with the suffix `A` indicating American style. It is based on `nUnifSpl` with three straightforward modifications. First, backward induction is used instead of forward induction. But a new issue arises that needs to be addressed. Backward induction requires two option values from the following time. In the algorithm, each of these two option values will be linearly interpolated from the option values at the round-up bucket and the round-down bucket using the proportions $\lambda$ and $1 - \lambda$ in Eq. (4.12). See Fig. 4.9 for illustration. Second, early exercise is considered at each bucket. Exercise generates value $P_b/(i + 1) - X$ for a bucket $b$ at time $i$ (See Eq. (2.8).) Third, $k_{ij}$ buckets are allocated at node $N(i, j)$ for the prefix-sum range $[\,\mathcal{R}_{\min}(i, j), \mathcal{R}_{\max}(i, j)\,]$ instead of the earlier range $[\,0, (n+1)X\,]$.

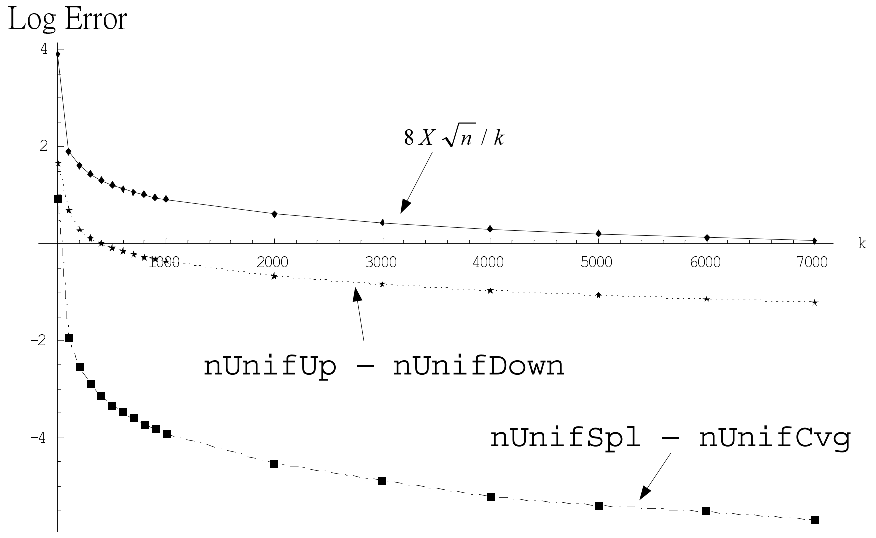


Figure 4.8: The number of buckets and accuracy.
This plot shows the log-plot (base 10) of the various error bounds vs. $k$. We use $n = 285$ here. The other parameters are identical to the ones used in Fig. 4.6.

### 4.7.4 The Second Upper-Bound Algorithm

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

**Lemma 4.7.1 (Contraction lemma)** *Suppose that $P_{b_1} > P_{b_2}$ at buckets $b_1$ and $b_2$ of the same node in the ideal lattice at time $m$. Then*

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} \geq E^I_{b_1} - E^I_{b_2}. \tag{4.14}$$

***Proof.*** Assume that bucket $b_i$ moves up to bucket $u(b_i)$ and down to bucket $d(b_i)$, $i = 1, 2$, in the ideal lattice. The lemma will be proved by induction on the level of the lattice. The base case involves the buckets allocated at the maturity date (at time $n$). There $E^I_{b_i} = \max(P_{b_i}/(n+1), X) - X$. 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 $P_{b_1} > P_{b_2}$. The induction step is divided into four cases.

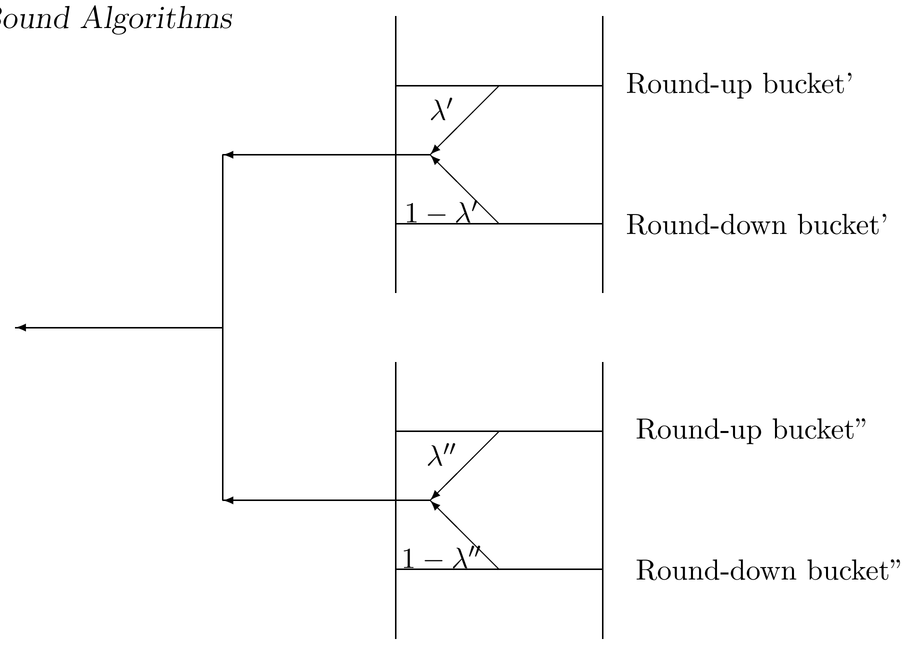


Figure 4.9: Interpolation of option values in `nUnifSplA` in backward induction. Linear interpolation is carried out at the successors. $\lambda'$ and $\lambda''$ are calculated from Eq. (4.12).

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

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

where the second inequality is by the induction hypothesis.

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

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

Subtract the inequality from the equality to obtain inequality (4.14).

**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 (4.14) 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

$$\frac{P_{b_1}}{m+1} - X \;<\; \left[\, P_u E^I_{u(b_1)} + P_d E^I_{d(b_1)} \,\right] e^{-r\Delta t}, \tag{4.15}$$

$$\frac{P_{b_2}}{m+1} - X \;\geq\; \left[\, P_u E^I_{u(b_2)} + P_d E^I_{d(b_2)} \,\right] e^{-r\Delta t}. \tag{4.16}$$

Subtracting inequality (4.16) from inequality (4.15) results in

$$\frac{P_{b_1}}{m+1} - \frac{P_{b_2}}{m+1} < \left\{ P_u [\, E^I_{u(b_1)} - E^I_{u(b_2)} \,] + P_d [\, E^I_{d(b_1)} - E^I_{d(b_2)} \,] \right\} e^{-r\Delta t}. \tag{4.17}$$

But

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

contradicting inequality (4.17). Q.E.D.

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

**Theorem 4.7.2** *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 $m$. Assume that it is optimal to exercise the option at bucket $b_2$. Then it is optimal to exercise the option at $b_1$.*

***Proof.*** We claim that

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

That the option value is at least the exercise value proves the first inequality. The second inequality is by Lemma 4.7.1. The equality holds because $b_2$ is an early-exercise bucket. As $E^I_{b_1} = P_{b_1}/(m+1) - X$, bucket $b_1$ is an early-exercise bucket. Q.E.D.

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

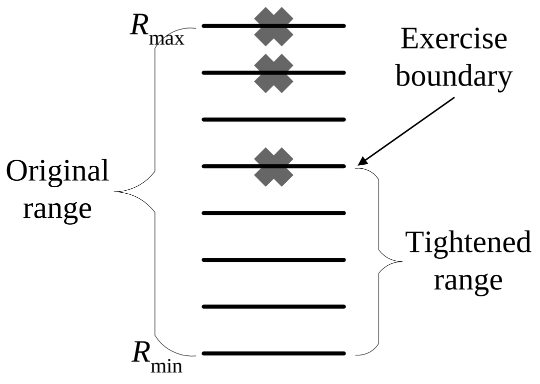


Figure 4.10: Determination of the exercise boundary.
Buckets marked with a "×" are early-exercise buckets under `nUnifSplA`.

Of course, it may happen that, in the practical lattice, the early-exercise buckets are scattered within a node without a clear boundary separating them from the non-early-exercise ones. We will settle with an approximate boundary at and above which all early-exercise buckets lie. See Fig. 4.10 for illustration. Theoretically, some non-early-exercise buckets may still lie above that boundary.

We now present a two-phase algorithm, called `nUnifSplA2`, that incorporates the idea of exercise boundary. Phase one uses `nUnifSplA` to estimate the exercise 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 `nUnifSplA` on the reduced prefix-sum ranges. The $k_{ij}$ need to be recalculated with Eq. (4.13) because ranges $R_{ij}$ have been reduced in phase one. A bucket with a prefix sum on or above the exercise boundary will be exercised in phase two.

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

### 4.7.5 A Lower-Bound Algorithm

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

## 4.8 The Range-Bound Proofs

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

**Theorem 4.8.1** `nUnifCvgA` $\leq$ `Desired`.

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

When a bucket is terminal, all the paths that pass through it terminate there. Let $\wp_b$ be the set of paths terminated at terminal bucket $b$ at time $t_b$ under `nUnifCvgA`. Every path in $\wp_b$ has length $t_b$. We now use those $\wp_b$ to define an exercise strategy on the 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 `nUnifCvgA`. This exercise strategy produces an option value $A$ that cannot exceed the desired option value because it may not be optimal.

We complete the proof by showing that `nUnifCvgA` generates an option value that cannot exceed $A$. Fix any terminal bucket $b$. Let prob$[\rho]$ denotes the probability of the path prefix $\rho$. The contribution of $\wp_b$ to the option value $A$ is

$$e^{-r\Delta t t_b} \sum_{\rho=(S_0,S_1,\ldots,S_{t_b})\in\wp_b} \text{prob}[\,\rho\,] \times \left\{ \frac{1}{t_b+1} \sum_{i=0}^{t_b} S_i - X \right\}^+ . \tag{4.18}$$

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

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

which is smaller than Eq. (4.18) by Jensen's inequality. By summing the contributions over all terminal buckets $b$, we conclude that `nUnifCvgA` gives a lower bound on $A$. Q.E.D.

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

**Lemma 4.8.2 (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 $\lambda$ satisfies $P_{b_2} = \lambda P_{b_1} + (1-\lambda)P_{b_3}$, then*

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

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

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

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

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

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

**Theorem 4.8.3** `Desired` $\leq$ `nUnifSplA`.

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

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

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

by Eq. (4.12), 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 4.8.2, and the second inequality is by the induction hypothesis. The equality holds because `nUnifSplA` 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. 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

$$E^I_b = \left[ P_u E^I_{u(b)} + P_d E^I_{d(b)} \right] e^{-r\Delta t} \leq \left[ P_u E^P_{u(b)} + P_d E^P_{d(b)} \right] e^{-r\Delta t} = E^P_b.$$

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

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

So it is also optimal to exercise $b$ in the ideal lattice as

$$(P_b/t) - X \geq \left[ P_u E^I_{u(b)} + P_d E^I_{d(b)} \right] e^{-r\Delta t}.$$

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, $E^I_b = (P_b/t) - X < E^P_b$.

Hence, $E^I_b \leq E^P_b$ in all cases, and the induction step is complete. Q.E.D.

Theorem 4.8.3 holds for a large class of algorithms, not just `nUnifSplA`. This is because the proof only requires that the option value at a non-existing bucket be linearly interpolated from the option values of its two bracketing buckets. Neither the number of buckets allocated per node nor the way the buckets are distanced in the prefix-sum range matters. It follows that the popular approximation algorithm in [26, 27] is an upper-bound algorithm.

**Corollary 4.8.4** *The Hull-White paradigm with linear interpolation is an upper-bound algorithm.*

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

**Corollary 4.8.5** *A bucket with a prefix sum equal to or larger than the exercised boundary determined by* `nUnifSplA` *must be an early-exercise bucket in the ideal lattices.*

***Proof.*** By case 2 in the proof of Theorem 4.8.3, an early-exercise bucket under `nUnifSplA` must also be an early-exercise bucket in the ideal lattice. Theorem 4.7.2 completes the proof. Q.E.D.

The above corollary implies that the exercise boundary determined by `nUnifSplA` 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 limited prefix sum ranges made possible by the exercise boundary estimated in phase one, it should outperform the original one-phase algorithm.

**Theorem 4.8.6** *An upper-bound algorithm that uses the estimated exercise boundary given by* `nUnifSplA` *remains an upper-bound algorithm as long as it works on the same 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 4.8.5. If bucket $b$ lies within the tightened prefix-sum range, then $E_b^I \leq E_b^P$ because of the algorithm being an upper-bound one and induction. Q.E.D.

For the upper-bound Hull-White algorithms in Corollary 4.8.4, for instance, the above theorem gives rise to two-phase versions. We finally prove that `nUnifSplA2` is an upper-bound algorithm.

**Corollary 4.8.7** `Desired` $\leq$ `nUnifSplA2`.

***Proof.*** It is immediate from Theorem 4.8.3 and Theorem 4.8.6. Q.E.D.

## 4.9 Numerical Results for American-Style Asian Options

Experimental results for `nUnifSplA`, `nUnifSplA2`, and `nUnifCvgA` will be presented below. All the experimental results reported here are based on an Athlon Thunderbird 1.33GHz PC with 1GB DRAM. Recall that `nUnifSplA` and `nUnifSplA2` provide upper bounds, whereas `nUnifCvgA` gives lower bounds (the claims will be proved in the next section). Fig. 4.11 plots the pricing errors of `nUnifCvgA:nUnifSplA` and `nUnifCvgA:nUnifSplA2` as functions of $n$. The pricing errors are measured by `nUnifSplA` $-$ `nUnifCvgA` and `nUnifSplA2` $-$ `nUnifCvgA`. Not surprisingly, both increase with $n$. But `nUnifSplA2` has a much smaller pricing error than `nUnifSplA`. Because the pricing errors of `nUnifCvgA:nUnifSplA2` are extremely small, the algorithm practically gives the desired option value.

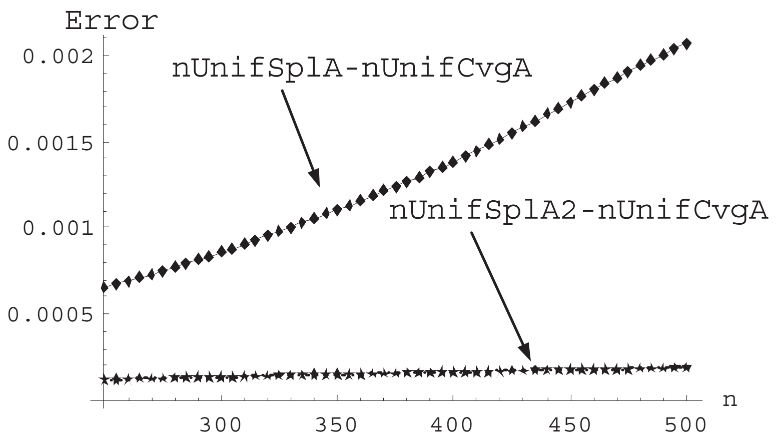


Figure 4.11: Pricing errors of `nUnifCvgA:nUnifSplA` and `nUnifCvgA:nUnifSplA2`. The data are: $S_0 = X = 100$, $\sigma = 30\%$, $r = 20\%$ per annum, and $T = 1$. All use an average of $k = 500$ buckets per node.

To further investigate the performance of `nUnifCvgA:nUnifSplA2` vs. `nUnifCvgA:nUnifSplA`, we perform a comprehensive test in Table 4.5. Although both algorithms perform well with small pricing errors, `nUnifCvgA:nUnifSplA2` is more competitive. Perhaps the most important lesson to draw from that table is that the pricing error rises as $\sigma$ increases. This phenomenon is most apparent in the case of `nUnifCvgA:nUnifSplA`. For the algorithm with a tighter range bound, `nUnifCvgA:nUnifSplA2`, the absolute pricing errors never exceed 0.000454. The advantage of the two-phase `nUnifCvgA:nUnifSplA2` again demonstrates the benefit of taking advantage of limited prefix sum ranges. Indeed, the data in Table 4.5 suggest that a $k$ of only 300 is sufficient in most cases to price the option accurately.

We next investigate the convergence behavior of `nUnifCvgA:nUnifSplA2`. The results tabulated in Table 4.6 are based on $k = 8n$; hence the running time is $O(n^3)$. To our knowledge, no papers in the literature on American-style Asian options mention numerical results for volatilities larger than 50%. Interestingly, Table 4.6 shows our $O(n^3)$ algorithm produces very tight range bounds for $\sigma$ as high as 100%. Even where the algorithm fails to converge when $\sigma = 100\%$ and $T = 5$ year, the relative errors are less than 0.14% up to $n = 400$.

As mentioned above, `nUnifCvgA:nUnifSplA2` fails to converge only when $\sigma$ and $T$ are both large ($\sigma = 100\%$ and $T = 5$ year in Table 4.6). This is in contrast to the case of European-style Asian options, where the related `nUnifCvg:nUnifSpl`'s convergence is not affected by a large $\sigma$ as shown in Table 4.4. The discrepancy can be attributed to the fact that prefix sum ranges in the case of European-style Asian options are limited from above by $(n + 1)X$ but no such limits are available in the case of American-style Asian options. Indeed, without this limit, the full-

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

Table 4.5: Comprehensive Tests for `nUnifSplA`, `nUnifSplA2`, and `nUnifCvgA`. The data are: $S_0 = 100$, $n = 300$, $k = 500$ and $T = 1$. Algorithms `nUnifCvgA`, `nUnifSplA2`, and `nUnifSplA` take an average of 4.60 seconds in generating their respective results.

range `nUnifCvg:nUnifSpl` also fails to converge for large $\sigma$ or large $T$. (Because $u = e^{\sigma\sqrt{T/n}}$, the prefix sum ranges depend on $\sigma$ and $T$ in an exponential manner.) This fact is confirmed in Table 4.7. We conjecture that, without some limits on prefix sums, deterministic algorithms will eventually fail when $\sigma$ is large enough unless $T$ is small or the number of buckets per node, $k$, is properly increased.

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

Table 4.6: Convergence of `nUnifCvgA:nUnifSplA2`.
Both algorithms use $k = 8n$. The data are: $S_0 = X = 100$ and $r = 10\%$ per annum.

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

Table 4.7: Convergence of the Full-Range `nUnifCvg:nUnifSpl`.
The setup is identical to Table 4.4 except that $k = 8n$. The full-range algorithm may fail to converge despite that more buckets are allocated than in Table 4.4

# Chapter 5

# Conclusions

Asian options are a kind of strongly path-dependent derivatives. How to price such derivatives efficiently and accurately has been a long-standing research and practical problem. Asian options can be priced on the lattice. Unfortunately, only exponential-time algorithms are currently available if such options are to be priced on the lattice exactly. Although efficient approximation methods are available, most of them lack convergence guarantees or error controls. This dissertation addresses the Asian option pricing problem with two different lattice methods to meet the efficiency and accuracy requirements. First, a new trinomial lattice for pricing Asian options is proposed, and the resulting exact pricing algorithm is proved to be the first one to break the exponential-time barrier. Second, range-bound algorithms are developed. These approximation algorithms are proved to converge to the true option value for pricing European-style Asian options. Extensive experiments also reveal that the extension of these algorithms work well numerically for American-style Asian options.

# Bibliography

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

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

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

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

[5] Bender, E.A. (1974). Asymptotic Methods in Enumeration. *SIAM Review*, Vol. 16, No. 4, pp. 485–515.

[6] Benhamou, E. (2002). "Fast Fourier Transform for Discrete Asian Options." *Journal of Computational Finance*, Vol. 6, No. 1, pp. 49–68.

[7] Black, F., and M. Scholes. (1973). "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy*, Vol. 81, No. 3, pp. 637-659.

[8] Bouaziz,L., E. Briys, and M. Crouhy, (1994). "The pricing of forward-starting Asian options." *Journal of Banking and Finance*, Vol. 18, pp. 823–839.

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

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

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

[12] Carverhill, A., and L. Clewlow. "Flexible Convolution." In *From Black Scholes to Black Holes*, Risk Books, London, pp. 165–171.

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

[14] Choa, H.Y., and H.Y. Lee. (1997). "A Lattice Model for Pricing Geometric and Arithmetic Average Options." *Journal of Financial Engineering,* Vol. 6, No. 3, pp. 179–191.

[15] Cox, J. C., A. Ross, and M. Rubinstein. (1979). "Option Pricing: a Simplified Approach." *Journal of Financial Economics,* Vol, 7, No. 3, pp. 229–263.

[16] Dai, T.-S., and Y.-D. Lyuu. (2002). "Efficient, Exact Algorithms for Asian Options Algorithms with Multiresolution Lattices." *Review of Derivatives Researches,* Vol. 5, No. 2, pp. 181–203.

[17] Dai, T.-S., and Y.-D. Lyuu. (2002). "Extremely Accurate and Efficient Algorithms for Asian Options with Error Bounds." *Quantitative Methods in Finance Conference*, Cairns, Australia, Dec. 2002.

[18] Dai, T.-S., and Y.-D. Lyuu. (2004). "An Exact Subexponential-Time Lattice Algorithm for Asian Options." In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, New Orleans, LA, January 11-13, 2004, to appear.

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

[20] Forsyth, P. A., K. R. Vetzal, and R. Zvan. (2002). "Convergence of Numerical Methods for Valuing Path-Dependent options Using Interpolation." *Review of Derivatives Research,* Vol. 5, pp. 273–314.

[21] Fu, M.C., D.B. Dilip, and T. Wang. (1998/9). "Pricing Continuous Asian Options: a Comparison of Monte Carlo and Laplace Transform Inversion Methods." *Journal of Computational Finance,* Vol. 2, No. 2, pp. 49–74.

[22] Geman, H., and A. Eydeland. (1995). "Domino Effect." *Risk,* Vol. 8, No. 4, pp. 65–67.

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

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

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

[26] Hull, J., and A. White. (1993)."Efficient Procedures for Valuing European and American Path-Dependent Options." *Journal of Derivatives,* Vol. 1, pp. 21–31.

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

[28] Kemna, A., and A. Vorst. (1990). "A Pricing Metohd for Options Based on Average Values." *Journal of Banking and Finance*, Vol. 14, pp. 113–129.

[29] Klassen, T. R. (2001). "Simple, Fast and Flexible Pricing of Asian Options." *Journal of Computational Finance*, Vol. 4, pp. 89-124.

[30] Klebaner, F. C. (1998). *Introduction to Stochastic Calculus with Applications.* London U.K.: Imperial College Press.

[31] Lapeyre, B., and E. Temam. (2001). "Competitive Monte Carlo Methods for the Pricing of Asian Options." *Journal of Computational Finance*, Vol. 5, No. 1, pp. 39–57.

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

[33] Longstaff, F.A., and E.S. Schwartz. (2001). "Valuing American Options by Simulation: a Simple Least-Squares Approach." *Review of Financial Studies*, Vol. 14, No. 1, pp. 113–147.

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

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

[36] Merton, R.C. (1994). *Continuous-Time Finance.* Revised ed. Cambridge, MA: Blackwell.

[37] Milevsky, M.A., and S.E. Posner. (1998). "Asian Options, the Sum of Lognormals, and the Reciprocal Gamma Distribution." *Journal of Financial and Quantitative Analysis*, Vol. 33, No. 3, pp. 409–422.

[38] Rogers, L.C.G., and Z. Shi. (1995). "The Value of an Asian Option." *Journal of Applied Probability,* Vol. 32, No. 4, pp. 1077–1088.

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

[40] Tsao, C. Y., C. C. Chang, and C, C, Lin. (2003). "Analytical Approximation Formulae for Pricing Forward-Starting Asian options." *The Journal of Futures Markets,* Vol. 23, pp. 487–516.

[41] Turnbull, S.M., and L.M. Wakeman. (1991). "A Quick Algorithm for Pricing European Average Options." *Financial and Quantitative Analysis,* Vol. 26, No. 3, pp. 377–389.

[42] Zhang, J.E. (2001). "Method for Pricing and Hedging Continuously Sampled Average Rate Options." *Journal of Computational Finance,* Vol. 5, No. 1, pp. 59–79.

[43] Zvan, R., and K. Vetzal. (1999). "Discrete Asian Barrier Options." *Journal of Computational Finance,* Vol. 3, pp. 41–68.

# Index