國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering & Computer Science

National Taiwan University

Master Thesis

評價巴黎選擇權之財務演算法：組合學、模擬法與平行處理

Pricing Parisian Options: Combinatorics, Simulation, and Parallel Processing

吳承瑋

Cheng-Wei Wu

指導教授：呂育道 博士

Advisor: Yuh-Dauh Lyuu, Ph.D.

中華民國 97 年 6 月

June, 2008

摘要

財務工程與金融創新在過去數十年蓬勃發展，設計出許多新金融商品，提供了風險管理所需的避險工具並促進市場效率與完整性。此財務領域的定價問題會嘗試建構數學模型推導公式解，但是由於大部分新奇衍生性金融商品契約複雜無公式能套用，必須借重電腦運算處理數值方法及模擬價格，因此計算機科學在此有其可施展之處。本篇論文探討巴黎選擇權評價方法即包含財務理論、機率統計、離散數學、計算複雜度、演算法設計與分析以及平行處理等議題。

巴黎選擇權爲路徑相關選擇權，迄今尚無封閉公式解存在，爲此我們提出兩種快速財務演算法。首先以 Costabile 於 2002 年建立在離散結構二項樹狀模型下的組合方法爲基礎去評價巴黎選擇權，再將此方法稍加修改可把原本的時間複雜度由$O\left(n^3\right)$成功推進至$O\left(n^2\right)$；若是組合數已給定的情況下，空間複雜度亦由$O\left(n^2\right)$減少到$O(n)$。另外，在蒙地卡羅模擬法方面，我們引進逆高斯機率分配並結合其抽樣，可減少時間切割的期數而縮短計算時間。因爲蒙地卡羅法的路徑模擬有各自獨立的特性，很容易使用平行運算處理。今日多核心處理器大行其道這不失爲提高效率的方法，對此我們也做了相關的說明並加以應用。

我們以 C 語言實作論文中的財務演算法，執行於自行建構的高效能叢集運算平台。同步處理模擬之計算量，俾能有效使用系統效能。

關鍵字：巴黎選擇權, 障礙選擇權, 選擇權評價, 演算法, 二項樹模型, 組合方法, 蒙地卡羅模擬法, 逆高斯機率分配, 平行處理

Abstract

Financial engineering and financial innovation flourished in last decades. We have developed many new financial products to provide hedge instruments for risk management, and promoted market efficiency and completeness. The pricing problems of this financial field will try to build mathematical models and derive analytic pricing formulas. But most exotic derivatives are too complicated to derive formulas. We must use computers to handle numerical methods and simulations, so computer science can give them a favor. This thesis discusses pricing of Parisian options and includes a lot of subjects: financial theory, probability & statistics, discrete mathematics, computational complexity, design & analysis of algorithms, and parallel processing.

Parisian options are path-dependent options and their closed-form solutions are not available up to now. We propose two fast financial algorithms to solve it. First we price Parisian options based on a combinatorial approach in binomial tree by Costabile in 2002. To refine Costabile's algorithm, time complexity $O\left(n^3\right)$ can be reduced to $O\left(n^2\right)$; If ignore binomial coefficients are given, the space complexity $O\left(n^2\right)$ could be reduced to $O\left(n\right)$. Second on Monte Carlo simulation, we introduce the inverse Gaussian distribution and its sampling method. To combine simulations and the inverse Gaussian distribution sampling, it can reduce divided time intervals to save computational time. Because the paths generated by Monte Carlo simulation are independent, it is easy to apply parallel processing. Nowadays multi-core processors are very popular, it is also a good idea to enhance computational efficiency. We give some descriptions and applications on it.

All financial algorithms in this thesis are implemented by the C programming language. We execute the programs in our high-performance computing clustered platform, and deal with simulation jobs synchronously. Then the system can be fully exploited.

Keywords: Parisian options, barrier options, option pricing, algorithm, binomial tree model, combinatorial method, Monte Carol simulation, inverse Gaussian distribution, parallel processing

Contents

List of Figures

List of Tables

Chapter 1
Introduction

A derivative (or derivative security) is a financial instrument whose value depends on, or is derived from, the values of other, more basic, underlying assets such as commodities, equities, bonds, interest rates, exchange rates, indexes, or other derivatives. The main types of derivatives are futures, forwards, options, and swaps.

1.1 Options

There are two basic types of option. A call option gives the holder the right to buy the underlying asset by a certain date for a certain price. A put option gives the holder the right to sell the underlying asset by a certain price. The date specified in the contract is known as the expiration date. The price specified in the contract is known as the strike price. American and European options differ in when they can be exercised. American options can be exercised at any time up to the expiration date, whereas European options can be excised only on the expiration date [9].

There are two sides to every option contract. On one side is the investor who has taken the long position. On the other side is the investor who has taken a short position. Let K be the strike price and S_T the final price of the underlying asset. There are four types of European option position and their payoff:

1. Long a call, payoff= $\max(0, S_T - K)$
2. Long a put, payoff= $\max(0, K - S_T)$
3. Short a call, payoff= $\min(0, S_T - K)$
4. Short a put, payoff= $\min(0, K - S_T)$

Figure 1.1 shows these payoffs.

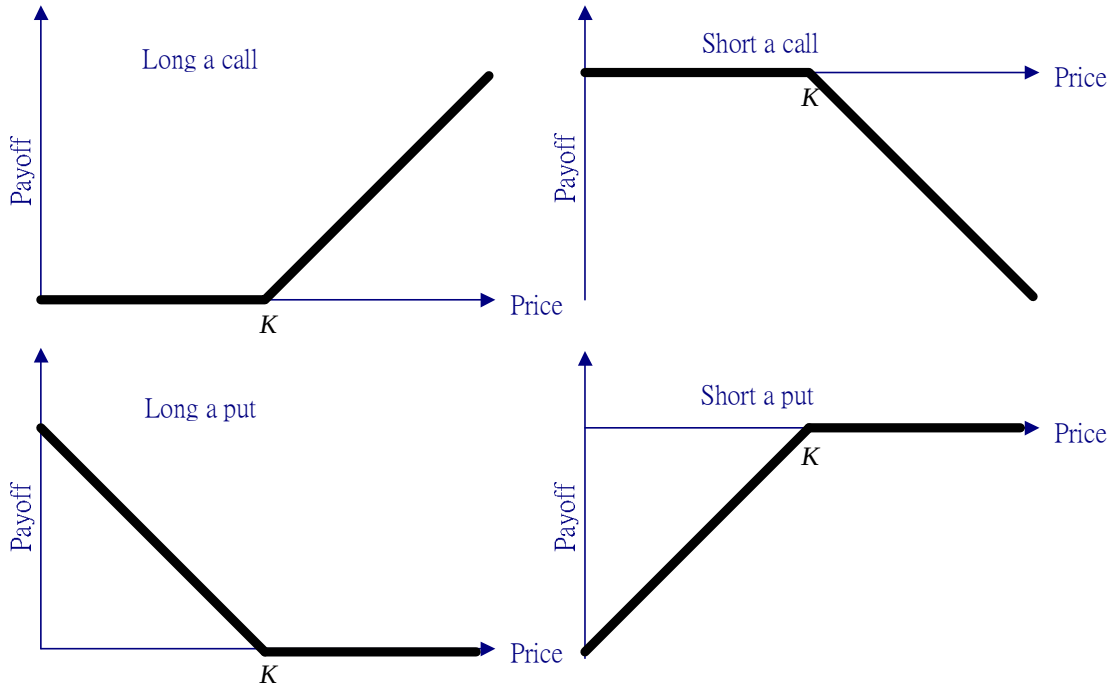


Figure 1.1: Payoff of options.

1.2 Barrier and Parisian Options

Options whose payoff depends on whether the underlying asset's price reaches a certain price H are called barrier options. These barrier options can be classified as either knock-out options or knock-in options. A knock-out option is like a standard European option except that it ceases to exist if barrier H is reached by the price of its underlying asset. A knock-in option, in contract, comes into existence if a certain barrier H is reached.

Let S_0 be the initial price of the underlying asset. A knock-out option is sometimes called a down-and-out option if $H < S_0$. A knock-out option is sometimes called an up-and-out option if $H > S_0$. A down-and-in option is a knock-in option that comes into existence only when the barrier is reached and $H < S_0$ (see Figure 1.2). An up-and-in option is a knock-in option that comes into existence only when the barrier is reached and $H > S_0$ [12]. Because there are two basic types of options, i.e., calls and puts, there are eight types of standard barrier option:

1. down-and-out call;
2. down-and-out put;
3. up-and-out call;
4. up-and-out put;
5. down-and-in call;

6. down-and-in put;
7. up-and-in call;
8. up-and-in put.

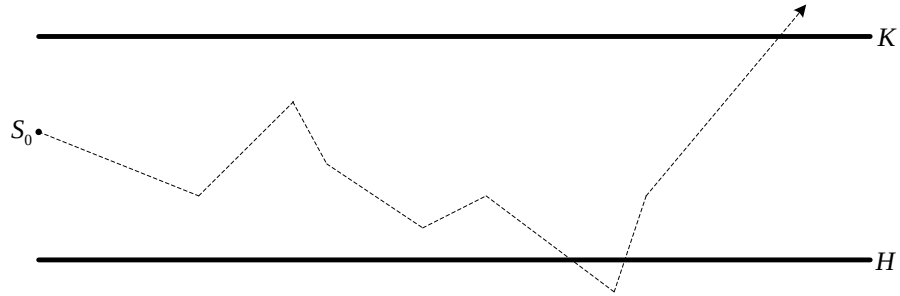


Figure 1.2: A sample underlying asset's price path of a down-and-in option.

Barrier options have many variations. If the barrier must be breached for a particular length of time, we have a Parisian option. Like common barrier options, it can be of the form of a down-and-out, up-and-out, down-and-in, up-and-in call or put. An up-and-out (up-and-in) Parisian option is an option that expires (comes into existence, respectively) if the underlying asset's price remains continuously at or above a given barrier over a pre-specified time interval, the so-called window period. Conversely, a down-and-out (down-and-in) Parisian option expires (is activated, respectively) if the underlying asset's price remains continuously at or below a lower barrier over the window period [6].

1.3 Thesis Structure

There are five chapters in this thesis. We introduce new financial products in this Chapter. Chapter 2 reviews the background knowledge and basic numerical techniques. Chapter 3 describes the combinatorial approach for pricing Parisian options by Costabile and how to speed up Costabile's algorithm. Chapter 4 discusses how to use Monte Carlo simulation with the inverse Gaussian distribution for pricing Parisian options. Chapter 5 concludes.

Chapter 2

Fundamental Concepts

In this chapter, we describe pricing models and methods. It covers the Wiener process, the Black-Scholes option pricing model, the risk-neutral valuation, and the binomial options pricing model.

2.1 Wiener Process

A Wiener process is a particular type of Markov stochastic process. It is sometimes referred to as normalized Brownian motion. A process, z, follows a Wiener process if it has the following two properties:

1. The change Δz during a small period of time Δt is
$$\Delta z = \varepsilon\sqrt{\Delta t},$$
where ε has a standardized normal distribution.
2. The value of Δz for any two difference short intervals of time, Δt, are independent.

That is a stochastic process where the change in a variable during each short period of time of length Δt has a normal distribution with a mean equal to zero and a variance equal to Δt.

We assume that our stock price follows the stochastic process:
$$dS = \mu S\, dt + \sigma S\, dz,$$
where σ is the volatility of the stock price, and μ is its expected rate of return. This equation is the most widely used model of stock price behavior and is also called the geometric Brownian motion [9].

2.2 Black-Scholes Option Pricing Model

In the year 1973, Fisher Black and Myron Scholes published the well-known option pricing model now universally known as the Black-Scholes option pricing model [2]. Several assumptions are made in the model as follows:

1. The stock price follows the process $dS = \mu S\, dt + \sigma S\, dz$ with constant μ and σ.
2. The short selling of securities with full use of proceeds is permitted.
3. There are no transactions costs or taxes.
4. All securities are perfectly divisible.
5. There are no dividends during the life of the derivative.
6. There are no riskless arbitrage opportunities.

7. Security trading is continuous
8. The risk-free rate of interest, r, is constant and the same for all maturities.

Black and Scholes derived the following formula for the price at time 0 of a European call and a European put:

$$C = S_0 N(d_1) - Ke^{-rT} N(d_2)$$
$$P = Ke^{-rT} N(-d_2) - S_0 N(-d_1),$$

where

$$d_1 = \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\ln(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

$N(x)$ = cumulative probability distribution function for a standardized normal distribution
S_0 = stock price at time zero
K = strike price
r = continuously compounded risk-free interest rate
σ = stock price volatility
T = time to maturity of the option

When the stock provides a continuous dividend yield at rate q, we obtain the price of a European call and a European put as, respectively,

$$C = S_0 e^{-qT} N(d_1) - Ke^{-rT} N(d_2)$$
$$P = Ke^{-rT} N(-d_2) - S_0 e^{-qT} N(-d_1),$$

and the parameters d_1 and d_2 are given by

$$d_1 = \frac{\ln(S_0/K) + (r - q + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\ln(S_0/K) + (r - q - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}.$$

These formulas, due to Merton [13], remain valid even if the dividend yield is not constant during the life of the option as long as q is replaced by the average annualized dividend yield during the life of the option.

2.3 Risk-Neutral Valuation

Risk-neutral valuation of derivatives assumes in a risk-neutral world where investors

are assumed to require no extra return on average for bearing risks. It gives the correct price for a derivative in all worlds, not just in a risk-neutral world. This means that for valuation purposes we can use the following procedure:

1. Assume that the expected return from the underlying asset is risk-free interest rate, r.
2. Calculate the expected payoff from the derivative.
3. Discount the expected payoff at risk-free interest rate.

Suppose the asset provides a yield of q. The expected return in the form of capital gain must be $r-q$.

In a risk-neutral world, the current price of a derivative then is given by

$$\text{Price}=e^{-rT}E\left[f\left(S_0,\cdots,S_T\right)\right],$$

where T is the maturity date of derivative, $f\left(S_0,\cdots,S_T\right)$ is the derivative's payoff, which may be dependent on entire price history of underlying asset, and $S_0,\cdots,S_T$ is the history of prices for the underlying asset from $t=0$ to T.

2.4 Binomial Option Pricing Model

John Cox, Stephen Ross and Mark Rubinstein developed the original version of the binomial option pricing model (BOPM) [7]. The BOPM is a binomial tree (lattice) algorithm. The binomial model is a discrete time and discrete variable pricing model. It suffers from two unrealistic assumptions:

1. The stock price takes on only two values in a time step.
2. Trading occurs at discrete points in times.

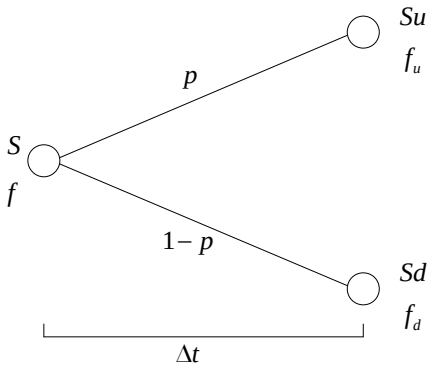


Figure 2.1: One time step in binomial tree.

First we assume the stock price follows the process $dS=\mu S\,dt+\sigma S\,dz$. For brevity, we use S for the current stock price here. Consider the stock at time

$\Delta t \equiv T/n$, where T is the time to maturity, and n is the number of time steps. It implies that

$$E\left[S(\Delta t)\right] = Se^{\mu \Delta t}$$
$$Var\left[S(\Delta t)\right] = S^2 e^{2\mu\Delta t}\left(e^{\sigma^2 \Delta t} - 1\right) \to S^2\sigma^2\Delta t$$

Second we assume that we are working in a risk-neutral world, so $\mu = r$. As in Figure 2.1, it limits the stock moves from its current price S to one of two new values, Su and Sd, in each time step. It will increase to Su with probability p or decrease to Sd with probability $1-p$. The expected stock price at time Δt converges to $Se^{r\Delta t}$. Then we get:

$$pSu + (1-p)Sd = Se^{r\Delta t}.$$

The variance of a variable X is defined as $E(X^2) - \left[E(X)\right]^2$ [28]. The variance of the stock price at time Δt converges to $S^2\sigma^2\Delta t$. Then we get:

$$p(Su)^2 + (1-p)(Sd)^2 - \left(Se^{r\Delta t}\right)^2 = S^2\sigma^2\Delta t$$

or

$$pu^2 + (1-p)d^2 - e^{2r\Delta t} = \sigma^2\Delta t.$$

Cox, Ross and Rubinstein (CRR) model imposes

$$u = \frac{1}{d},$$

we can get the solution

$$u = e^{\sigma\sqrt{\Delta t}}$$
$$d = e^{-\sigma\sqrt{\Delta t}} \quad .$$
$$p = \frac{e^{r\Delta t} - d}{u - d}$$

The option on the stock whose current price is f. When stock price moves up to Su, the option price is f_u; when stock price moves down to Sd, the option price is f_d. The risk-neutral valuation gives

$$f = e^{-r\Delta t}\left(pf_u + (1-p)f_d\right).$$

Using the two values of the option in the next time step, we can evaluate the option price at a given node. This procedure is called backward induction because it works backward in time.

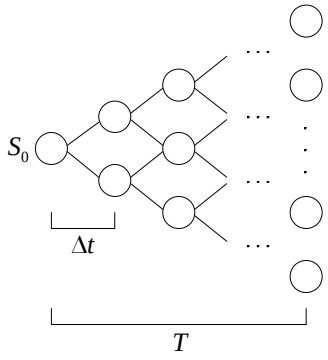


Figure 2.2: Binomial tree.

Consider binomial trees with more than one time step (see Figure 2.2). Note that the tree recombines in the sense that an up move followed by a down move leads to the same asset price as a down move followed by an up move. Note also that u and d are the same at each node of the tree and the time steps are of the same length, so the risk-neutral probability p is the same at each node.

By risk-neutral valuation principle, the option price is equal to its expected payoff in a risk-neutral world discounted at the risk-free interest rate. The BOPM is a three-step procedure:

1. Generate the price tree.
2. Calculate the payoffs in terminal nodes.
3. Iterating backward induction from the end of a tree to its beginning, we are able to obtain the value of the option at time zero.

In Figure 2.2 there are quadratic, i.e., $O\left(n^2\right)$, nodes. So the BOPM must take $O\left(n^2\right)$ steps to apply backward induction by visiting every node. We can reuse a one-dimensional array of size $n+1$ instead of a two-dimensional array of size $\left(n+1\right)\times\left(n+1\right)$ for storing node values; so the memory requirement is reduced to $O\left(n\right)$.

As the number of time steps increases, the stock price ranges over ever-larger numbers of possible values, and trading takes place nearly continuously. It can be proved that the price computed by the BOPM converges to the price from the Black-Scholes option pricing model as $n\to\infty$.

Chapter 3
Trees with Combinatorial Method in Pricing

Combinatorial methods use combinatorics to solve problems. It has been widely applied in many fields [10]. This chapter deals with using combinatorial methods to price financial options. Under the popular tree model, combinatorial methods can often speed up the pricing of European options by an order of magnitude [12].

3.1 Combinatorial Methods

Once we have the probability distribution of the terminal nodes, these options can be priced by simply summing the products of the probability and the payoff function at each terminal node to obtain their expected payoff. The theoretical price then equals the discounted expected payoff. We give some examples below.

In the n time step case, the value of a European call is

$$C = e^{-rT}\sum_{j=0}^{n}\binom{n}{j}p^{j}(1-p)^{n-j}\times\max\left(0, S_0u^{j}d^{n-j}-K\right).$$

The number of paths from S_0 to the terminal price $S_0u^{j}d^{n-j}$ is given by the binomial coefficient $\binom{n}{j}$, where n is the number of time steps used for price computation and j is the number of up moves of the underlying asset's price during the option's lifetime. Each path to the terminal price $S_0u^{j}d^{n-j}$ has the same probability $p^{j}(1-p)^{n-j}$. To adapt the equation to price European puts, simply replace the call payoff, $\max\left(0, S_0u^{j}d^{n-j}-K\right)$, with the put payoff, $\max\left(0, K-S_0u^{j}d^{n-j}\right)$. The resulting formula becomes

$$P = e^{-rT}\sum_{j=0}^{n}\binom{n}{j}p^{j}(1-p)^{n-j}\times\max\left(0, K-S_0u^{j}d^{n-j}\right).$$

Let a be the minimum number of upward price moves for the call to finish in the money, i.e., a is the smallest nonnegative integer such that $S_0u^{a}d^{n-a}\geq K$, or

$$a = \left\lceil \frac{\ln\left(K/S_0 d^n\right)}{\ln\left(u/d\right)} \right\rceil .$$

Hence,

$$C = e^{-rT} \sum_{j=a}^{n} \binom{n}{j} p^j \left(1-p\right)^{n-j} \left(S_0 u^j d^{n-j} - K\right).$$

Also, let b be the maximum number of upward price moves for the put to finish in the money, i.e., b is the largest integer such that $S_0 u^b d^{n-b} \le K$, or

$$b = \left\lfloor \frac{\ln\left(K/S_0 d^n\right)}{\ln\left(u/d\right)} \right\rfloor .$$

Hence,

$$P = e^{-rT} \sum_{j=0}^{b} \binom{n}{j} p^j \left(1-p\right)^{n-j} \left(K - S_0 u^j d^{n-j}\right).$$

It implies a linear-time, $O(n)$, and constant-space, $O(1)$, algorithm for pricing standard European options [12].

Consider the down-and-in call with barrier $H < K$. We have to calculate the number of admissible paths that lead to a particular terminal node. Towards that end, we pick the barrier level, h to be one of the $n+1$ possible terminal underlying asset's prices that is closed to, but does not exceed, H. We call it the effective barrier. This condition $S_0 u^h d^{n-h} \le H$ implies

$$h = \left\lfloor \frac{\ln\left(H/S_0 d^n\right)}{\ln\left(u/d\right)} \right\rfloor .$$

We effectively use $S_0 u^h d^{n-h}$ not H as the barrier on the binomial model. The option value equals

$$e^{-rT} \sum_{j=a}^{2h} \binom{n}{n-2h+j} p^j \left(1-p\right)^{n-j} \left(S_0 u^j d^{n-j} - K\right).$$

Not all n give acceptable numerical results [4]. For convergence reasons, the effective barrier level should be close to H, i.e., $H \approx S_0 d^j = S_0 e^{-j\sigma\sqrt{T/n}}$ for some integer j. But the effective barrier level $S_0 d^j$ corresponds to a terminal underlying

asset's price only when $n-j$ is even. To close this gap, we decrement n by one, if necessary, to make $n-j$ an even number. The preferred n's are finally

$$n=\begin{cases} d, & \text{if } d-j \text{ is even} \\ d-1, & \text{otherwise} \end{cases}, \quad d \equiv \left\lfloor \frac{j^2\sigma^2 T}{\ln^2\left(S_0/H\right)} \right\rfloor.$$

It also implies a $O(n)$-time, $O(1)$-space algorithm [11].

3.2 Technique for Handling Large Numbers

When the life of the option is divided into ever more time steps, the number of underlying asset's price paths increases exponentially. So the combinatorial method rapidly generates large numbers. The primitive data types of typical programming languages cannot store numbers larger than a certain magnitude. For example, the largest number the data type double can represent is $1.7976931348623157\times10^{308}$. The problem that occurs when the value to be represented is greater in magnitude is called overflow. To solve this problem, we will process data in the log domain. We calculate $\ln(p_1\cdot p_2)$ from $\ln p_1$ and $\ln p_2$ as follows:

$$\ln(p_1\cdot p_2)=\ln p_1+\ln p_2.$$

We calculate $\ln(p_1+p_2)$ from $\ln p_1$ and $\ln p_2$ as follows:

$$\begin{aligned} \ln(p_1+p_2) &= \ln\left(p_1\left(1+\frac{p_2}{p_1}\right)\right) \\ &= \ln(p_1)+\ln\left(1+\frac{p_2}{p_1}\right) \\ &= \ln(p_1)+\ln\left(1+e^{\ln p_2-\ln p_1}\right) \end{aligned}.$$

When the number in the decimal domain is needed, simply take the exponential function of the final number for the desired number. In the following algorithms, we will assume the large numbers technique implicitly.

3.3 Costabile's Algorithm

To evaluate Parisian options, Costabile proposed an algorithm in the CRR model and applied the combinatorial method [6]. In this section, we describe Costabile's algorithm for European up-and-out Parisian options.

We consider a particle whose dynamics is described by a random walk in the two

dimensional space (i, j). From (i, j), it moves to $(i+1, j+1)$ with probability p if an up move occurs or to $(i+1, j-1)$ with probability $1-p$ if a down move takes place. Without loss of generality we assume that the particle starts from the origin $P \equiv (0,0)$.

The number of paths, $N(i, j)$, from the origin P to (i, j) has the well-known recursive relation, $N(i, j) = N(i-1, j-1) + N(i-1, j+1)$. $N(i, j)$ is given by the binomial coefficient $\binom{i}{(i+j)/2}$. Suppose the barrier level in the binomial tree is at level m. Let $g(i, j)$ denote the number of the paths from P to (i, j) so that the paths spend fewer than l consecutive time steps at or above the barrier level, $l \in \mathbb{N}$, which is an input parameter. Such paths are said to be admissible. Later we will describe how to derive m from the barrier H and l from the window period w.

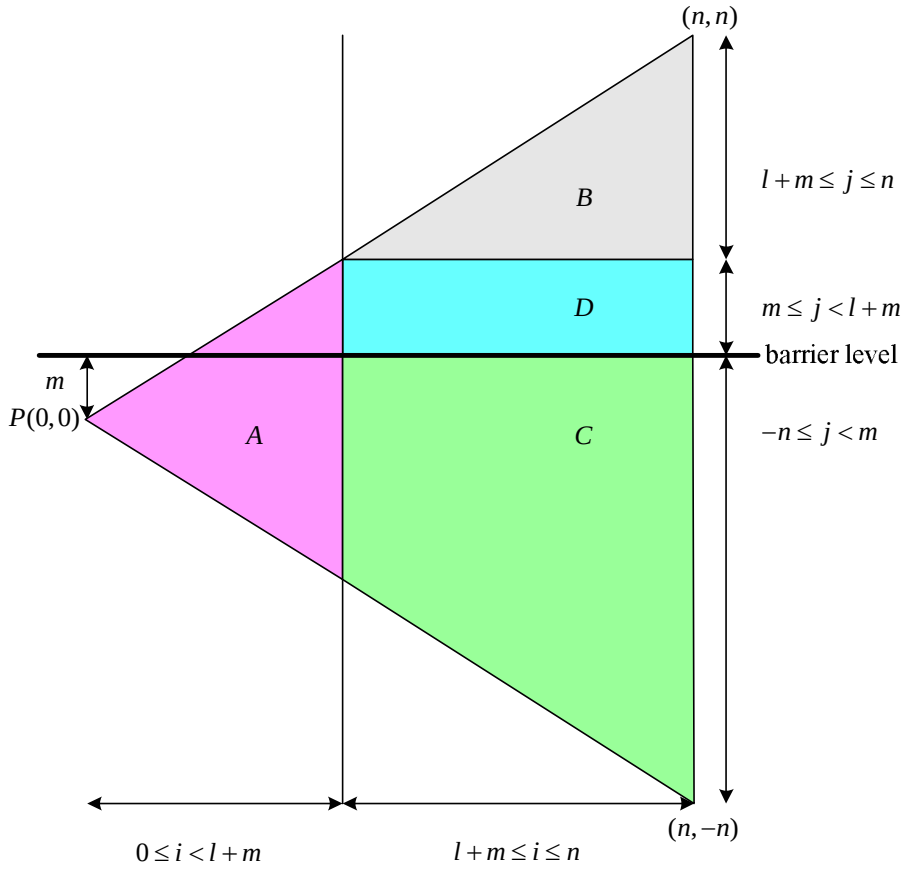


Figure 3.1: Partitions of the areas of computation. This figure is due to Prof. Jr-Yan Wang.

We derive $g(i, j)$ for (i, j) in different regions of the tree (see Figure 3.1). The first step considers the nodes (i, j) such that $0 \le i < l + m$ called Region A. In this case $g(i, j) = N(i, j)$. Indeed, such nodes are characterized by at most i up moves and $i - m < l$. In the second step we observe that, for the nodes (i, j) such that $l + m \le i \le n, l + m \le j \le n$ (Region B), $g(i, j) = 0$. Indeed, in such cases, a path starting form P can reach the level $l + m \le j$ only if the particle spends at least l successive time steps at or above the barrier level. The third step concerns the nodes (i, j) located below the barrier level (i.e., $l + m \le i \le n, -n \le j < l + m$ or Region C). Each path from P to (i, j) in Region C makes its last move below the barrier level.

This implies $g(i,j) = g(i-1,j-1) + g(i-1,j+1)$.

In the final step we are left to calculate $g(i,j)$ of the nodes (i,j) such that $l+m \le i \le n, m \le j \le l+m$ or Region D. Note that in this case the recursive relation $g(i,j) = g(i-1,j-1) + g(i-1,j+1)$ is no longer holds. Indeed some of the $g(i-1,j+1)$ paths from P to $(i-1,j+1)$ may have spent $l-1$ time steps at or above the barrier level already, in which case the move from time step $i-1$ to i is made above the barrier level. Such paths have to be excluded from $g(i-1,j+1)$.

Each admissible path from P to (i,j) in Region D can be divided into two different parts. The first part is an admissible path from P to a node which is located at level $m-1$ and then makes an up move to a node $(i-2k-(j-m),m)$ on the barrier level. The number of the first partial path is given by the quantity $g(i-2k-(j-m)-1,m-1)$. The second part is a partial path from the final node of first partial path, $(i-2k-(j-m),m)$, to the node (i,j) with the number of moves strictly smaller than l, and this part should be never cross the barrier level. The parameter k should be a nonnegative integer and the condition $i-(i-2k-(j-m)) < l$ implies:

$$0 \le k < \frac{l-(j-m)}{2}.$$

The number of the second partial paths can be computed as the difference between the following two terms:

1. The first is the number of paths from the node $(i-2k-(j-m),m)$ on the barrier level to (i,j). It is given by $\binom{2k+j-m}{k+j-m}$.
2. The second is the number of paths from the same node to (i,j) which cross the

barrier level. Using the reflection principle, it is given by $\binom{2k+j-m}{k+1+j-m}$ with the convention $\binom{a}{b}=0$ for $a<b$.

Finally we get the number of admissible paths as

$$g(i,j)=\sum_{0\le k<\frac{l-(j-m)}{2}} g\left(i-2k-(j-m)-1,m-1\right)\cdot\left[\binom{2k+j-m}{k+j-m}-\binom{2k+j-m}{k+1+j-m}\right]. \quad (1)$$

See [6] for additional discussions.

We assume that the underlying asset's price dynamics is described by the CRR model and also use the notations as before. Recall that w denotes the window period measured in years and it must be limited to $0\le w\le T$. When a Parisian option with $w=0$, this is a standard barrier option. When $w=T$, we have either a standard European option or a option with zero value.

In order to drive the binomial algorithm for evaluating Parisian options, we need to determine the total number of time steps, n, to be used in the price computations. As pointed out by Boyle and Lau, at first we build up a binomial tree such that the barrier is close to but just below a layer of horizontal nodes in the tree [4]. Recall that m is the number of successive up moves for the underlying asset's price, starting from S_0, must make to touch or cross the barrier H. This condition $S_0u^m\ge H$ implies

$$n=\left\lfloor\frac{m^2\sigma^2T}{\ln^2\left(H/S_0\right)}\right\rfloor,$$

and

$$m=\left\lceil\frac{\ln\left(H/S_0\right)}{\sigma\sqrt{\Delta t}}\right\rceil.$$

The next step is to determine the number of time steps, l, corresponding to the window period w. Because, in general, $w/\Delta t$ is not an integer, we set l equal to $\left[w/\Delta t\right]$, the integer closest to $w/\Delta t$. The price of the up-and-out Parisian call option can now be easily derived:

$$PC=e^{-rT}\sum_{j=a}^{n} g\left(n,2j-n\right)p^j\left(1-p\right)^{n-j}\left(S_0u^jd^{n-j}-K\right).$$

Recall that $g(n, 2j-n)$ is the number of paths which spend fewer than l consecutive time steps at or above the barrier level and these paths are made up of j up moves and $n-j$ down moves.

Together with the initial conditions:

$$\binom{n}{0} = \binom{n}{1} = 1,$$

Pascal's triangle gives a method for calculating the binomial coefficients by using the following well-known identity:

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}$$

where n and k are positive integers with $n \geq k$ [17]. The binomial coefficients can be computed in $O(n^2)$ time. Assume the binomial coefficients have been computed from now on. See Figure 3.2 for Costabile's algorithm.

1: $\Delta t := T/n$;

2: $u := \exp\left(\sigma\sqrt{\Delta t}\right)$;

3: $d := 1/u$;

4: $p := \dfrac{\exp\left((r-q)\Delta t\right) - d}{u-d}$;

5: $m := \left\lceil \dfrac{\ln(H/S_0)}{\sigma\sqrt{\Delta t}} \right\rceil$;

6: $l := \lceil w/\Delta t \rceil$;

7: $a := \left\lceil \ln\left(K/S_0 d^n\right) / \ln(u/d) \right\rceil$;

8: **for** $i := 0$ to n **do**

9: **for** $j := i$ to $-i$ **step** -2 **do**

10: **if** $i < l + m$ **then** {Region A}

11: $g[i][j] := \binom{i}{(i+j)/2}$;

12: **else if** $j \geq l + m$ **then** {Region B}

13: $g[i][j] := 0$;

14: **else if** $j < m$ **then** {Region C}
15: **if** $j = -i$ **then**
16: $g[i][j] := g[i-1][j+1]$;
17: **else**
18: $g[i][j] := g[i-1][j-1] + g[i-1][j+1]$;
19: **end if**
20: **else** {Region D}
21: $g[i][j] := 0$;
22: **for** $k := 0$ to $\left\lfloor \frac{l-(j-m)}{2} - 0.5 \right\rfloor$ **do**
23: **if** $2k + j - m < k + 1 + j - m$ **then**
24: $g[i][j] := g\left[i - 2k - (j-m) - 1\right][m-1] \cdot \binom{2k+j-m}{k+j-m}$;
25: **else**
26: $g[i][j] := g\left[i - 2k - (j-m) - 1\right][m-1] \cdot \left[\binom{2k+j-m}{k+j-m} - \binom{2k+j-m}{k+1+j-m} \right]$;
27: **end if**
28: **end for**
29: **end if**
30: **end for**
31: **end for**
32: $C := 0$;
33: **for** $j := a$ to n **do**
34: $C := C + g[n][2j-n] \cdot p^j (1-p)^{n-j} \cdot \left(S_0 u^j d^{n-j} - K\right)$;
35: **end for**
36: return $C \cdot e^{-rT}$;

Figure 3.2: Costabile's algorithm for up-and-out Parisian options.

We proceed to count the number of operations needed to compute the $g(i, j)$. For simplicity, the number of time steps, n, is odd for the option evaluation and the parameters m and l are even. Because the $g(i, j)$ in Region A and Region B

needed no arithmetic operations, Table 3.1 only shows the number of mathematical operations needed to implement Costabile's algorithm in Region C and Region D. See [6] for more details. Recall that l was defined as the integer closed to $w/\Delta t = w \cdot n/T$. It follows that the computational cost is a cubic function of n, or $O(n^3)$. The numbers $g(i,j)$ are stored in a two-dimensional array of size $(n+1)\times(2n+1)$ so the memory requirement is $O(n^2)$.

	Region C	Region D
Additions	$\frac{(n+m)^2+2m+2n+1-(l+2m)^2}{4}$	$\left[\frac{n+1-(l+m)}{2}\right]\left(\frac{l}{2}-1\right)\left(\frac{l}{2}\right)$
Subtractions	0	$\left[\frac{n+1-(l+m)}{2}\right]\left(\frac{l}{2}\right)\left(\frac{l}{2}+1\right)$
Multiplications	0	$\left[\frac{n+1-(l+m)}{2}\right]\left(\frac{l}{2}\right)\left(\frac{l}{2}+1\right)$

Table 3.1: Number of operations needed in Region C and Region D for Costabile's algorithm.

3.4 An Improved Algorithm

For nodes (i,j) in Region D, $g(i,j)$ depends on the $g(i-2k-(j-m)-1, m-1)$ located at level $m-1$ in Region A and Region C (recall Equation (1)). So we need not calculate all $g(i,j)$ in Region D. In Region C when $j = m-1$, the recursive relation $g(i,j) = g(i-1, j+1) + g(i-1, j+1)$ is also dependent on the $g(i-1,m)$ located at level m. So we only need to calculate the $g(i,j)$ in Region D which are

1. nodes located at level m, and
2. nodes located at the n'th time step.

It is not hard to observe that other $g(i,j)$ are not needed for calculating $g[n][2j-n]$ of Line 34 in Figure 3.1.

The memory requirement can be reused if the space is reused. Specifically, replace the two-dimensional array $g[n+1][2n+1]$ with a one-dimensional array of size $2n+1$. But, we should use another one-dimensional array $t[n+1]$ to store a copy of $g(i, m-1)$ located at level $m-1$ for calculating $g(i, j)$ in Region D.

These crucial points can be used to improve Costabile's algorithm, and the results must be identical to Costabile's algorithm. See Figure 3.3 for the improved algorithm.

1: $\Delta t := T/n$;

2: $u := \exp\left(\sigma\sqrt{\Delta t}\right)$;

3: $d := 1/u$;

4: $p := \dfrac{\exp\left((r-q)\Delta t\right) - d}{u-d}$;

5: $m := \left\lceil \dfrac{\ln(H/S_0)}{\sigma\sqrt{\Delta t}} \right\rceil$;

6: $l := \lceil w/\Delta t \rceil$;

7: $a := \left\lceil \ln\left(K/S_0 d^n\right) \Big/ \ln(u/d) \right\rceil$;

8: **for** $i := 0$ to n **do**

9: **for** $j := i$ to $-i$ **step** -2 **do**

10: **if** $i < l + m$ **then** {Region A}

11: $g[j] := \begin{pmatrix} i \\ (i+j)/2 \end{pmatrix}$;

12: **else if** $j \geq l + m$ **then** {Region B}

13: $g[j] := 0$;

14: **else if** $j < m$ **then** {Region C}

15: **if** $j = -i$ **then**

16: $g[j] := g[j+1]$;

17: **else**

18: $g[j] := g[j-1] + g[j+1];$

19: **end if**

20: **else if** $j = m$ or $i = n$ **then** {Partial Region D}

21: $g[j] := 0;$

22: **for** $k := 0$ to $\left\lfloor \frac{l-(j-m)}{2} - 0.5 \right\rfloor$ **do**

23: **if** $2k + j - m < k + 1 + j - m$ **then**

24: $g[j] := t[i - 2k - (j-m) - 1] \cdot \binom{2k + j - m}{k + j - m};$

25: **else**

26: $g[j] := t[i - 2k - (j-m) - 1] \cdot \left[\binom{2k + j - m}{k + j - m} - \binom{2k + j - m}{k + 1 + j - m} \right];$

27: **end if**

28: **end for**

29: **end if**

30: **if** $j = m - 1$ **then**

31: $t[i] := g[j];$

32: **end if**

33: **end for**

34: **end for**

35: $C := 0;$

36: **for** $j := a$ to n **do**

37: $C := C + g[2j - n] \cdot p^j (1-p)^{n-j} \cdot (S_0 u^j d^{n-j} - K);$

38: **end for**

39: return $C \cdot e^{-rT}$;

Figure 3.3: Improved algorithm for up-and-out Parisian options.

Recall that we assume the parameters n is odd, and m and l are even. We observe that there are $\lfloor n+1-(l+m) \rfloor / 2$ nodes at each level in Region D. According to Equation (1), at levels m and $m+1$, each $g(i, j)$ is calculated with $l/2-1$

additions, $l/2$ multiplications and $l/2$ subtractions. At levels $m+2$ and $m+3$, each $g(i,j)$ is calculated with $l/2-2$ additions, $l/2-1$ multiplications and $l/2-1$ subtractions. Continuing in this manner we can compute the total number of additions needed to compute $g(i,j)$ located at level m and at the n'th time step in Region D as

$$\underbrace{\left(\frac{l}{2}-1\right)\left[\frac{n+1-(l+m)}{2}\right]}_{\text{located at level } m}+\underbrace{\sum_{j=1}^{l/2-1} j}_{\text{located at the } n\text{'th time step}}=\left(\frac{l}{2}-1\right)\left(\frac{2n-2m+2-l}{4}\right).$$

The number of multiplications and subtractions needed is

$$\underbrace{\left(\frac{l}{2}\right)\left[\frac{n+1-(l+m)}{2}\right]}_{\text{located at level } m}+\underbrace{\sum_{j=1}^{l/2} j}_{\text{located at the } n\text{'th time step}}=\left(\frac{l}{2}\right)\left(\frac{2n-2m+4-l}{4}\right).$$

Table 3.2 shows the number of mathematical operations needed to implement the improved algorithm in Region C and Region D.

The computational cost of the improved algorithm is less than Costabile's algorithm, which is $O(n^3)$. The reduction of nodes that need to be computed in Region D leads to $O(n^2)$ computational time. This improved algorithm could be also speeded up somewhat, but the total time complexity remains $O(n^2)$. Because binomial coefficients are given and we only use two one-dimensional arrays for the improved algorithm, the memory requirement is $O(n)$.

	Region C	Region D
Additions	$\frac{(n+m)^2+2m+2n+1-(l+2m)^2}{4}$	$\left(\frac{l}{2}-1\right)\left(\frac{2n-2m+2-l}{4}\right)$
Subtractions	0	$\left(\frac{l}{2}\right)\left(\frac{2n-2m+4-l}{4}\right)$
Multiplications	0	$\left(\frac{l}{2}\right)\left(\frac{2n-2m+4-l}{4}\right)$

Table 3.2: Number of operations needed in Region C and Region D for the improved algorithm.

3.5 General Case

The $g(i, j)$ of the down-and-out barrier level, $-m$ is the same as the $g(i, -j)$ of the up-and-out barrier level, m with an identical window period. To price a down-and-out Parisian call with a barrier H', it is not necessary to discuss where Region A, Region B, Region C, and Region D are. We can reuse the up-and-out Parisian options pricing algorithm in Figure 3.3 and replace Line 5 with

$$m := -\left\lfloor \frac{\ln(H'/S_0)}{\sigma\sqrt{\Delta t}} \right\rfloor;$$

and replace Line 37 with

$$C := C + g\left[-(2j-n)\right] \cdot p^j (1-p)^{n-j} \cdot \left(S_0 u^j d^{n-j} - K\right);$$

Then all knock-out Parisian calls could be priced.

The $g(i, j)$ of knock-in options is the same as $N(i, j)$ subtract the $g(i, j)$ of knock-out options with identical barrier and window period. For example, to price an up-and-in Parisian call, replace Line 37 with

$$C := C + \left[\binom{n}{j} - g\left[2j-n\right]\right] \cdot p^j (1-p)^{n-j} \cdot \left(S_0 u^j d^{n-j} - K\right);$$

Use the BOMP in Section 2.4 or combinatorial method's formula for the standard European call in Section 3.1 to subtract the up-and-out Parisian call, it can also get equal value. It is just the in-out parity. Then all Parisian calls could be priced.

To price a Parisian put, refer to Section 3.1 for b, the maximum number of upward price moves for the put to finish in money. For example, to price an up-and-out Parisian put, replace Line 7 with

$$b := \left\lfloor \ln\left(K/S_0 d^n\right) \Big/ \ln(u/d) \right\rfloor;$$

Line 35 as

$$P := 0;$$

Line 36 as

for $j := 0$ to b **do**

and Line 37 as

$$P := P + g\left[2j-n\right] \cdot p^j (1-p)^{n-j} \cdot \left(K - S_0 u^j d^{n-j}\right);$$

Then all Parisian options could be priced.

3.6 Numerical Results

Next we will price an up-and-out Parisian call option with the underlying asset given by the exchange rate between US dollar (USD) and Japanese yen (JPY). Consider a spot exchange rate $S_0 = 1/120.5$, time to maturity $T = 0.5$ year, strike rate $K = 1/125$, and barrier $H = 1/110$. The US risk-free interest rate is $r_{us} = 0.056$ per year, the Japan risk-free interest rate is $r_{jp} = 0.007$, and volatility is $\sigma = 0.13$ per year.

The number of trading days in a year is assumed to be 250. Table 3.3 illustrates the numerical results for different values of the window period w provided by Costabile. Compared with Table 3.4 computed by our program and Table 4.1 generated from the fast Monte Carlo simulation in Chapter 4, Table 3.3 is wrong when $w = 5, 10, 15$ days.

		$w = 0$ days		$w = 5$ days		$w = 10$ days		$w = 15$ days	
m	n	l	PC	l	PC	l	PC	l	PC
10	101	0	0.000142	4	0.000202	8	0.000244	12	0.000279
20	406	0	0.000140	16	0.000214	32	0.000257	49	0.000297
32	1041	0	0.000141	42	0.000214	83	0.000254	125	0.000287
40	1626	0	0.000140	65	0.000213	130	0.000255	195	0.000284
50	2541	0	0.000141	102	0.000215	203	0.000252	305	0.000280

Table 3.3: Numerical results of different window periods by Costabile.

		$w = 0$ days		$w = 5$ days		$w = 10$ days		$w = 15$ days	
m	n	l	PC	l	PC	l	PC	l	PC
10	101	0	0.000142	4	0.000205	8	0.000246	12	0.000282
20	406	0	0.000140	16	0.000214	32	0.000258	49	0.000297
32	1041	0	0.000141	42	0.000222	83	0.000265	125	0.000301
40	1626	0	0.000140	65	0.000224	130	0.000267	195	0.000304
50	2541	0	0.000141	102	0.000225	203	0.000269	305	0.000305
100	10166	0	0.000141	407	0.000229	813	0.000273	1220	0.000308
200	40664	0	0.000141	1627	0.000230	3253	0.000274	4880	0.000310
300	91495	0	0.000141	3660	0.000231	7320	0.000275	10979	0.000311
400	162659	0	0.000141	6506	0.000231	13013	0.000275	19519	0.000311

Table 3.4: Numerical results of different window periods from Costabile's algorithm

and the improved algorithm.

Avellaneda and Wu priced the same Parisian options by formulating a partial differential equation (PDE) which is solved numerically on a trinomial lattice, but they assumed the number of trading days in a year is 360 [1]. Table 3.5 compares their method and ours. The last row illustrates the numerical results obtained with the Avellaneda-Wu (AW) model. As the number of time steps increases, our numerical results converge to the Avellaneda-Wu model's.

		$w = 0$ days		$w = 5$ days		$w = 10$ days		$w = 15$ days	
m	n	l	PC	l	PC	l	PC	l	PC
10	101	0	0.000142	3	0.000197	6	0.000227	8	0.000246
20	406	0	0.000140	11	0.000201	23	0.000237	34	0.000262
32	1041	0	0.000141	29	0.000206	58	0.000240	87	0.000269
40	1626	0	0.000140	45	0.000207	90	0.000242	135	0.000271
50	2541	0	0.000141	71	0.000209	141	0.000244	212	0.000273
100	10166	0	0.000141	282	0.000212	565	0.000247	847	0.000276
200	40664	0	0.000141	1130	0.000213	2259	0.000249	3389	0.000278
300	91495	0	0.000141	2542	0.000214	5083	0.000249	7625	0.000278
400	162659	0	0.000141	4518	0.000214	9037	0.000250	13555	0.000278
AW model			0.000141		0.000215		0.000251		0.000279

Table 3.5: Numerical results of different window periods compared with the Avellaneda-Wu model.

Chapter 4

Monte Carlo Simulation

Monte Carlo (MC) simulation is a sampling scheme. The first application to option pricing was by Phelim Boyle [3]. Monte Carlo simulation can be a procedure for randomly sampling changes in market variables in order to value a derivative. It uses the risk-neural valuation result to approximate the expectation of the derivative's terminal cash flows with a simple arithmetic average of the cash flows taken over a finite number of simulated paths:

$$\text{Price} \approx e^{-rT}\left(\frac{1}{M}\sum_{i=1}^{M} f\left(S_0^i,\cdots,S_T^i\right)\right),$$

where $S_0^i,\cdots,S_T^i$ is the i'th simulated price path of underlying asset over the life of the derivative and $f\left(S_0^i,\cdots,S_T^i\right)$ is the derivative's terminal cash flow from this path.

For a large sample of simulated price paths, the mean of the sample will closely approximate the derivative's true price. The statistical error of sample mean of the price grows as $1/\sqrt{M}$, where M is the number of replications (or independent trials).

4.1 Crude Monte Carlo Simulation

Suppose that the process followed by the underlying market variable in a risk-neutral world is

$$dS = \hat{\mu} S\, dt + \sigma S\, dz,$$

where $\hat{\mu}$ is the expected return in a risk-neutral world, and σ is the volatility. From Ito's lemma process followed by $\ln S$ is

$$d\ln S = \left(\hat{\mu} - \frac{\sigma^2}{2}\right)dt + \sigma\, dz.$$

Since $\hat{\mu}$ and σ are constant, this equation indicated that $\ln S$ follows a Brownian motion. It has constant drift rate $\hat{\mu} - \sigma^2/2$ and constant variance rate σ^2. This equation is used to construct a path for S. To simulate the path follow by S, we can divide the life of the derivative into n short intervals of length Δt. So that

$$\ln S\left(t+\Delta t\right) - \ln S\left(t\right) = \left(\hat{\mu} - \frac{\sigma^2}{2}\right)\Delta t + \sigma\varepsilon\sqrt{\Delta t},$$

or equivalently

$$S\left(t+\Delta t\right)=S\left(t\right)\exp\left[\left(\hat{\mu}-\frac{\sigma^2}{2}\right)\Delta t+\sigma\varepsilon\sqrt{\Delta t}\right]$$

where $S\left(t\right)$ denotes the value of S at time t, ε is a random sample from a normal distribution with mean zero and standard deviation of 1.0 and can be generated by the Box-Muller algorithm [22]. This enables $S\left(\Delta t\right)$ to be calculated from $S\left(0\right)$, $S\left(2\Delta t\right)$ to be calculated from $S\left(\Delta t\right)$, and so on [9]. For the pricing of up-and-in Parisian options which have the payoff $V\left(S\left(T\right)\right)$, we may proceed as Figure 4.1.

1: $\Delta t := T/n$;
2: $C := 0$; {Accumulated option value.}
3: **for** $i := 1$ to M **do**
4: $St := S_0$;
5: $flag := 0$;
6: $l := 0$;
7: **for** $j := 1$ to n **do**
8: $St := St \cdot \exp\left[\left(\hat{\mu}-\frac{\sigma^2}{2}\right)\cdot\Delta t+\sigma\varepsilon\sqrt{\Delta t}\right]$;
9: **if** $St \geq H$ **then**
10: $l := l+\Delta t$;
11: **if** $l \geq w$ **then**
12: $flag := 1$;
13: **end if**
14: **else**
15: $l := 0$;
16: **end if**
17: **if** $flag = 1$ **then**
18: $C := C+V\left(S\left(T\right)\right)\cdot e^{-rT}$;
19: **end if**
20: **end for**

21: **end for**
22: return C/M ;

Figure 4.1: Crude Monte Carlo simulation for up-and-in Parisian options.

4.2 Inverse Gaussian Distribution

Tweedie introduced the inverse Gaussian distribution (IG) [19]. The inverse Gaussian distribution is the distribution over $[0,\infty)$ with the probability density function (p.d.f.) given by

$$f(x;\mu,\lambda)=\sqrt{\frac{\lambda}{2\pi x^3}}\,\mathrm{e}^{-\frac{\lambda(x-\mu)^2}{2\mu^2 x}},\quad x>0$$

where $\mu>0$ is the mean and $\lambda>0$ is a scale parameter. As λ tends to infinity, the inverse Gaussian distribution becomes more like a normal (Gaussian) distribution. If random variable X has inverse Gaussian distribution with location parameter μ and scale parameter λ, we write

$$X\sim IG(\mu,\lambda)$$

where $\sim$ means equality in distribution.

The inverse Gaussian describes the distribution of the time a Brownian motion with positive drift takes to reach a fixed positive level. That is the first-passage time distributions of Brownian motion with positive drift [20]. In particular, the relationship between the inverse Gaussian distribution and Brownian motion is as follows: The Brownian motion for a variable $X(t)$ is given by

$$dX=\nu\,dt+\sigma\,dz,$$

where dz is a normalized Brownian motion with drift ν and variance σ^2. Refer to Figure 4.2, consider the first-passage time F of $X(t)$ to the fixed level $a>x_0$, called a barrier. In order for F to be the first-passage time, we require

$$\begin{aligned}X(0)&=x_0\\ X(t)&<a,\ 0<t<F\ .\\ X(F)&=a\end{aligned}$$

When the drift $\nu>0$, we have $\mathrm{Prob}(F<\infty)=1$ and the first-passage time to the

barrier $a > x_0$ is the inverse Gaussian $IG(\mu, \lambda)$, where

$$\mu = \frac{a - x_0}{v}$$

and

$$\lambda = \frac{(a - x_0)^2}{\sigma^2}.$$

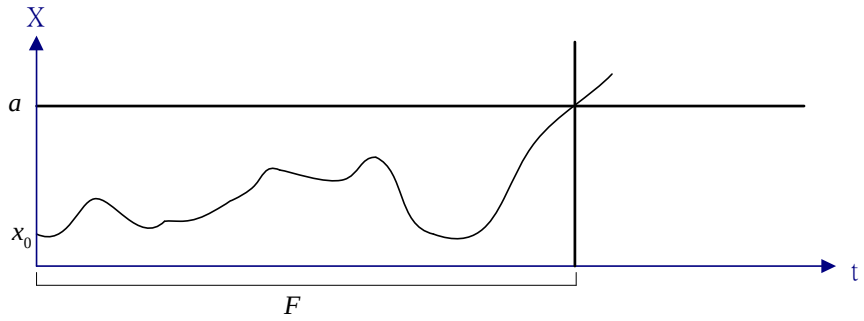


Figure 4.2: First passage time.

The random variable Y has a uniform distribution if its probability density function is equal to a constant on its support. In particular, if the support is the interval $[a,b]$, then

$$f(x) = \frac{1}{b-a}, \quad a \le x \le b.$$

Moreover, we shall say that Y is UNIFORM(a,b) [8]. Let $\chi^2_{(1)}$ be a chi-square distribution with one degree of freedom, i.e.,

$$\chi^2_{(1)} = \left(\frac{X - \mu}{\sigma}\right)^2 = Z^2, \quad X \sim N(\mu, \sigma^2).$$

$\chi^2_{(1)}$ are easily generated as the squares of standard normals. Michael *et al*. gave a method of generating random variates from the inverse Gaussian distribution using a transformation [14]. Figure 4.3 shows how to generate the inverse Gaussian variate X.

1: sample $V \sim \chi^2_{(1)}$;

2: $x_1 := \mu + \frac{\mu^2 V}{2\lambda} - \frac{\mu}{2\lambda}\sqrt{4\mu\lambda V + \mu^2 V^2}$;

3: $P := \frac{\mu}{\mu + x_1}$;

4: sample $Y \sim \text{UNIFORM}(0,1)$;

5: $X := x_1$;

6: **if** $Y > P$ **then**

7: $X := \frac{\mu^2}{x_1}$;

8: **end if**

9: return X ;

Figure 4.3: Michael's sampling method for the inverse Gaussian distribution.

Whitmore discussed what he terms a defective inverse Gaussian distribution (DIG):

$$f(x;\mu,\lambda) = \sqrt{\frac{\lambda}{2\pi x^3}}\, \mathrm{e}^{-\frac{\lambda(x-\mu)^2}{2\mu^2 x}}$$

with $x > 0$, $\lambda > 0$, $-\infty < \mu < \infty$, $\mu \neq 0$ [21]. The inverse Gaussian distribution does not allow $\mu < 0$, but the defective inverse Gaussian distribution does. With $\mu < 0$, f does not integrate to unity so it is not a density function in the ordinary sense. It is not hard to verify that

$$f(x;\mu < 0,\lambda) = e^{2\lambda/\mu} f(x;-\mu,\lambda)$$

and

$$\int_0^\infty f(x;\mu < 0,\lambda)\,dx = e^{2\lambda/\mu}.$$

Such a definition takes into account the possibility of the drift being negative in a Brownian motion. Consider the first-passage time F of $X(t)$. When drift $\nu < 0$ and barrier $a > x_0$, we have $\text{Prob}(F < \infty) = e^{2\lambda/\mu}$ and the conditional distribution of F given $F < \infty$ is $IG(-\mu > 0,\lambda)$, where

$$\mu = \frac{a - x_0}{\nu}$$

and

$$\lambda = \frac{\left(a - x_0\right)^2}{\sigma^2}.$$

So the probability of never hitting the barrier is $1 - e^{2\lambda/\mu}$ as $\text{Prob}\left(F = \infty\right) = 1 - e^{2\lambda/\mu}$.

We can also use Michael's sampling method for generating the defective inverse Gaussian variate. We emphasize that the sampling variate is also conditional distribution of ever hitting the barrier again.

Regardless of whether drift ν is positive or negative and whether barrier a is large than x_0 or not, their first-passage time distribution is the same as the first-passage time distribution with drift $-\nu$, barrier $2x_0 - a$ and initial position x_0. Consider the first-passage time F of $X(t)$. When drift v and barrier $a < x_0$, the conditional distribution of F given $F < \infty$ is $IG\left(|\mu|, \lambda\right)$, where

$$\begin{aligned}\mu &= \frac{\left(2x_0 - a\right) - x_0}{-v} \\ &= \frac{a - x_0}{v}\end{aligned}$$

and

$$\begin{aligned}\lambda &= \frac{\left(\left(2x_0 - a\right) - x_0\right)^2}{\sigma^2} \\ &= \frac{\left(a - x_0\right)^2}{\sigma^2}\end{aligned}.$$

In summary, the first-passage time distribution F of the Brownian motion given $F < \infty$ is $IG\left(|\mu|, \lambda\right)$.

4.3 Simulation with the Inverse Gaussian Distribution

Now we can use IG and DIG to price up-and-in Parisian options with the initial stock price S_0, the strike price K, the time to expiration T, the volatility of the stock price σ, the risk-free interest rate r, the dividend yield q, the barrier H, and the window period w.

We now introduce our fast simulation algorithm. First, use Michael's sampling method to get the first-passage time $F \sim IG\left(|\mu|, \lambda\right)$ of the underlying asset's price

process with barrier H. Set the Brownian motion parameters $a = \ln H$, $x_0 = \ln S_0$, and $\nu = \hat{\mu} - \sigma^2/2$ to get the Inverse Gaussian distribution parameters μ and λ. If $F + w$ is less than duration T, it means that the underlying asset's price S at the time F is H:

$$S(F) = H, \quad F \le T.$$

Second, we can use the Monte Carlo method to simulate the following partial path whether continuously at or above the given barrier over the window period from the time F. If it did, use the Monte Carlo method directly to simulate the final underlying asset's price at the time T and calculate the payoff. Otherwise if $F + w$ is greater than duration T, the path will not be admissible. So this path is not taken and the payoff is zero.

Finally, if μ is less than 0, it is a defective inverse Gaussian distribution and the sampling variate F is a conditional distribution given $F < \infty$. We should multiply $e^{2\lambda/\mu}$ to the price to adjust the first-passage time distributions. For standard knock-in barrier options, i.e., $w = 0$, their fast simulation algorithm is similar to knock-in Parisian options'. See Figure 4.4 for the complete algorithm.

1: $\Delta t := T/n$;
2: $\hat{\mu} := r - q$;
3: $\mu := \dfrac{\ln H - \ln S_0}{\hat{\mu} - \sigma^2/2}$;
4: $\lambda := \dfrac{(\ln H - \ln S_0)^2}{\sigma^2}$;
5: $C := 0$; {Accumulated option value.}
6: **for** $i := 1$ to M **do**
7: sample $F \sim IG(|\mu|, \lambda)$;
8: **if** $F \le T$ and $w = 0$ **then** {Standard knock-in barrier options.}
9: $St := H \cdot \exp\left[\left(\hat{\mu} - \dfrac{\sigma^2}{2}\right) \cdot (T - F) + \sigma\varepsilon\sqrt{T - F}\right]$;
10: $C := C + V(S(T)) \cdot e^{-rT}$;
11: **else if** $F + w \le T$ **then**
12: $St := H$;

13: $l := 0$;

14: **for** $j := \left\lceil \frac{igT}{\Delta t} \right\rceil$ to n **do**

15: $St := St \cdot \exp\left[\left(\hat{\mu} - \frac{\sigma^2}{2}\right) \cdot dt + \sigma\varepsilon\sqrt{\Delta t}\right]$;

16: **if** $St \geq H$ **then**

17: $l := l + \Delta t$;

18: **if** $l \geq \mathrm{w}$ **then**

19: $St := St \cdot \exp\left[\left(\hat{\mu} - \frac{\sigma^2}{2}\right) \cdot (n-j) \cdot \Delta t + \sigma\varepsilon\sqrt{(n-j) \cdot \Delta t}\right]$;

20: $C := C + V\left(S(T)\right) \cdot e^{-rT}$;

21: **break**;

22: **end if**

23: **else**

24: $l := 0$;

25: **end if**

26: **end for**

27: **end if**

28: **end for**

29: **if** $\mu < 0$ **then**

30: $C := C \cdot e^{2\lambda/\mu}$;

31: **end if**

32: return C/M;

Figure 4.4: Fast simulation algorithm for up-and-in Parisian options.

If the payoff is the same as the standard call or put, we also can just use the Black-Scholes formula rather than the Monte Carlo method in Line 9, Line 10, Line 19 and Line 20 in Figure 4.4 for variance reduction. Replace Line 9 and Line 10 with

$$C := C + BS\left(H, K, r, T-F, \sigma, q\right) \cdot e^{-rF};$$

The condition of this problem is equivalent to pricing with the current stock price $S_0' = H$, the strike price K, the time to expiration $T-F$, the volatility of the stock price σ, the risk-free interest rate r, and the dividend yield q. When using the Black-Scholes formula, it should discount with e^{-rF} not e^{-rT}. We can also replace

Line 19 and Line 20 with

$$C := C + BS\left(St, K, r, \left(n - j\right) \cdot \Delta t, \sigma, q\right) \cdot e^{-r \cdot j \cdot \Delta t};$$

See Figure 4.5 for the revised algorithm.

1: $\Delta t := T/n$;
2: $\hat{\mu} := r - q$;
3: $\mu := \dfrac{\ln H - \ln S_0}{\hat{\mu} - \sigma^2/2}$;
4: $\lambda := \dfrac{\left(\ln H - \ln S_0\right)^2}{\sigma^2}$;
5: $C := 0$; {Accumulated option value.}
6: **for** $i := 1$ to M **do**
7: sample $F \sim IG\left(|\mu|, \lambda\right)$;
8: **if** $F \le T$ and $w = 0$ **then** {Standard knock-in barrier options.}
9: $C := C + BS\left(H, K, r, T - F, \sigma, q\right) \cdot e^{-rF}$;
10: **else if** $F + w \le T$ **then**
11: $St := H$;
12: $l := 0$;
13: **for** $j := \left\lceil \dfrac{igT}{\Delta t} \right\rceil$ to n **do**
14: $St := St \cdot \exp\left[\left(\hat{\mu} - \dfrac{\sigma^2}{2}\right) \cdot \Delta t + \sigma\varepsilon\sqrt{\Delta t}\right]$;
15: **if** $St \ge H$ **then**
16: $l := l + \Delta t$;
17: **if** $l \ge$ w **then**
18: $C := C + BS\left(St, K, r, \left(n - j\right) \cdot \Delta t, \sigma, q\right) \cdot e^{-r \cdot j \cdot \Delta t}$;
19: **break**;
20: **end if**
21: **else**
22: $l := 0$;
23: **end if**
24: **end for**
25: **end if**

26: **end for**
27: **if** $\mu < 0$ **then**
28: $C := C \cdot e^{2\lambda/\mu}$;
29: **end if**
30: return C/M;

Figure 4.5: Fast simulation algorithm for up-and-in Parisian options with the Black-Scholes formula.

4.4 General Case

To price down-and-in Parisian options, we reuse the algorithm in Figure 4.5 and just replace Line 15 with

if $St \leq H$ **then**

Then all knock-in Parisian options could be priced. From the in-out parity, we knew a standard European option is equivalent to a portfolio of a European knock-out option and a European knock-in option with an identical barrier. To price knock-out options, we can use the Black-Scholes formula to subtract knock-in options we simulated.

All above Parisian options that we discussed are consecutive Parisian options. There is another kind of Parisian-type options. The cumulative Parisian feature counts the cumulative time that underlying asset's price spends at or above (at or below) the barrier throughout the whole life of the option. It is trivial to modify our fast Monte Carlo simulation to evaluate cumulative Parisian options. For example, to price an up-and-in cumulative Parisian option, delete Line 21 and Line 22 in Figure 4.5. For other types of cumulative Parisian option, their fast simulation algorithm could be discussed similar to consecutive Parisian options' in the previous paragraph.

4.5 Parallel Processing

Multiprocessor systems (also known as tightly coupled systems) have two or more processors in close communication, sharing the computer bus and sometimes the clock, memory, and peripheral devices. They have three main advantages:

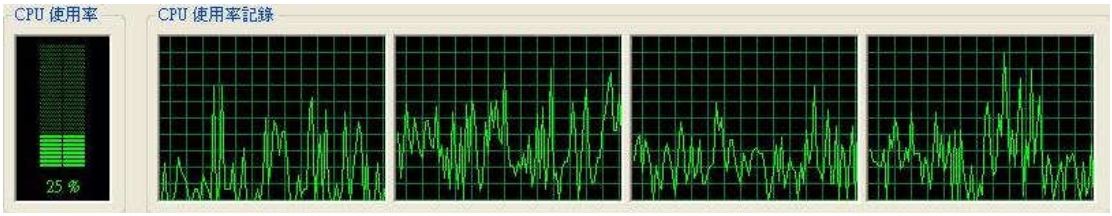
1. Increased throughput.
2. Economy of scale.
3. Increased reliability.

Nowadays the central processing unit (CPU) often combines two or more independent cores into a single package composed of a single integrated circuit (IC). A multi-core CPU functions as a chip-level multiprocessor. Aside from architectural considerations such as cache, memory, and bus contention, these multi-core CPUs appear to the

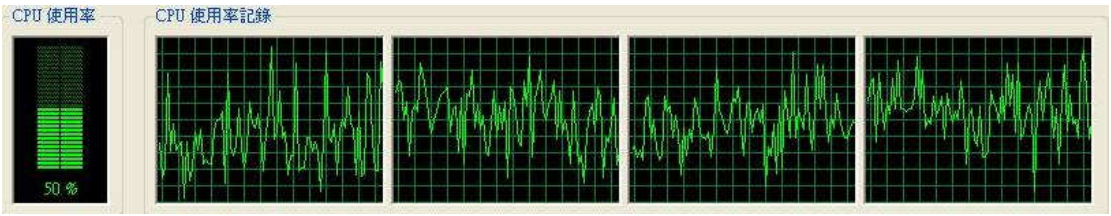
operating system as N standard processors [18].

If we execute a single-thread process, it can only utilize one core; as a result, the multi-core CPU is not fully exploited. Parallel programming techniques can benefit from multiple cores directly (see Figure 4.6). Some existing parallel programming models such as Message Passing Interface (MPI) can be used on multi-core systems. It can also apply to a multiprocessor system or a distributed system (also known as loosely coupled systems) which is a collection of physically separate, possibly heterogeneous computer systems interconnected by a communication network.

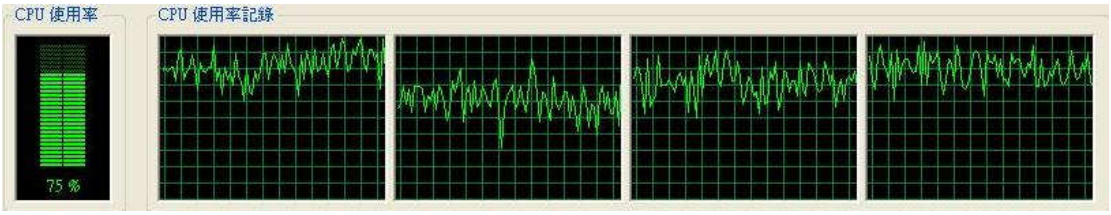
One single-thread process executed:



Two single-thread processes executed:



Three single-thread processes executed:



Four single-thread processes executed:

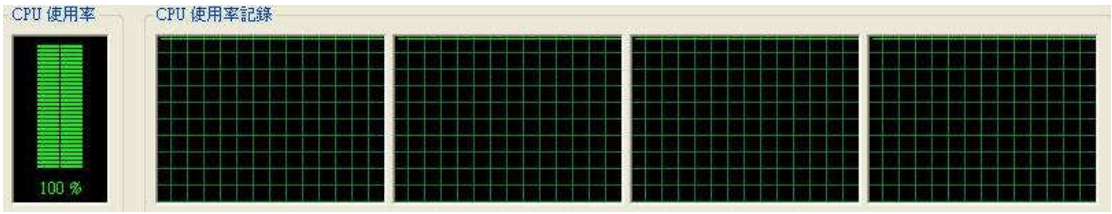


Figure 4.6: A quad-core processor usage for one, two, three, and four single-thread processes simultaneously.

Some computation-intensive tasks can take advantage of parallel processing for

much faster performance [12]. The paths generated by Monte Carlo simulation are independent. So it is straightforward to apply parallel programming. We can build a high-performance computing (HPC) clustered system to accomplish computational work and take the core as the unit by MPI platform. Our HPC system includes three dual-core PCs, i.e., six cores. The parallel processing for our simulation algorithm is described as follows:

1. Partition the number of paths into few paths for each core.
2. Each core computes the simulation job sequentially.
3. When all cores finish the jobs, we collect the meta-data and average them to get the final result.

Note that once the work has been divided, no communication among the jobs is needed before the collection stage. Good speed-ups have been obtained. Numerical results in next section are provided from our HPC system.

4.6 Numerical Results

According to Section 4.4 to price the same Parisian options in Table 3.4, we can use the Black-Scholes formula to subtract the up-and-in Parisian option we simulated. Table 4.1 is generated from the fast simulation algorithm with $M = 1000000$ and $n = 1000000$. The parameter $count$ means the average of the actually divided time intervals in the fast simulation algorithm. If using crude Monte Carlo simulation, $count$ must be equal to n. So we have saved divided time intervals.

$w = 0$ days		$w = 5$ days		$w = 10$ days		$w = 15$ days	
count	*PC*	*count*	*PC*	*count*	*PC*	*count*	*PC*
1.394	0.000141	79930.35	0.000232	108568.70	0.000275	121904.71	0.000312

Table 4.1: Numerical results of different window periods from the fast simulation algorithm.

Because we use Michael's sampling method, the time before first-passage time is only divided one time interval, i.e., F. Consider that the barrier H is equal to the initial underlying asset's price S_0, the first-passage time F is zero. Our fast simulation algorithm will degenerate to Monte Carlo simulation. But we also combine Monte Carlo simulation and the Black-Scholes formula to save the time for simulating the payoff. The Black-Scholes formula is regarded as just one time interval.

Chapter 5
Conclusions

Trees and Monte Carlo simulation are two main kinds of numerical method for valuing derivatives. This thesis proposes two fast algorithms based on them to pricing Parisian options, respectively.

About trees, we review Costabile's algorithm and try to refine it. Observe combinatorial methods deeply and use programming techniques, not only its running time but also required space can be saved. Actually, the time complexity and space complexity are both reduced by an order successfully.

About simulations, we create processes to simulate underlying asset’s price processes. Monte Carlo Simulation is easy to apply when payoffs are path-dependent. To combine simulations and the inverse Gaussian distributions, we save divided time intervals. To combine simulations and Black-Scholes formula, we reduce the variance of sample price. Monte Carlo simulation can also be sped up by parallel processing.

Complete algorithms for all types of Parisian option are available. They are easy to implement in practice. As a result, numerical results are given to suggest the correctness of these two fast algorithms.

Bibliography

[1] Avellaneda, M., and Wu, L., 1999, Pricing Parisian-style options with a lattice method, *International Journal of Theoretical and Applied Finance*, 2, 1–16.

[2] Black, F., and Scholes, M., 1973, The pricing of options and corporate liabilities, *Journal of Political Economy*, 81, 637–659.

[3] Boyle, P.P., 1977, Option: a Monte Carlo approach, *Journal of Financial Economics*, 4, 323–338.

[4] Boyle, P.P., and Lau, S.H., 1994, Bumping up against the barrier with the binomial method, *The Journal of Derivatives*, 1(4), 6–14.

[5] Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C., 2001, *Introduction to Algorithms*, 2nd edition. Cambridge, MA: The MIT Press.

[6] Costabile, M., 2002, A combinatorial approach for pricing Parisian options, *Decisions in Economics and Finance*, 25(2), 111–125.

[7] Cox, J.C., Ross, S.A., and Rubinstein, M., 1979, Option pricing: a simplified approach, *Journal of Financial Economics*, 7, 229–263.

[8] Hogg, R.V., and Tanis, E.A., 2001, *Probability and Statistical Inference*, 6th edition. Upper Saddle River, NJ: Prentice-Hall.

[9] Hull, J.C., 2006, *Options, Futures, and Other Derivatives*, 6th edition. Upper Saddle River, NJ: Prentice-Hall.

[10] Lint, J.H. Van, and Wilson, R.M., 2001, *A Course in Combinatorics*, 2nd edition. New York, NY: Cambridge University Press.

[11] Lyuu, Y.-D., 1998, Very fast algorithms for barrier options pricing and ballot problem, *The Journal of Derivatives*, 5(3), 68–79.

[12] Lyuu, Y.-D., 2002, *Financial Engineering and Computation: Principles, Mathematics, Algorithms*. New York, NY: Cambridge University Press.

[13] Merton, R.C., 1973, Theory of Rational Option Pricing, *Bell Journal of Economics and Management Science*, 4, 141–83.

[14] Michael, J.R., Schucany, W.R., and Haas, R.W., 1976, Generating random variates using transformations with multiple roots, *American Statistician*, 30, 88–90.

[15] Papadimitriou, C.H., 1995, *Computational Complexity*. Reading, MA: Addison-Wesley.

[16] Patterson, D.A., and Hennessy, J.L., 2005, *Computer Organization and Design: The Hardware/Software Interface*, 3rd edition. San Francisco, CA: Morgan Kaufmann.

[17] Rosen, K.H., 2003, *Discrete Mathematics and Its Applications*, 5th edition. New York, NY: McGraw-Hill.

[18] Silberschatz, A., Galvin, P.B., and Gagne, G., 2006, *Operating System Principles*, 7th edition. Hoboken, NJ: John Wiley & Sons.

[19] Tweedie, M.C.K., 1947, Functions of a statistical variate with given means, with special reference to Laplacian distributions, *Proceedings of the Cambridge Philosophical Society*, 43, 41–49.

[20] Wasan, M.T., 1969, First passage time distribution of Brownian motion with positive drift (Inverse Gaussian Distribution), Queen's Papers in Pure and Applied Mathematics, No.19, Queen's University, Kingston, Ontario.

[21] Whitmore, G..A., 1979, An inverse Gaussian model for labour turnover, *Journal of the Royal Statistical Society*, A142, 468–478.

[22] William, H.P., Brain, P.F., Saul, A.T., and William T.V., 1992, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd edition. New York, NY: Cambridge University Press.

[23] 冼鏡光，2002，*名題精選百則：使用 C 語言—技巧篇*，第二版，臺北市：儒林。

[24] 陳威光，2001，*選擇權：理論、實務與應用*，臺北市：智勝文化。

[25] 陳威光，2001，*衍生性金融商品：選擇權、期貨與交換*，臺北市：智勝文化。

[26] 黃達業，2004，*選擇權、期貨與其他衍生性商品*，臺北縣：普林斯頓國際。

[27] 鄭守成，2002，*C 語言 MPI 平行計算程式設計*，新竹市：國家高速網路與計算中心。

[28] 顏月珠，2004，*應用統計學—最新課程含 Microsoft Excel、SAS 範例*，修訂版，臺北市：臺大法律學院圖書文具部。