Multilevel Monte Carlo Simulation for Options Pricing

Pei Yuen Lee
MSc Computing and Management
Session 2010/2011

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

_____

Summary

Mathematical models developed to price options pose challenges in producing solutions that can achieve a high level of accuracy and efficiency. In the current research, much attention has been drawn to numerical methods of pricing options, particularly in the binomial options pricing model and the Monte Carlo simulation. This project evaluates the two models by calculating the errors produced from the approximated solutions and verifying that the solutions produced converges in some way to an analytical solution. A multillevel method for the Monte Carlo simulation has been introduced to reduce the computational complexity, which we will also investigate. The results of which show a significant amount of computational savings due to the multilevel setting.

This project is organised as follows. Chapter 1 outlines the overview of the project, which includes the aim, project methodology and the problem description. Chapter 2 provides a background research to options pricing and highlights current research interests in this area. In this chapter, we also introduce the Black-Scholes model for the valuation of European options to provide a foundation for the evaluation of other options pricing models. In Chapter 3, we study the binomial options pricing model through implementation, and then apply a test case. We then investigate the behaviour of this model by comparing the results from this model with the solution of the Black-Scholes model. In Chapter 4, we implement another options pricing model: the Monte Carlo simulation and similarly, we apply a test case and study the behaviour of the model under certain change of variables. In addition to the standard Monte Carlo simulation, we also investigate its discretised versions: the Euler and Milstein schemes, both of which are applied in the next chapter. In Chapter 5, we study the multilevel Monte Carlo simulation for pricing options in terms of efficiency. In the first section, we implement the method using the Euler scheme whilst in the second section, we implement the method using the Milstein scheme. Results show that there is an increased efficiency of using Milstein scheme as compared to using the Euler scheme. In Chapter 6, we summarise the evaluation that was done on individual model in the previous chapters and also evaluate the implementation of the models. All models demonstrated in this report are implemented using Java.

Acknowledgements

First and foremost, my utmost gratitude goes to my project supervisor, Matthew Hubbard, for his support and insight that steers the direction of the project. His invaluable guidance has helped me overcome the challenges especially in the later development of the project and his dedication in the weekly meetings has been motivational. I would sincerely like to thank my project assessor, Hamish Carr, for providing constructive feedback for my interim report as well as during the progress meeting.

I would also like to thank Daphne Lai Teck Ching for taking time off her busy schedule to proofread my report. To my coursemates who helped in one way or another throughout the course of my project I thank you all. Special thanks are due to Philipp Schroeder for his tips in making good use of the tool for the write-up. Also, thanks are due to my flatmates for their inspiration. Finally, my heartfelt thanks go to my family, without their encouragement and support I couldn't have gone this far.

Contents

Chapter 1

Introduction

1.1 Overview

Finance is a rapidly developing area in the corporate world today. Thus, new challenges posed to financial models are inevitably becoming important in order to obtain more accurate results that is close to real world scenarios. Our interest is in the valuation of options. In particular, we first study an options pricing (OP) model that produces an analytical solution, and then analyse two numerical options pricing models in terms of accuracy. Finally, we focus on improving the efficiency of one of the models.

1.2 Aim and objectives

The original aim was to develop and evaluate a computational tool for simulating the binomial OP model and the Monte Carlo (MC) simulation for the valuation of European options. However, as the work progressed and as more background materials are gathered, a need to steer the project in a different direction came into light for several reasons as summarised below.

1. Current research interests in the area of financial modelling are prominent in finding ways to improve options pricing models, by introducing alternative mathematical methods to price options or by modifying current models. A project related to evaluating methods to improve options pricing models would make a good contribution to this research area.

2. Developing a computational tool requires a programming language that can call a plotting library or a separate tool for plotting static graphs. This initial set up had already proven to be time-consuming for the author so there is a risk of not completing the project in time if an attempt is taken to build a tool to price options.

Thus, the project took a different direction towards a modelling approach. In the earlier development, we aim to investigate several options pricing models and find ways to improve the models and later on, we focus on improving the MC simulation. Therefore, we present here the current modified aim of the project, which is to improve the efficiency of the MC options pricing model by simulating a multilevel MC simulation for different styles of options.

The objectives are to:

- Understand the different models proposed in options pricing and additional extensions or modifications made to improve the models.
- Implement the algorithms for the binomial OP model and the MC simulation.
- Test and evaluate the accuracies of the binomial and MC simulation relative to the solution obtained from the BS formula.
- Implement the algorithm for the multilevel MC simulation.
- Apply the multilevel MC simulation to the European option.
- Apply the multilevel MC simulation to the Asian option.
- Evaluate the efficiency of the multilevel MC simulation for the European and Asian options.

1.3 Minimum Requirements

The following minimum requirements were defined at the beginning of the project:

1. A binomial simulation of the European option.
2. A MC simulation of the European option.
3. A prototype of a static interface to visualize the results.

A further extension to the minimum requirements was added as follows:

1. A multilevel MC simulation.

1.4 Project Methodology

This project is divided into small groups of subprojects, where for each group, we investigate the behaviour of an options pricing model in terms of how well it approximates a solution or converges towards one. The project begins with background research on options pricing models, which includes the Black Scholes (BS) model, binomial OP model and the MC simulation. The background reading also includes current research in improving these models.

The next step is to study the behaviour of binomial options pricing model. This includes implementing the model and providing a test case to test the model. The evaluation is then done by comparing the accuracy of this model relative to the BS model.

The MC simulation is then investigated. Similarly, the model is implemented and tested with the same test case for consistency. Again, the evaluation is carried out by showing convergence of this MC value to the BS value. Two discretised methods are then introduced: Euler and Milstein schemes, to which implementation, testing and evaluation are carried out.

Finally, an improved MC method in terms of efficiency is investigated. Similarly, the model is implemented. Testing and evaluation includes finding the computational costs and the root-mean-square error of the model. As an extension, the Milstein scheme is introduced to this model to further improve the efficiency. The same methodology applies for this multilevel Monte Carlo simulation with the Milstein scheme.

1.5 Project Schedule

Revision of the original project schedule is necessary due to the change of project direction. The original traditional approach (see Appendix C.5) taken at the very early stage is discarded due to a change in the structure of the methodology. The revised project schedule is shown in the Gantt chart below. Since the models in this report have already been designed and developed by researchers, the project focuses on implementing, testing and evaluating these models.

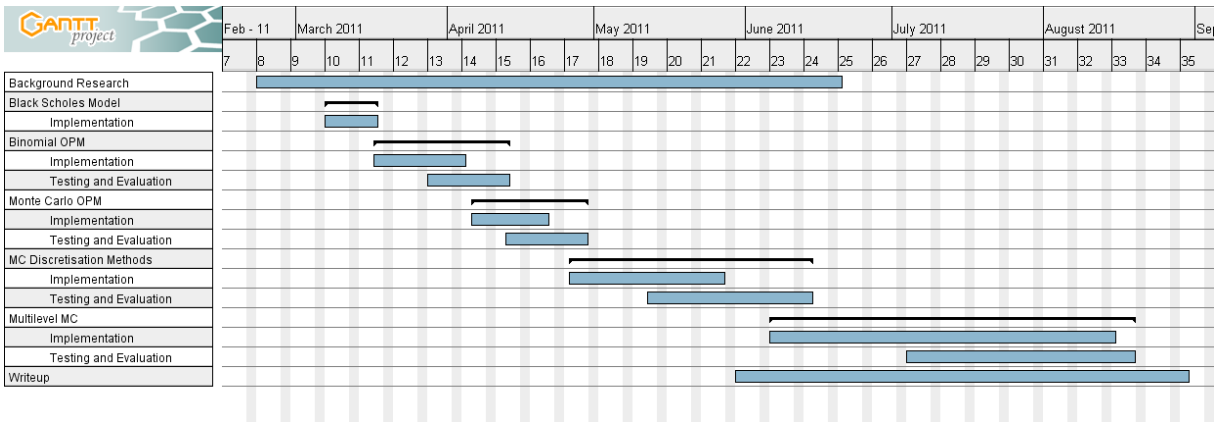


Figure 1.1: A revised Gantt chart outlining the project schedule.

Chapter 2

Background Research

2.1 Literature Review

In the world of finance, mathematical models can be used as approximations to value complex real market derivatives. The modelling of financial options gained its popularity when Fisher Black and Myron Scholes (1973) introduced the BS model, which later became the foundation of the literature on options pricing where various studies are made on extending the model and developing alternative approaches to the valuation of options. A recent literature by Broadie and Detemple (2004) focuses on the trends and development of financial options modelling with emphasis on the development of models that depart from the assumptions of the classic BS model, since empirical evidence suggests that the BS prices tend to differ from the market prices of options due to the assumption that sharp changes in stock prices are negligible (MacBeth and Merville, 1979; Vasile and Armeanu, 2009). Several modifications of the model have been made to reduce discrepancies between these assumptions and the real world. Examples are the extension of the BS model with illiquidity (Cetin et al., 2004), the inclusion of transaction costs through adjusting the volatility (Leland, 1985), and also extensions to include jump-diffusion models and stochastic volatility models.

Since financial markets undergo stochastic fluctuations, numerical methods such as MC methods become useful tools to price options. Alternatively, binomial methods are discrete numerical approaches that can value options at any point in time until expiration. The literature has also expanded beyond the basics of these numerical methods, such as Giles (2007)'s work on improving efficiency by introducing a multilevel approach to the MC method, and most recently, Kyoung and Hong (2011) presented an improved binomial method that uses cell averages of payoffs around each node in addition to the standard method. Essentially in this literature, the goal is to improve both the accuracy and efficiency in approximating values of OP models.

For the convenience of further discussion, the notations used throughout the paper are summarized below:

S	price of underlying asset
K	strike or exercise price
C	value of the European call option
r	risk-free interest rate
t	time in years
T	maturity date
σ	volatility of returns of the underlying asset
μ	drift rate
p	a probability measure

Common abbreviations used are:

GBM	geometric Brownian motion
OP	options pricing
BS	Black-Scholes
MC	Monte Carlo
PDE	partial differential equation
SDE	stochastic differential equation

2.2 Options

An *option* is a derivative security that grants the buyer of the option the right, but not the obligation, to buy or sell an underlying asset, S (such as a stock, a bond or an index portfolio) on or before an expiration date, T, for an *exercise* or *strike price*, K. A *call option* is the right to buy, while a *put option* gives the right to sell. Let's take an example of a call option. Say a company holds 100 shares of a stock priced at \$20 each. An investor believing the price will go up in a month's time enters into a contract with the company to buy the stock at, say \$19 after one month. All the investor needed to pay is the premium of (stock - strike)$= 20 - 19 =$ \$1 per share. If the price did go up on the exercise date, the investor will exercise the option and gain the profit of buying cheap and selling high in the market. If the price goes down, the contract will expire and becomes worthless so he will only lose the premium price he paid to enter into the contract in the first place.

2.2.1 Option Styles

Exercising the options can be of several styles and some common ones are listed below. The first two are plain vanilla options. The third option is a non-vanilla option and the rest are exotic options.

- European options: Options that can only be exercised on the expiration date.
- American options: Options that can be exercised on or before the expiration date.
- Bermudan options: Options that can be exercised at any fixed period of time.
- Asian options: Options whose payoff depends on the average underlying asset over a certain period of time.
- Barrier options: Options either come into existence after a barrier is breached (up-and-in or down-and-in) or drop out of existence as a result of breaching the barrier (up-and-out or down-and-out).
- Lookback options: Options that depend on the minimum (for call) or maximum (for put) value of the stock price over a certain period of time.
- Digital options: Options whose payoff is fixed after the underlying asset exceeds the exercise price.

2.2.2 The Put-Call Parity

For the European style option, the relationship between call and put options can be derived from their payoffs when they have the same underlying asset price, strike price and expiration date. The no-arbitrage assumption, which places a bound on the options, is important for this principle so that the same payoff is maintained for both the call and put options. The idea is that if a portfolio containing a call option has the same payoff at expiration as a portfolio containing a put option, then they must have the same value at any given time before the expiration. This is known as the put-call parity.

The derivation begins with both options having an equal payoff (Wilmott, Howinson and Dewynne, 1998). Let C and P be the value of the call and put options at any time t respectively, and let T be the time at expiration, K the strike price and S the stock price at time t. Then, the payoffs at expiration are

$$\begin{aligned} C &= \max(S-K,0), \quad \text{and} \\ P &= \max(K-S,0). \end{aligned}$$

The payoff at expiry is

$$\begin{aligned} C - P &= \max(S-K,0) - \max(K-S,0) \\ &\begin{cases} (S-K) - 0 & \text{if } S \geq K \\ (0-(K-S) & \text{if } S \leq K \end{cases} \\ &= S - K. \end{aligned}$$

Finally, discounting the value of the portfolio, the put-call parity is defined as

$$C - P = S - K.e^{-r(T-t)},$$

where r is the discounted risk-free rate. The basic idea of the put-call parity can be applied to the Black-Scholes model that values call and put options independently, which we will derive in 2.3.

2.2.3 Risk-Neutral Valuation

A risk-neutral measure is a measure applied to arbitrage-free option valuation where the growth rate μ is replaced by the risk-free rate r. For example, for a continuous-time measure, we define a stochastic process, that is, a geometric Brownian motion (GBM) with the following stochastic differential equation (SDE):

$$dS = \mu S dt + \sigma S dW, \tag{2.1}$$

where σ is the volatility and W is a Brownian motion. To make the equation risk-neutral, dW is redefined with a new measure so that we get

$$dW = d\tilde{W} + \frac{\mu - r}{\sigma} dt.$$

This equation is a result of applying the Girsanov theorem, which calculates the likelihood ratio of the original measure and the risk-neutral measure (refer Seydel (2005)). Hence, Equation (2.1) yields

$$\begin{aligned} dS &= \mu S dt + \sigma S \left(d\tilde{W} - \frac{\mu - r}{\sigma} dt \right), \\ &= \mu S dt - (\mu - r) S dt + \sigma S_t d\tilde{W}, \\ &= r S dt + \sigma S d\tilde{W}. \end{aligned}$$

2.3 The Black-Scholes Model

The BS model is a classic example of an options pricing model that was developed by Fisher Black and Myron Scholes in their seminal work in 1973. Their approach to options pricing

problems is to solve a partial differential equation (PDE) with a final condition at $t = T$ to obtain a unique solution. The fundamental idea is to find a closed-form solution to the Black-Scholes PDE by first using the Ito's calculus from Ito's lemma to obtain the BS equation, then transform it to the heat equation to get the unique solution, and finally transform the solution back to find the corresponding solution of the Black-Scholes PDE.

2.3.1 Assumptions

As mentioned earlier, the no-arbitrage assumption is an important assumption in options pricing, but in the world of option valuation, there are several other assumptions to follow. The assumptions for this model which can be applied to other subsequent models are:

1. There is no arbitrage opportunity so there is a premium price for buying the option.
2. The price follows a GBM with constant drift and volatility.
3. The underlying stock does not pay dividends.
4. There is a constant interest rate.
5. There is no transaction cost and tax.
6. It is possible to buy or sell any amount of options at any given time.

2.3.2 The Black-Scholes PDE

The Black-Scholes PDE is an important part of the BS model. This PDE describes the option over time and is used to obtain the BS formula for pricing options. The underlying asset is assumed to follow the GBM with an SDE as defined in Equation (2.1). Itô's lemma states that for the SDE defined and any twice differential function, C, of S and t, we have

$$dC = \left(\mu S\frac{\partial C}{\partial S} + \frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 C}{\partial S^2}\right)dt + \sigma S\frac{\partial C}{\partial S}dW. \tag{2.2}$$

The Wiener process dW is random so we want to eliminate this variable in order to obtain the PDE. This can be achieved by constructing a portfolio Π consisting of a long call for an option and a short Δ shares of the underlying asset. A long call is the purchase of a call option while a short call is the selling of the underlying asset. Therefore, the portfolio is defined as

$$\Pi = C - \Delta S.$$

A small change in the portfolio for a time period of $[t, t + \Delta t]$ results in

$$d\Pi = dC - \Delta dS.$$

Applying Equations (2.1) and (2.2) into the equation yields

$$d\Pi = \left(\mu S\frac{\partial C}{\partial S} + \frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{d^2C}{dS^2}\right)dt + rS\frac{\partial C}{\partial S}dW - \Delta(\mu S dt + \sigma S dW). \tag{2.3}$$

To eliminate any risk of price movement, we apply delta hedging, which simply means that $\Delta = \frac{\partial C}{\partial S}$, to Equation (2.3) to get

$$d\Pi = \left(\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 C}{\partial S^2}\right)dt.$$

The assumption of no-arbitrage defines the rate of return of the portfolio as $d\Pi = r\Pi dt$. Therefore, the Black-Scholes PDE is given by

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 C}{\partial S^2} + \frac{\partial C}{\partial S}rS - rC = 0.$$

2.3.3 The Black-Scholes Formula

For a call option, the boundary condition is $\max(S - K, 0)$. If we set this boundary condition to the Black-Scholes PDE, we obtain the following BS formula

$$C = SN(d_1) - Ke^{-rt}N(d_2), \tag{2.4}$$

with

$$\begin{aligned} d_1 &= \frac{ln(\frac{S}{K}) + (r + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}} \quad \text{and} \\ d_2 &= d_1 - \sigma\sqrt{t}, \end{aligned}$$

where $N(d_1)$ and $N(d_2)$ denote the cumulative distribution functions of the standard normal distribution for d_1 and d_2 respectively. The detailed derivation of the BS formula can be found in many standard books such as Wilmott, Howinson and Dewynne (1998), Ugur (2008) and Lovelock, Mendel and Wright (2007).

We take a test case for pricing a European call option to illustrate this model. Suppose that a stock price, S, of a company is currently \$250 per share, and that in a year's time ($T = 1$), the price either rises or falls by 20% ($\sigma = 0.2$). A European call option is to buy the stock at an exercise price, K, of \$200 at the expiration time with a risk-free rate of 5% ($r = 0.05$). To price the call option, we apply the given set of variables into Equation (2.4), and obtain $C = \$61.472091898474446$. This value was calculated from the implementation of the BS model in Java. The algorithm for this model is trivial and hence will not be elaborated here. The BS value obtained is the market value of the call option for the underlying stock. Mathematically, it is a closed-form solution of the call option that will form the basis of our result for comparisons with other methods of pricing options, in particular the binomial and Monte Carlo OP models.

One of the limitations of the BS model is it can only be used to price European options. To price American options, numerical methods such as the binomial OP method (Cox, Ross and Rubenstein, 1979) and finite difference methods (Schwartz, 1977) are required. For exotic options (e.g. Asian, Lookback and Barrier), the MC method is normally used. We shall focus on two models in detail: the binomial OP model and the MC simulation. These models are alternative methods that are approximations to the BS solution for European options. Running approximations of the BS value help in validating results and reducing errors of the models so that they can be applied to the valuations of more complex options such as American and Asian options.

Chapter 3

Binomial Options Pricing Model

The binomial OP model is a lattice tree model that approximates a continuous random walk in discrete time with a fixed number of periods. A direct relationship of this model with the BS model may not be immediately evident but in the case of European options, the binomial value converges to the BS value as the number of periods increases. This model shares the same basic assumptions as the BS model and assumes an asset price path that follows a GBM.

3.1 Valuing the Options

In essence, the binomial OP model divides the time line into m equally-spaced intervals, where for each period $\delta t = T/m$, the price either goes up by an up-factor u or down by a down-factor d. Thus, if the current stock price is S, the stock price at the next period is either Su or Sd. For the next period, Su goes up to Suu or down to Sud and similarly, Sd goes to Sdu or Sdd. Notice that the stock price recombines at this stage since $Sdu = Sud$ (see Figure 3.1), therefore this reduces the number of possible prices so that after m periods, there are only $m+1$ possible prices. We next define values for the parameters u and d. The Cox, Ross and Rubenstein (1979) (CRR) method assumes that u and d are determined by the volatility σ, such that

$$\begin{aligned} u &= e^{\sigma\sqrt{\delta t}}, \quad \text{and} \\ d &= \frac{1}{u} = e^{-\sigma\sqrt{\delta t}}. \end{aligned}$$

Another important assumption is the risk-neutrality measure. Under this assumption, an investor's risk preferences are not taken into account so therefore we assume that the return on the investment is a risk-free interest rate r. The steps involved in finding the option value is quite straight forward. We shall define the steps for finding a call option C. For a one-period

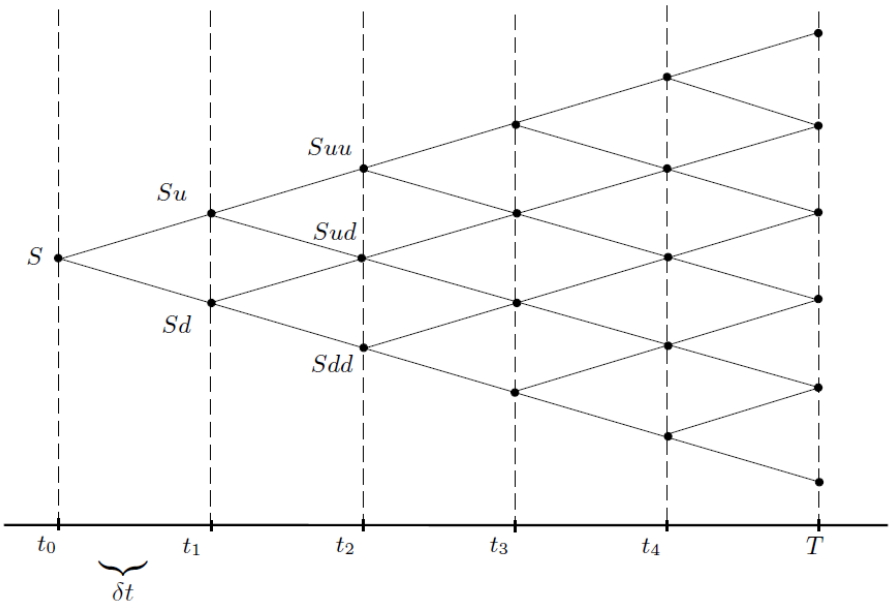


Figure 3.1: A binomial tree with $m = 5$ of possible stock prices.

binomial tree, the option will be C_u if the stock price goes to Su and C_d if the stock price goes to Sd. Hence, from the intrinsic value formula, we can define C_u and C_d as

$$\begin{aligned} C_u &= \max(0, Su - K), \text{ and} \\ C_d &= \max(0, Sd - K). \end{aligned}$$

Suppose we build a portfolio that stores shares of a stock for investment. Let Δ be the number of shares and B be the price invested in the bonds of the stock. The portfolio payoff is thus $\Delta S + B$. We can equate this to the option payoff, in this case the call option C, so that the up and down options become

$$\Delta Su + e^{r\delta t}B = C_u, \text{ and} \tag{3.1}$$
$$\Delta Sd + e^{r\delta t}B = C_d. \tag{3.2}$$

Solving Equations (3.1) and (3.2), we find that

$$\Delta = \frac{C_u - C_d}{(u-d)S} \quad \text{and} \quad B = \frac{uC_d - dC_u}{(u-d)e^{r\delta t}}.$$

Therefore, the call option is

$$C = \Delta S + B = \left[\frac{e^{r\delta t} - d}{u-d}C_u + \frac{u - e^{r\delta t}}{u-d}C_d\right] / e^{r\delta t}.$$

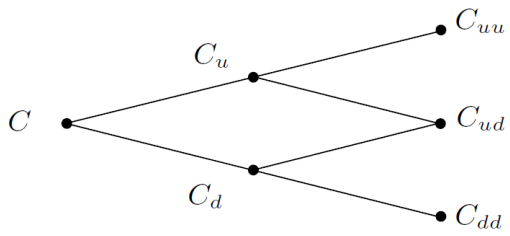


Figure 3.2: Possible option prices for a 2-period binomial tree.

To simplify the term, we let

$$p = \frac{e^{r\delta t} - d}{u - d} \quad \text{and} \quad 1 - p = \frac{u - e^{r\delta t}}{u - d},$$

and hence, we can write the call option as

$$C = [pC_u + (1 - p)C_d]e^{-r\delta t}.$$

Now we consider a call option with two periods. After the first period, C_u either goes up to C_{uu} or down to C_{ud} (see Figure 3.2). C_d is analogous. From the previous derivation, we find that

$$\begin{aligned} C_u &= [pC_{uu} + (1 - p)C_{ud}]e^{-r\delta t}, \quad \text{and} \\ C_d &= [pC_{du} + (1 - p)C_{dd}]e^{-r\delta t}. \end{aligned}$$

Algorithm 3.1: An algorithm for a binomial options pricing model.

```
function BinomialOPM (T, S, K, r, sigma, n) {
    deltaT:= T/ n;
    up:=     exp(sigma* sqrt(deltaT));
    down:=   1/up;

    cu:=     (up* exp(-r* deltaT)- exp(-q* deltaT))* up/ (up^ 2- 1);
    cd:=     exp(-r* deltaT)- cu;

    for i:= 0 to n {
        c(i):= S* up^i* down^(n-i); if c(i)< 0 then c(i)=0;
    }

    for j:= n-1 to 0 step -1 {
        for i:= 0 to j {
            c(i):= cu* c(i)+ cd* c(i+1);
        }
    }
    return c(0); }
```

Generally for m periods, the equation is given by

$$C = \left[\sum_{j=a}^{m} \left(\frac{m!}{(m-j)!j!} \right) p^j (1-p)^{m-j} u^j d^{m-j} S - K \right] e^{-r\delta t}. \tag{3.3}$$

where a is the minimum number of upward moves such that the strike price falls below the stock price upon expiry so that it can be exercised. In other words, we require that $u^a d^{m-a} S > K$ (Cox, Ross and Rubenstein, 1979). In implementing the binomial model, the multi-period steps are computed recursively where the first step involves calculating the options at the terminal nodes and then working backwards to obtain the value of the first node. Algorithm 4.1 summarises the steps in obtaining the binomial value of a call option.

We can test the algorithm using the test case from the previous section where $S = 250$, $K = 200$, $T = 1$, $r = 0.05$ and $\sigma = 0.2$, with an additional parameter m for the number of periods. If we choose m to be 10 and run the algorithm, we obtain a binomial value of approximately 61.536162. Now that we have the binomial value, we want to verify that the result is correct so we compare it with the exact solution found from the BS formula. As it turns out, the binomial value converges to the BS value as the number of periods increases. We shall look at this in detail next.

3.2 Convergence of the binomial OP model to the BS model

First, we investigate the relationship between the binomial OP model and the BS model. Equation (3.3) with m periods can be rewritten as

$$C = S\left[\sum_{j=a}^{m} \frac{m!}{(m-j)!j!} p^j (1-p)^{m-j} \frac{u^j d^{m-j}}{e^{r\delta t}}\right] - Ke^{-r\delta t}\left[\sum_{j=a}^{m} \frac{m!}{(m-j)!j!} p^j (1-p)^{m-j}\right].$$

Replacing the two parts in parentheses with functions $\phi(a; m, p')$ and $\phi(a; m, p)$ respectively, we obtain a simpler equation of the form

$$C = S\left[\phi(a; m, p')\right] - Ke^{-r\delta t}\left[\phi(a; m, p)\right],$$

where $p' = ue^{-r\delta t}p$. From Cox, Ross and Rubenstein (1979)'s work on the convergence of the binomial formula to the BS formula, as m tends to infinity,

$$\phi(a; m, p') \to N(d_1) \quad \text{and} \quad \phi(a; m, p) \to N(d_2).$$

Hence, the BS formula is a limiting case of the binomial OP model (Cox, Ross and Rubenstein, 1979; Lee and Lin, 2010).

Next, we investigate the convergence of the binomial OP model to the BS model. This can be easily demonstrated with a plot of the number of periods m against the option values found using the binomial OP model (see Figure 3.3). Cox, Ross and Rubenstein (1979) provided a proof for the convergence as m tends to infinity. Their proof uses a special case of the central limit theorem which imposes restrictions on u and d. However, the proof provided is too specific. Hsia (1983) applied a more general proof for the convergence of the Binomial OP model to the

BS formula without restricting u and d, using the DeMoivre-Laplace limit theorem with the only condition being $mp \to \infty$ as $m \to \infty$. Qu (2010) further demonstrated that there is a direct proof of the Binomial OP model converging to BS formula as m tends to infinity with the use of direct approximation of binomial probability from the normal distribution.

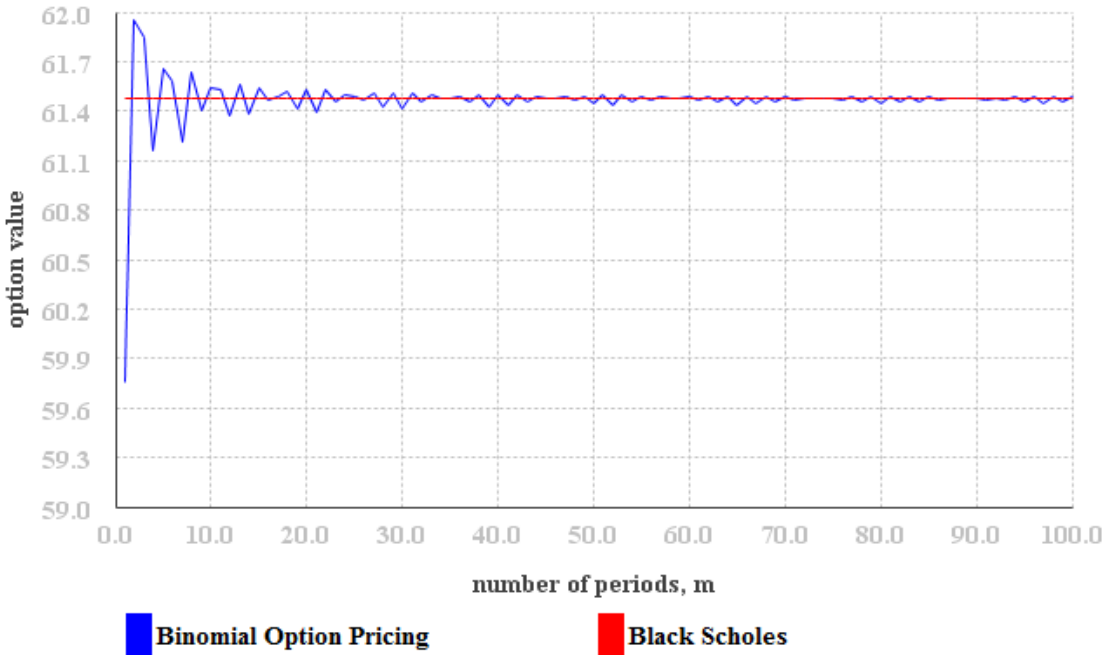


Figure 3.3: A plot demonstrating the convergence of the binomial model to the BS model as m increases

Based on Chang and Palmer (2007)'s paper, the rate of convergence from the Binomial OP model to the BS formula was found to be $1/m$. In the evaluation of the binomial OP model, we verify that this statement is true. Taking the same test case, we test the convergence with different values of m. Recall that the BS value was found to be 61.472091898474446. For each value of m, we find the absolute error such that

$$\text{error} = \text{binomial value} - \text{BS value}.$$

m	Binomial value	|error|	$1/m$	ratio = |error|/$(1/m)$
10	61.53616204233657	0.06407014386210	0.1	1.56078937820459
50	61.443894450462025	0.02819744801241	0.02	0.70928404553492
100	61.48373974924799	0.01164785077350	0.01	0.85852748240485
200	61.468107803401594	0.00398409507290	0.005	1.25499013163825
500	61.47445853544964	0.00236663697520	0.002	0.84508102466116
1000	61.47304425642073	0.00095235794630	0.001	1.05002536482089
5000	61.47232014677787	0.00022824830339	0.0002	0.87623871470794

Table 3.1: Table of absolute errors for different m for a European call option.

The model is valid if its absolute error is proportional to the convergence rate,

$$|\text{error}| \propto \frac{1}{m}.$$

The result can be seen in Table 3.1. The absolute error is found to be approximately equal to the convergence rate of $1/m$, since the ratio is roughly 1. Hence, the implementation of this model is correct.

3.3 Computational Effort

In terms of computational effort, comparisons can be made between this model and the BS model by computing the execution time when running them on the same machine. The execution time for the BS model earlier was 0.000775551 seconds, which is relatively fast compare to those of the binomial model (see Table 3.2). Furthermore, we observe that as m increases, the execution time also increases. For example, a 1000-period binomial model takes about 10 times longer to run than a 10-period binomial model.

m	execution time (s)
10	0.000831498
50	0.001185654
100	0.001634766
200	0.003117604
500	0.004586584
1000	0.008521827
5000	0.14583653

Table 3.2: Table of execution time for different m for a European call option.

We have applied the simplest option style to this model so that the validity of this model can be verified since we can compare the results with the exact solution obtained from the BS model for the European call option. With the assurance that this model is valid, we can then apply it to other option styles such as the American option where early exercises of the options are possible. In the next chapter, we shall look at another OP model that uses random numbers to generate option values after a fixed number of simulation runs.

Chapter 4

Monte Carlo Simulation

Valuing of options is not limited to European and American options, which are the most basic styles of options. There are also exotic options with complicated features, such as the Asian option that takes the average underlying asset price over a predetermined period of time, and they cannot be easily valued using binomial OP model or the BS model due to their inflexibility in implementation. Therefore, in this chapter, we present another popular approach to valuing these options: the Monte Carlo simulation. This technique can easily simulate the stochastic process using random numbers and is flexible in terms of combining multiple sources of uncertainties. Hence, it is practical for options that suffer from the *curse of dimensionality*, such as the real option. For the interest of this report, we will only apply the standard Monte Carlo simulation to European-style option to demonstrate its convergence to the BS model. Then, we will introduce the discretisation methods in this chapter as a preparation for the chapter that follows.

4.1 Valuing the Options

The MC simulation, which was first proposed by Boyle (1977), uses pseudorandom numbers to simulate price paths. It is a useful method to price options that has multiple uncertainties. We shall derive a sample path for the MC simulation. Recall in Equation (2.1) that the underlying asset is assumed to follow the GBM given by the SDE,

$$dS = \mu S dt + \sigma S dW,$$

where μ is the drift rate and σ is the volatility. Since the risk-neutrality assumption also applies here, we let $\mu = r$, where r is the risk-free interest rate. To obtain the sample path, we need to use the Itô's formula in Equation (2.2).

Algorithm 4.1: An algorithm for a standard Monte Carlo simulation.

```
function MonteCarlo {
    % m --> number of timesteps
    % n --> number of simulation paths
    timestep:= T/m;
    sum:= 0;
    for i:= 1 to n {
        for j:= 1 to m {
            S = S * exp[(r-0.5*sigma^2)*timestep+
                                    sigma*sqrt(timestep)*rand];
        }
           sum := sum + max(S-K,0);
    }
    value := sum/n*exp(-r*timestep);
    return value;
}
```

Using the properties of lognormal distribution, we let $C = \log S(t)$ and apply it to the Itô's formula to get

$$\begin{aligned} d\log S(t) &= (r - \frac{1}{2}\sigma^2)dt + \sigma dW \\ S(t) &= S(t-1)\exp\left[\left(r - \frac{\sigma^2}{2}\right)\delta t + \sigma dW\right]. \end{aligned}$$

We generate the sample path for m periods by dividing the time period $[0, T]$ into m intervals of δt to produce a sample path of

$$S(t_j) = S(t_{j-1})\exp\left[\left(r - \frac{\sigma^2}{2}\right)\delta t + \sigma\sqrt{\delta t}Z_j\right], \qquad Z_j \sim N(0,1) \quad \text{and} \quad j = 1, ..., m.$$

The payoff, $X(\omega)$, for a European call option is $max(S(t) - K, 0)$ for a sample path ω. To sample n stock price paths, we find the sample mean of the payoffs discounted to present using the risk free rate, r, to obtain

$$X = \frac{1}{n}\sum_{i=1}^{n} X(\omega_i)e^{-r\delta t}. \tag{4.1}$$

This simple iteration can be seen in Algorithm 4.1. The pseudorandom number used for this implementation is a normally distributed value from the normal distribution $N \sim (0, 1)$.

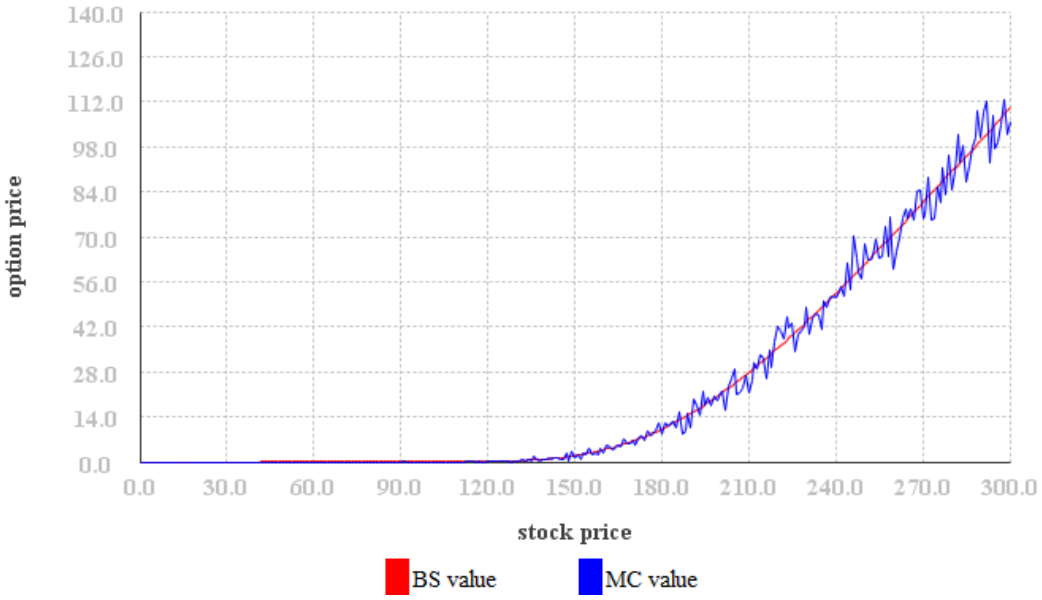


Figure 4.1: A plot comparing the Monte Carlo simulation and the BS model for $n = 100$.

We begin our test with the test case we used in the earlier example. Setting the parameters $S = 250$, $K = 200$, $r = 0.05$ and $\sigma = 0.2$, we can generate a MC value with predetermined n and m. We demonstrate the results obtained from this simulation by plotting a graph of stock price against the option price. Figure 4.1 shows this result for $n = 100$ and $m = 100$, with comparison to the BS values. However, because this simulation returns a random value with every run, we want to set the seed so that we can control the outcome when we run it with different values of n and m. As part of the evaluation, we will investigate the effect of changing n in the next section.

4.2 Convergence Test

Table 4.1 shows the effect of increasing n which was tested with three seeds. Standard textbook's convergence test suggests that the MC values tend to the BS closed-form solution when n increases. Although the convergence in Table 4.1 does not seem conclusive, there is an indication that the values are getting closer to the BS value of 61.4720918984744 (see Figure 4.2). For example, for $n = 10^2$, the difference between the highest and the lowest MC values within the three seeds are approximately 9.494630, whereas for $n = 10^6$, the difference is approximately 0.099661, indicating that there is a significant decrease in the standard deviation of the MC values as n increases.

n	MC value		
	Seed 1	Seed 2	Seed 3
10^2	59.1854875330700	55.66077308258579	65.15540374696462
10^3	62.3879080724478	60.420893879950015	60.806343515915735
10^4	61.3979056884734	60.716972462469144	61.06141350337705
10^5	61.5693015528772	61.44086461566363	61.612622484545696
10^6	61.5366498823280	61.43698915939219	61.502473728570706

Table 4.1: A table of mean MC values with $m = 1000$ and different values of n.

To find the approximation error, we first need to calculate the estimated variance. Let $a = \mathbb{E}(X)$ and $b^2 = VarX$ be the expectation of X and the variance respectively. If we obtain n samples X_i for $i = 1, 2, ..., n$, then the approximation of a is

$$\hat{a} = \frac{1}{n} \sum_{i-1}^{n} X_i.$$

Therefore, the estimated variance is

$$\hat{b}^2 = \frac{\sum_{i=1}^{n} (X_i - \hat{a})^2}{n - 1}.$$

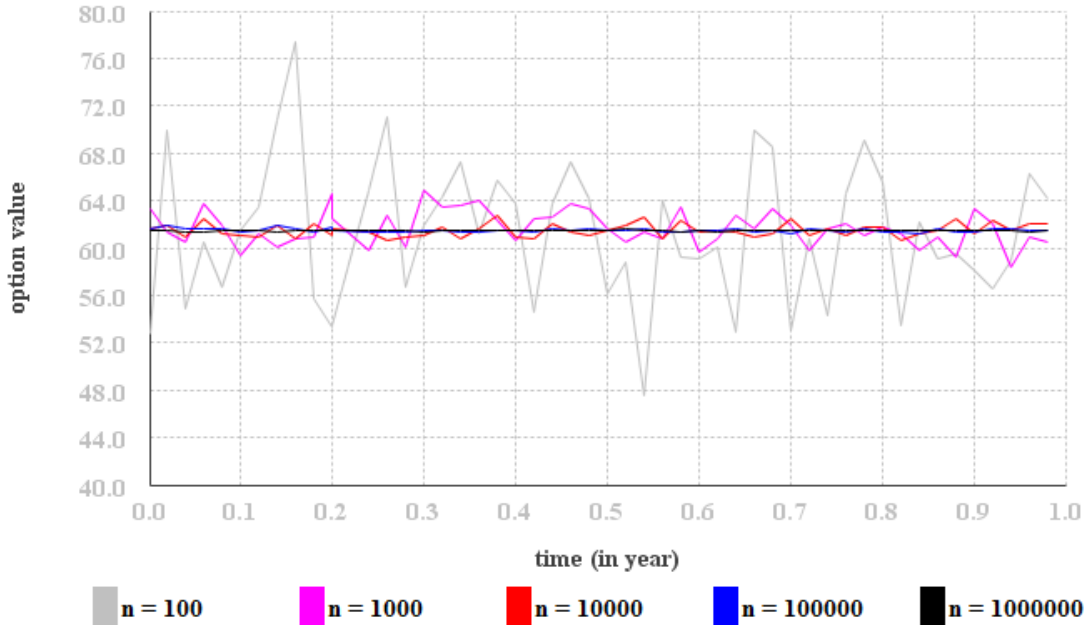


Figure 4.2: A plot of the Monte Carlo simulations for different values of n.

The central limit theorem implies that the error is approximately normally distributed with mean 0 and variance $\frac{b^2}{n}$, that is with a standard deviation of $b/\sqrt{n}$. Therefore, the rate of convergence is $\frac{1}{\sqrt{n}}$. Based on the confidence interval built in Palczewski (2009), the expected

value a should lie in the 95% interval

$$\left[\hat{a} - \frac{1.96\hat{b}}{\sqrt{n}}, \hat{a} + \frac{1.96\hat{b}}{\sqrt{n}}\right].$$

$1.96\hat{b}/\sqrt{n}$ is the measure of the error of the MC simulation. We observe in Table 5.2 that the error bound decreases at a rate of $1/\sqrt{n}$ as n increases, therefore ideally we want an n big enough to reduce the error.

n	$1.96\hat{b}/\sqrt{n}$	execution time (s)
10^1	12.5621587068127	0.008329864
10^2	9.8028069826697	0.038385436
10^3	3.0804327504420	0.300104554
10^4	0.9358969040374	2.913556035
10^5	0.2987591240892	29.08993831
10^6	0.0940535776040	289.7469993

Table 4.2: The error bound and execution time in seconds for different n.

However, the simulation becomes computationally slower as the number of paths increases as seen in Table 5.2. Moreover, when we compare this model to the binomial OP model, for the same number of intervals of 1000 and the same computational time of roughly 0.008s, the error for the binomial OP model is ~ 0.0095236 while the approximate standard deviation for the MC simulation is ~ 12.5621587 for 10 simulation paths, indicating that the MC simulation requires many runs in order to reach the level of accuracy of a binomial OP model. The model is nevertheless useful especially for option styles where the binomial OP model is not practical. Various methods have been proposed to improve the MC simulation; examples are variance reduction techniques and quasi-MC methods. In the next section, we introduce approximations of the MC simulation through discretisation. The next chapter will be built based on the discretisation methods analysed here.

4.3 Discretisation Methods

Discretisation is a method of approximating solutions to SDEs, which can be done in several ways, for example the use of Taylor series expansion to get time-discretised solutions. The approximate solution to a stochastic process X is assumed to follow the SDE

$$dX_t = a(X_t)dt + b(X_t)dW_t, \tag{4.2}$$

where a and b are coefficient functions satisfying the conditions for the existence and uniqueness of a solution to the SDE (see B.2 Glasserman (2004)). The solution X is approximated over a time interval of $[0, t_m]$ with a time step of $h = t_i/i$ for $i = 0, 1, ...m$. Integrating Equation (4.2)

from t to $t+h$ produces

$$X_{t+h} = X_t + \int_t^{t+h} a(X_u)du + \int_t^{t+h} b(X_u)dW_u. \tag{4.3}$$

4.3.1 Euler Scheme

The simplest discretisation method is the Euler scheme. In this scheme, the integrals of Equation (4.3) are approximated as products of the integrands at time t and the size of the integration domain from t to $t+h$. The approximations are therefore given by

$$\begin{aligned} \int_t^{t+h} a(X_u)du &\approx a(X_t)\int_t^{t+h} du \\ &= a(X_t)h \end{aligned}$$

and

$$\begin{aligned} \int_t^{t+h} b(X_u)dW_u &\approx b(X_t)\int_t^{t+h} dW_u \\ &= b(X_t)(W_{t+h} - W_t) \\ &= b(X_t)\sqrt{h}Z, \end{aligned}$$

where Z is the standard normal distribution. Thus, the Euler scheme can be written as

$$X_{t+h} \approx X_t + a(X_t)h + b(X_t)\sqrt{h}Z.$$

In the MC path simulation that follows a GBM for generating a stock price, S, set $X_t = S_t$, $h = \delta t$, $a(X_t) = rS$ and $b(X_t) = \sigma S$. Hence, the Euler scheme is

$$S_{t+\delta t} \approx S_t + rS\delta t + \sigma S\sqrt{\delta t}Z.$$

Algorithm 4.2: A standard Euler Scheme to obtain the payoff of the option.

```
1  function EulerScheme {
2      T := 1, S:= 250, K := 200, r := 0.05, sigma := 0.2;
3      m := 100, n := 100, sum:=0, timestep := T/m;
4
5      for i:=1 to n {
6          for j:=1 to m{
7              S = S+r*S*timestep + sigma*S*sqrt(timestep)*Z;
8          }
9          sum := sum + max(S - K, 0);
10       }
11       value = sum/n*exp(-r*T);
12       return value;
13 }
```

4.3.2 Milstein Scheme

An alternative method of discretisation that improves the speed of convergence from a strong order of 0.5 for the Euler scheme to 1 is the Milstein scheme, which was derived by Milshtein (1976). This scheme adds a correction term to the Euler scheme. To derive the Milstein scheme with a general setting X, we begin with the two integral terms of the Euler scheme, given by

$$\int_t^{t+h} a(X_u)du \approx a(X_t)h, \qquad \text{and}$$

$$\int_t^{t+h} b(X_u)dW_u \approx b(X_t)(W_{t+h} - W_t).$$

The idea is to improve the accuracy of the discretisation by considering expansions of order $O(h)$ to the last term of the Euler scheme. This can be achieved by applying Itô's formula to that term. The Itô 's formula yields

$$\begin{aligned} db(X_t) &= b^{'}(X_t)dX_t + \frac{1}{2}b^{''}(X_t)b^2(X_t)dt \\ &= b^{'}(X_t)[a(X_t)dt + b(X_t)dW_t] + \frac{1}{2}b^{''}(X_t)b^2(X_t)dt \\ &= [b^{'}(X_t)a(X_t) + \frac{1}{2}b^{''}(X_t)b^2(X_t)]dt + b^{'}(X_t)b(X_t)dW_t \end{aligned}$$

Applying the Euler approximation to $b(X_t)$ in approximation of $b(X_u)$ where $t < u < t + h$ by

$$b(X_u) \approx b(X_t) + [b^{'}(X_t)a(X_t) + \frac{1}{2}b^{''}(X_t)b^2(X_t)](u-t) + [b^{'}(X_t)b(X_t)][W_u - W_t].$$

The order for $W_u - W_t$ is $O(\sqrt{u-t})$ whereas $(u-t)$ has a higher order. We remove the higher order term, we get a simpler approximation

$$b(X_u) = b(X_t) + [b^{'}(X_t)b(X_t)][W_u - W_t] \tag{4.4}$$

Integrating Equation (4.4) with respect to W_u, we get

$$\begin{aligned} \int_t^{t+h} b(X_u)dW_u &\approx \int_t^{t+h} (b(X_t) + [b^{'}(X_t)b(X_t)][W_u - W_t])dW_u \\ &\approx b(X_t)[W_{t+h} - W_t] + b^{'}(X_t)b(X_t)\int_t^{t+h}(W_u - W_t)dW_u \end{aligned}$$

Simplifying the remaining integral, we get

$$\int_t^{t+h}(W_u - W_t)dW_u = \int_t^{t+h}(W_u)dW_u - \int_t^{t+h}(W_t)dW_u \tag{4.5}$$

Let $Y_t = \int_0^t W_t dW_t$ with $Y_0 = 0$, so that $dY_t = W_t dW_t$. To obtain a solution to this SDE, we apply Ito's formula and get

$$Y_t = \frac{1}{2}W_t^2 - \frac{1}{2}t.$$

Applying this back to Equation (4.5) yields

$$\begin{aligned}
\int_t^{t+h}(W_u)dW_u - \int_t^{t+h}(W_t)dW_u &= Y_{t+h} - Y_t - W_t(W_{t+h} - W_t) \\
&= \frac{1}{2}W_{t+h}^2 - \frac{1}{2}(t+h) - \frac{1}{2}W_t^2 + \frac{1}{2}t - W_t(W_{t+h} - W_t) \\
&= \frac{1}{2}W_{t+h}^2 - W_t W_{t+h} + \frac{1}{2}W_t^2 - \frac{1}{2}h \\
&= \frac{1}{2}[W_{t+h} - W_t]^2 - \frac{1}{2}h
\end{aligned}$$

Now we substitute it back into the equation,

$$\int_t^{t+h} b(X_u)dW_u \approx b(X_t)[W_{t+h} - W_t] + b'(X_t)b(X_t)\frac{1}{2}[W_{t+h} - W_t]^2 - \frac{1}{2}h \tag{4.6}$$

Finally, substituting Equation (4.6) into Equation (4.3) yields the Milstein approximation

$$X_{t+h} \approx X_t + a(X_t)h + b(X_t)(W_{t+h} - W_t) + \frac{1}{2}b'(X_t)b(X_t)[(W_{t+h} - W_t)^2 - h],$$

where $W_{t+h} - W_t = \Delta W = \sqrt{h}Z$ with $Z \sim \mathcal{N}(0,1)$. Again, for a Monte Carlo path simulation that follows a geometric Brownian motion for generating a stock price, S, set $X_t = S_t$, $h = \delta t$, $a(X_t) = rS$ and $b(X_t) = \sigma S$. Hence,

$$S_{t+\delta t} \approx S_t + rS\delta t + \sigma S\sqrt{\delta t}Z + \frac{1}{2}\sigma^2 S\left[(\sqrt{\delta t}Z)^2 - \delta t\right].$$

Algorithm 4.3: A standard Milstein Scheme to obtain the payoff of the option.

```
function MilsteinScheme {
    T := 1, S:= 250, K := 200, r := 0.05, sigma := 0.2;
    m := 100, n := 100, sum:=0, timestep := T/m;

    for i:=1 to n {
        for j:=1 to m{
            S = S+r*S*timestep + sigma*S*sqrt(timestep)*Z
                  +0.5*sigma*sigma*S*((sqrt(timestep)*Z)^2-timestep);
        }
        sum := sum + max(S - K, 0);
     }
     value = sum/n*exp(-r*T);
     return value;
}
```

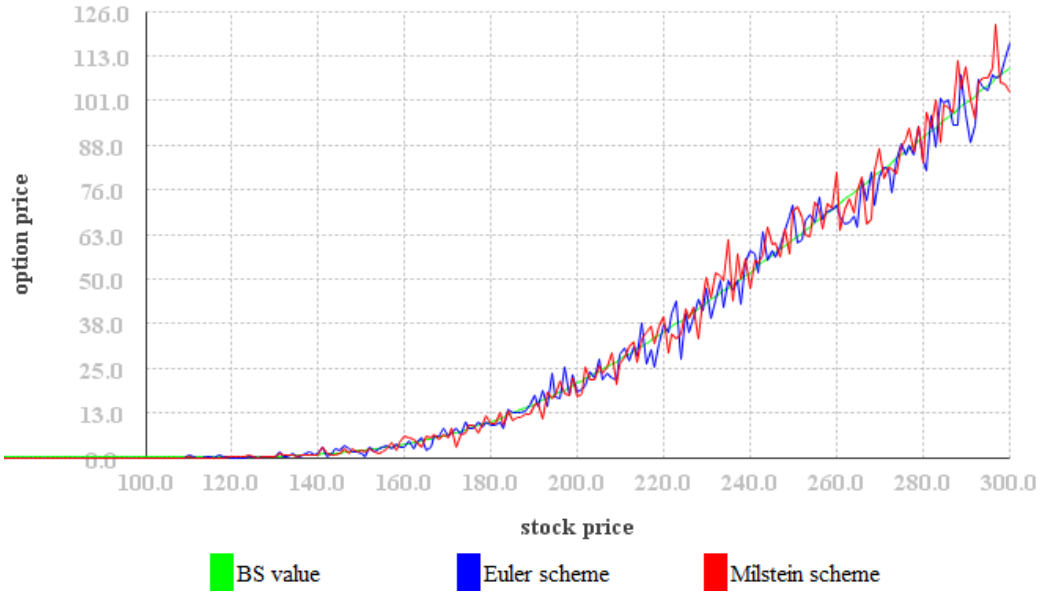


Figure 4.3: A comparison of the Milstein and Euler Scheme as a function of stock (S) for n=100.

Using the same iterative algorithm applied in the standard MC simulation, we generate iteration of $S_{t+\delta t}$ at $\delta t, 2\delta t, ...$ for the Milstein scheme (see Algorithm 4.3) as well as the Euler scheme (see Algorithm 4.2) for comparison later on. The algorithms return discretised approximations of the option value for the SDE. In contrast, a standard Monte Carlo simulation gives an exact solution of the stochastic process, S. To compare the results of the Euler and Milstein schemes, a simulation of 100 runs is generated, as shown in Figure 4.3. An improvement of the Milstein scheme to the Euler scheme may not be obvious from this graph because the accuracy depends on how the error is measured. Hence the errors obtained from discretising the SDE need to be calculated.

4.3.3 Convergence Order

Two types of errors associated with dicretisation methods are considered: the sampling error and the bias obtained from discretising the continuous process. For the sampling error, consider a stochastic process $S_{t+\delta t}$ of a standard MC simulation. $S_{t+\delta t}$ has a solution of the form

$$S_{t+\delta t} = S_t \exp\left[\left(r - \frac{\sigma^2}{2}\right)\delta t + \sigma\sqrt{\delta t}Z\right].$$

We have earlier defined the convergence rate for the MC simulation, which is $1/\sqrt{n}$. This applies to its discretised method as well. So increasing the number of simulation paths n decreases the sampling error at this convergence rate. However, due to discretisation, another error called the bias emerges. Assume that $\hat{S}$ is the estimator of S. We define the bias as

$$\text{bias}(\hat{S}) \approx \mathsf{E}(\hat{S}) - S \neq 0.$$

The expected mean square error (MSE) is the sum of the sampling error $\text{Var}(\hat{S})$ and the square of the bias, that is

$$\begin{aligned} \text{MSE} &\approx \text{Var}(\hat{S}) + (\text{bias}(\hat{S}))^2 \\ &= c_1 \frac{1}{n} + c_2(\delta t)^2, \end{aligned}$$

where c_1, c_2 are positive constants and $\delta t = \frac{T}{m}$ for m-steps. To calculate the bias, take the mean absolute error of $S_{t+\delta t} - S^{\delta t}_{t+\delta t}$ for each simulated path, where $S^{\delta t}_{t+\delta t}$ is the solution obtained from the Euler scheme with a step size of δt (see Algorithm 4.4). A discrete measure of the absolute error (Seydel, 2005) is

$$\tilde{\epsilon}(\delta t) = \frac{1}{n} \sum_{i=1}^{n} |S_{t+\delta t,i} - S^{\delta t}_{t+\delta t,i}|.$$

The convergence order can be defined for strong and weak convergence. The scheme is said to be strongly convergent with order $\gamma > 0$ if

$$\mathbb{E}(|S_{t+\delta t} - S^{\delta t}_{t+\delta t}|) = \text{O}((\delta t)^{\gamma}), \tag{4.7}$$

and weakly convergent with respect to polynomials g with order $\gamma > 0$ if

$$|\mathbb{E}(g(S_{t+\delta t})) - \mathbb{E}(g(S^{\delta t}_{t+\delta t}))| = \text{O}((\delta t)^{\gamma}). \tag{4.8}$$

Algorithm 4.4: An algorithm to obtain the mean absolute error of the Euler scheme.

```
1  function EulerSchemeError{
2      sum := 0, e := 0;
3
4      for i:=1 to n {
5          for j:=1 to m {
6              S := S+r*S*timestep + sigma*S*sqrt(timestep)*Z;
7
8              %   std_S is the exact solution generated by
9              %   a standard MC simulation
10             std_S := std_S*exp((r-0.5*sigma*sigma)*timestep
11                     + sigma*sqrt(timestep)*Z);
12         }
13         e := e + abs(std_S-S);
14     }
15     error := e/n;
16     return error;
17 }
```

The Euler scheme has a weak convergence of order 1 and strong convergence of order 0.5. The Milstein scheme, on the other hand, has the same weak convergence but with an improved

strong convergence of order 1 (Palczewski, 2009). This implies that, for the strong convergence, the error for Euler scheme is

$$\epsilon \approx O\left(\frac{T}{m}\right)^{0.5} = O\left(\frac{1}{m}\right)^{0.5}. \tag{4.9}$$

Error	$m = 100$	$m = 1000$	$m = 10000$
Seed 1	0.597312767636658	0.178602797821815	0.0614883837645385
Seed 2	0.713334963844081	0.170026343577513	0.0571792534279899
Seed 3	0.535792622477826	0.192392546740131	0.0599758175936207
execution time (s)	0.026315263	0.123001487	0.875772804
Error	$m = 100000$	$m = 1000000$	$m = 10000000$
Seed 1	0.01670048913320240	0.00568751863932249	0.0020911765693703
Seed 2	0.01817524350137770	0.00562403830202470	0.0018639849044890 8
Seed 3	0.02066917988648350	0.00567992441663136	0.0018572184786046 3
execution time (s)	8.355924078	84.05672546	836.446252838

Table 4.3: Table of mean absolute error of the Euler scheme

m	$(1/m)^{0.5}$	mean abs error	ratio
100	0.10000	0.59731276763665800	5.97312767636658
1000	0.03162	0.17860279782181500	5.64791637595495
10000	0.01000	0.06148838376453850	6.14883837645385
100000	0.00316	0.01670048913320240	5.28115836998107
1000000	0.00100	0.00568751863932249	5.68751863932249
10000000	0.00032	0.00209911765693703	6.63799287259696

Table 4.4: Table of the ratio of mean absolute error and ϵ for the Euler scheme

The convergence is tested for different m-steps on three seeds and the result can be seen in Table 4.3. As expected, increasing m reduces the error. The result obtained is checked against ϵ from the strong order convergence of Equation (4.9). The result from Table 4.3 should be consistent with the ϵ value and to verify, the ratio between the mean absolute error and ϵ is found (see Table 4.4). The ratio shows consistency in the mean absolute errors obtained from Algorithm 4.4 with the equation of the strong convergence. In a similar fashion, for the Milstein scheme, the additional correction term is added to Algorithm 4.4 to produce the mean absolute errors and from Equation (4.7), we have

$$\epsilon \approx O\left(\frac{T}{m}\right)^{1} = O\left(\frac{1}{m}\right)^{1}. \tag{4.10}$$

Error	$m = 100$	$m = 1000$	$m = 10000$
Seed 1	0.02272292010973670	0.00221953267967961	0.00020831336164974
Seed 2	0.01986182858192390	0.00214778291577914	0.00023217745088317
Seed 3	0.02138403075735730	0.00203093795191222	0.00018211846211557
execution time (s)	0.026315263	0.123001487	0.875772804
Error	$m = 100000$	$m = 1000000$	$m = 10000000$
Seed 1	0.00001980099067197	0.00000244459023406	0.00000024554289610
Seed 2	0.00002353406093249	0.00000231741006814	0.00000021906750192
Seed 3	0.00002433720133524	0.00000198317872702	0.00000020873203255
execution time (s)	8.355924078	84.05672546	836.446252838

Table 4.5: Table of mean absolute error of the Milstein scheme

m	$(1/m)^1$	mean abs error	ratio
100	0.0100000	0.02272292010973670	2.27229201097367
1000	0.0010000	0.00221953267967961	2.21953267967961
10000	0.0001000	0.00020831336164974	2.08313361649743
100000	0.0000100	0.00001980099067197	1.98009906719676
1000000	0.0000010	0.00000244459023406	2.44459023406307
10000000	0.0000001	0.00000024554289610	2.45542896095685

Table 4.6: Table of the ratio of mean absolute error and ϵ for the Milstein scheme

We test the convergence with the same three seeds for different m (see Table 4.5) and then find their corresponding ratios (see Table 4.6). The result also shows consistency where the ratio across different m remains relatively the same. This scheme converges at a faster rate than the Euler scheme, therefore the additional correction term produces a higher order of convergence $O(h)$ which in essence is an improvement to the Euler scheme.

4.3.4 Runge-Kutta Methods

One problem of applying Taylor series methods is the use of the derivative b' as seen in the Milstein scheme. In this section, we will discuss in theory an alternative method that eliminate the use of that derivative. The second order Runge-Kutta (RK2) method, considered as a refinement of the Euler scheme, takes a trial step at the midpoint of a time-step interval. We will discuss this method in a general setting. Consider an initial value problem of

$$y' = f(t, y), \quad y(t_0) = y_0.$$

This technique begins by first finding the initial derivative of y at the initial step, set to k_1. Then, using k_1, we find the midpoint between this step and the next step, and calculate the derivative of y at this point. Finally, we use this value to estimate y at the next step, $y_{n+1} = y_n + k_2$.

Therefore, the RK2 method produces the following equations:

$$\begin{aligned} k_1 &= hf(x_n, y_n), \\ k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1), \\ y_{n+1} &= y_n + k_2 + O(h^3). \end{aligned}$$

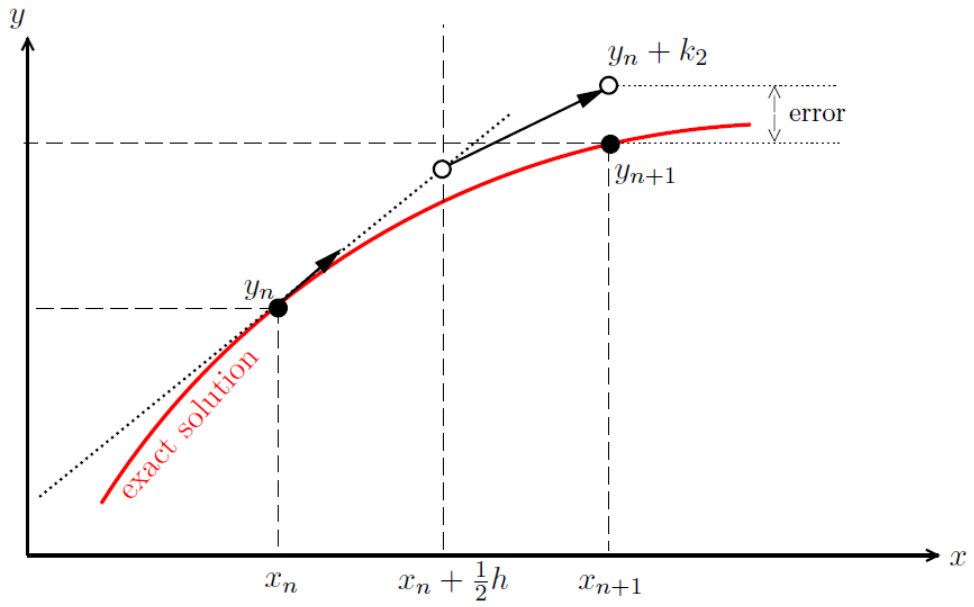


Figure 4.4: An illustration of the 2nd-order Runge Kutta method.

The advantage of using this method is the error associated with estimating y_{n+1} decreases due to a more accurate slope (see Figure 4.4). In addition, the first-order error is eliminated since this method is of second order. Applying this method to the factor b' of the Milstein scheme, Schaffter (2010) found that the RK2 method for the stochastic process X is

$$X_{t+h} \approx X_t + ah + b\Delta W_t + \frac{1}{2\sqrt{h}}[b(\bar{X}_t) - b(X_t)][(\Delta W_t)^2 - h].$$

where $\bar{X}_t = X_t + ah + b\sqrt{h}$ is an approximate solution to X. In fact, $\bar{X}_t$ is the Euler scheme. This method has the same strong order of convergence as the Milstein scheme, which is an order of 1. Implementation of the RK2 method was not carried out due to the limited time available. This method would reduce the computational cost because only evaluates a and b and not their derivatives.

Chapter 5

Improving the Monte Carlo Simulation

In this chapter, we consider a method to improve the efficiency of the MC simulations, which is the multilevel method introduced by Giles (2008) and applied to discretised versions of the SDEs. We first introduce the multilevel method to the MC simulation using the Euler scheme and test the complexity and evaluate the root-mean-square (RMS) error. Then, we apply the Milstein scheme to the multilevel method to get a better strong convergence.

5.1 Multilevel Monte Carlo Simulation

The multilevel MC approach is a result of the work by Giles (2008), which is based on the multigrid idea for finding iterative solutions of PDEs. The objective is to reduce the computational cost of estimating the payoff value obtained using MC simulations from $O(\epsilon^{-3})$ to $O\left(\epsilon^{-2}(\log\epsilon)^2\right)$ for the Euler discretisation with an RMS accuracy ϵ.

In a standard MC simulation, the timestep is fixed at $h = TM^{-1}$, where M is the refinement factor. For the multilevel MC simulation, consider MC simulations with different timesteps $h_l = TM^{-l}$ for different levels of refinements, $l = 0, ..., L$, with L being the finest refinement. Let P be the payoff and $\hat{P}_l$ the approximation of the payoff on level l. According to Giles (2008), the expectation for $\hat{P}_L$ is

$$E[\hat{P}_L] = E[\hat{P}_0] + \sum_{l=1}^{L} E[\hat{P}_l - \hat{P}_{l-1}]. \tag{5.1}$$

Algorithm 5.1: An algorithm for a multilevel Monte Carlo simulation.

```
function mlmc (M, eps, mlmc_l) {
    %   L = the finest level
    %   Vl = the variance at level l
    %   Pf = the payoff at the finer level(l)
    %   Pc = the payoff at the coarser level(l-1)
    %   N = initial number of simulation paths
    %   Nl = optiomal number of simulation paths
    %   mlmc_l = a function to calculate Pf and Pc at l,
    %            returns 4 sets of sums
    L := -1, converged := 0;

    while ~conver {
        %   Starts from L=0 and iterate while convergence is false.
        L := L+1;

        sums := mlmc_l (M,L,N);
        N[L+1] := 10^4;
        sum1[L+1] := sums[0];
        sum2[L+1] := sums[1];

        %   Estimate Vl and find optimal Nl.
        for i := 1 to L+1 {
            Vl := sum2[L+1]/N[L+1]- sum1[L+1]/N[L+1])^2;
            Nl := ceil(2*sqrt(Vl/M^L)*sum(sqrt(Vl*M^L)/eps^2)); }

        %   Compare Nl with the number of samples used in the current level l.
        %   If Nl is larger, then the additional number of samples dNl
        %   is calculated.
        for l := 1 to L {
            dNl := Nl-N[l+1];
            sums := mlmc_l (M,L,dNl);
            N[l+1] := N[l+1] + dNl;
            sum1[l+1] := sums[0];
            sum2[l+1] := sums[1]; }

        %   Test for convergence if L>=2.
        if L > 1 && M^L >= 16 {
            con1 := M^(-1)*suml1[L]/N[L];
            con2 := suml1[L+1]/N[L+1];
            converged := max(con1, con2);
            if (converged < (M-1)*eps/sqrt(2) || M^L>1024) {
                conver := true;
            } else conver := false;
        }
        P := P + sum1[L+1] / N[L+1];
     }
     return [P, Nl];     }
```

In other words, the expectation of the finest level is equal to the expectation of the coarsest level with an additional sum of the difference between expectations with different levels. The expectation $E[\hat{P}_l - \hat{P}_{l-1}]$ is a result obtained from two discrete approximations that follow the same GBM path. Now we want to estimate the payoff value in a way that will minimise the variance for a given computational cost. Let $\hat{Y}_l$ be the estimator of $E[\hat{P}_l - \hat{P}_{l-1}]$ for $l > 0$ using N_l samples. The simplest estimator is to take the mean of these independent samples, which is

$$\hat{Y}_l = N_l^{-1} \sum_{i=1}^{N_l} \left(\hat{P}_l^{(i)} - \hat{P}_{l-1}^{(i)} \right).$$

Once we obtain the estimator, the variance is calculated. This estimator has a variance $V[\hat{Y}_l] = N_l^{-1} V_l$ where V_l is the variance of a single sample. The combined variance for the combined estimator $\hat{Y} = \sum_{l=0}^{L} \hat{Y}_l$ is therefore

$$V[\hat{Y}] = \sum_{l=0}^{L} N_l^{-1} V_l. \tag{5.2}$$

Next, the value of the estimator is tested for convergence. The criterion for convergence is

$$\max \left\{ M^{-1} \left| \hat{Y}_{L-1} \right|, \left| \hat{Y}_L \right| \right\} < \frac{1}{\sqrt{2}} (M-1)\epsilon. \tag{5.3}$$

To minimise the variance, we must choose an optimal number of samples N_l to be proportional to $\sqrt{V_l h_l}$. Hence, the equation to determine optimal N_l (Giles, 2008) is given by

$$N_l = \left\lceil 2\epsilon^{-2} \sqrt{V_l h_l} \left(\sum_{l=0}^{L} \sqrt{\frac{V_l}{h_l}} \right) \right\rceil. \tag{5.4}$$

This optimal N_l takes into account the effect of the computational cost across all levels. The accuracy is chosen in such a way to ensure that the variance of the combined estimator in Equation (5.2) is less than $\frac{1}{2}\epsilon^2$. Equation (5.3) makes sure that the bias is less than $\frac{1}{\sqrt{2}}\epsilon$ so that the combined error should be less than ϵ^2. The numerical algorithm for the multilevel MC method is summarised in Algorithm 5.1. The algorithm begins by initially setting $L = 0$ and $N = 10^4$. Then we estimate Vl and find the optimal N_l using Equation (5.4). This optimal N_l is compared to the number of sample paths at that current level and if N_l is larger, then the extra samples are calculated. Then, we test for convergence. If $L < 2$, then go to the next level $L = L + 1$ and repeat the steps. If $L \geq 2$, then we take the convergence test which has the condition in Equation (5.3). If the condition for convergence is not met, go to the next level and repeat the steps. Finally, when L converges, we calculate the estimated payoff $\hat{P}$. This is generally how the multilevel MC method works and it is applicable to different types of option styles.

5.1.1 Computational Complexity

In order to increase efficiency in estimating the expected option value, a certain level of computational savings needs to be achieved. The computational complexity measure is governed by the complexity theorem, as stated by Giles (2008) in a general setting so that it can be applied to different option styles including path-dependent styles. The theorem gives bounds to the mean-square-error (MSE) and the computational complexity C of the multilevel estimator $\hat{Y}$. This theorem states that there exists a positive constant c_1 such that for any $\epsilon < e^{-1}$, we have the bounds

$$MSE < \epsilon^2 \qquad \text{and} \qquad C \leq c_1 \epsilon^{-2} (log\epsilon)^2. \tag{5.5}$$

The computational cost is proportional to $\sum_{l=0}^{L} N_l h_l^{-1}$. Therefore, the cost is calculated as the total number of timesteps across all the levels, with each level performing the coarser and finer timesteps. Thus, we can write the computational cost as

$$C = N_0 + \sum_{l=1}^{L} N_l (M^l + M^{l-1}). \tag{5.6}$$

For the standard MC simulation, the computational cost is defined as

$$C^* = \sum_{l=1}^{L} N_l^* M_l, \tag{5.7}$$

where $N_l^* = 2\epsilon^{-2} V[P_l]$. This is chosen so that the variance of the estimator is $\frac{1}{2}\epsilon^2$ similar to that of the multilevel MC simulation. To calculate the accuracy, we find the RMS error. The error is the difference between the estimated payoff obtained from the multilevel algorithm and an exact payoff value obtained from the BS formula. This RMS error is then compared to the desired accuracy ϵ.

5.1.2 Application to the European Option

Figure 5.1 shows the numerical results for parameters $S(0) = 1$, $K = 1$, $T = 1$, $\sigma = 0.2$ and $r = 0.05$ for a geometric Brownian motion with European option, which has a payoff value of

$$P = e^{-rT} \max(S - K, 0).$$

The top left shows the behaviour of the logarithm base M variance of $\hat{P}_l$ and $\hat{P}_l - \hat{P}_{l+1}$ at different grid levels. The slope $\log|\hat{P}_l - \hat{P}_{l+1}|$ has a gradient of -1, which indicates that the variance is proportional to M^{-1}, therefore $V_l = O(h_l)$. The top right is a plot of the log of mean at different levels. The mean $E[\hat{P}_l - \hat{P}_{l+1}]$ has a slope of -1, which again indicates a convergence of $O(h_l)$.

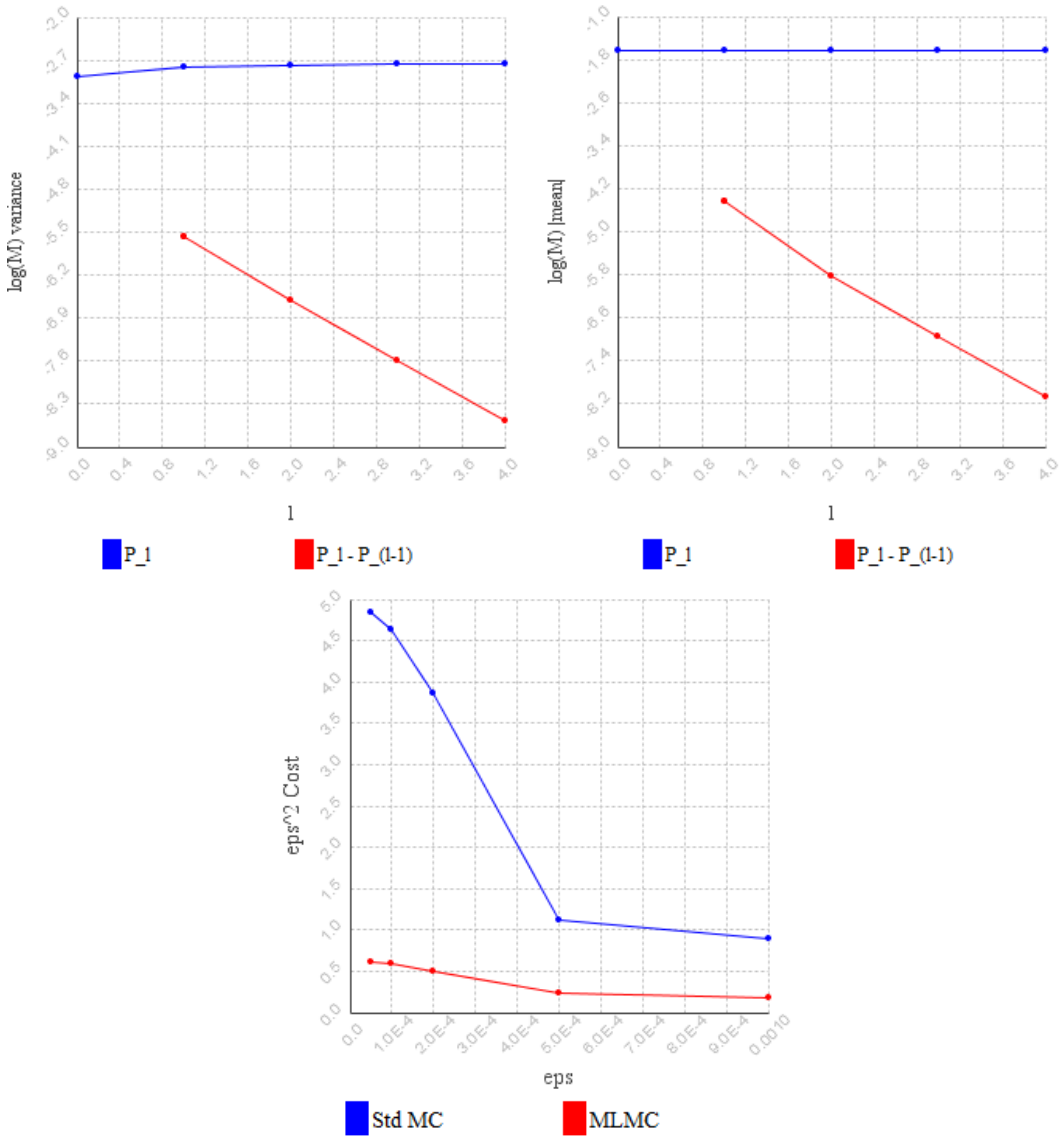


Figure 5.1: Geometric Brownian Motion with European Option (option value ≈ 0.104632)

$\epsilon^2 C$	$(log\epsilon)^2$	ratio
0.181524	47.71708299430558	0.003804172187592912
0.2264535	57.773718199472874	0.003919662903089145
0.50385836	72.54257985990712	0.006945691219874478
0.59348669	84.83036976765435	0.006996158234669104
0.61589781	98.07906570323802	0.006279605189791959

Table 5.1: Table showing the ratio $\epsilon^2 C/(log\epsilon)^2$.

The plot at the bottom is a result of obtaining five sets of multilevel calculations for different user-specified accuracies, ϵ. By observation, we can see a significant decrease in the computational cost for the multilevel MC method as compared to the standard MC method.

The computational cost is calculated as $\epsilon^2 C$ because from the complexity theorem, C is proportional to $\epsilon^{-2}(\log\epsilon)^2$, which means that $\epsilon^2 C$ should be proportional to $(\log\epsilon)^2$. We check this by calculating the ratio $\epsilon^2 C/(log\epsilon)^2$ for the multilevel method as seen in Table 5.1. The table shows consistency in the ratio. The reason for the slight difference in ratio between the first two values and the last three is due to the different number of L calculated at each ϵ. For ϵ values of 0.001 and 0.0005, $L = 2$, while for $\epsilon = 0.0002, 0.0001, 0.00005$, $L = 3$. From the the plot in Figure 5.1, it can be observed that there is a slow decrease in $\epsilon^2 C$ as the accuracy increases for the multilevel method. On the other hand, for the standard MC method, $\epsilon^2 C$ is approximately proportional to ϵ^{-1}. The significant decrease from the third ϵ value to the fourth is again due to the fact that $L = 3$ for the first three accuracies and $L = 2$ for the last two accuracies as seen in the plot.

eps	std cost	mlmc cost	savings
0.001	901779.6278575546	181524.0	4.967825895515494
0.0005	4508898.1392877735	905814.0	4.9777306812301125
0.0002	9.640097101142982E7	1.2596459E7	7.653021457175371
0.0001	4.63969262499998E8	5.9348669E7	7.817686063018498
0.00005	1.9342424284542708E9	2.46359124E8	7.851312332374874

Table 5.2: Table of standard MC costs, multilevel MC (mlmc) costs and their subsequent savings.

Table 5.2 shows the computational savings achieved from calculating the computational costs for the standard MC simulation and the multilevel MC simulation (from Equations (5.7) and (5.6) respectively), that is, savings = standard cost / multilevel cost. As the accuracy increases, the computational savings increases. At the highest accuracy, $\epsilon = 0.00005$, the multilevel MC method is roughly 8 times more efficient than the standard MC method.

5.1.3 Application to the Asian Option

The Asian option is an option style where the payoff is determined by the average asset price over a fixed period of time, unlike the European option where the payoff is determined by the asset price at the expiration date. Therefore, the Asian option has a payoff of

$$P = e^{-rT}\max(\bar{S} - K, 0),$$

where $\bar{S} = T^{-1}\int_0^T S(t)dt$. The simplest approximation of $\bar{S}$ is given by Giles (2008) as

$$\bar{S} \approx T^{-1}\sum_{n=1}^{n_T}\frac{1}{2}h(\hat{S}_n + \hat{S}_{n+1}),$$

where $n_T = T/h$ is the number of timesteps. This approximation takes the average value of the coarser and finer values of the approximated prices found with the Euler discretisation. In the numerical calculation, the option prices for the coarser and finer grids are found separately and

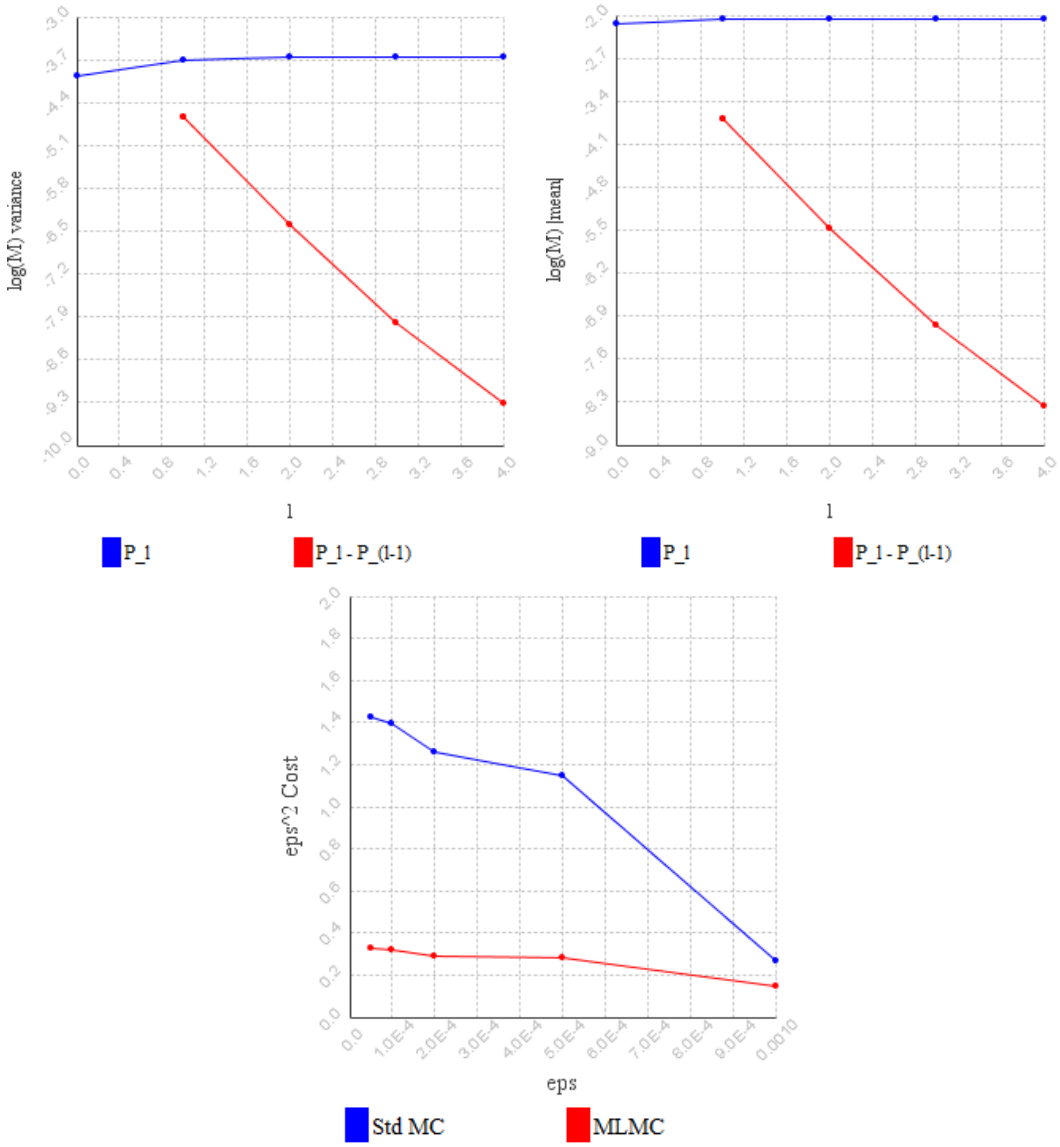


Figure 5.2: Geometric Brownian Motion with Asian Option (option value ≈ 0.0576613)

then the final payoff is found by subtracting the payoff at the finer grid and the payoff at the coarser grid discounted at e^{-rT}.

$\epsilon^2 C$	$(log\epsilon)^2$	ratio
0.14595	47.71708299430558	0.0030586530198716726
0.282945	57.773718199472874	0.004897469105642253
0.2915706	72.54257985990712	0.004019302877883248
0.3193017	84.83036976765435	0.003764002218480829
0.32615585	98.07906570323802	0.0033254379786494255

Table 5.3: Table showing the ratio $\epsilon^2 C/(log\epsilon)^2$ for the Asian option.

Figure 5.2 presents results from the multilevel MC simulation for the Asian option with

Euler discretisation. The behaviours of the variance and the mean $E[\hat{P}_l - \hat{P}_{l+1}]$ are similar to those of the European option. The slopes for the two plots at the top for $\log|\hat{P}_l - \hat{P}_{l+1}|$ are both equal to -1, hence the convergence is $O(h_l)$ in both cases. The bottom plot shows the result of $\epsilon^2 C$ against ϵ. Similar to the European option, $\epsilon^2 C$ should be proportional to $(\log\epsilon)^2$. We demonstrate this in Table 5.3.

eps	std cost	mlmc cost	savings
0.001	264281.929053998	145950.0	1.8107703258238987
0.0005	4576876.039467748	1131780.0	4.043962642446189
0.0002	3.1530589229553685E7	7289265.0	4.325619829921629
0.0001	1.3934544198989743E8	3.193017E7	4.364068277428445
0.00005	5.706048530312724E8	1.3046234E8	4.373713157615235

Table 5.4: Table of standard MC costs, mlmc costs and their savings.

Similarly, the savings for the Asian option are also calculated as standard cost/multilevel cost. The results indicate that there is a significant amount of savings as ϵ decreases (see Table 5.4).

5.2 Multilevel MC Simulation using Milstein Scheme

In the multilevel MC simulation, Giles (2008) has shown that the computational cost can be reduced from $O(\epsilon^{-3}$ to $O(\epsilon^{-2}(log\epsilon)^2)$. In his next paper, he showed that the Milstein scheme can be used to further improve the computational cost to $O(\epsilon^{-2})$, which we will examine in this section. However, the paper addresses the more complex exotic options such as Asian, lookback and barrier options while this section only investigates the European option because of time constraint. Recall that the approximation for X using the Milstein scheme is

$$X_{t+h} \approx X_t + a(X_t)h + b(X_t)(W_{t+h} - W_t) + \frac{1}{2}b'(X_t)b(X_t)[(W_{t+h} - W_t)^2 - h].$$

We apply this scheme with the correction term to the multilevel MC simulation implemented earlier, therefore we apply the same algorithm for this method (see Algorithm **??**). The refinement factor is chosen as $M = 2$, meaning that the next level is has twice the number of timesteps as the current level.

The complexity theorem from Giles (2007) defines a bound for the computational complexity C of a multilevel method using a Milstein scheme such that for any $\epsilon < e^{-1}$, there exists a positive constant c_2 such that

$$C \leq c_2\epsilon^{-2}.$$

5.2.1 Application to the European Option

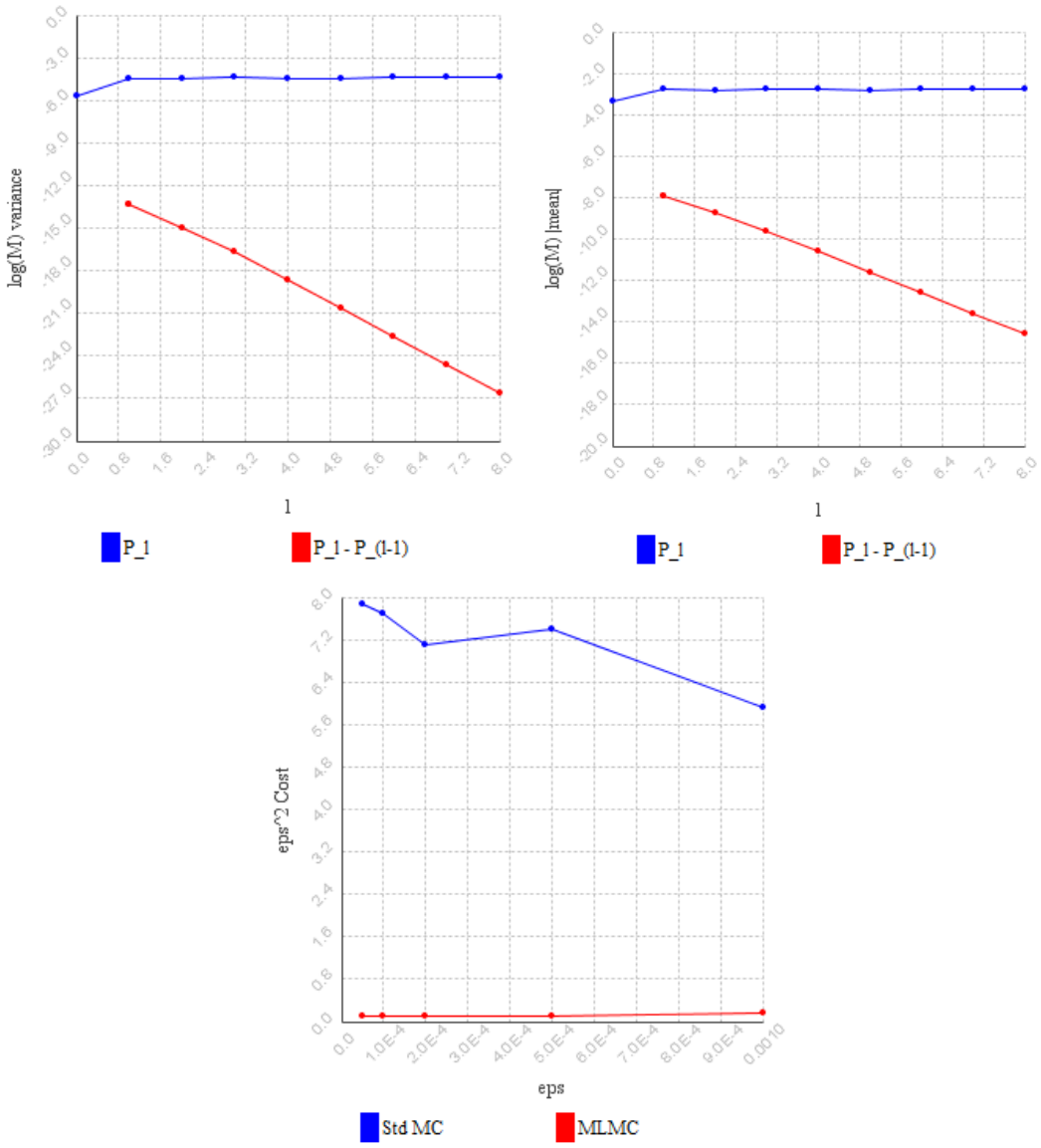


Figure 5.3: Geometric Brownian Motion with European Option (option value ≈ 0.145465)

Similar to the previous multilevel method, the test case supplied has the parameters $S = 1$, $K = 1$, $r = 0.05$, $\sigma = 0.2$ and $T = 1$. Figure 5.3 shows the plots of the results from the multilevel MC simulation using the Milstein scheme. The top left is a plot of log base M variance versus the level l. The slope for $\log|\hat{P}_l - \hat{P}_{l-1}|$ is observed to be approximately -2, which indicates that $V_l = O(h_l^2)$. The top right shows a plot of the absolute mean in log base M against l. At $l = 8$ and $l = 4$, we observe that the slope for $\log|\hat{P}_l - \hat{P}_{l-1}|$ is approximately -1, indicating that the mean $E[\hat{P}_l - \hat{P}_{l+1}]$ converges at $O(h_l)$.

The bottom plot shows results from five sets of accuracies, ϵ. As expected, the computational cost for the multilevel method is much lower. For example, for the finest accuracy, $\epsilon = 0.00005$,

the computational cost for the multilevel method is reduced to almost 8 times compare to the standard method. From the complexity theorem stated earlier, the computational cost for the Milstein scheme is $O(\epsilon^{-2})$, therefore $\epsilon^2 C$ should be proportional to 1. Table 5.5 shows that the results of $\epsilon^2 C$ for different accuracies are approximately equal, indicating that the convergence order of C is ϵ^{-2}.

ϵ	$\epsilon^2 C$
0.001	0.15372
0.0005	0.10935225
0.0002	0.0885906
0.0001	0.09317259
0.00005	0.0943719525

Table 5.5: Table showing the values of $\epsilon^2 C$ for the European option.

Also, if we compare this plot with that of the Euler scheme in Figure 5.1, we observe that for the most accurate case, $\epsilon = 0.00005$, the computational cost for the multilevel method using the Milstein scheme is reduced by almost 7 times. Table 5.6 shows that there is a significant amount of computational savings attained for the multilevel MC simulation.

eps	std cost	mlmc cost	savings
0.001	5926783.0	153720.0	38.555705178246164
0.0005	2.9633798E7	437409.0	67.74848711389112
0.0002	1.77802467E8	2214765.0	80.28051147638689
0.0001	7.70477004E8	9317259.0	82.69352649743878
0.00005	5.3.141175009E9	3.7748781E7	83.21262106450537

Table 5.6: Table of standard MC costs, mlmc costs and their savings.

The numerical results demonstrated here is only for the most basic option style. Application to other option styles has not been implemented. More option styles should be applied in order to investigate the extent to which the efficiency of the Milstein scheme can be achieved.

Chapter 6

Evaluation

In this chapter, two types of evaluation will be discussed. They are: the evaluation of the models and the evaluation of the implementation. The first part gives a brief summary of all the evaluations of the models explained in earlier chapters. The next part explores the more technical side of the evaluation, giving insights on the choice of programming language used and then discussing the implementation of the algorithms.

6.1 Evaluation of Models

Both the solutions obtained from the BS model and the binomial OP model can be easily verified by comparing them against option calculators available on the internet. The MC simulation is much harder to compare since the model generates random values. For the BS model, we can take, for example, the following website with a BS calculator, which returns call and put values accurate to 4 decimal places: `http://www.money-zine.com/Calculators/Investment-Calculators/Black-Scholes-Calculator/`. Different test cases can simply be applied on our implementation of the BS model and that of the BS calculator to verify that the results are the same. The only problem is that this BS calculator is only accurate up to 4 decimal places, while our result is accurate up to 15 decimal places. For the binomial OP model, the following interactive website is used: `http://jpja.net/interactive/binomial.php`. This calculator prints the entire binomial tree, hence is it is possible to verify that all values in our binomial tree are correct. Again, the result from this calculator is accurate up to 3 decimal places. However, a 3-decimal place accuracy of a result for comparison should provide enough information to verify that our model returns the correct result.

Now, referring to the evaluation in earlier chapters, we verified that the binomial OP value converges to the BS value with an increased in the number of periods, as demonstrated in

Section 3.2.

For the MC simulation, we rely on the BS model, which was found to produce correct results, to verify that the MC value is correct. The result is seen in Section 4.2, which explores the effect of increasing the number of simulation paths. For the Milstein and Euler schemes, we verify their results by calculating their mean absolute errors relative to the MC simulation. We found that they indeed converge strongly at their orders 1 and 0.5 respectively (see Section 4.3.3).

The evaluation for the multilevel method involves verifying that efficiency is achieved by calculating the computational costs for different accuracies (see Sections 5.1.2, 5.1.3 and 5.2.1).

6.2 Evaluation of Implementation

All implementation of the models are executed on the same machine to eliminate any error in the variation of the performances of different machines. The test cases are also kept consistent so that comparisons of results can be made between the models. For the choice of programming language, we do not require a fast language since we want to compare the performance of different models through observations of their execution times. Therefore, although slower than C, Java is chosen as the programming language because it is substantial, robust and is platform-independent. It is also interoperable as extensions or bridges to other programming languages can be easily added. For producing the graphical results seen in this report, a Java library was added called the JMathPlot. This tool provides interactive 2D and 3D plots and is simple to import and use.

A round off issue may emerge especially in implementations where calculations are required in many runs. We rely on the precision of the double values used in the implementation, although roundoff errors cannot be completely eliminated.

The codes implemented in this project are not fully tested, each only having one test case since they are only used for analysing. Nevertheless, the results from the test case are verified.

Chapter 7

Conclusions

In this report, we first outline two standard options pricing models and evaluate them based on their accuracy and efficiency, and then apply the multilevel MC simulation as a method to improve the efficiency of the MC simulation. The result for the binomial options pricing model shows that the accuracy relative to the BS model can be achieved with a large number of periods, m. The convergence rate from this result is verified as $1/m$, which is a theoretical convergence rate presented by Chang and Palmer (2007).

The MC OP model is the next model to be investigated. We test the standard MC simulation by initially running a simulation of n paths for three seeds. This three seeds are taken to show how the result varies when they are run on different number of paths n. Due to its random nature, there is a sampling error associated with taking random variables to estimate the payoff, which we want to minimise. The result shows that increasing the number of paths will reduce the sampling error at a rate of $1/\sqrt{n}$. However, to achieve the level of accuracy of a binomial OP model takes a much greater computational effort for the MC simulation.

Since discretised methods are applied in the multilevel MC simulation, a study of these methods is necessary. The convergence order takes into account the bias due to discretisation. The result from the Euler scheme shows a strong convergence order of 1/2. To improve the strong convergence, a correction term is added to the Euler scheme. This refinement is the Milstein scheme and the result of this scheme has a strong convergence order of 1. The error is thus significantly reduced.

The results presented for the improved MC simulation show an increased in the efficiency of the model. The computational cost of estimating the payoff is observed to be $O(\epsilon^{-2}(\log\epsilon)^2)$ for the Euler scheme. An even better computational saving can be observed for the Milstein

scheme, with $O(\epsilon^{-2})$.

7.1 Future Work

Due to time constraint, several problems that have been originally planned were not carried out. These are thus set as the future work. We have investigated two discretisation methods for the MC simulation. Another method we could consider is the Runge-Kutta method that simplifies calculation of the asset price by replacing the derivative term of the Milstein scheme with a simpler term while still keeping the same convergence order.

Application of the multilevel MC simulation to the Bermudan option proved to be quite a challenge since this option style is path-dependent. In addition, the multilevel method adds to the complexity of pricing the American option so it is also wise to price the American option using the standard MC simulation. One method to price options of this style is to use the Longstaff-Schwartz's least square approach (Longstaff and Schwartz, 2001).

For the multilevel MC simulation, application to other exotic option styles such as barrier and look back options can be tested to observe the behaviour of this method in estimating payoffs of different option styles.

Here, we also present ideas on potential areas of options pricing for future work. We have seen two standard models that price options numerically. We may use another popular numerical method, finite-difference methods, to compare its performance with the other models. In addition, we could take a different direction and look at extensions to the models. For example, we can reduce the assumptions of the models and introduce much complex methods to price the options.

In conclusion, the models presented in this project are standard models that can be readily applied in the real world. The binomial options pricing model proves to converge faster to the BS model compare to the MC simulation, although it is much less flexible due to the assumption that there are only two possible price movements. The multilevel methods introduced to the MC simulation shows an increased efficiency, albeit not by much, but with promising results. The application of the Bermudan option would make for an interesting case for future work.

Glossary

arbitrage	The simultaneous buying and selling of securities from two different markets in order to take advantage of a price discrepancy.
call option	An option that grants the buyer the right, but not the obligation, to buy an underlying asset at a strike price on or before an expiration date.
dividend	A portion of a company's profit paid out to shareholders.
drift rate	The average rate of increase of a value in a stochastic process.
illiquidity	The condition in which an asset is difficult to buy or sell.
intrinsic value	An absolute value of the difference between the stock price and the strike price of an option.
option	A financial derivative whose price is derived from an underlying asset such as a stock.
portfolio	An investment that includes a collection of several assets in order to reduce risk.
premium	The amount per share that an option buyer pays to the seller in order to enter the option contract.
put option	An option that grants the seller the right, but not the obligation, to sell an underlying asset at a strike price on or before an expiration date.
risk-free bond	A theoretical bond that pays out interest and principal in a given period of time with certainty.
risk-free interest rate	A rate of return of an investment with zero risk.
risk-neutral measure	A probability measure for an underlying asset that is discounted at a risk-free rate.
stock price	The current price of a security that represents a claim on part of the company's assets and earnings.
strike price	A fixed price at which a derivative contract can be exercised.
transaction cost	A fee for buying or selling options or stocks.
vanilla options	Options that are not exotic.
volatility	A measure for the variation of price of an underlying asset over time.

Bibliography

Boyle, Phelim P. 1977. "Options: A Monte Carlo approach." *Journal of Financial Economics* 4(3):323–338.

Brandimarte, Paolo. 2002. *Numerical Methods in Finance: A MATLAB-Based Introduction.* Wiley Series in Probability and Statistics John WIley & Sons, Inc.

Broadie, Mark and Jerome B. Detemple. 2004. "Option Pricing: Valuation Models and Applications." *Management Science* 50(9):pp. 1145–1177.

Cetin, U., R. Jarrow, P. Protter and M. Warachka. 2004. "Pricing Options in an Extended Black Scholes Economy with Illiquidity: Theory and Empirical Evidence." *Review of Financial Studies* 19(2):493.

Chang, Lo-Bin and Ken Palmer. 2007. "Smooth convergence in the binomial model." *Finance and Stochastics* 11:91–105(15).

Cox, John C., Stephen Ross and Mark Rubenstein. 1979. "Option pricing: A simplified approach." *Journal of Financial Economics* 7(3):229–263.

Giles, Michael B. 2007. Improved multilevel Monte Carlo convergence using the Milstein scheme. In *Monte Carlo and Quasi-Monte Carlo Methods.* Springer pp. 343–358.

Giles, Michael B. 2008. "Multi-level Monte Carlo path simulation." *Operations Research* 56(3):607–617.

Glasserman, Paul. 2004. *Monte Carlo Methods in Financial Engineering.* Springer.

Hsia, Chi Cheng. 1983. "On Binomial Option Pricing." *The Journal of Financial Research* 6:41–46.

Jabbour, George M and Yi Kang Liu. 2005. "Option Pricing and Monte Carlo Simulations." *Journal of Business and Economics Research* 3(9).

Kyoung, Sook Moon and Joong Kim Hong. 2011. "An Improved Binomial Method using Cell Averages for Option Pricing." *IEMS* 10(2):170–177.

Lee, Cheng-Few and Carl Shu-Ming Lin. 2010. Two Alternative Binomial Option Pricing Model Approaches to Derive Black-Scholes Option Pricing Model. In *Handbook of Quantitative Finance and Risk Management.*

Leland, Hayne E. 1985. "Option Pricing and Replication with Transaction Costs." *The Journal of Finance* 40(5):1283–1301.

Longstaff, Francis A. and Eduardo S. Schwartz. 2001. "Valuing American Options by Simulation: A Simple Least-Squares Approach." *The Review of Financial Studies* 14(1):113–147.

Lovelock, David, Marilou Mendel and A Larry Wright. 2007. *An Introduction to the Mathematics of Money: Saving and Investing*. Springer.

MacBeth, James D. and Larry J. Merville. 1979. "An Empirical Examination of the Black-Scholes Call Option Pricing Model." *The Journal of Finance* 34(5):1173.

Milshtein, G. N. 1976. "A method of Second-Order Accuracy integration of Stochastic Differential Equations." *Theory of Probablility and its Applications* 23(2):396–401.

Palczewski, Jan. 2009. "Milstein Scheme and Convergence, Computations in Finance: MATH5350.".
URL: *http://www.maths.leeds.ac.uk/ jp/leeds/CIF/CIFLecture6.pdf*

Qu, Xianggui. 2010. "A Direct Justification of the Binomial Pricing Model as an Approach of the Black-Scholes Formula." *Pakistan Journal of Statistics* 26(1):187–193.

Schaffter, Thomas. 2010. "Numerical Integration of SDEs: A Short Tutorial." Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland, Unpublished manuscript.

Schwartz, Eduardo. 1977. "The Valuation of Warrants: Implementing a New Approach." *Journal of Financial Economics* 4(1):79–93.

Seydel, Rudiger. 2005. *Tools for Computational Finance*. Springer.

Ugur, Omur. 2008. *An Introduction to Computational Finance*. Imperial College Press.

Vasile, Emilia and Dan Armeanu. 2009. "Empirical Study on the Performances of Black-Scholes Model for Evaluating European Options." *Romanian Journal of Economic Forecasting* 10(1):48–62.

Wilmott, Paul, Sam Howinson and Jeff Dewynne. 1998. *The Mathematics of Financial Derivatives*. Cambridge University Press.

Appendix A

Personal Reflection

This project has been quite a challenge for me, especially in terms of giving a clear context of what it is I want to achieve from an early stage. The project started off well in early February when I began with the background research, albeit still not having a clear direction. By the time the interim report was due, I had successfully implemented two options pricing models. However, the problem was I only had a fairly vague aim and I needed an element that will push this project to the desired master's level. The exact term that my supervisor used was to "broaden and focus". Hence, there was a slight change in the project direction, although I still keep my earlier work which eventually became Chapter 3 and the first half of Chapter 4 of this report. The second half of the project period involves more intensive programming and evaluation as I started to work on the main model that reflects the project aim. It is therefore advisable to have a clear aim from day one because a change in the project direction especially midway through the project can be a risky affair.

Methodology-wise, the structure of this project is not the usual methodology that you see in many other projects. Basically, I work on one model at a time and for each of these models, I did some background research, implement the model, then test and evaluate it. The reason for this choice of methodology is due to the way this report is structured. It would be difficult to introduce all the models in one section and then have separate sections to discuss their implementations and evaluations because it can get very messy. Hence, I decided to keep it simple by working on one model at a time while still maintaining a good balance between the breadth and the depth of the project.

The supervisor's guidance throughout the course of the project has been extremely helpful and as a result, I've overcome many challenges. The weekly meetings were consistent and the advises given were invaluable. The opportunity to get the assessor's feedback is also important

because not only will the assessor mark the report, it is also a golden opportunity to get a third person's perspective on the project. The feedback given by the assessor has been very constructive, and it gave me the opportunity to properly structure this report.

Towards the end of the project, time became really precious and my original plan of completing the implementation part for the last section well before the deadline fell apart as I had some trouble with the implementation, thus allowing little time for the final evaluation. A word of advise for students looking to work on this area is to keep a good time management schedule and to start early as soon as the opportunity arise, because errors and bugs in implementation can throw you off guard when you least expected them to. You will be surprised at what you can achieve if you plan your project well.

Another advise that you constantly hear from lecturers and seniors alike is to choose a project that you are genuinely interested in, and I couldn't agree more. I found myself naturally drawn to this area of computing and I can't imagine myself working on a different area now, although if given the opportunity, I would no doubt take up the challenge. Successful delivery of your project is just as important as finding your passion working on it.

Finally, the knowledge and experience I have gained through working on this project is rewarding. I thoroughly enjoy my time spent on it.

Appendix B

Record of Materials Used

For the multilevel Monte Carlo simulation, the algorithms are obtained from Giles (2008)'s "Multilevel Monte Carlo path simulation" and Giles (2007)'s "Improved multilevel Monte Carlo convergence using the Milstein Scheme". In this project, however, all the work done is in Java so the implementation of the multilevel method in Java is based on the MATLAB code provided by Mike Giles, which can be found in `http://people.maths.ox.ac.uk/gilesm/mlmc.html`.

Appendix C

Interim Report

School of Computing, University of Leeds

MSc INTERIM PROJECT REPORT

All MSc students must submit an interim report on their project to the CSO ***by 9 am Friday 17th June.*** Note that it may require two or three iterations to agree a suitable report with your supervisor, so you should let him/her have an initial draft well in advance of the deadline. The report should be a maximum of 15 pages long and be attached to this header sheet. It should include:

- the overall aim of the project
- the objectives of the project
- the minimum requirements of the project and further enhancements
- a list of deliverables
- resources required
- project schedule and progress report
- proposed research methods
- a draft chapter on the literature review and/or an evaluation of tools/techniques

The report will be commented upon both by the supervisor and the assessor in order to provide you with feedback on your approach and progress so far.

Student:	Pei Yuen Lee
Programme of Study:	**MSc Computing and Management**
Title of project:	Modelling and Simulation of Options Pricing
Supervisor:	**Matthew Hubbard**
External Company (if appropriate):	**-**

Signature of student:  **Date: 14/06/2011**

Supervisor's and Assessor's comments overleaf

Supervisor's comments on the Interim Report

The report is well-structured and clearly written, showing a good understanding of the background material and a promising start to the project. The main issue is the lack of a clear direction from this point onwards - a challenging issue needs to be addressed if a distinction-level grade is to be considered. A specific goal should be identified which demonstrates sufficiently advanced skills in computation. This has the potential to be a very interesting and challenging project.

Assessor's comments on the Interim Report

The report submitted could serve as a model of clarity for other students, and it is easy to see that the student has the project well in hand. The planning is reasonable, and delivery is not likely to be an issue.

It is of some concern, however, that the amount of computation envisaged is small. More attention should be paid to communicating the role of computation, and what technical hurdles are involved

C.1 Aim

The aim of the project is to study several options pricing models to determine areas where improvements in pricing options can be made, and then to develop an efficient method to price options.

C.2 Objectives

The objectives of the project are to:

- Design and implement the algorithms for binomial and Monte Carlo models.
- Test the algorithms with different test cases by changing parameters.
- Visualize the models from the results obtained.
- Develop a method to improve these models.

C.3 Minimum Requirements

The original minimum requirements are:

1. A binomial simulation of the European option.
2. A Monte Carlo simulation of the European option.
3. A prototype of a static interface to visualize the results.

Further extensions of the minimum requirements are:

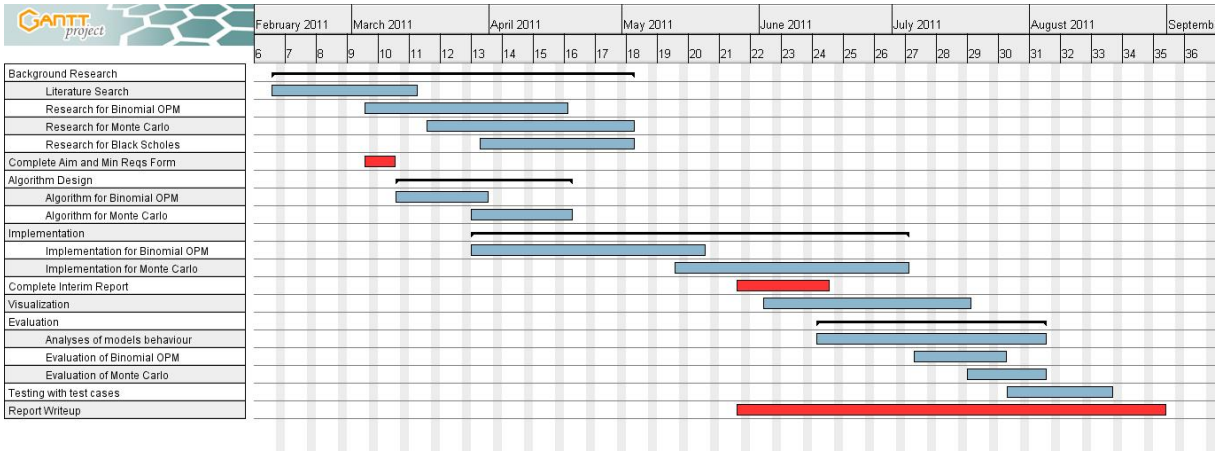
1. A modified options pricing model.
2. An implied volatility surface of the European option.

C.4 Deliverables

The deliverable for this project will be the written report and the modified options pricing model.

C.5 Project Schedule

The original project schedule is laid out in the Gantt chart below.



The progress at the current stage includes the continuous literature research on two option pricing models: binomial and Monte Carlo, and also the Black-Scholes formula which is used for validating the results from the two models. Algorithms for the two models were then designed and successfully implemented with Java.

Since the original aim was to develop a tool that can visualize the behaviour of the two models, a graph plotting tool or extension was needed. JMathPlot is an essential Java library that can plot 2D and 3D graphs so it is used for this project. As of this writing, graphs that have been successfully plotted include the stock-option graph, the option pricing surface and the graph of convergence of the binomial model to the Black-Scholes model.

C.6 Proposed research methods

The research method begins with background research on financial modelling to determine current research interests on options pricing, including the different styles of options, the Black-Scholes model, the binomial options pricing model and the Monte Carlo option pricing model. Next, the basic algorithms for binomial OPM and Monte Carlo OPM will be desinged to prepare for implementation and testing. Java will be the primary programming language of choice to draw on the programming experience gained from related modules. Currently, the JMathPlot package for Java is used as the graphical extension for visualizing the models but other possible tools such as MATLAB will be considered if there is a need for visualizing more complex models. A further extension to the project would be a method to improve the current option pricing models.

C.7 Literature Review

The modelling of financial options became significant in the early seventies when Fischer Black and Myron Scholes [1973] published a paper on pricing European call and put options using stochastic differential equations which subsequently led to the founding of the Chicago Board of Options Exchange (CBOE) in 1973. Since then, various studies on extending the model and

developing alternative approaches to the valuation of options have emerged, including numerical approaches such as finite difference methods, Monte Carlo methods and binomial methods. A recent literature by (Broadie and Detemple, 2004) focuses on the trends and development of financial options modelling with emphasis on the development of models that depart from the assumptions of the classic Black-Scholes (BS) model, since empirical evidence suggests that the BS prices tend to differ from the market prices of options due to the assumption that sharp changes in stock prices are negligible (MacBeth and Merville, 1979; Vasile and Armeanu, 2009).

Several modifications of the model have been made to reduce discrepancies between these assumptions and the real world. Examples are the extension of the Black-Scholes model with illiquidity (Cetin et al., 2004), the inclusion of transaction costs through adjusting the volatility (Leland, 1985), and also extensions to include jump-diffusion models and stochastic volatility models. In terms of option styles, the Black-Scholes model can only be used to price European options, where exercising can only take place at the expiration date. American options, on the other hand, can be exercised on or before the expiration date. Hence, the binomial lattice model is suitable for valuing this type of options (for example, the Cox, Ross and Rubenstein (CRR) model). For more complex options like exotic options with multiple uncertainties, the Monte Carlo model is introduced.

C.8 Options pricing

An *option* is a derivative security that grants the buyer of the option the right, but not the obligation, to buy or sell an underlying asset, S (such as a stock, a bond or an index portfolio) on or before an expiration date, T, for an *exercise* or *strike price*, K. A *call option* is the right to buy, while a *put option* gives the right to sell. The most common styles of standard plain vanilla options are the European option and the American option. Exercising for the European option can only be done at the expiration time, T, but the option can be exercised at any time, t, up to the expiration date for the American option. Since exercising the option is a right, the exercise payoff, for a call option, is $max\{S-K,0\}$, and for a put, $max\{K-S,0\}$, in which case 0 is the situation where the option becomes worthless upon expiration. Due to the non-negative payoff, an investor must pay a premium to purchase the option.

For the convenience of further discussion, the notations used throughout the paper are summarized below:

S = price of underlying asset

K = strike or exercise price

C = value of the European call option

r = risk-free interest rate

t = time in years

T = maturity date

σ = volatility of returns of the underlying asset

μ = drift rate

p = a probability measure

q = probability that the price will move upwards

C.8.1 Black-Scholes Model

The most popular method of pricing options is the classic Black-Scholes model (Black and Scholes, 1973). The assumptions for this model are:

1. There is no arbitrage opportunity so there is a premium price for buying the option.
2. The price follows a geometric Brownian motion with constant drift and volatility.
3. The underlying stock does not pay dividends.
4. There is a constant interest rate.
5. There is no transaction cost and tax.
6. It is possible to buy or sell any amount of options at any given time.

The underlying asset follows the geometric Brownian motion (GBM), which applies the following stochastic differential equation (SDE):

$$dS = \mu S dt + \sigma S dW \tag{C.1}$$

Using Itô's lemma,

$$dC = \left(\mu S \frac{\partial C}{\partial S} + \frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2}\right) dt + \sigma S \frac{\partial C}{\partial S} dW$$

and from the risk-neutrality measure (setting $\mu = r$), the Black-Scholes partial differential equation (PDE) can be derived as

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{C.2}$$

Solving the PDE, we obtain the following Black-Scholes formula

$$C = SN(d_1) - Ke^{-rt}N(d_2) \tag{C.3}$$

with

$$d_1 = \frac{ln(\frac{S}{K}) + (r + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}} \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{t}$$

where $N(d_1)$ and $N(d_2)$ denote the cumulative distribution functions of the standard normal distribution for d_1 and d_2 respectively. The complete derivation can be found in (Ugur, 2008), (Wilmott, Howinson and Dewynne, 1998) and (Brandimarte, 2002).

We shall take a test example problem for pricing a European call option to illustrate this model. Suppose that a stock price, S, of a company is currently \$250 per share, and that in a year's time ($T = 1$), the price either rises or falls by 20% ($\sigma = 0.2$). A European call option is to buy the stock at an exercise price, K, of \$200 at the expiration time with a risk-free rate of 5% ($r = 0.05$). To price the call option, we substitute the given set of variables into (C.5), and obtain $C = \$61.472091898474446$. This is the market value of the call option for the underlying stock. Mathematically, it is a closed-form solution of the call option that will form the basis of our result for comparisons with other methods of pricing options, in particular the binomial and Monte Carlo option pricing models.

One of the limitations of the Black Scholes model is it can only be used to price European options. To price American options, numerical methods such as the binomial method (Cox, Ross and Rubenstein, 1979) and finite difference methods (Schwartz, 1977) are required. For exotic options (e.g. Asian, Lookback and Barrier), the Monte Carlo method is normally used. We shall focus on two models in detail: the binomial model and the Monte Carlo model. These models are alternative methods that are approximations to the Black Scholes for European options. Running approximations of the Black-Scholes value help in validating results and reducing errors of the models so that they can be applied to the valuations of more complex options such as American and Asian options.

C.8.2 Binomial Options Pricing Model

(Cox, Ross and Rubenstein, 1979) (CRR) developed a discrete-time lattice model for valuing options that is useful for pricing American options. This binomial model follows the same basic assumptions as the Black-Scholes model. At each step, it is assumed that the stock price either moves up by an up factor of u or down by a down factor of d. Thus, if the current stock price is S, the stock price at the next period will either be uS or dS. CRR method assumes that u and d are determined by the volatility σ, such that

$$\begin{aligned} u &= e^{\sigma\sqrt{\delta t}} \\ d &= \frac{1}{u} = e^{-\sigma\sqrt{\delta t}} \\ q &= \frac{1}{2} + \frac{1}{2}\left(\frac{\mu}{\sigma}\right)\sqrt{\delta t} \end{aligned} \tag{C.4}$$

where q is the probability of the price moving upwards.

Let r be the risk-free interest rate. We require that $u > e^{r\delta t} > d$, where $e^{r\delta t}$ is the discounted riskless bond at time δt according to the risk-neutrality measure. Let C be the current call option. At the end of one period, the option will be C_u if the stock price goes to uS and C_d if

the stock price goes to dS. Hence, we have

$$C_u = max(0, uS - K) \tag{C.5}$$
$$C_d = max(0, dS - K) \tag{C.6}$$

Suppose Δ is the number of shares[1] of stock and B is the amount invested in risk-free interest bond (a debt security), the value of the portfolio is $\Delta S + B$, which we equate to C. Then, the up and down options become

$$\Delta uS + e^{r\delta t}B = C_u \tag{C.7}$$
$$\Delta dS + e^{r\delta t}B = C_d \tag{C.8}$$

Solving (C.7) and (C.8), we find that

$$\Delta = \frac{C_u - C_d}{(u-d)S} \quad \text{and} \quad B = \frac{uC_d - dC_u}{(u-d)e^{r\delta t}}$$

Therefore, the call option is

$$C = \Delta S + B = \left[\frac{e^{r\delta t} - d}{u-d}C_u + \frac{u - e^{r\delta t}}{u-d}C_d\right] / e^{r\delta t}$$

Let

$$p = \frac{e^{r\delta t} - d}{u-d} \quad \text{and} \quad 1 - p = \frac{u - e^{r\delta t}}{u-d}$$

Hence, we can write the call option as

$$C = [pC_u + (1-p)C_d]e^{-r\delta t} \tag{C.9}$$

Now we consider a call option with two periods. After the first period, C_u either goes up to C_{uu} or down to C_{ud}. C_d is analogous. CRR method ensures that the price that moves up and then down is equivalent to the price that moves down and then up. From the previous derivation, we find that

$$C_u = [pC_{uu} + (1-p)C_{ud}]e^{-r\delta t}$$
$$C_d = [pC_{du} + (1-p)C_{dd}]e^{-r\delta t}$$

Generally, for n periods, the equation is

$$C = \left[\sum_{j=a}^{n}\left(\frac{n!}{(n-j)!j!}\right)p^j(1-p)^{n-j}u^j d^{n-j}S - K\right]e^{-r\delta t} \tag{C.10}$$

[1] Δ is not to be confused with a "change". It is the number of shares of stock to buy for a call option. It is also called a riskless hedge ratio and is in the range $0 \leq \Delta \leq 1$.

where a is the smallest non-negative integer such that $u^a d^{n-a} S > K$.

Algorithm for the binomial option pricing model

To price options using the binomial model, we have to design an algorithm which we will implement in Java. Here, we designed a pseudocode to show the iterations involved in building a lattice tree to obtain the binomial value of a European call option.

Algorithm C.1: An algorithm for a multilevel Monte Carlo simulation.

```
function europeanCall(T, S, K, r, sigma, q, n) {
    deltaT:= T/ n;
    up:=     exp(sigma* sqrt(deltaT));
    down:=   1/up;
    c0:=     (up* exp(-r* deltaT)- exp(-q* deltaT))* up/ (up^ 2- 1);
    c1:=     exp(-r* deltaT)- c0;

    for i:= 0 to n {
        c(i):= S* up^i* down^(n-i); if c(i)< 0 then c(i)=0;
    }

    for j:= 0 to n step -1 {
        for i:= 0 to j {
            c(i):= c0* c(i)+ c1* c(i+1);
        }
    }
    return europeanCall:= c(0); }
```

After implementation in Java, we test the result with the same test example for a 2-period binomial tree. The binomial value is 61.9480957352685. This result is validated by checking it against an online binomial calculator.[2] For different stock price values, we can generate a range of option prices. To visualize these results, we need to import a plotting tool in Java as an extension. JMathPlot is used in this case because it is relatively simple to implement and can plot both 2D and 3D graphs.

Figure C.1 is a stock-option graph produces using the binomial model for $n = 2$ that illustrates how the option prices changes with different stock prices. Included in the graph are the maximum and minimum values. The minimum value of the option (intrinsic value) is the value at which a call option is in-the-money (i.e. the strike price is below the stock price). In other words, it is the actual value of the stock as opposed to the option value and is calculated by taking the difference between the strike price and the stock price. Option prices, on the other hand, are calculated using the equation (C.10). The time value (extrinsic value) is the difference between the option price and the intrinsic value. As an option moves closer to maturity, the values of the options move closer to the intrinsic value, which means that the time value decays and eventually becomes worthless when it reaches maturity (Ugur, 2008).

[2]This can be found at http://jpja.net/interactive/binomial.php. This application also displays the binomial tree with a value at each node, which is useful for checking the value of each iteration.

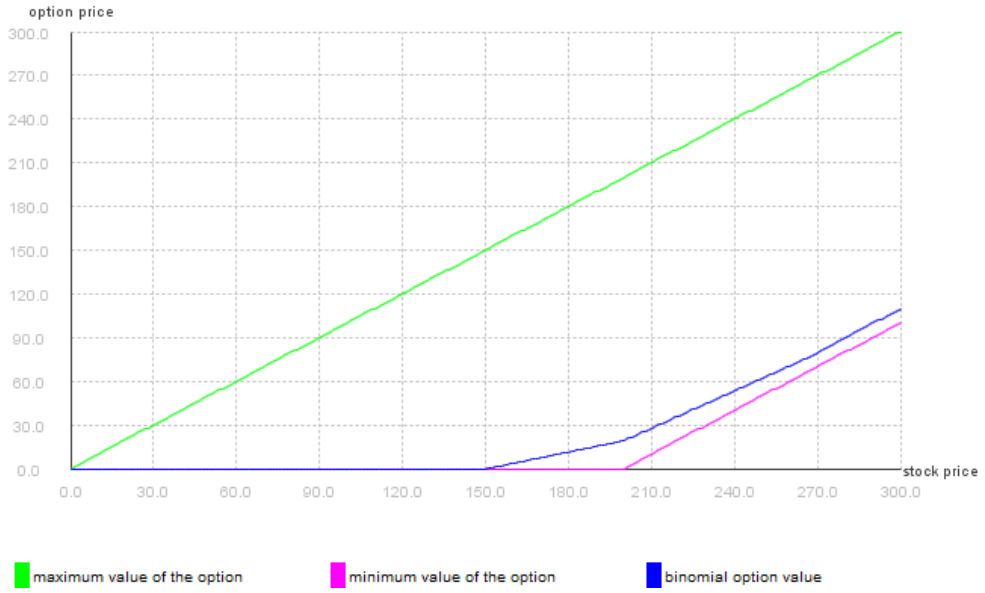


Figure C.1: A graph of stock price against option price with maximum and minimum (intrinsic) values

As the strike price increases, the option value decreases since the strike price moves closer to the stock value. Combining Figure C.1 and Figure C.2, we plot a snapshot of the option pricing surface as a function of the stock price and the strike price. (see Figure C.3). The other variables remain fixed, that is, the risk-free rate at $r = 0.05$, the expiration time at $T = 1$, and volatility at $\mu = 0.2$.

Convergence of the binomial formula to the Black-Scholes formula

The model, implemented in Java, is run with the same test example that was used for the Black-Scholes formula but this time, for the binomial, we test it with different number of periods, n. The following table shows the binomial values obtained for different n.

no. of period, n	Binomial value
10	61.53616204233657
50	61.443894450462025
100	61.48373974924799
200	61.468107803401594
500	61.47445853544964
1000	61.47304425642073
5000	61.47232014677787

As the number of period increases, the binomial value seems to converge to the Black-Scholes value of 61.472091898474446. To demonstrate this behaviour, we implement the convergence for the given test example in Java using the Java package JMathPlot (see Figure C.4). There is, in fact, a close analogy between the binomial formula and the Black-Scholes formula in (C.5).

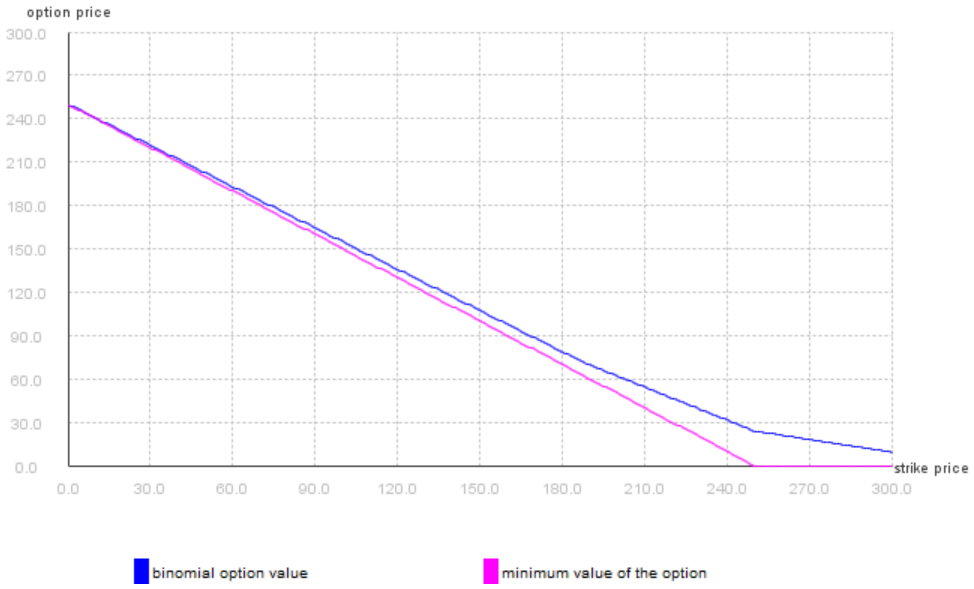


Figure C.2: A graph of strike price against option price

The call option obtained at (C.10) can be rewritten as

$$C = S\left[\sum_{j=a}^{n} \frac{n!}{(n-j)!j!} p^j (1-p)^{n-j} \frac{u^j d^{n-j}}{e^{r\delta t}}\right] - Ke^{-r\delta t}\left[\sum_{j=a}^{n} \frac{n!}{(n-j)!j!} p^j (1-p)^{n-j}\right]$$

Replacing the two parts in parentheses with functions $\phi(a; n, p')$ and $\phi(a; n, p)$ respectively, we obtain a simpler equation of the form

$$C = S\phi(a; n, p') - Ke^{-r\delta t}\phi(a; n, p)$$

where $p' = ue^{-r\delta t}p$

From (Cox, Ross and Rubenstein, 1979) work on the convergence of the binomial formula to the Black-Scholes formula, as n tends to infinity,

$$\phi(a; n, p') \to N(d_1) \quad \text{and} \quad \phi(a; n, p) \to N(d_2)$$

Hence, the Black Scholes formula is a limiting case of the Binomial OPM (Cox, Ross and Rubenstein, 1979; Lee and Lin, 2010). However, the proof given by CRR imposed restrictions on u, d and q (see equation (C.4)). Hsia (1983) applied a general proof for the convergence of the Binomial OPM to the Black Scholes formula without restricting u, d and q, using the DeMoivre-Laplace limit theorem with the only condition being $np \to \infty$ as $n \to \infty$. Qu (2010) further demonstrated that there is a direct proof of the Binomial OPM converging to Black Scholes formula as n tends to infinity with the use of direct approximation of binomial probability from the normal distribution. The rate of convergence from the Binomial pricing model to the Black Scholes formula was found to be $\frac{1}{n}$ (Chang and Palmer, 2007).

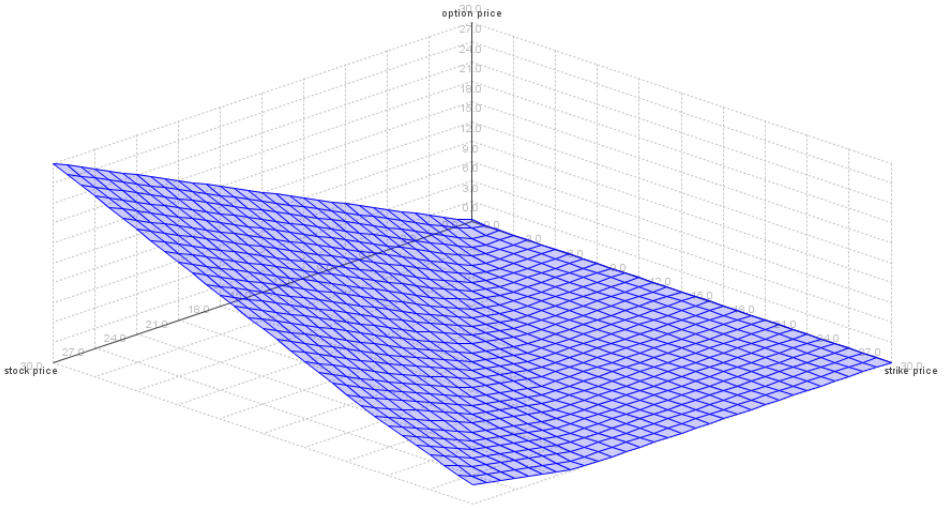


Figure C.3: A pricing surface with stock and strike as the independent variables

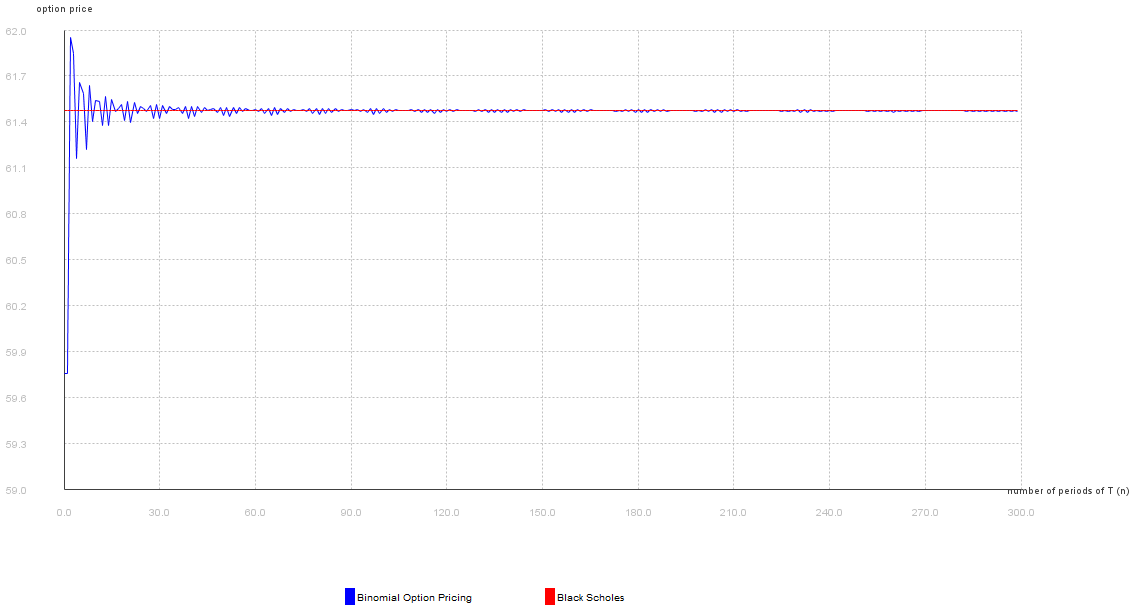


Figure C.4: A graph showing convergence of the binomial model to the BS model

C.8.3 Monte Carlo Simulation

Monte Carlo simulation, which was first proposed by Boyle (1977), uses pseudorandom numbers to simulate price paths. It is a useful method to price options that has multiple uncertainties when the Black-Scholes and the binomial tree become infeasible. Like the Black-Scholes model, the underlying asset is assumed to follow the geometric Brownian motion (GBM) given by the stochastic differential equation (SDE):

$$dS = \mu S dt + \sigma S dW \tag{C.11}$$

where μ is the drift rate and σ is the volatility. Since the risk-neutrality assumption is made for the pricing of options, we let $\mu = r$, where r is the risk-free interest rate. We can generate a sample path by dividing the time period $[0, T]$ into M intervals of δt and using Itô's Lemma with the properties of lognormal distribution to obtain a sample stock price path of

$$S(t + \delta t) = S(t)\exp\left[\left(r - \frac{\sigma^2}{2}\right)\delta t + \sigma\sqrt{\delta t}Z_j\right] \qquad , j = 1, ..., M \tag{C.12}$$

Z_j is a standard normal random variable for $j = 1, ..., M$. The payoff, $C(\omega)$, for a European call option is $max(S_t - K, 0)$ for a sample path ω. To sample N stock price paths, we find the sample mean of the payoffs discounted to present using the risk free rate, r, to obtain

$$C = \frac{1}{N}\sum_{i=1}^{N} C(\omega_i)e^{-r\delta t}$$

Algorithm for the Monte Carlo simulation

Similar to the previous model, we will show the steps involved in valuing options using the Monte Carlo simulation and implement them in Java. The following pseudocode is the algorithm we have designed for the Monte Carlo simulation of a European call option.

Algorithm C.2: An algorithm for a multilevel Monte Carlo simulation.

```
1
2  function europeanCall(T, S, K, r, sigma) {
3      deltaT:= T/M;
4      sum:= 0;
5      for i:= 1 to N {
6          for j:= 1 to M {
7              S(t+deltaT) = S(t) * exp[(r-0.5*sigma^2)*timestep+
8                          sigma*sqrt(timestep)*rand];
9          }
10         sum := sum + max(S–K,0);
11     }
12     c := sum/N*exp(-r*timestep);
13     return europeanCall:= c;
14 }
```

Convergence of the Monte Carlo simulation to the Black-Scholes formula

In this context, the error generated by Monte Carlo simulation can be reduced by increasing the number of simulation runs, N.

no. of runs, N	mean Monte Carlo value
100	59.455676789370784
10000	61.62079134557369
100000	61.37954293238363

Jabbour and Liu (2005) tested the convergence of the Monte Carlo price to the closed-form solution of $C = 61.472091898474446$, which is measured by taking the difference of the sample means of both models, and applying the t-test. The evidence suggests that the accuracy of the price is increased with an increase in the number of executions. However, increasing the runs causes this method to be computationally slow.