VŠB — TECHNICAL UNIVERSITY OF OSTRAVA

FACULTY OF ECONOMICS

DEPARTMENT OF FINANCE

Srovnání vybraných metod pro oceňování opcí pomocí C++

Comparison of Selected Methods for Option Pricing Using C++

Student: Lun Gao

Supervisor of the diploma thesis: doc. Ing. Tomáš Tichý, Ph.D.

Ostrava 2018

VŠB - Technical University of Ostrava
Faculty of Economics
Department of Finance

Diploma Thesis Assignment

Student:	**Bc. Lun Gao**
Study Programme:	N6202 Economic Policy and Administration
Study Branch:	6202T010 Finance
Title:	Comparison of Selected Methods for Option Pricing Using C++ Srovnání vybraných metod pro oceňování opcí pomocí C++
The thesis language:	English

Description:

1. Introduction
2. Analysis of Financial Derivatives and Their Pricing
3. Description of Selected Methods for Option Pricing
4. Evaluation of Selected Approaches
5. Conclusion

Bibliography
List of Abbreviations
Declaration of Utilisation of Results from the Diploma Thesis
List of Annexes
Annexes

References:

DUFFY, Daniel J. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach.* Wiley, 2013. 464 p. ISBN 9781118856482.
PEÑA, Alonso. *Advanced Quantitative Finance with C++.* Packt Publishing, 2014. 124 p. ISBN 9781782167235.
WILMOTT, P., S. HOWISON and J. DEWYNNE. *The Mathematics of Financial Derivatives: A Student Introduction.* Cambridge University Press, 1995. 317 p. ISBN 9780521497893.

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor:	**doc. Ing. Tomáš Tichý, Ph.D.**
Date of issue:	24.11.2017
Date of submission:	27.04.2018

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA OSTRAVA
Ekonomická fakulta
1

[illegible]
Ing. Iveta Ratmanová, Ph.D.
Head of Department

[illegible]
prof. Dr. Ing. Zdeněk Zmeškal
Dean

The declaration

"Herewith I declare that I elaborated the entire thesis, including all annexes, independently."

Ostrava dated 23.04.2018..............................

[illegible]

Signature

Contents

Chapter 1. Introduction

As we all know, the world financial market is developing rapidly. Various types of transactions are complex and increasingly sophisticated. After going through several global financial crises, how to properly price financial derivatives has become an increasingly important topic in the financial sector today. Among them, the study of options is even more hot. In the study of option pricing, the pricing methods are mainly divided into two major categories of backward stochastic differential equations and martingale method. The widely used Black-Scholes partial differential equation is a special backward stochastic differential equation. Because its complexity determines the pricing of options is very difficult.

Since the scope of this topic is very broad, there are many aspects to be studied and it is impossible to cover everything. It is important to study the direction and ideas. So the main objective of this diploma thesis is to compare different options pricing methods and their application in the real market.

This thesis is divided into five main sections:

The first part of thesis is an introduction that focuses on explaining the main goals of this thesis and the structure of the article.

The second part of thesis will provide basic preliminary knowledge of option pricing such as the origin and development of the Black-Scholes model, the relationship between the option price and the underlying asset price, etc.

The third part of the article will focus on the methods that thesis chose to use. It mainly includes the establishment background of the Black-Scholes equation, the derivation and solution of partial differential equations, and the intrinsic deficiencies of the Black-Scholes option pricing model. At the same time, several common numerical methods in option pricing are introduced in detail, such as binomial tree method, Monte Carlo simulation method and finite difference method.

The fourth chapter of thesis can be divided into two parts. The first part focuses on using the sample data of the Chicago Board of Trade to calculate with different pricing methods. The results from different numerical methods are compared with the analytical solutions, and the most accurate method in the numerical solution can be obtained. At the same time, sensitivity analysis is also used to determine the impact of simulation times on the accuracy of the pricing model. The second part will use the Hong Kong Stock Exchange's stock and stock option data to calculate the theoretical price of European and American options. Then compare the resulting theoretical price with the real market price.

The fifth part is the conclusion which will explain the deviation between the market price and the theoretical price, also the comparison of several pricing models will be concluded.

Chapter 2. Analysis of financial derivatives and their pricing

A derivative is a financial contract that derives its value from an underlying asset. The buyer agrees to purchase the asset on a specific date at a specific price.

Derivatives are often used for commodities, such as oil, gasoline or gold. Another asset class is currencies, often the U.S. dollar. There are derivatives based on stocks or bonds. Still others use interest rates, such as the yield on the 10-year Treasury note.

The contract's seller doesn't have to own the underlying asset. He can fulfill the contract by giving the buyer enough money to buy the asset at the prevailing price. He can also give the buyer another derivative contract that offsets the value of the first. This makes derivatives much easier to trade than the asset itself.

Many different types of derivatives have different pricing mechanisms. The most common derivative types are futures contracts, forward contracts, options and swaps. More exotic derivatives can be based on factors such as weather or carbon emissions. A derivative is a financial contract with a value based on an underlying asset.

Options on stocks and exchange-traded funds are also common derivative contracts. Options give the buyer the right, as opposed to the obligation, to buy or sell 100 shares of a stock at a strike price for a predetermined amount of time. The best-known pricing model for options is the Black-Scholes method. This method considers the underlying stock price, option strike price, time until the option expires, underlying stock volatility and risk-free interest rate to provide a value for the option.

There are many types of options. Divided by exercise time, there are three types of European options, American options, Bermuda options. European option is an option that can only exercise at the end of its life, at its time. European options tend to sometimes trade at a discount to their comparable American option because American options allow investors more opportunities to exercise the contract. European options Normally trade over the counter, while American options usually trade on standardized exchanges. An American option is an option that can be exercised anytime during its life. American options allow option holders to exercise the option at any time prior to and including its maturity date, thus increasing the value of the option to the holder relative to European options, which can only be exercised at maturity. The majority of exchange-traded options are American. A Bermuda option is a type of exotic option that can be exercised only on predetermined dates, typically every month. Bermuda options are a combination of American and European options; they are exercisable at the date of expiration, and on certain specified dates that occur between the purchase date and the date of expiration. In addition, there are more complex derivative than regular options (standard European or American options). These are the exotic options just mentioned (barrier options, lookback options, shout options, Asian options…).

We often use two different methods in the pricing of options: series of methods derived from the normal distribution of the underlying asset prices and series of methods derived from the fat tail distribution and skewness distribution of the underlying asset price.

Classically, the factors affecting the pricing of options are price of underlying asset, exercise price, expiry date, volatility of underlying asset price, risk free rate and size of the proposed dividend. These factors have different influence on different kinds of options. In fact, price of underlying asset is the key variable influence the option price.

2.1 Wiener process

The change in the stock price is uncertain so it is suitable to be described in a stochastic process. First of all we will introduce the Markov process. In Markov process the change of a variable depends only on the state of the variable in the first instant. When variables follow a Markov process, the variances of the variables in the adjacent time are additive, but the standard deviation does not have additivity. The most important feature of Markov process is independent and identically distributed of random changes of variables.

The Wiener process could be seen as a special form of Markov process. If the variable obeys the Wiener process the expected value of the variable is 0 and the variance is 1. The stock price model is usually expressed in the Wiener process. In physics this process is also called the Brownian movement.

If the variable $z = z(t)$ obey the Wiener process its increment Δz must met the following two basic properties.

Property 2.1.1

The relationship between Δz and Δt satisfaction:

$$\Delta z = \varepsilon\sqrt{\Delta t} \tag{2.1.1}$$

The ε is a random value extracted from the standard normal distribution. The simplest case of a normal distribution is known as the standard normal distribution. This is a special case when $\mu = 0$ and $\sigma = 1$, and it is described by this probability density function:

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

The normal distribution is the only absolutely continuous distribution whose cumulants beyond the first two (i.e., other than the mean and variance) are zero. It is also the continuous distribution with the maximum entropy for a specified mean and variance. Assume that the mean and variance are finite, that the normal distribution is the only distribution where the mean and variance calculated from a set of independent draws are independent of each other.

Property 2.1.2

The value of Δz and Δt at any two different time intervals is independent.

From the properties 2.1.1 we can get that Δz is the normal distribution which obey the expected value is 0, variance equals to Δt, the standard deviation is $\sqrt{\Delta t}$. The properties 2.1.2 means the variable $z = z(t)$ obey the Markov process.

Again, by properties 2.1.2, when $\Delta T \to 0$ the differential form of Δz is:

$$dz = \varepsilon\sqrt{dt} \tag{2.1.2}$$

where ε is a random value extracted from the standard normal distribution.

2.2 General Wiener process

Variable x are subject to the general Wiener process as follows:

$$dx = adt + bdz \tag{2.2.1}$$

Among them, a and b are constant. a is the expected drift rate of the general Wiener process and b is the volatility.

The formula (2.2.1) is made up of two parts, if do not consider the bdz, then exist:

$$dx = adt \quad \text{or} \quad x = x_0 + at$$

The x_0 is the value of the x at the time *0*, after t time, the increment of the x is at.

If only bdz is considered, then exist:

$$dx = bdz$$

bdz can be seen as a noise or fluctuation attached to the trail of a variable x, these noises or fluctuations are the b times of the Wiener process.

Take adt and bdz into consideration, there exist:

$$dx = adt + bdz$$

After the time increment of Δt, the increment of the x is:

$$\Delta x = a\Delta t + b\Delta z \tag{2.2.2}$$

Bring the formula (2.2.1) into (2.2.2) can get:

$$\Delta x = a\Delta t + b\varepsilon\sqrt{\Delta t} \tag{2.2.3}$$

As mentioned in the previous article, ε is random sampling value derived from the standardized normal distribution. So the x obeys the normal distribution. Its average value is $a\Delta t$, variance is $b^2\Delta t$ and the standard deviation is $b\sqrt{\Delta t}$.

From the above discussion, we can conclude that after at any time t, the change of the x also obeys the normal distribution of average value is $a\Delta t$, variance is $b^2\Delta t$ and the standard deviation is $b\sqrt{\Delta t}$.

2.3 Ito calculus and Ito's Lemma

If the a and b are functions of x and t in stochastic process which mentioned in chapter 2.2. We can get Ito calculus:

$$dx = a(x,t)dt + b(x,t)dz \qquad (2.3.1)$$

The expected drift rate and volatility in the Ito process vary with time.

Theorem 2.3.1 (Ito's Lemma)

Assume that the variable x obeys the Ito calculus:

$$dx = a(x,t)dt + b(x,t)dz$$

dz is the Wiener process, suppose that the $G = G(x,t)$ is the twice continuously differentiable function of the x, then the $G = G(x,t)$ follows the following process:

$$dG = \left(\frac{\partial G}{\partial x} a + \frac{\partial G}{\partial t} + \frac{1}{2} \cdot \frac{\partial^2 G}{\partial x^2} b^2 \right) dt + \frac{\partial G}{\partial x} b dz \qquad (2.3.2)$$

Mathematical proof:

From the Taylor expansion formula of the binary function

$$\Delta G = \frac{\partial G}{\partial x} \Delta x + \frac{\partial G}{\partial t} \Delta t + \frac{1}{2} \cdot \frac{\partial^2 G}{\partial x^2} \Delta x^2 + \frac{\partial^2 G}{\partial x \partial t} \Delta x \Delta t + \frac{1}{2} \cdot \frac{\partial^2 G}{\partial t^2} \Delta t^2 + \cdots \qquad (2.3.3)$$

$\because$

$$\Delta x = a(x,t)\Delta t + b(x,t)\varepsilon\sqrt{\Delta t} \qquad (2.3.4)$$

$\therefore$

$$\Delta x^2 = b^2 \varepsilon^2 \Delta t + o(\Delta t) \qquad (2.3.5)$$

From formula (2.3.4), can get

$$\Delta x \Delta t = a(x,t)\Delta t^2 + b(x,t)\varepsilon\sqrt{(\Delta t^3)} = o(\Delta t) \qquad (2.3.6)$$

Take formula (2.3.4), (2.3.5) and (2.3.6) into (2.3.3), can get

$$\Delta G = \frac{\partial G}{\partial x} \Delta x + \frac{\partial G}{\partial t} \Delta t + \frac{1}{2} \cdot \frac{\partial^2 G}{\partial x^2} b^2 \Delta t + o(\Delta t)$$

Make $\Delta t \to 0$, can get

$$\Delta G = \frac{\partial G}{\partial x}dx + \frac{\partial G}{\partial t}dt + \frac{1}{2}\cdot\frac{\partial^2 G}{\partial x^2}b^2 dt \tag{2.3.7}$$

Then take $dx = a(x,t)dt + b(x,t)dz$ into formula (2.3.7),can get

$$dG = \left(\frac{\partial G}{\partial x}a + \frac{\partial G}{\partial t} + \frac{1}{2}\cdot\frac{\partial^2 G}{\partial x^2}b^2\right)dt + \frac{\partial G}{\partial x}bdz \tag{2.3.8}$$

Quod erat demonstrandum.

From the Ito lemma will know, if x, t obey the Ito calculus, in that way the function of x, t. G also obey the Ito calculus, but the drift rate and the fluctuation rate are

$$\frac{\partial G}{\partial x}a + \frac{\partial G}{\partial t} + \frac{1}{2}\cdot\frac{\partial^2 G}{\partial x^2}b^2 \text{ and } \left(\frac{\partial G}{\partial x}b\right)^2,$$

respectively.

2.4 The behavior process of stock price without dividend

Assume that the stock price obeys the general Wiener process, there is a constant expected drift rate and volatility, which is not fix the reality. So, it is generally assumed that the proportion of the stock price changes dS/S obeys the general Wiener process, that is

$$\frac{dS}{S} = \mu dt + \sigma dz \tag{2.4.1}$$

Therefore, the stock price S can be described by the Ito calculus of the drift rate μS and the volatility σS.

That is

$$\Delta S = \mu S dt + \sigma S dz \tag{2.4.2}$$

Its dispersed form is

$$\Delta S = \mu S \Delta t + \sigma S \Delta z \tag{2.4.3}$$

If μ and σ are constant, the formula (2.4.2) is called the geometric Brownian motion which is the most widely used model to describe the behavior of stock prices.

If S obeys the Ito calculus, the function G of S and t also subject to the Ito calculus :

$$\begin{aligned} dG &= \left(\frac{\partial G}{\partial x} a + \frac{\partial G}{\partial t} + \frac{1}{2} \cdot \frac{\partial^2 G}{\partial x^2} b^2 \right) dt + \frac{\partial G}{\partial x} b dz \\ &= \left(\frac{\partial G}{\partial S} \mu S + \frac{\partial G}{\partial t} + \frac{1}{2} \cdot \frac{\partial^2 G}{\partial S^2} \sigma^2 S^2 \right) dt + \frac{\partial G}{\partial S} \sigma S dz \end{aligned} \tag{2.4.4}$$

Where both S and G are affected by dz. Define $G = lnS$.

$\because$

$$\frac{\partial G}{\partial S} = \frac{1}{S}, \quad \frac{\partial^2 G}{\partial S^2} = -\frac{1}{S^2}, \quad \frac{\partial G}{\partial t} = 0$$

$\therefore$ Simplify the formula (2.4.4)

$$dG = (\mu - \frac{\sigma^2}{2}) dt + \sigma dz \tag{2.4.5}$$

$\because$ μ and σ and y are constant

$\therefore$ Formula (2.4.5) is also the Wiener process, the drift rate and the fluctuation rate are

$$\mu - \frac{\sigma^2}{2} \text{ and } \sigma .$$

Therefore the change of the lnS between the t and the T times follows the normal distribution. The expectation and variance are

$$dG = \left(\mu - \frac{\sigma^2}{2} \right) dt + \sigma dz \text{ and } \sigma^2 (T - t) .$$

Which means

$$\ln S_T - \ln S \sim N\left((\mu - \frac{\sigma^2}{2})(T-t), \sigma^2(T-t) \right)$$

or

$$\ln S_T \sim N\left(\ln S + (\mu - \frac{\sigma^2}{2})(T-t), \sigma^2(T-t) \right) \tag{2.4.6}$$

Where $N(m, s)$ means the normal distribution which the expected value is m and the variance is s.

The above is the analysis and inference of the option pricing methods based on the underlying asset price fix the Gaussian distribution. With the development of mathematical sciences, a deeper level of research has found that the price of the underlying asset does not strictly adhere to the standard normal distribution. The data often shows partial peaks and fat tails.

2.5 Levy process

More and more studies can prove that the price fluctuation and the rate of return of financial assets are contrary to the geometric Brownian motion. At the same time, the assumption of the Black-Scholes model is too strict. So we will introduce the Levy process which is a stochastic process with independent, stationary increments: it represents the motion of a point whose successive displacements are random and independent, and statistically identical over different time intervals of the same length. A Levy process may thus be viewed as the continuous-time analog of a random walk.

The definition of Levy process: The stochastic process $LL_t\{L_t : t \geq 0\}$ defined in the probability space (Ω, F, P) which satisfies the following three conditions called Levy process.

1. L_t has independent increments, i.e. $L_t - L_s$ is independent of F_s for any $0 \le s < t \le T$.
2. L_t has stationary increments, i.e. for any $0 \le s, t \le T$ the distribution of $L_{s+t} - L_t$ does not depend on t.
3. L_t is stochastically continuous, i.e. for every $0 \le t \le T$ and $\epsilon > 0$: $\lim_{s \to t} P(|L_t - L_s| > \epsilon) = 0$.

If L is Levy process, the distribution $L = L_t - L_0$ is infinite divisible.

2.6 Normal Inverse Gaussian Distribution

The normal inverse Gaussian distribution (NIG) is a continuous probability distribution, which is defined as the normal variance-average mixture with an inverse Gaussian distribution (IG). The IG process is a normal stochastic time distribution process. The time increment of a random variable with a normal distribution for the first time to a certain critical value is used as the distribution of the new random variable. Its density function is:

$$f(x) = \frac{ae^{ab}}{\sqrt{2\pi}} x^{-\frac{3}{2} - \frac{1}{2}(a^3 x^{-1} + b^2 x)} . \qquad (2.6.1)$$

Its characteristic function is:

$$E(e^{iuX}) = \varphi(u; a, b) = e^{-a(\sqrt{-2iu+b^2} - b)} \qquad (2.6.2)$$

NIG model use IG process as a dependent process and use it to drive the Brownian motion of the time variable. The NIG model can be defined by two ways, as well. The first one utilize the characteristic function $(\alpha > 0, -\alpha < \beta < \alpha, \delta > 0)$:

$$\varphi NIG(x,t;\alpha,\beta,\delta)=\exp\left[-t\delta(\sqrt{\alpha^2-(\beta+\iota x)^2}-\sqrt{\alpha^2-\beta^2})\right]. \quad (2.6.3)$$

Then, the density function is given as follows:

$$f_{NIG}(x,t;\alpha,\beta,\delta)=\frac{\alpha\delta}{\pi}\exp\left(\delta\sqrt{\alpha^2-\beta^2}+\beta x\right)\frac{K_1(\alpha\sqrt{\delta^2+x^2})}{\sqrt{\delta^2+x^2}}, \quad (2.6.4)$$

where $K_\lambda(x)$ is modified Bessel function:

$$K_\lambda(x)=\frac{1}{2}\int_0^{-\infty} y^{\lambda-1}\exp\left(-\frac{1}{2}x(y+y^{-1})\right)dy\,. \quad (2.6.5)$$

Alternatively, following the definition of the Brownian motion driven by inverse Gaussian process, i.e. process $L(t;v)$ with drift v, which at time $L \sim IG[t;v]$ reaches level t, as follows:

$$NIG(L(t;v);\theta;\vartheta)=\theta L_t+\vartheta Z(L_t)=\theta L_t+\vartheta\sqrt{L_t}\varepsilon\,. \quad (2.6.6)$$

In this case we can formulate the characteristic function as follows:

$$\phi NIG(x;v,\theta,\vartheta)=\exp\left[\frac{1}{v}-\frac{1}{v}(\sqrt{1+x^2\vartheta^2 v-2\theta v\iota})\right], \quad (2.6.7)$$

which result into:

$$\theta=\frac{\delta\beta}{\sqrt{\alpha^2-\beta^2}},\ \vartheta=\frac{\sqrt{\delta\sqrt{\alpha^2-\beta^2}}}{\sqrt{\alpha-\beta}\sqrt{\alpha+\beta}}\ \text{ and }\ v=\frac{1}{\delta\sqrt{\alpha^2-\beta^2}}. \quad (2.6.8)$$

Similarly to variance gamma model also in the case of the NIG model particular parameters allows us to fit the skewness and kurtosis. We can see on next table.

Model	NIG
Parameter	$NIG(L(t;v);\theta,\vartheta)$
Mean	θ
Variance	$\vartheta^2+v\theta^2$
Skewness	$3\theta v(\vartheta^2+v\theta^2)^{-\frac{1}{2}}$
Kurtosis	$3\frac{v\theta^2(1+5v)+\vartheta^2(1+v)}{\vartheta^2+v\theta^2}$

Table 2.6.1 Comparison of basic moments for NIG model.

While Black-Scholes model is based on the geometric Brownian motion, and thus the unrealistic assumption of Gaussian distribution, more advance NIG model allows us to fit also the skewness and excess kurtosis of the returns. Recall NIG process $NIG(L(t;v);\theta;\vartheta)$:

$$NIG_t=\theta L_t+\vartheta Z(L_t)=\theta L_t+\vartheta\sqrt{L_t}\varepsilon\,. \tag{2.6.9}$$

The above is the analysis of two kinds of distributions that are commonly used in option pricing. In this diploma thesis we will only price options for hypothetical underlying asset price distributions that conform to Gaussian distributions.

Since the B-S model was first published in the Journal of Political Economy in 1973, the traders at the Chicago Board Options Exchange immediately realized its importance, and soon programmed the B-S model into computers for use in the newly opened Chicago Options Exchange. The application of this formula expands with the advancement of computer and communication technology. To this day, the model and some of its variants have been widely used by options dealers, investment banks, financial managers, insurers, and so on. The expansion of derivatives has made the international financial market more efficient, but it has also made the global market more volatile. The creation of new technologies and new financial instruments has strengthened the interdependence of markets and market participants, not only in one country but also in other countries or even multiple countries. The result is that a market or a country's volatility or financial crisis is most likely to be rapidly transmitted to other countries and even the entire world economy. The result is that a market or a country's volatility or financial crisis is most likely to be rapidly transmitted to other countries and even the entire world economy. Therefore, it is necessary to cultivate risk-averse financial derivatives markets. It is also necessary to explore derivative markets. Although there are many advantages in the Black-Scholes option pricing model, its derivation process is difficult for people to accept. In 1979, Ross et al. used a relatively simple method to design a pricing model for options, known as the Binomial tree method.

In thesis Monte Carlo simulations, least-squares Monte Carlo simulations, binomial methods, and finite difference methods will be used to analyze the options price.

Chapter 3. Description of selected methods for option pricing

Financial option is a kind of contract which gives the buyer (the owner or holder of the option) the right, but not the obligation, to buy or sell an underlying asset or instrument at a specified strike price on a specified date, depending on the form of the option. The holder of call option has the right to buy the underlying asset or instrument; the holder of put option has right to sell the underlying asset or instrument. The holder of European option can only exercise the option at the end of its life, at its maturity. American options allow option holders to exercise the option at any time prior to and including its maturity date. Through the introduction of the partial differential equations and Wiener process in Chapter 2, we can draw the Black-Scholes model.

3.1 Black-Scholes option pricing theory

The price of derivatives of non-dividend paying stock must be satisfied the Black Scholes partial differential equation. The Black Scholes partial differential equation is based on the following hypothesis:

1. The stock price follows the geometric Brownian motion.
2. Allow short selling of derived securities.
3. Without transaction costs or taxes, all securities are highly separable.
4. In the period of validity of derived securities, the underlying assets will not pay dividends.
5. There is no chance of risk-free arbitrage.
6. The transaction of securities is continuous.
7. The riskless interest rate r is constant and same for all maturity days.

According to hypothesis 1:

$$dS = \mu S dt + \sigma S dz \tag{3.1.1}$$

In formula (3.1.1) z is a Wiener process, μ is the expected rate of return on stock prices, σ is the volatility of the stock price.

Suppose that the derivative securities price f depends on the underlying asset price S, so f must be a function of S and time t.

From Ito's lemma:

$$df = \left(\frac{\partial f}{\partial S}\mu S + \frac{\partial f}{\partial t} + \frac{1}{2} \cdot \frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 \right) dt + \frac{\partial f}{\partial S}\sigma S dz \, . \tag{3.1.2}$$

The discrete forms of formula (3.1.1) and (3.1.2) are:

$$\Delta S = \mu S \Delta t + \sigma S \Delta z \, , \tag{3.1.3}$$

$$\Delta f = \left(\frac{\partial f}{\partial S}\mu S + \frac{\partial f}{\partial t} + \frac{1}{2} \cdot \frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 \right) \Delta t + \frac{\partial f}{\partial S}\sigma S \Delta z \, , \tag{3.1.4}$$

In these two formulas, Δf and ΔS are variation of f and S after a short interval of time Δt. Since f and S comply with the same Wiener process, the Δz of two formula (3.1.3) and (3.1.4) should be the same. So, a proper selection of stock and derivative portfolio can eliminate the uncertainty Δz.

In order to eliminate the Δz, we can build a portfolio with one unit derived securities short position and $\frac{\partial f}{\partial S}$ units of securities long position. Π represents the value of the portfolio, and there is a result:

$$\Pi = -f + \frac{\partial f}{\partial S} S \, .$$

After Δt time, the value of the portfolio changes to:

$$\Delta\Pi = -\Delta f + \frac{\partial f}{\partial S}\Delta S \tag{3.1.5}$$

Substituting ΔS and Δf into formula (3.1.5), get:

$$\Delta\Pi = \left(-\frac{\partial f}{\partial t} - \frac{1}{2}\cdot\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 \right)\Delta t\,. \tag{3.1.6}$$

Because formula (3.1.7) does not contain Δz, the value of the portfolio after the time interval Δt must be no risk, the instantaneous rate of return after Δt is equal to the risk-free rate. Otherwise, the arbitrage can gain a risk-free rate by arbitrage, so the result should be:

$$\Delta\Pi = r\Pi\Delta t\,. \tag{3.1.7}$$

Take formula (3.1.7) into (3.1.6), get:

$$\left(\frac{\partial f}{\partial t} + \frac{1}{2}\cdot\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 \right)\Delta t = r\left(f - \frac{\partial f}{\partial S}S \right)\Delta t$$

After finishing, get:

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\cdot\sigma^2 S^2\frac{\partial^2 f}{\partial S^2} = rf\,. \tag{3.1.8}$$

The formula (3.1.8) is the Black-Scholes partial differential equation. This equation applies to all derivative securities pricing that depends on the price of the underlying asset price S. There are many solutions to the equation. To ensure that it has a unique solution, we need to give the boundary conditions that meet the derivative securities.

For European call options, the key boundary conditions are:

$$c(S,t) = \max\left\{S - X, 0\right\},\quad t = T \tag{3.1.9}$$

When $S(t) = 0$, the options have no value, so the boundary condition is:

$$c(0,t) = 0\,. \tag{3.1.10}$$

When $S(t) \to +\infty$, $c(S,t) \to +\infty$ the value of the option becomes the value of the stock. That is:

$$c(S,t) \sim S\,,\ \ S \to \infty \qquad (3.1.11)$$

According to the boundary condition formula (3.1.9), (3.1.10) and (3.1.11), the equation (3.1.8) can be solved.

The equation (3.1.8) is similar to the diffusion equation, but it has more items. For the convenience of getting solution, we set:

$$S = Xe^x, t = T - \frac{1}{2}\tau\sigma^2, f = X\upsilon(x,\tau).$$

The equation (3.1.8) changes to:

$$\frac{\partial \upsilon}{\partial t} = (k-1)\frac{\partial \upsilon}{\partial x} + \frac{\partial^2 \upsilon}{\partial x^2} - k\upsilon\,,\ \ k = r\Big/\left(\frac{1}{2}\sigma^2\right) \qquad (3.1.12)$$

At this time the termination condition is transformed into the initial condition.

$$\upsilon(x,0) = \max\left\{e^x - 1, 0\right\}$$

The equation (3.1.12) only have one parameter k, so we make:

$$\upsilon = e^{\alpha x + \beta\tau}u(x,\tau)\,.$$

Here α and β are undetermined constants, take into (2.5.12) so that can get the new equation:

$$\beta u + \frac{\partial u}{\partial \tau} = \alpha^2 u + 2\alpha\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial x^2} + (k+1)\left(\alpha u + \frac{\partial u}{\partial x}\right) - ku \qquad (3.1.13)$$

Now choose α and β and make them satisfied:

$$\beta = \alpha^2 + (k-1)\alpha - k\,,$$

$$0 = 2\alpha - (k-1)\,.$$

So that can get:

$$\alpha = -\frac{1}{2}(k-1)\,,\ \ \beta = -\frac{1}{4}(k-1)^2\,.$$

So

$$\upsilon(x,\tau) = \exp\left[-\frac{1}{2}(k-1)x - \frac{1}{4}(k+1)^2\tau\right]u(x,\tau)\,, \qquad (3.1.14)$$

the u in equation (3.1.14) satisfied:

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2}\,,\ \ -\infty < x < +\infty\,,\ \ \tau > 0\,,$$

subject to

$$u(x,0) = u_0(x) = \max\left\{\exp\left[\frac{1}{2}(k+1)x\right] - \exp\left[\frac{1}{2}(k-1)x\right],0\right\}.$$

From the knowledge of differential equation, get:

$$u(x,\tau) = \frac{1}{2\sqrt{\pi\tau}}\int_{-\infty}^{+\infty} u_0(s)\exp\left[-\frac{1}{4\tau}(x-s)^2\right]ds\,.$$

Change to $x' = (x-s)/\sqrt{2\tau}$, so:

$$\begin{aligned}
u(x,\tau) &= \frac{1}{2\sqrt{\pi}}\int_{-\infty}^{+\infty} u_0(x'\sqrt{2\tau}+x)\exp\left[-\frac{1}{2}x'^2\right]ds \\
&= \frac{1}{2\sqrt{\pi}}\int_{-x/\sqrt{2\tau}}^{+\infty}\exp\left[\frac{1}{2}(k+1)(x+x'\sqrt{2\tau}) - \frac{1}{2}x'^2\right]dx' \\
&\quad - \frac{1}{2\sqrt{\pi}}\int_{-x/\sqrt{2\tau}}^{+\infty}\exp\left[\frac{1}{2}(k-1)(x+x'\sqrt{2\tau}) - \frac{1}{2}x'^2\right]dx' \\
&= I_1 - I_2\,, \qquad (3.1.15)
\end{aligned}$$

here

$$\begin{aligned}
I_1 &= \frac{1}{2\sqrt{\pi}}\int_{-x/\sqrt{2\tau}}^{+\infty}\exp\left[\frac{1}{2}(k+1)(x+x'\sqrt{2\tau}) - \frac{1}{2}x'^2\right]dx' \\
&= \frac{\exp\left[\frac{1}{2}(k+1)x\right]}{2\sqrt{\pi}}\int_{-x/\sqrt{2\tau}}^{+\infty}\exp\left[\frac{1}{4}(k+1)^2\tau - \frac{1}{2}(x' - \frac{1}{2}(k+1)\sqrt{2\tau})^2\right]dx'
\end{aligned}$$

$$= \frac{\exp\left[\frac{1}{2}(k+1)x + \frac{1}{4}(k+1)^2\tau\right]}{2/\sqrt{\pi}} \int_{-x/2\sqrt{\tau}-\frac{1}{2}(k+1)\sqrt{2\tau}}^{+\infty} \exp(-\frac{1}{2}\rho^2)\,d\rho$$

$$= \exp\left[\frac{1}{2}(k+1)x + \frac{1}{4}(k+1)^2\tau\right] N(d_1),$$

at this place

$$N(d_1) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{d_1} \exp(-\frac{1}{2}s^2)\,ds ,$$

$$d_1 = \frac{x}{2\sqrt{\tau}} + \frac{1}{2}(k+1)\sqrt{2\tau}$$

is the cumulative distribution function of the Gaussian distribution. Change $(k+1)$ to $(k-1)$ can get

$$I_2 = \exp\left[\frac{1}{2}(k-1)x + \frac{1}{4}(k-1)^2\tau\right] N(d_2),$$

$$d_2 = \frac{x}{2\sqrt{\tau}} + \frac{1}{2}(k-1)\sqrt{2\tau} .$$

Take I_1 and I_2 into equation (3.5.15), then use

$$x = \ln(\frac{S}{X}),\ \ \tau = \frac{1}{2}\sigma^2(T-t),\ \ c = X\upsilon(X,\tau),$$

$$\upsilon(x,\tau) = \exp\left[-\frac{1}{2}(k-1)x - \frac{1}{4}(k+1)^2\tau\right] u(x,\tau),$$

so

$$d_1 = \frac{\ln(S/X) + (r+\sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = \frac{\ln(S/X) + (r-\sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$$

$$= d_1 - \sigma\sqrt{T-t} .$$

From the formula above, we can get the Black-Scholes option pricing formula. For the European call option that obeys the geometric Brownian movement which expiration time is *T*, exercise price is *X*, underlying asset price is *S*, the pricing formula is:

$$c = SN(d_1) - Xe^{-r(T-t)}N(d_2) \tag{3.1.16}$$

According to the parity relationship between European call options and put options, it is easy to get the pricing formula of European put option:

$$p = Xe^{-r(T-t)}N(-d_2) - SN(-d_1) \tag{3.1.17}$$

Before using formula (3.1.16) and formula (3.1.17), we need to solve the calculate of $N(x)$. The $N(x)$ is the cumulative distribution function of the standard normal distribution. In this thesis we will apply it in C++.

The code (Program 1) is an approximate solution way to the cumulative distribution function of normal normal distribution. Then we can programe the formula (3.1.16) and formula (3.1.17). Save the C++ programme of cumulative distribution function of normal normal distribution as ‘normdist.h’ so we can directly invoke the cumulative distribution function of normal normal distribution 's header file in programming.

3.2 The numerical method of option pricing

Sometimes some complex derivative securities cannot give analytical solutions, so the numerical method is needed.

3.2.1 Monte Carlo method

The Monte Carlo method is a numerical method to solve the option price by simulating the movement of the underlying asset price. The basic ideal of Monte Carlo

Method is: in the risk nature situation, we randomly generate the possible path of the underlying asset price and get the expected value of the option earning. After that, discount the price by risk free rate then we can get the option price.

Assume that in the world of risk nature, the variable θ obey the geometric Brownian motion with the standard deviation of s and the expected rate of return is $\widehat{m}$, that is:

$$d\theta = \widehat{m}\theta dt + s\theta\varepsilon\sqrt{dt} \tag{3.2.1}$$

where ε is one random sample extracted from the normal distribution.

In order to simulate the path of the variable θ and considering the discrete form of (3.2.1), we divide the life of derivative security into n fragments with length Δt:

$$d\theta = \widehat{m}\theta\Delta t + s\theta\varepsilon\sqrt{\Delta t} \tag{3.2.2}$$

From this formula, we can get a path of the variable θ, its final value corresponds to a sample final value of the derivative price. It could be seen as a random sample in a set of final values. Using the same method, we can get a large number of sample final prices, and get the average value of the number, then get the approximate value of the final price of the derivatives. The price of derivative securities can be obtained by discounting the final value at risk free interest rate.

Assuming the European call option which price of the underlying asset is S, exercise price is X at the date of expiry the price is

$$c_T = \max\left\{0, S_T - X\right\} \tag{3.2.3}$$

In a risk neutral world, we use the risk-free rate r to discount to get the price of the option at the t moment

$$c_T = e^{-r(T-t)} E(\max\{0, S_T - X\}) \tag{3.2.4}$$

In formula (3.2.4), only S_T has relationship with c_T. The value of the underlying asset price during $T - t$ is independent from c_T. So just simulate $S_T$to get a series of values: S_T^1, S_T^2, S_T^3, …,S_T^n. Then replace $S_T^i (i = 1,2,3, \dots, n)$ into formula (3.2.4) to get all value of c_t. Then calculate the arithmetic mean of c_t, after that use risk free rate to discount to get the price of European call option

$$\hat{c}_T = \frac{e^{-r(T-t)}}{n} \sum_{i=1}^{n} E(\max\{0, S_{T_i} - X\}) \tag{3.2.5}$$

The same method can get the price of a European put option

$$\hat{p}_T = \frac{e^{-r(T-t)}}{n} \sum_{i=1}^{n} E(\max\{S_{T_i} - X, 0\}) \tag{3.2.6}$$

To apply the program of Monet Carlo Method we need to apply the program of Random number function. See in appendix (Program 4) The above Monte Carlo method can only be used to price European-style options, but in recent years, with the development of mathematical finance, there have been some algorithms that use the Monte Carlo method to simulate the pricing of American options. The most widely used is the Least Squares Monte Carlo simulation proposed by Longstaff and Schwartz. The basic principle is: at a limited number of discrete time points, according to the cross-sectional data of the simulated sample path of the target asset price at each moment, use least squares regression to find the expected return on continued holding options. And compare it with the proceeds that were immediately exercised at that moment. If the immediately exercise is greater than continued holding, it will immediately exercise or it will continue to hold. Suppose the option expiration date is T, the exercise time is T^e. The basic steps of Least Squares Monte Carlo simulation are similar as the European option. But should notice that the European option can only be exercised at expiry date that is $T = T^e$ but for American option $T^e \in [0, T]$, that is, the option can be exercise at any time before the expiration date. As the proceeds at the time of exercise are not only affected by the asset price, but also affected by the path taken by the asset price from the issue date $(t = 0)$ to the maturity date T. For European options, it has been mentioned how to calculate. But for American options, we need to compare the instantaneous income (intrinsic value) immediately exercise at that moment and the expected return to continue holding when determining the optimal exercise time. What needs to be established is the value to continued holding the option $F(\omega, t_k)$. According to no arbitrage principle:

$$F(\omega,t_k) = E_Q[\sum_{j=k+1}^{K} \exp(-\int_{t_k}^{t_j} r(\omega,s)\,ds C(\omega,t_j;t_k,T) \big| F_{tk}] \qquad (3.2.7)$$

Where $r(\omega, s)$ is riskless discount rate, the expectation is taken conditional on the information set. F_{t_k} at time t_k,. With this representation, the problem of optimal exercise reduces to comparing the immediate exercise value with this conditional expectation, and then exercising as soon as the immediate exercise value is positive and greater than or equal to the conditional expectation. The LSM method is to calculate the expected condition in formula (3.2.7). For example, if this conditional expectation function belongs to the Hilbert space L^2, the value of continuing to hold the option $F(\omega, t_{k-1})$ can be expressed as follows:

$$F(\omega, t_{k-1}) = \sum\nolimits_{j=0}^{\infty} a_j L_j(X) \tag{3.2.8}$$

Where is X a Markov process, a_j is a constant and L_j is a set of basic functions. In practical applications, the infinite-dimensional space will not be discussed. The usual choice is based on the previous M basis functions to calculate $F_M(\omega, t_{k-1})$ instead of $F(\omega, t_{k-1})$. The statistical estimate $\breve{F}_M(\omega, t_{k-1})$ can be calculated by the $C(\omega, s; t_{k-1}, T)$ through a mapping or regression. In the following use, weighted Lagrange polynomials will be used as regression basis functions.

$$L_0(X) = \exp(-\frac{X}{2})$$

$$L_1(X) = \exp(-\frac{X}{2})(1-X)$$

$$L_2(X) = \exp(-\frac{X}{2})(1-2X+\frac{X^2}{2})$$

$$\cdots$$

$$L_n(X) = \exp(-\frac{X}{2})\frac{e^X}{n!}\frac{d^n}{dX^n}(X^n e^{-X}). \tag{3.2.9}$$

Once the function $F_M(\omega, t_{k-1})$ is determined, the coefficients before each basis function are determined accordingly. From this, the value of $\breve{F}_M(\omega, t_{k-1})$ can be calculated and compared immediately with the gain of the execution of the option and make the decision on whether to exercise American options here. Then continue to iterate until the initial moment to find an optimal execution moment. Then discount it to get the value of the option.

The LSM algorithm provides a simple and elegant way of approximating the optimal early exercise strategy for an American-style option. While the ultimate test of the algorithm is how well it performs using a realistic number of paths and basic functions, it is also useful to examine what can be said about the theoretical convergence of the algorithm to the true value $V(X)$ of the American option.

The first convergence result addresses the bias of the LSM algorithm and is applicable even when the American option is continuously exercisable.

Proposition 1. For any finite choice of M, K and $vector\theta \in R^{M\times(K-1)}$ representing the coefficients for the M basis functions at each of the $K-1$ early exercise dates, let $LSM(\omega; M, K)$ denote the discounted cash flow resulting from following the LSM rule of exercising when the immediate exercise value is positive and greater than or equal to $\widehat{F_M}(\omega_i; t_k)$ as defined by θ. Then the following inequality holds almost surely,

$$V(X) \geq \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} LSM(\omega_i; M, K) \,. \qquad (3.2.10)$$

The intuition for this result is easily understood. The LSM algorithm results in a stopping rule for an American-style option. The value of an American-style option, however, is based on the stopping rule that maximizes the value of the option; all other stopping rules, including the stopping rule implied by the LSM algorithm, result in values less than or equal to that implied by the optimal stopping rule.

This result is particularly useful since it provides an objective criterion for convergence. For example, this criterion provides guidance in determining the number of basic functions needed to obtain an accurate approximation; simply increase W until the value implied by the LSM algorithm no longer increases. This useful and important property is not shared by algorithms that simply discount back functions based on the estimated continuation value.

By its nature, providing a general convergence result for the LSM algorithm is difficult since we need to consider limits as the number of discretization points K, the number of basic functions M, and the number of paths N go to infinity. In addition, we need to consider the effects of propagating the estimating stopping rule backwards through time from t_{K-1}, to t_1. In the case where the American option can only be exercised at $K = 2$ discrete points in time, however, convergence of the algorithm is more easily demonstrated. As an example, consider the following proposition.

Proposition 2. Assume that the value of an American option depends on a single state variable X with support on $(0, \infty)$ which follows a Markov process. Assume further that the option can only be exercised at times t_1, and t_2, and that the conditional expectation function $F(\omega; t_1)$ which is absolutely continuous and

$$\int_0^\infty e^{-X} F^2(\omega; t_1)\, dX < \infty$$

$$\int_0^\infty e^{-X} F_X{}^2(\omega; t_1)\, dX < \infty$$

Then for any $\in > 0$, there exists an $M < \infty$ such that

$$\lim_{N\to\infty} \Pr\left[\left| V(X) - \frac{1}{N}\sum_{i=1}^{N} LSM(\omega_i; M, K)\right| > \in \right] = 0 \qquad (3.2.11)$$

Intuitively this result means that by selecting M large enough and letting $N \to \infty$, the LSM algorithm results in a value for the American option within $\in$ of the true value. Thus the LSM algorithm converges to any desired degree of accuracy since $\in$ is arbitrary. The key to this result is that the convergence of $F_M(\omega, t_1)$ to $F(\omega; t_1)$ is uniform on $(0,\infty)$ when the indicated integrability conditions are met. In summary the American put option can be written as

$$PutAme(0,S_{0,}K,T)=PutEur(0,S_{0,}K,T)E[K-S_{t^*}e^{-rt^*}-PutEur(t^*,S_{t^*},K,T)e^{-rt^*}]$$

(3.2.12)

For technical reasons, programming here is performed by R.

3.2.2 Binomial tree method

The basic principle of the binomial tree method is: Assume that the probability and magnitude of the motion of the target variable only move up or down. It is also assumed that the probability and magnitude of each upward or downward movement of the target variable does not change throughout the investigation period. Divide the period into several stages. According to the historical volatility of the target variable, we simulate all possible development paths of the target variables in the whole inspection period, the price of the 0 moment is obtained at the same time by the discounting method. If face the problem of advance exercise, it is necessary to check at each node of the binomial tree to see if it is more advantageous than the next node on this point, and then repeat the process.

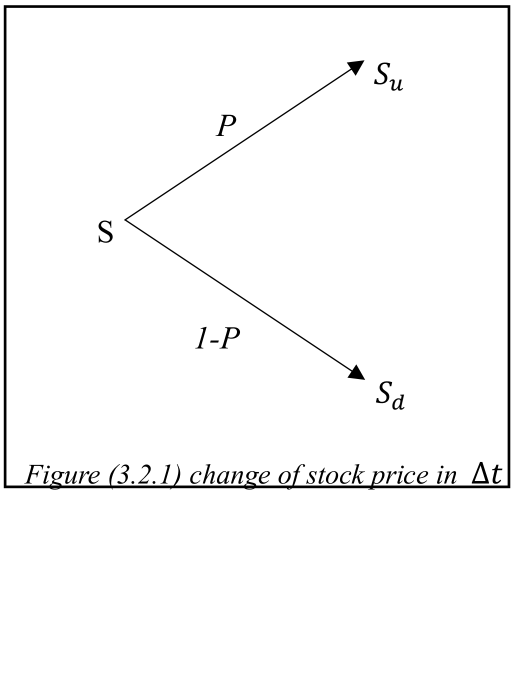

Figure (3.2.1) change of stock price in Δt

Consider a stock option that does not pay dividends. We divide the period of maturity of the option into many small time intervals, each of the intervals is Δt. Assume that in each interval the stock price changes from the beginning of S to two new prices S_u and S_d, and also assume $u > 1$, $d < 1$, so S to $\mathrm{S_u}$ is a process of rising prices, and the probability of rising is P; S to $\mathrm{S_d}$ is a process of falling prices, and the probability of rising is $1 - P$.

In a risk neutral world, the expected return rate of stock is risk free rate r. Then the expected value of the stock price at the end of the time interval Δt is $Se^{r\Delta t}$ where S is the initial stock price of the time interval. So, we have a result:

$$Se^{r\Delta t} = PSu + (1-P)Sd$$

which can be written as:

$$e^{r\Delta t} = Pu + (1-P)d \,. \tag{3.2.13}$$

Since the previous hypothesis is the behavior model of the stock price, the variance of stock price change in the time interval $\Delta \mathrm{t}$ is $S^2\sigma^2\Delta t$. According to the definition of variance, the variance of the variable X is equal to $\mathrm{E}(X^2) - [E(X)]^2$. So, then

$$\sigma^2\Delta t = Pu^2 + (1-P)d^2 - \left[Pu + (1-P)d\right]^2 \tag{3.2.14}$$

The formula (3.2.7) and (3.2.8) provide two conditions and the third condition

$$u = \frac{1}{d} \,, \tag{3.2.15}$$

So, we get the result:

$$P = \frac{a-d}{u-d} \tag{3.2.16}$$

$$u = e^{\sigma\sqrt{\Delta t}} \tag{3.2.17}$$

$$d = e^{-\sigma\sqrt{\Delta t}} \tag{3.2.18}$$

where

$$a = e^{r\Delta t} \,. \tag{3.2.19}$$

From formula (3.2.9) to (3.2.13), the tree structure of the stock price can be constructed, which is called the binomial tree of the stock. As shown in figure (3.2.2). In the picture, the stock price of the 0 moment is S, and at Δt time, there are two possibilities for the stock price: S_u and S_d; at $2\Delta t$ time, there are three possibilities for the stock price Su^2, S_{ud} and Sd^2. By analogy, in general, in the $i\Delta t$ moments, the price of the stock are $i+1$ possibilities,

$$Su^j d^{i-j}, j = 0,1,2,...,i \qquad (3.2.20)$$

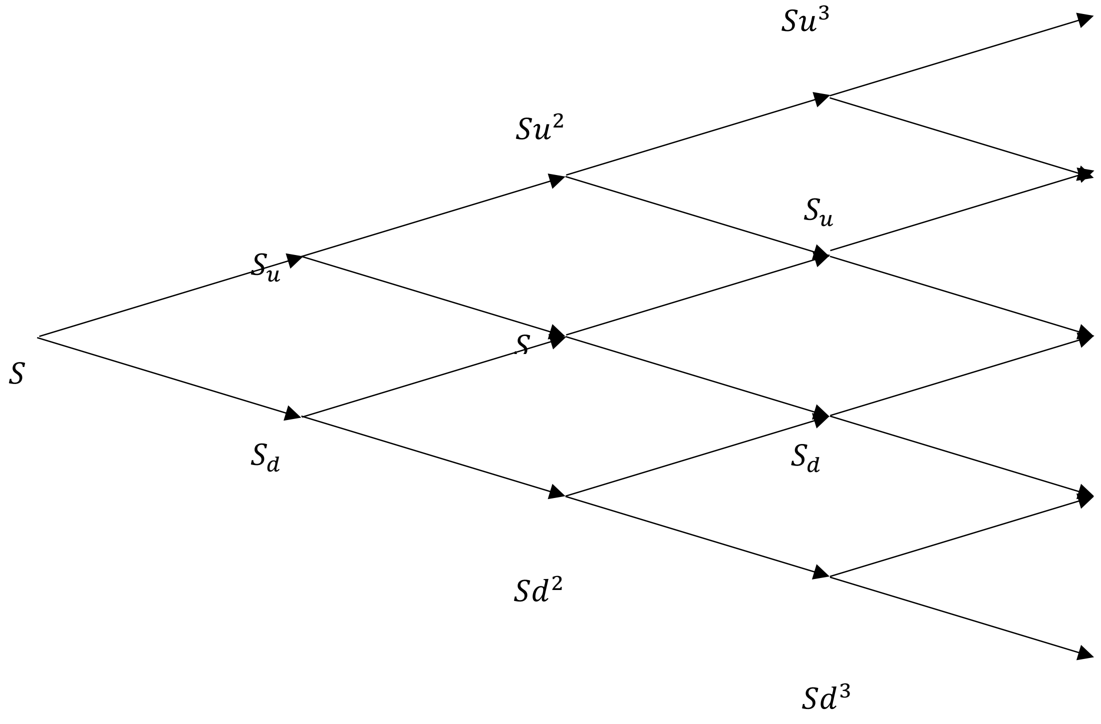

Figure 3.6.2 Binomial tree of stock price

Assumed that the period of an American put option that does not pay dividends is divided into a small time period of *N* length of Δt. Suppose f_{ij} is the option price of the stock price of $Su^j d^{i-j}, (0 \leq i \leq N, 0 \leq j \leq i)$ at $\mathrm{i}\Delta\mathrm{t}$ moment, also known as the option price at node (i,j). Because the price of American put option at maturity is $max\{X - S_T, 0\}$, so

$$f_{ij} = \max\left\{X - Su^j d^{N-i}, 0\right\}, j = 0,1,\ldots,N \qquad (3.2.21)$$

Assumed that the probability of moving the node $(i+1, j+1)$ from the node (i,j) to the $(i+1)\Delta t$ moment at the $i\Delta t$ moment is P; the probability of moving the node $(i+1, j)$ from the node (i,j) to the $(i+1)\Delta t$ moment at the $i\Delta t$ moment is $(1-P)$. Without exercise in advance and in the risk neutral world, the price of the option is

$$f_{ij} = e^{-r\Delta t}\left[Pf_{i+1,j+1} + (1-P)f_{i+1,j}\right], 0 \le i \le N-1, 0 \le i \le j. \qquad (3.2.22)$$

If consider the exercise in advance, f_{ij} must be compared to the intrinsic value of the put option

$$f_{ij} = \max\left\{X - Su^j d^{j-i}, e^{-r\Delta t}\left[Pf_{i+1,j+1} + (1-P)f_{i+1,j}\right]\right\} \qquad (3.2.23)$$

According to the above basic principles and analytical expressions, we get the basic steps of the binomial tree method:

1. Divide the validity time of the derivative securities into N equal interval time periods, step length is Δt. So we need to consider $N+1$ time points: $0, \Delta t, 2\Delta t, \ldots, T$.
2. Calculating the parameters P, u and d of the binomial tree.
3. Construction of binomial tree.
4. Calculating the price of option by discount binomial trees.

American option has a problem of exercise in advance, therefore, on the basis program of the above European option pricing procedure, the statement of checking exercise in advance is needed.

3.2.3 Finite-difference method

In most cases, it is almost impossible to require an exact solution of a partial differential equation. At this time it is necessary to use the finite difference approximation. The basic idea of the difference method is to replace the partial derivative in a partial differential equation by Taylor expansion in a certain point. According to the definition:

$$\frac{\partial u}{\partial \tau}(x,\tau)=\lim_{\delta\tau\to 0}\frac{u(x,\tau+\delta\tau)}{\delta\tau}$$

Now $\delta\tau$ is not regarded as a variable that tends to 0, but as a small amount which greater than 0, an approximate. We can obtain an approximate

$$\frac{\partial u}{\partial \tau}(x,\tau)\approx\frac{u(x,\tau+\delta\tau)-u(x,\tau)}{\delta\tau}+O(\delta\tau) \qquad (3.2.24)$$

This is called the finite difference approximation of $\frac{\partial u}{\partial \tau}$. The smaller the time interval, the more accurate the approximation is. What considered here is the time change from τ to $\tau+\delta\tau$, often referred to as the forward difference.

If do the following approximation:

$$\frac{\partial u}{\partial \tau}(x,\tau)\approx\frac{u(x,\tau)-u(x,\tau-\delta\tau)}{\delta\tau}+O(\delta\tau) \qquad (3.2.25)$$

then it is called the backward difference.

Also, the central difference can be defined as

$$\frac{\partial u}{\partial \tau}(x,\tau)\approx\frac{u(x,\tau+\delta\tau)-u(x,\tau-\delta\tau)}{2\delta\tau}+O((\delta\tau)^2). \qquad (3.2.26)$$

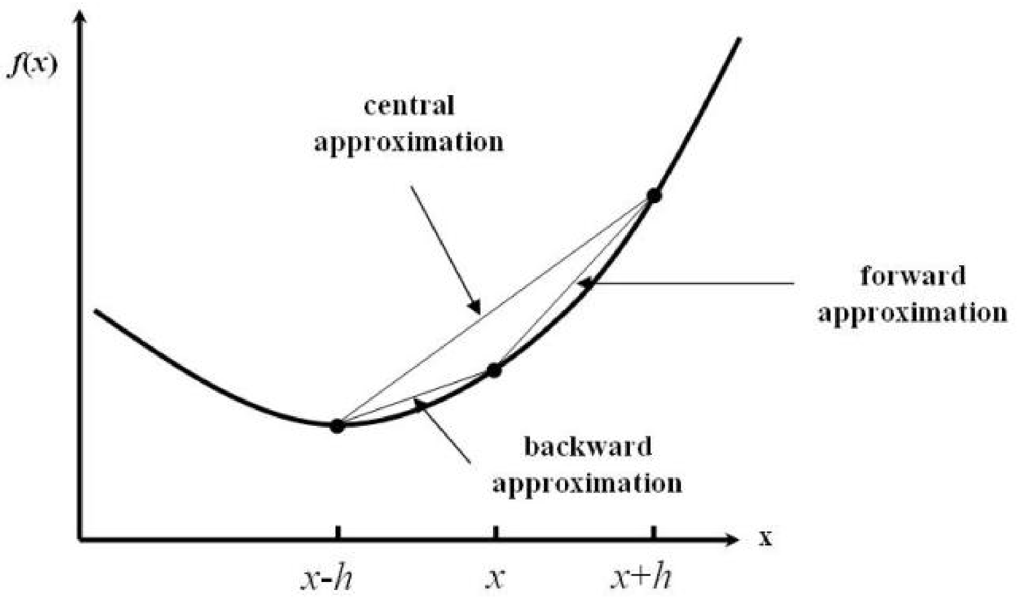

Figure 3.2.3 Forward, backward and central difference

When applied to the diffusion equation, the forward difference approximation leads to the explicit difference method, and the backward difference approximation leads to the full implicit difference method. The center difference approximation shown in formula (3.2.20) is rarely used because it often causes bad behavior in the solution process. In the commonly used Crank-Nicolson difference method, the central difference defined by the next formula

$$\frac{\partial u}{\partial \tau}(x,\tau) \approx \frac{u(x,\tau+\delta\tau/2)-u(x,\tau-\delta\tau/2)}{2\delta\tau} + O((\delta\tau)^2). \qquad (3.2.27)$$

In the same way, for x, the central difference approximate of one partial derivative is

$$\frac{\partial u}{\partial x}(x,\tau) \approx \frac{u(x+\delta\tau,\tau)-u(x-\delta\tau,\tau)}{2\delta x} + O((\delta x)^2). \qquad (3.2.28)$$

And the symmetric center difference of the second derivative is

$$\frac{\partial^2 u}{\partial x^2}(x,\tau) \approx \frac{u(x+\delta\tau,\tau)-2u(x,\tau)-u(x-\delta\tau,\tau)}{(\delta x)^2} + O((\delta x)^2).$$

(3.2.29)

The difference method is equivalent to dividing the x axis into a space segment with an equidistance of δx and the τ axis into a time interval with $\delta\tau$ as an equidistance. Thus, the (x,τ) plane is divided into a grid.

Considering a stock option that does not pay dividends, the partial differential equation of the option price is (3.1.8). Suppose it's the 0 moment at present, we divide time from 0 to the expiration date T into N interval time intervals, each step is $\Delta t = T/N$, so there is a total of $N+1$ time points.

$$0, \Delta t, 2\Delta t, 3\Delta t, \ldots, T.$$

Assuming S is the maximum value that the stock price can reach, the price step is defined as $\Delta S = S/M$, and M is a given price step, so there is a total of $M+1$ price points.

$$0, \Delta s, 2\Delta s, 3\Delta s, \ldots, S.$$

The above price points and time points form a grid of $(M+1) \times (N+1)$ coordinate points. For any point (i,j) in the grid the corresponding time is $i\Delta t$ and the stock price is $j\Delta S$.

We use f_{ij} to indicate the option price of point (i,j), in this way, we can use discrete operators to approach $\frac{\partial f}{\partial t}, \frac{\partial f}{\partial S}, \frac{\partial^2 f}{\partial S^2}$ so the partial differential equation is converted into a discrete equation.

By performing differential processing on the Black-Scholes partial differential equation, we can derive the expression of the explicit finite difference method.

$$a_j f_{i+1,j-1} + b_j f_{i+1,j} + c_j f_{i+1,j+1} = f_{ij}$$

(3.2.30)

where

$$a_j = \frac{1}{1+r\Delta t}\left(-\frac{1}{2}rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right),$$

$$b_j = \frac{1}{1+r\Delta t}(1-\sigma^2 j^2 \Delta t),$$

$$c_j = \frac{1}{1+r\Delta t}\left(\frac{1}{2}rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right).$$

Next we will use the explicit finite difference method to solve the American option. First of all we need to set the key boundary conditions.

For the American call option:

The value of the option at expiration date is $max\{S_T - X, 0\}$, in which S_T is the stock price at the time of T. So

$$f_{Nj} = \max\{j\Delta S - X, 0\}.$$

When the stock price is 0, the price of the call option is 0. So

$$f_{i0} = 0, i = 0,1,2,...,N;$$

When the stock price is $S = S_{max}$, the price of the call option is S_{max}. So

$$f_{iM} = S_{\max}, i = 0,1,2,...,N.$$

For the American put option:

The value of the option at expiration date is $max\{X - S_T, 0\}$, in which S_T is the stock price at the time of T. So

$$f_{Nj} = \max\{X - j\Delta S, 0\}.$$

When the stock price is 0, the price of the call option is X. So

$$f_{i0} = X, i = 0,1,2,\dots,N;$$

When the price of stock tends to infinity, the price of the put option is 0. So

$$f_{iM} = 0, i = 0,1,2,\dots,N.$$

From the key boundary conditions of American option we can do the program. See appendix (Program 9).

Because of the existence of the rounding error in the explicit finite difference method, so sometimes the solution of the difference equation does not converge to the solution of the partial differential equation. To solve this problem we will introduce the implicit finite difference method. It can solve more X nodes at the same time step. Through the differential treatment of the Black Scholes partial differential equation, we can get the expression of the implicit finite difference method.

$$a_j f_{i,j-1} + b_j f_{ij} + c_j f_{i,j+1} = f_{i+1,j}, \tag{3.2.31}$$

where

$$a_j = \frac{1}{2} rj\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t,$$

$$b_j = 1 + \sigma^2 j^2 \Delta t + r\Delta t,$$

$$c_j = \frac{1}{1+r\Delta t}\left(\frac{1}{2} rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right).$$

In the implicit finite difference method, the calculation of $f_{i,j}$ by $f_{i+1,j}$ needs to solve $M+1$ equations at the same time the amount of calculation is very large. So the matrix library needs to be introduced in the program. The details are showing in appendix (Program 10)

Similar with the explicit finite difference method, we use the same key boundary conditions in the implicit finite difference method. So we can apply the program.See appendix (Program 11)

In addition, the Crank-Nicolson finite difference method can also be used, which is essentially the mean of explicit and implicit finite difference methods. In this thesis we do not introduce too much about this method.

Chapter 4. Evaluation of selected method

In this chapter we will use the method introduced in the last chapter to calculate the price of options. The paper uses data download from the sample data base of Chicago Board of Trade. First, we will calculate the analytical solution of the Black-Scholes model. In the programs we set *S* as the price of underlying asset which means the stock price of this option, *X* as the exercise price means the price at which an underlying security can be purchased (call option) or sold (put option). *r* as the risk free rate which is the rate of return of a hypothetical investment with no risk of financial loss, over a given period of time. *sigma* as the implied volatility which is the estimated volatility, or gyrations, of a security's price and is most commonly used when pricing options. In general, implied volatility increases while the market is bearish, when investors believe the asset's price will decline over time, and decreases when the market is bullish, when investors believe that the price will rise over time. This is due to the common belief that bearish markets are riskier than bullish markets. Implied volatility is a way of estimating the future fluctuations of a security's worth based on certain predictive factors. Implicit volatility is usually calculated by the stock price, but the data collected in this paper already contains the implied volatility, so it is no needed to calculate. In the mathematical sense, the movement of financial asset prices is random, and the volatility reflects the volatility of this stochastic path. It describes the statistical distribution characteristics of asset returns and is usually represented by the standard deviation of asset returns. And *t* as the period of rights. The programs described in appendix can only calculate the price of one option once a time. In order to allow the program to continuously calculate option prices in a loop, we design the following loops. The column data shown in this chapter is only the first six groups of sample data.

4.1 The analytical solution of Black-Scholes model

After finishing the programming of the loop we can calculate the options price. The first method we apply is the analytical solution of B-S partial differential equation.

Call option

S	X	r	sigma	time	Black-Scholes
14.575	10	0.21	1.4094	0.02	4.64252
14.575	10.5	0.21	1.2383	0.02	4.14305
14.575	11	0.21	1.0838	0.02	3.64458
14.575	11.5	0.21	0.9506	0.02	3.14833
14.575	12	0.21	0.7994	0.02	2.649
14.575	12.5	0.21	0.6621	0.02	2.15129

Put option

S	X	r	sigma	time	Black-Scholes
14.575	10	0.21	2.2815	0.02	0.311095
14.575	10.5	0.21	1.8378	0.02	0.152188
14.575	11	0.21	1.5005	0.02	0.11019
14.575	11.5	0.21	1.2454	0.02	0.0890074
14.575	12	0.21	0.9886	0.02	0.0645449
14.575	12.5	0.21	0.7804	0.02	0.0509957

Table 4.1 Analytical solutions of the Black-Scholes

The above example tables show the analytical solutions of the Black-Scholes partial differential equation obtained by program (2.5.2), (2.5.3). After checking with Excel, we can conclude that the computational accuracy of C++ meets the research needs. At the same time the calculation is very fast compare to the using of Excel. The specific data is in appendix (table 1). Accurately speaking, volatility describes the degree of fluctuation of financial asset prices, which is a measure of the uncertainty of asset returns and is commonly used to reflect the level of risk of financial assets. The higher the volatility, the greater the volatility of financial asset prices and the greater the uncertainty of asset returns. The lower the volatility, the smoother the fluctuation of the financial asset price, and the stronger the certainty of asset returns. From the table above we get when the exercise price rises and the implied volatility falls, the estimated price of either the call or the put option will decline.

4.2 The numerical solution of Black-Scholes model

Because there are many quite complex derivative securities that cannot give analytical solutions we will use the numerical method of option pricing to calculate the numerical solution of the option price. Normally we think the numerical solution is a value calculated by approximate calculation under certain conditions and the analytical solution is the analytic formula of the function, and any corresponding value can be calculated from the expression of the solution. We usually think that analytical solutions are more accurate than numerical solutions. Next we will perform sensitivity analysis on numerical methods, then compare it with the analytical solution of partial differential equations.

4.2.1 The Monte Carlo method

First we will start on the Monte Carlo Method, and we will use the different simulation times to do the sensitivity analysis to evaluate the accuracy of numerical solutions derived from Monte Carlo simulations.

European Call option

B-S	MC50	MC1000	MC10000	MC50000
4.64252	4.65235	4.65235	4.65235	4.65235
4.14305	4.18337	4.05996	4.1538	4.15782
3.64458	3.56816	3.70136	3.6403	3.64479
3.14833	3.08868	3.1357	3.14918	3.15287
2.649	2.72959	2.64618	2.64103	2.65809
2.15129	2.15344	2.17803	2.15466	2.15032

Table 4.2.1 European call option Black-Scholes numerical solution in Monte Carlo

It can be seen from the data that the more the number of simulation, the smaller the deviation and the analytical solution of the numerical solution. But we can't get a complete and accurate conclusion from a small number of samples. Here we calculate the price of the put option. Then calculate and compare the average error values and maximum error of different simulate steps. Which the error means the absolute value of the difference between the numerical solution and the analytical solution. The average error is the mean of absolute value error and the maximum error is the maximum of absolute value error (the maximum value of the absolute value of the difference between the analytical solution and the numerical solution).

European put option

B-S	MC50	MC1000	MC10000	MC50000
0.311095	0.2137	0.242663	0.219319	0.225429
0.152188	0.277804	0.145098	0.15777	0.152949
0.11019	0.134721	0.100421	0.112334	0.108977
0.0890074	0.119097	0.0903439	0.0902059	0.088398
0.0645449	0.0981901	0.0484329	0.0650068	0.0630125
0.0509957	0.0821978	0.0490903	0.052366	0.050724

Table 4.2.2 European put option Black-Scholes numerical solution in Monte Carlo

The specific data is in appendix (table 2). By observing the above data, it is found that the error of the numerical solution will be significantly reduced when the number of simulation is increased. The numerical solution given by Monte Carlo method gradually approximated the analytical solution given by the option pricing formula. However, the computing speed of the C++ program also drops dramatically at same time.

Call option	50 steps	1000steps	10000steps	50000steps
Maximum error	0.07642	0.08309	0.00797	0.00526
Mean	0.006115083	0.003010634	-0.003130817	-0.000473584
Put option	50 steps	1000steps	10000steps	50000steps
Maximum error	0.3973	0.0965	0.091776	0.085666
Mean	0.017662173	0.008184492	0.001335953	0.001660199

Table 4.2.3 Maximum error and mean of error of Monte Carlo method

Analysis of data through Microsoft Excel can get the conclusions that the calculation accuracy increases with the number of simulated times, because the average error is significantly reduced and the maximum absolute error is significantly reduced. Through the sensitivity analysis, we can predict that the error of the numerical solution and the analytical solution can be ignored when the Monte Carlo simulation step trends to be an infinite value.

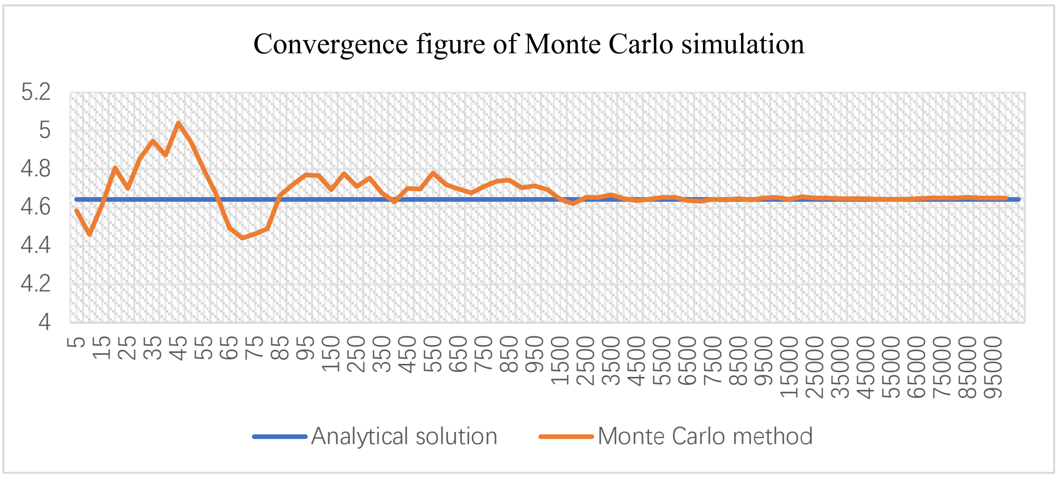

Figure 4.2.1 Convergence figure of Monte Carlo simulation

The above figure shows the convergence of the Monte Carlo simulation method. It reflects the difference between the numerical solution and the analytical solution as the number of simulations increases. It can be seen from the figure that when the number of simulations is less than 1500 times, the difference between the numerical solution and the analytical solution fluctuates greatly, but with the increase of the number of simulations, the fluctuation gradually becomes stable. When the number of simulations is greater than 25,000, the numerical solution is basically stable at or very close to the analytical solution.

4.2.2 The binomial tree method

Next we will use the binomial tree method to calculate the numerical solution of the option price. The binomial tree method is similar to the Monte Carlo method which Simulate the possible path of the price and then discount it. Same as the previous steps, first introduce Loop (4.1) to solve the loop calculation problem, then perform multiple sets of calculations to detect sensitivity of binomial tree method.

European call option

Black-Scholes	BinTree50	BinTree1000	BinTree10000	BinTree50000
4.64252	4.64235	4.64255	4.6426	4.6426
4.14305	4.14293	4.14304	4.14305	4.14305
3.64458	3.64457	3.64452	3.64458	3.64458
3.14833	3.14831	3.1483	3.14833	3.14833
2.649	2.64896	2.64897	2.64899	2.649
2.15129	2.15128	2.15125	2.15128	2.15129

Table 4.2.4 European call option Black-Scholes numerical solution in binomial tree

Through observation, it can be found that the higher the number of binary tree steps, the closer the numerical solution to the analytical solution. Also, because the sample size is too small, the price of the put option needs to be calculated for analysis. Next we will use the binomial method to calculate the price of a put option.

European put option

Black-Scholes	BinTree50	BinTree1000	BinTree10000	BinTree50000
0.311095	0.222508	0.225031	0.224938	0.224932
0.152188	0.153442	0.152029	0.152176	0.152185
0.11019	0.110673	0.110048	0.110191	0.110189
0.0890074	0.0889809	0.0890055	0.0890071	0.0890078
0.0645449	0.063288	0.0644442	0.0645472	0.0645451
0.0509957	0.0499729	0.0509105	0.0509972	0.0509958

Table 4.2.4 European put option Black-Scholes numerical solution in binomial tree

Comparing the numerical solutions between the analytic solution of the partial differential equations and the binomial tree method, it can be found that as the number of binomial tree steps increases, the numerical solution will more closely approximate the analytical solution. However, because the binary tree method is essentially an exhaustive method, and you need to check on each node whether the exercise is more favorable than exercise in the next node. This will consume a lot of Central Processing Unit (CPU) power count. So, as the number of steps increases, the efficiency drops significantly. The C++ program does not reduce computing time when large amounts of data require large number of steps. Because the binomial tree model itself does not involve very complicated calculation formulas. Therefore, if using the binomial tree method to calculate option prices, it is recommended to use a more computationally intensive Graphics Processing Unit (GPU). At the same time, compared to the Monte Carlo model described above, binomial tree model is more accurate and more approximates the value of the analytical solution.

call option	50 steps	1000steps	10000steps	50000steps
Maximum error Mean	0.00100 0.000125	0.00050 0.000071	0.00003 0.000002	0.00001 0.000000
put option	50 steps	1000steps	10000steps	50000steps
Maximum error Mean	0.088587 0.001273	0.086064 0.001124	0.086157 0.001117	0.086163 0.001119

Table 4.2.5 Maximum error and mean of error of binomial tree method

From the table 4.2.2 above. In call option pricing, the accuracy of the binomial tree method is very high and can be further increased with the increase in the number of steps. However, when the number of steps exceeds 50,000 steps, the efficiency is greatly reduced but the accuracy improved little. Therefore, in the calculation of pricing, the number of simulation steps should be less than 50,000 or even less than 20,000. At this time, we can still obtain very high accuracy. In the calculation of the put option price, there is a slight deviation in the C++ program. It does not come to the result that the expected precision gradually increases with the increase of the number of steps. At this point the accuracy increases into the bottleneck period but the calculation efficiency still decreases as the number of steps increases. After a more detailed calculation and sensitivity analysis, the put option price calculation in binomial tree method step should be less than 15000 steps in order to achieve a balance between accuracy and efficiency. It is precisely because of the extremely high accuracy of the binomial tree method, and it is possible to add a code to check the advance exercise in the program so the binomial tree method can be used to calculate the price of American options which will be apply in the following after the analysis of numerical Solution of European Options.

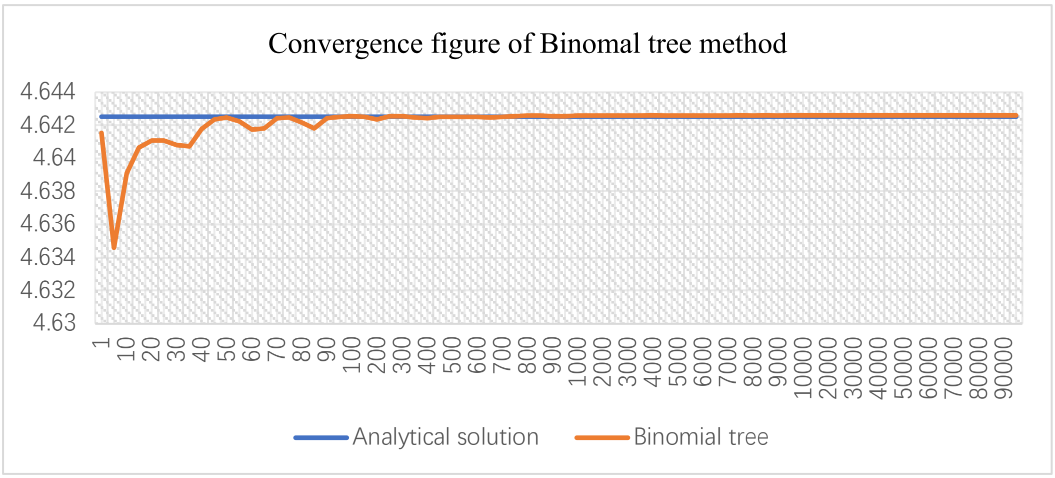

Figure 4.2.2 Convergence figure of Binomial tree method

Figure (4.2.2) is a convergence plot for the binomial tree method, which reflects the error between the numerical solution of the Black-Scholes partial differential equation solved by the binomial tree method and the analytical solution. It can be seen from the figure that the fluctuation of the error has stabilized at 90 steps, and the numerical solution calculated by the binomial tree method after 450 steps is already very close to the analytical solution. It can be said that the error of the binomial tree method is very small.

4.2.3 The finite difference method

The Black-Scholes partial differential equation numerical solution is not limited to the Monte Carlo method and the binomial tree method, but also includes the finite difference method. Next, we will use the finite difference method to calculate the numerical solution of BS partial differential equations.

At first we will apply the explicit finite difference method in European option pricing. The calculation of explicit difference method is relatively simple. Adding loop (4.1) to the main program it is easy to get the conclusion. Here we assume that the steps of the prices is equal to the time step, so it is convenient for the sensitivity analysis.

Call option

Black-Scholes	explicit10 10	explicit20 20	explicit200 200	explicit2000 2000
4.64252	4.6676	4.64012	1.73E+164	nan
4.14305	4.16763	4.14979	1.06E+140	nan
3.64458	3.66051	3.65428	2.85E+114	nan
3.14833	3.15274	3.15095	2.61E+88	nan
2.649	2.68306	2.66202	1.13E+52	nan
2.15129	2.20735	2.16505	2.20E+08	nan

Table 4.2.6 European call option s numerical solution in explicit finite difference

Where nan means the program can not calculate the result and inf means the conclusion trends to infinite. It can be seen directly from the results that when the number of simulated steps is relatively small, the numerical solution is similar to the analytic solution. The accuracy of the numerical solution at this time is acceptable. However, with the increase of the number of simulation steps, the numerical solutions have overflow. Because the calculation process of the difference scheme is pushed by layer by layer, the approximate value of the N layer is used in the calculation of the approximate value of the N + 1 layer, until it is related to the initial value. If there are rounding errors in the preceding layers, it will inevitably affect the values of the latter layers. If the influence of errors is bigger and bigger, the appearance of the exact solutions of the difference schemes will be completely concealed. This is why the numerical solution overflow. So in this case we think that explicit finite difference method is unstable. The same result will also appear in the pricing of the put option.

Put option

Black-Scholes	explicit 10 10	explicit 20 20	explicit 200 200	explicit 2000 2000
0.311095	0.271463	0.224174	3.63E+188	nan
0.152188	0.190018	0.157399	-1.38E+151	nan
0.11019	0.125888	0.121602	-6.57E+115	nan
0.0890074	0.0807608	0.08757	1.99E+81	nan
0.0645449	0.0965918	0.0769244	3.18E+34	nan
0.0509957	0.113626	0.0658148	5.10E-02	nan

Table 4.2.7 European put options numerical solution in explicit finite difference

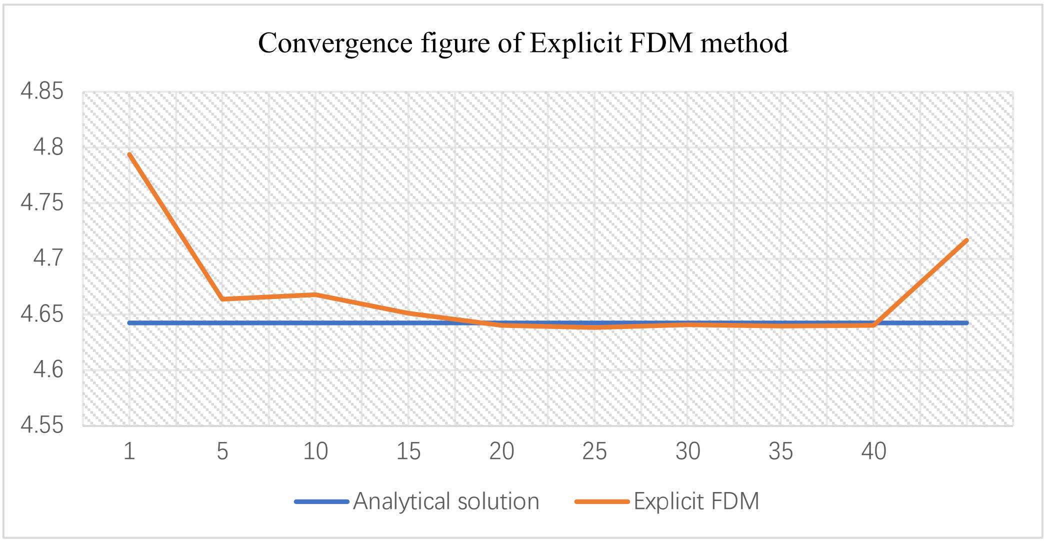

Figure 4.2.3 Convergence figure of Explicit FDM method

The converging graph shown in the above figure has an overflow effect when the calculated time steps and price steps is 41 steps, because the error will accumulate as the number of steps increases, so only the data before 41 steps are plotted. It can be clearly seen from Figure 4.2.3 that the analytical solution of the explicit finite difference method is more volatile than other methods.

By the price of the call and put options, we conclude that the calculation of explicit difference method will have overflow effects when the number of simulated steps is too large, so the method is unstable for calculate. The maximum and mean value of the error can not be compared at this time. Because under the limited number of steps, the explicit difference method calculation precision can not reach the requirement we turn to use implicit difference method to calculate. First we will calculate the European put option.

Put option

Black-Scholes	implicit 10,10	implicit 20,20	implicit 100, 100	implicit 200, 200
0.31110	0.31234	0.29999	0.22543	0.22494
0.15219	0.13253	0.13108	0.15237	0.15154
0.11019	0.07891	0.09889	0.11055	0.11034
0.08901	0.05985	0.08102	0.08907	0.08919
0.06454	0.04548	0.06061	0.06372	0.06439
0.05100	0.03596	0.04794	0.05032	0.05087

Table 4.2.8 European put option s numerical solution in implicit finite difference

From the put option calculation, implicit method has a good precision and still has no overflow when the number of simulated steps is 200. Now we will calculate the call option to do more analysis.

Call option

Black-Scholes	implicit10 10	implicit20 20	implicit100 100	implicit200 200
4.64252	4.65235	4.64235	4.64212	4.64235
4.14305	4.13678	4.14098	4.14292	4.14291
3.64458	3.64055	3.642	3.64424	3.64425
3.14833	3.14701	3.14579	3.14809	3.14833
2.649	2.64873	2.64786	2.649	2.64885
2.15129	2.15184	2.15139	2.15109	2.1513

Table 4.2.9 European call option s numerical solution in implicit finite difference

By analyzing the above calculation results, we can conclude that implicit difference method can achieve quite high accuracy at less simulated steps. At the same time, unlike explicit difference method, implicit difference method can eliminate the limit of stability. It means in the same time steps, more nodes can be solved if used the implicit difference method. The shortcomings of implicit difference method are also obvious. Because the matrix is introduced in the calculation, the calculation is very cumbersome. In the writing of C++ programs, the problem is debug for the external matrix library, at present, two widely used matrix libraries are NEWMAT and IT++ and this thesis also introduces the two matrix Libraries.

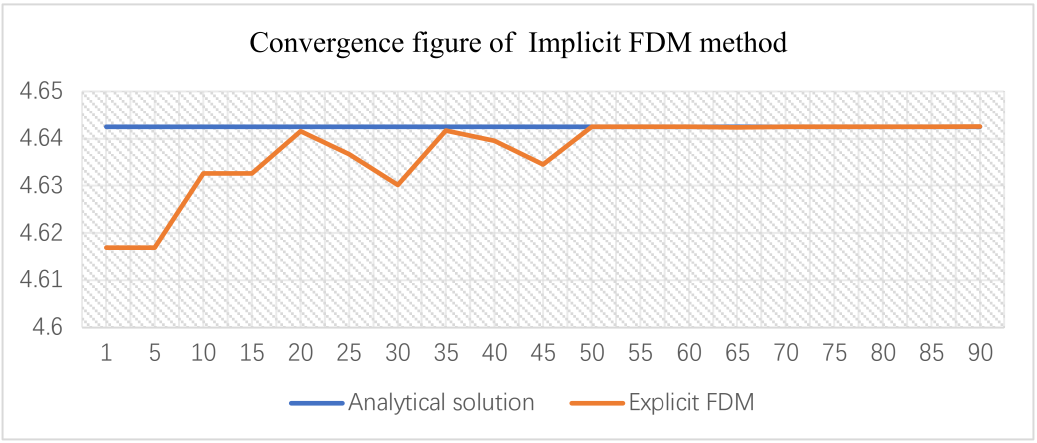

Figure 4.2.4 Convergence figure of Implicit FDM method

From the above figure, we can see that when the number of simulations is less than 50 steps, that is, the number of time steps and price steps is less than 50 steps, the numerical solution fluctuates greatly. When the number of simulation steps is greater than 50 steps, the numerical solution approaches the analytical solution.

Next we will perform sensitivity analysis on the infinite difference method. Compare the average error and the maximum error between the analytical solution and the numerical solution.

call option	implicit 10	implicit 20	implicit 100	implicit 200
Average error Max error	0.0056 0.0455	0.0026 0.0218	0.0004 0.0040	0.0002 0.0023
put option	implicit 10	implicit 20	implicit 100	implicit 200
Average error *Max error*	0.0022 *0.0655*	0.0025 *0.0456*	0.0010 *0.0857*	0.0011 *0.0862*

Table 4.2.10 Maximum error and mean of error of finite difference method

From the maximum error and the average error, the error of implicit difference method is fully conformed to the requirements. This thesis only focuses on full implicit difference and explicit difference methods. There are also the semi implicit difference and the central difference method in the finite difference method. In the paper, we will not to say more about it. What needs to be mentioned is the mean of explicit difference and implicit difference called Crank-Nicolson method. In general, the Crank-Nicolson method is stable when the explicit method is overflow, and is more accurate than the implicit method. This method has not been realized in this paper for technical reasons. Through the above three numerical methods of calculation and analysis. And after comparing with the Black-Scholes partial differential equation's analytical solution. We can conclude that as the number of calculations increases, the accuracy of the numerical solution increases. However, due to the accumulation of errors, the explicit finite difference method does not meet the above conclusions. At the same time, because the binomial tree method is a computational approximation similar to the exhaustive method. So the binomial tree method has the highest accuracy. But at the same time, because the calculation is too cumbersome, it leads to the lowest efficiency. In actual production and life, it is recommended to use a small number of simulations of the binomial tree method or implicit finite difference method to let the calculation accuracy and calculation efficiency reach the optimal combination.

4.3 The option pricing of American options

The difficulty of American option calculation is that due to the existence of advance exercise, the traditional Black-Scholes model cannot be used to solve the American put option price. Because the exercise date is a very important parameter in the Black-Scholes model, unlike the European option, the exercise date of the American option is more flexible. The usual calculation of American option prices can use Barone Adesi & Whaley Model (BAM model), least-squares Monte Carlo simulation(LSMC model), binomial tree method and finite difference method. Because BAM model requires the introduction of a bivariate normal distribution and the cumulative distribution function to get the approximate analytical solution.

The program design is too cumbersome, so it will not be introduced here. In the previous section, we know that the binomial method has a high accuracy, so we will first calculate the price of the American option using the binomial tree method. We assume the most exact solution is the 50,000 step simulation of the binomial tree. For comparison, here we will use the same data as the previous European options.

American call option

S	X	r	sigma	time	BinTree50000
14.575	10	0.21	1.4094	0.02	4.6426
14.575	10.5	0.21	1.2383	0.02	4.14305
14.575	11	0.21	1.0838	0.02	3.64458
14.575	11.5	0.21	0.9506	0.02	3.14833
14.575	12	0.21	0.7994	0.02	2.649
14.575	12.5	0.21	0.6621	0.02	2.15129

Table 4.3.1 American call option price

Similar to previous conclusions, the computational efficiency of the binomial tree method is very low at large simulation times. Next is the American put option pricing.

American put option

S	X	r	sigma	time	BinTree50000
14.575	10	0.21	2.2815	0.02	0.225212
14.575	10.5	0.21	1.8378	0.02	0.152414
14.575	11	0.21	1.5005	0.02	0.11039
14.575	11.5	0.21	1.2454	0.02	0.0892056
14.575	12	0.21	0.9886	0.02	0.0647286
14.575	12.5	0.21	0.7804	0.02	0.0511868

Table 4.3.2 American put option price

As we can above is the American put option price. At this time we take the same method but the European option price to compare the difference. The table below shows the specific differences between American option and the European option.

Comparison of European and American option

European call	American call	European put	American put
4.6426	4.6426	0.224932	0.225212
4.14305	4.14305	0.152185	0.152414
3.64458	3.64458	0.110189	0.11039
3.14833	3.14833	0.0890078	0.0892056
2.649	2.649	0.0645451	0.0647286
2.15129	2.15129	0.0509958	0.0511868

Table 4.3.3 Comparison between European and American option

Although the data given in Table 4.2.13 is only a small part of the complete data (Detailed data see in appendix). We can still see that there is no difference between the price of the American call option and the price of the European call option when using the same calculation method. This is because the non-dividend American call options generally do not exercise before the maturity. Strictly speaking, not only the American option, but all the "convex" interest free American call derivatives are all the sub-martingale, that is, their expectation at $T+1$ is greater than or equal to the value of T time, so they should never be exercised advance. Which is:

Define convex:

$$\forall t \in [0,1], f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2) \qquad (4.3.1)$$

The payoff $\text{Max}\,(S-K,0)$ of American call options is obviously convex. For a function with an initial value of 0 (which $\text{Max}\,(0-K,0) = 0$), the convex definition can also be "linearized":

$$\forall t \in [0,1], f(tx) \le tf(x) \qquad (4.3.2)$$

We give a x as a martingale discounting measure, that is, there is no excess return except $1/P[0,1]-1$. The $T$of the upper form can be replaced by a discount factor belonging to $[0,1]$,

$$\forall t \in [0,1], f(P(0,t)x_t) \leq P(0,t)f(x_t) \tag{4.3.3}$$

Let's give a filtration $\widetilde{F_s}, s<t$ so there are:

$$\forall t \in [0,1], \tilde{E}(f(P(s,t)x_t) \mid \widetilde{F_s}) \leq \tilde{E}(P(s,t)f(x_t) \mid \widetilde{F_s}) \tag{4.3.4}$$

For convex functions, the Jensen inequality can be used

$$\forall t \in [0,1], f(\tilde{E}(P(s,t)x_t \mid \widetilde{F_s})) \leq \tilde{E}(f(P(s,t)x_t) \mid \widetilde{F_s}) \leq \tilde{E}(P(s,t)f(x_t) \mid \widetilde{F_s}) \tag{4.3.5}$$

The most left function of the equation is already under the X discounting measure, so it is a directly martingale:

$$\tilde{E}(P(s,t)x \mid \widetilde{F_s}) = x_s \tag{4.3.6}$$

So in the s filtration, even if the discounted measure is given, the expected value of the future is larger than the current period:

$$\forall t \in [0,1], f(x_s) \leq \tilde{E}(P(s,t)f(x_t) \mid \widetilde{F_s}) \tag{4.3.7}$$

At this point, f is a sub-martingale, which means that its expectation value will only add, so it will not be exercise advance.

This is why there is no difference between American call options and European call options, because there is no exercise advance.

Now let us compare the European put options and the American put options, from the table 4.3.3 or the data in appendix we can conclude under the same condition the price of an American put option is always higher than the price of the European put option.

Since this paper focuses on the pricing of non-dividends stock options, and American call options do not exercise advance. Therefore, the following calculations will only focus on American put options.

Next, we will use the finite difference method to price American options. In the same way likes European option we will use the explicit finite difference method and implicit difference method to calculate the option prices. The design of the program is also quite simple. It is only necessary to add a statement that checks the exercise on the basic of the original European option pricing model.

American put option

FDM EX 10	FDM EX 20	FDM EX 100	FDM EX200
0.271636	0.224332	1.71E+44	3.83E+148
0.190094	0.157585	-4.075	-4.075
0.125917	0.121694	-3.575	-3.575
0.0808651	0.0876486	0.272587	1.22E+55
0.0967486	0.0770379	0.0647022	1.98E+20
0.113696	0.0658942	0.0509992	0.0511809

Table 4.3.4 American put option price in explicit finite difference method

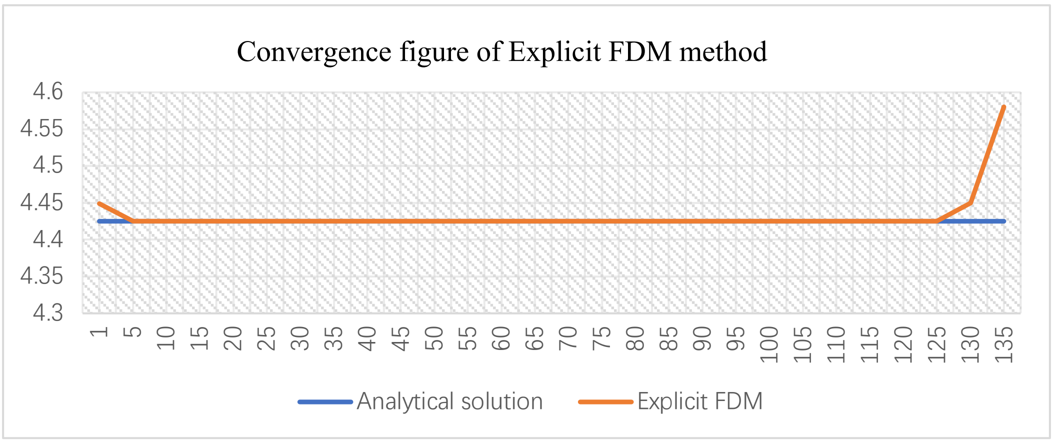

Figure 4.3.1 Convergence figure of Explicit FDM method in American option

From the above figure, it could be seen that comparing with European option the explicit finite difference method is more suitable for calculating the price of American options because the value calculated when the time step and the price step simulation number is 3 is equal to the 50,000 times simulation binomial tree method (without considering small errors). Of course, the disadvantages of the explicit finite difference method also exist at the same time. That is, the overflow effect begins when the time step is longer than the price step and the simulation is to 130 steps.

Similar to the conclusion obtained in the calculation of European option prices, when the number of simulation steps is small, the calculation accuracy is in an acceptable range, but as the number of simulation steps increases, the cumulative error increases and the calculation result overflows. Table 4.3.4 shows that the results have been significantly different from the real values in 200 steps. From the above results, we can use the explicit finite difference method to divide the right period into small steps when the calculation accuracy requirement is not particularly high. However, this method cannot be used if there is a high requirement for calculation accuracy. The following will use another form of the finite difference method the implicit finite difference method. Also, because the calculation volume is too large, it is necessary to introduce an exogenous matrix library and some codes about check whether or not to exercise in advance. Since the principle of implicit finite difference method is similar to the principle of explicit finite difference method, it is only slightly different in the calculation method. Due to the implicit finite difference method use the computationally intensive matrix operations to avoids the overflow effect and the precision is similar to the explicit FDM. So no convergence analysis will be done here.

American put option

FDM IM 10	FDM IM20	FDM IM 100	FDM IM 200
0.0246463	0.0177993	0.0242112	0.0256316
0.023181	0.0178893	0.022428	0.023972
0.0226994	0.0195644	0.0213696	0.0235232
0.0238995	0.0239306	0.0236072	0.0251829
0.0221137	0.0235546	0.0232184	0.0237446
0.0212015	0.0245754	0.0242078	0.0239952

Table 4.3.5 American put option price in implicit finite difference method

The calculation of the implied difference method of the American option has no overflow similar to that of the European option. Also, the calculation accuracy also reached expectations. However the finite difference method and the binomial tree method have common disadvantages, that is, the calculation process is too complicated and inefficient. With the development of mathematics and financial science, scientists have also studied the use of least square Monte Carlo method to calculate American put option prices. Assume that the discrete time interval number is 10 which means that there are 10 time points during the rights period to exercise to simulate American options. Due to technical limitations and the too short expiration time, it is not possible to simulate too many times.

American put option

S	X	r	sigma	time	LSM
14.575	10	0.21	2.2815	0.02	0.2266601
14.575	10.5	0.21	1.8378	0.02	0.1520834
14.575	11	0.21	1.5005	0.02	0.1117332
14.575	11.5	0.21	1.2454	0.02	0.09138384
14.575	12	0.21	0.9886	0.02	0.0637418
14.575	12.5	0.21	0.7804	0.02	0.04974608

Table 4.3.6 American put option price in Least Square Monte Carlo

After using the same data for calculation, we can compare with the results obtained by other methods before. It can be found that the LSM model can obtain better calculation accuracy and the calculation speed is faster at smaller simulation times.

4.4 Application of option pricing model in real market

First we will apply the pricing of European options. We have downloaded relevant data on the stock options of China Construction Bank (HK.0939) from the derivatives database of the official website of the Hong Kong Stock Exchange. The stock price of HK.0939 in 1.04.2018 is 8.06HK$. We selected several stock options with different execution prices and active trading volume for empirical analysis. Through the query of relevant data we set the risk-free interest rate to 0.015 meanwhile the three different expiration dates are April 18 (0.0439 years till now), May 18 (0.1315 years till now) and June 18 (0.21643 years till now). Implied volatility of options is in Appendix.

T	Strike price	Real price	BS	MC	BT	FDM
t=0.0439	7.75	0.505	0.42910	0.42103	0.42036	0.42434
	8	0.38	0.25207	0.24264	0.24229	0.19459
	8.25	0.212	0.13500	0.12515	0.12543	0.10784
	8.5	0.124	0.06351	0.05437	0.05480	0.06655
t=0.1315	7.75	0.572	0.52946	0.53049	0.52983	0.53188
	8	0.418	0.37795	0.37797	0.37837	0.34656
	8.25	0.314	0.25715	0.25694	0.25753	0.24368
t=0.2164	8	0.481	0.46959	0.47114	0.47007	0.44764
	8.25	0.407	0.35958	0.35950	0.35979	0.35133
	8.5	0.28	0.25412	0.25430	0.25400	0.25654

Table 4.4.1 Call option price of CCB

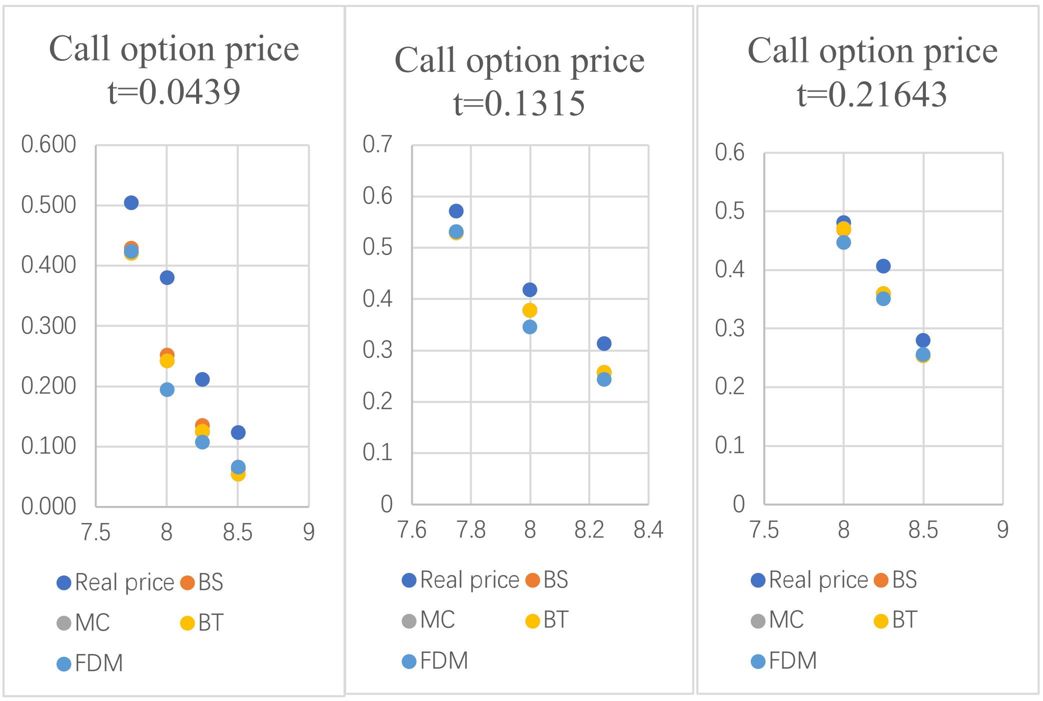

Chart 4.4.1 Call option price of CCB

T	Strike price	Real price	BS	MC	BT	FDM
t=0.0439	7.75	0.151	0.09788	0.09764	0.09789	0.102205
	8	0.244	0.18229	0.18265	0.18229	0.13506
	8.25	0.352	0.30343	0.30385	0.30342	0.286317
	8.5	0.59	0.49657	0.49646	0.49657	0.50744
t=0.1315	7.75	0.24	0.21057	0.21014	0.21056	0.213006
	8	0.314	0.30218	0.30162	0.30218	0.270798
	8.25	0.501	0.42859	0.42876	0.42858	0.414984
t=0.21643	8	0.424	0.37922	0.37943	0.37921	0.356878
	8.25	0.535	0.50347	0.50396	0.50347	0.494331
	8.5	1.055	1.05590	1.05449	1.05590	1.04975

Table 4.4.2 Put option price of CCB

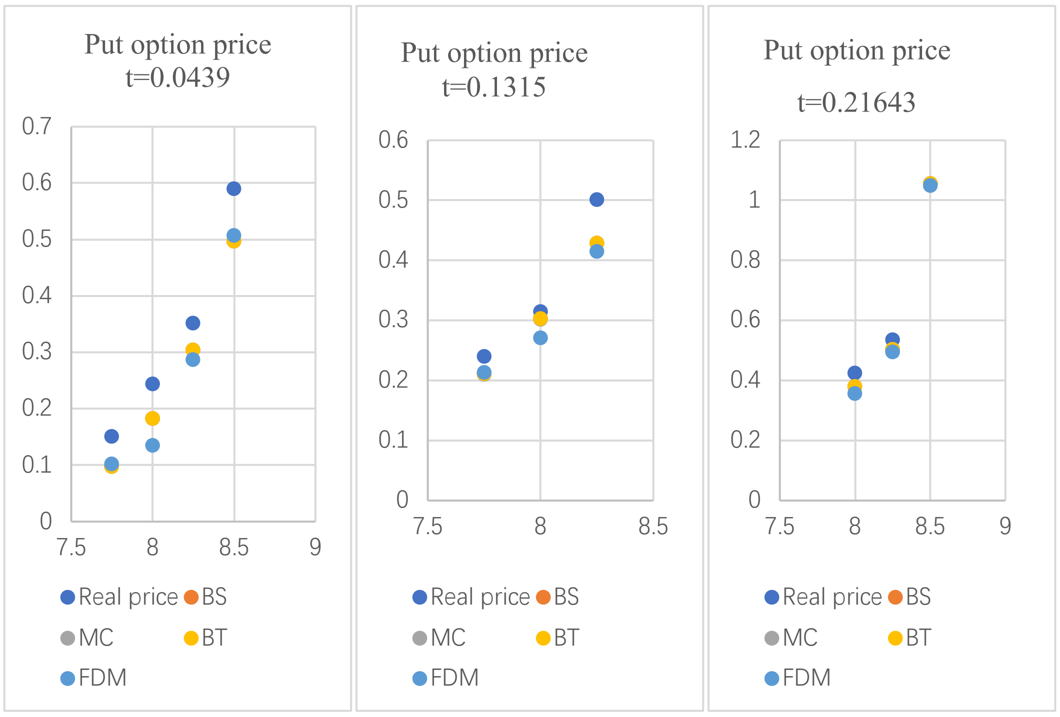

Chart 4.4.2 Put option price of CCB

Calculate the degree of deviation based on the theoretical price and the actual price. The definition of the degree of deviation is as follows: $BI = Q/P - 1$ where BI is deviation degree, Q is the market price and P is theoretical price. We will compare the maximum, minimum, and mean deviations between various methods in both the call and put options.

T	Strike price	BS	MC	BT	FDM
t=0.0439	7.75	-0.15030	-0.16628	-0.16761	-0.15972
	8	-0.33666	-0.36148	-0.36241	-0.48791
	8.25	-0.36321	-0.40968	-0.40837	-0.49133
	8.5	-0.48779	-0.56156	-0.55803	-0.46331
t=0.1315	7.75	-0.07437	-0.07257	-0.07372	-0.07014
	8	-0.09582	-0.09576	-0.09481	-0.17091
	8.25	-0.18106	-0.18172	-0.17985	-0.22396
t=0.21643	8	-0.02373	-0.02050	-0.02273	-0.06936
	8.25	-0.11650	-0.11670	-0.11600	-0.13679
	8.5	-0.09241	-0.09178	-0.09284	-0.08380
	Maximum	-0.02373	-0.02050	-0.02273	-0.06936
	Minimum	-0.48779	-0.56156	-0.55803	-0.49133
	Mean	-0.19219	-0.20780	-0.20764	-0.23572

Table 4.4.3 Call option degree of deviation *of CCB*

T	Strike price	BS	MC	BT	FDM
t=0.0439	7.75	-0.35176	-0.35340	-0.35175	-0.32315
	8	-0.25292	-0.25143	-0.25293	-0.44648
	8.25	-0.13800	-0.13680	-0.13800	-0.18660
	8.5	-0.15836	-0.15855	-0.15835	-0.13993
t=0.1315	7.75	-0.12265	-0.12442	-0.12266	-0.11248
	8	-0.03763	-0.03944	-0.03764	-0.13759
	8.25	-0.14454	-0.14419	-0.14455	-0.17169
t=0.21643	8	-0.10562	-0.10513	-0.10563	-0.15831
	8.25	-0.05894	-0.05802	-0.05894	-0.07602
	8.5	0.00085	-0.00048	0.00085	-0.00498
	Maximum	0.00085	-0.00048	0.00085	-0.00498
	Minimum	-0.35176	-0.35340	-0.35175	-0.44648
	Mean	-0.13696	-0.13719	-0.13696	-0.17572

Table 4.4.4 Put option degree of deviation *of CCB*

From the analysis of the simulation results and deviation degree methods, our theoretical prices tend to be the same as the market prices. On the premise of different maturity dates and different exercise prices, the degree of deviation between the analytical solution and the approximate solution of the B-S partial differential equation is mostly negative. This shows that the market price is higher than the theoretical price. The market price is overestimate to some extent. The deviation of the analytical solution of the B-S partial differential equation is the smallest, and it can be considered that using the analytical solution in the pricing of the European option can get the theoretical price closest to the market value.

In the following, we will transfer the research object to the empirical analysis of American put option pricing. Here we choose HSBC Holdings (HK.0005) as our object. We will study the prices of eight HSBC stock American put options. The stock price of HK.0005 in 1.04.2018 is 74 HK$. Through the query of relevant data we set the risk-free interest rate to 0.015 meanwhile the expiration dates is September18 (0.41918 years till now), Implied volatility of options is in Appendix.

T	Strike price	Real price	LSM MC	BT	FDM
t=0.41918	65	0.91	0.5037872	0.513894	0.51127
	67.5	1.44	0.8369717	0.844566	0.847237
	70	2.07	1.36331	1.36444	1.36591
	72.5	3.11	2.231524	2.23591	2.22645
	75	4.47	3.384697	3.40177	3.40199
	77.5	6.11	4.897231	4.90927	4.90949
	85	12.35	11.07543	11.1263	11.1254

Table 4.4.5 American Put option of HSBC Holding

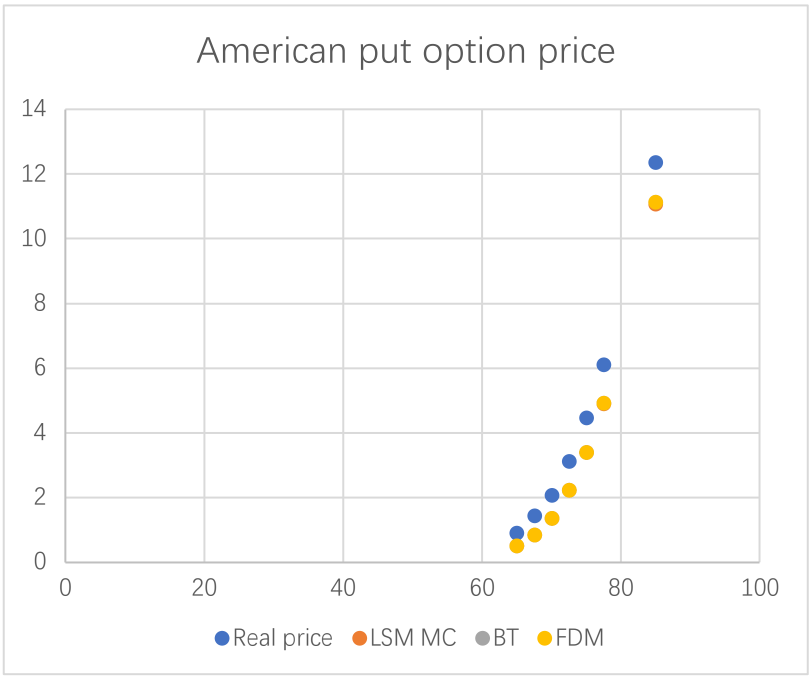

Chart 4.4.2 Put option price of CCB

It can be seen from Tables 4.4.5 and 4.4.2 that the actual value of HSBC Options is higher than the theoretical value predicted by the algorithm, but the actual value is close to the theoretical value, which means that the algorithm is real and effective. However, there are some problems in the selection of calculation data. When the exercise date is too short, the price of the American put option cannot be calculated by the least squares Monte Carlo simulation. The reason for this problem caused by the debug analysis should be because in the program, the optimal stopping time (similar to Bermuda option) is determined by artificially setting the number of exercise. When the distance to maturity date is too short, the computer cannot calculate the too small value. Resulting in an error.

Strike price	LSM MC	BT	FDM
65	-0.44639	-0.43528	-0.43816
67.5	-0.41877	-0.41350	-0.41164
70	-0.34140	-0.34085	-0.34014
72.5	-0.28247	-0.28106	-0.28410
75	-0.24280	-0.23898	-0.23893
77.5	-0.19849	-0.19652	-0.19648
85	-0.10320	-0.09909	-0.09916
Maximum	-0.10320	-0.09909	-0.09916
Minimum	-0.44639	-0.43528	-0.43816
Mean	-0.29050	-0.28647	-0.28694

Table 4.4.5American Put option degree of deviation *of HSBC Holding*

Table 4.4.5 shows the degree of deviation between the real market value and the approximate solution of the least square Monte Carlo Simulation, binomial tree model and the finite difference method is mostly negative. This also shows that the market price is high than the theoretical price. The market price is overestimate to some extent. From the average deviation of degree, the simulate option price calculate by binomial method of large steps have the smallest deviation with the real market price. This means that the binomial method has a very high precision in the pricing of American options.

Chapter 5. Conclusion

In order to study the method of option pricing and its calculation accuracy, this thesis introduces the stochastic process and the Black-Scholes partial differential equation that can be used for European option pricing, and uses C++ programming to calculate its analytic solution. The Monte Carlo method and the binomial tree method, as well as the finite difference method and their calculation methods are also introduced. Because Black-Scholes partial differential equations and general Monte Carlo simulation can not consider the advance exercise problems. It is not suitable to solve the American option price.

The thesis also introduces the least squares Monte Carlo simulation together with the binomial tree method and the finite difference method calculate the American option price. In the empirical analysis, several stock options of Construction Bank (HK.0939) and HSBC Holdings (HK.0005) were selected for research. The article uses several models to simulate the European option path of China Construction Bank's stock and the American option path of HSBC Holdings to obtain the option price.

The experimental results show that the direct solution to the analytical solution of partial differential equations in the pricing of European options yields the option price closest to the real value of the option. However, due to the inability to directly obtain analytical solutions in the face of more complex partial differential equations, a binomial method with large step numbers is required because the accuracy of the binomial method is very close to that of BS partial differential equations at larger steps. In addition, in the American option price simulation, it is also the binomial method with the highest accuracy, and the closest to the market real value.

However, in empirical analysis simulations, we find that the theoretical price of derivatives is always lower than the real market price, which means that the market is overestimated to some extent. We think the reasons may have the following:

1. In recent years, the stock prices of China Construction Bank and HSBC Holdings have been at a high level and the return on net assets has risen (Construction Bank (HK.0939): 13.16% and HSBC Holdings (HK.0005): 5.68% in 2017). Investors are optimistic about the stock's outlook and the option price is overvalued.
2. According to modern financial theory, when the price of financial assets seriously deviates from the theoretical price, the market will have opposite expectations. If the issuer or market makers are allowed to sell short, it may create a constraint on the price trend of the financial asset. The model of option pricing itself assumes that there is a short selling mechanism in the market. However, virtually no short-selling mechanism exists in the securities markets of the Chinese mainland or the Hong Kong Special Administrative Region. The unilateral nature of the market mechanism has made it difficult for the securities market to form an effective spontaneous restraint mechanism. As a result, effective suppression of market prices has not been applied.
3. Even though the Hong Kong Special Administrative Region is an international financial center with an extremely sophisticated financial system, many individual investors still have a lot of blinding and following trends in investment behavior,

the ability to identify and judge risks is insufficient, so there is inevitable blindness in the investment process.

Due to the limited ability of author, there are still many places that are worth discussing. It can be perfected through deeper research. First, the thesis considers only Black-Scholes partial differential equations. However, with the development of mathematical sciences, more and more studies tend to think that financial asset price distribution has fat tails and skewness, which are not suitable for normal distribution. Due to the programming technique reasons, the article did not include a variance gamma distribution model (VG) or a normal inverse Gaussian distribution model (NIG) for comparison studies. Finally, this thesis does not correct some of the limitations of the Monte Carlo simulation method.

Bibliography

1. BARNDORFF-NIELSEN OLE E. Normal Inverse Gaussian Distributions and Stochastic Volatility Modelling. *Scandinavian Journal of Statistics* 24: 1–13 1997.
2. BOYLE Phelim P. Options: A Monte Carlo approach. *Journal of Financial Economics*, 4:323–38, 1977.
3. CHEN Yu. *Options pricing theory and its applications. China Financial Publishing House*, 1998, 183p. ISBN: 9787504920287.
4. DUFFY Daniel J. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*. Wiley, 2013, 464p. ISBN: 9781118856482.
5. DUFFY Daniel J., J KIENITZ. *Monte Carlo Frameworks: Building Customisable High-performance C++ Applications.*Wiley,2009,776p ISBN: 9780470060698.
6. DUFFY Daniel J. *Introduction to C++ for Financial Engineers*. Wiley, 2006, 438p. ISBN:9780470015384.
7. DUFFY Daniel J. *Financial Instrument Pricing Using C++*. Wiley, 2004, 432p. ISBN:9780470855096.
8. HULL John C. *Options, Futures and Other Derivatives 8th Edition*. Pearson, 2011, 872p. ISBN: 9780273759072.
9. IACUS Stefano M. *Option Pricing and Estimation of Financial Models with R*. John Wiley & Sons, 2011, 472p. ISBN:9781119990208.
10. JOSHI M. S. *C++ Design Patterns and Derivatives Pricing. Cambridge*, 2008, 306p. ISBN:9780521721622.

11. JOY Corwin, Phelim P. BOYLE, KEN Seng Tan. Quasi-Monte Carlo methods in numerical finance. *Management Science*, vol. 42, pp. 926-938, June 1996.
12. LONGSTAFF Francis A., Eduardo S. SCHWARTZ. Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies*, Vol. 14, No. 1. (Spring, 2001), pp. 113-147.
13. LONDON Justin. *Modeling Derivatives in C++*.John Wiley & Sons, 2005, 840p. ISBN:9780471681892.
14. MEYER Christian. Recursive Numerical Evaluation of the Cumulative Bivariate Normal Distribution. *Journal of Statistical Software*, March 2013.Volume 52, Issue 10.
15. MOROKOFF W. J., R. E. Caflisch. Quasi-Monte Carlo Integration. *Journal of Computational Physics*, 1995, 122,218-230.
16. MERTON Robert C. An analytic derivation of the efficient portfolio frontier. *Journal of Financial and Quantitative Analysis*, 7:1851–72, September 1972.
17. MCDONALD Robert L. *Derivatives Markets*. Pearson, third edition, 2013, 904p. ISBN:9781292021256.
18. ODEGAARD Bernt Arne. *Financial Numerical Recipes in C++*. 2014.89-98.
19. PENA Alonso. *Advanced Quantitative Finance with C++*. Packt Publishing, 2014, 124p. ISBN: 9781782167235.
20. PLATEN Eckhard, David HEATH. *A Benchmark Approach to Quantitative Finance. Springer Finance*, 2006, 700p.ISBN:9783540478560.

21. PHILIP Peter. *Numerical Methods for Mathematical Finance*. August 7, 2016. Lecture Notes.
22. PRATA Stephen. *C++ Primer Plus (6th Edition)*. Addison-Wesley Professional, 2011, 1200p. ISBN:9780321776402.
23. SHREVE, Steven. E. *Stochastic Calculus for Finance II: Continuous-Time modeling*. Springer-Verlag New York, 2004, 550p. ISBN: 9781441923110.
24. TICHY Tomas. L*attice Models- Pricing and Hedging at (In)complete Markets.* VSB-Technical University, Faculty of Economics, 2008, 150p.ISBN:9788024817033.
25. TICHY Tomas. Levy Processes in Finance: Selected Applications with Theoretical Background. VSB-Technical University, Faculty of Economics, 2011. ISBN: 9788024825366
26. TANKOV Peter. *Financial Modelling with Jump Processes*. CRC Press, 2003, 552p. ISBN:9781135437947.
27. TANKOV Peter. *Financial modeling with Levy processes*. Lecture notes.
28. TIAN Wenzhao. *Pricing Theory and Numerical Calculation of Financial Assets*. Peking University Press, 2011, 277P. ISBN:9787301159903.
29. WILMOTT Pau, Sam HOWISON, Jeff DEWYNNE. *The Mathematics of Financial Derivatives: A Student Introduction*. Cambridge University Press, 1995, 317p. ISBN: 9780521497893.
30. WILMOTT Pau. *Paul Wilmott Introduces Quantitative Finance*. John Wiley & Sons, 2013,728p. ISBN:9781118836798.

31. YANG Haijun, YANG Lei. Valuing American Options by Weighted Least-Squares Quasi-Monte Carlo. *Journal of systems engineering*, vol.23 No.5 Oct.2008.
32. ZMESKAL Zdenek, Dana DLUHOSOVA, Tomas TICHY. *Financial models*. Ostrava: VSB-Technical University, Faculty of Economics, 2004, 254p. ISBN:9788024807546.

Electronic literature

1. http://itpp.sourceforge.net/4.3.1/
2. http://www.robertnz.net/nm_intro.htm
3. http://www.hkex.com.hk/Market-Data/Futures-and-Options-Prices/Equity-Index/Hang-Seng-Index-Futures-and-Options?sc_lang=zh-HK#&product=HSI
4. http://www.cmegroup.com/market-data.html/
5. https://github.com

List of abbreviations

X: strike price.

S_0: current stock price.

S_t: stock price at time t.

T: time to maturity.

Sigma/σ: implied volatility.

c: European call option price.

p: European put option price.

C: American call option price.

P: American put option price.

B-S: Black-Scholes function.

LSM: least square Monte Carlo simulation method.

Declaration of Utilization of Result from the Diploma Thesis

Herewith I declare that

- I am informed that Act No. 121/2000 Coll. – the Copyright Act, in particular, Section 35 – Utilization of the Work as a part of Civil and Religious Ceremonies, as a Part of School Performances and the Utilization of a School Work – and Section 60 – School Work. Fully applies to my diploma thesis;
- I take account of the VSB - Technical University of Ostrava (hereinafter as VSB-TUO) having the right to utilize the diploma thesis (under Section 35(3)) unprofitably and for own use;
- I agree that the diploma thesis shall be archived in the electronic from in VSB-TUO's Central Library and one copy shall be kept by the supervisor of the diploma thesis. I agree that the bibliographic information about the diploma thesis shall be published in VSB-TUO's information system;
- It was agreed that, in case of VSB-TUO's interest, I shall enter into a license agreement with VSB-TUO, granting the authorization to utilize the work in the scope of Section 12(4) of the Copyright Act;
- It was agrees that I may utilize my work, the diploma thesis, or provide a license to utilize it only with the consent of VSB-TUO, which is entitled, in such a case, to claim an adequate contribution from me to cover the cost expended by VSB-TUO for producing the work (up to its real amount),

Ostrava dated 23.04.2018

[illegible]

Student's name and surname

79

List of annexes

Table 7. Numerical solution in explicit FDM method of American option

Table 8. Numerical solution in implicit FDM method of American option

Annexes

1.Code of the pricing program

Program 1. Normal distribution.

```
#include <math.h>

     #include <iostream>

double N(const double &x)

{

    if (x>6.0) {return 1.0;};if (x<-6.0) {return 0.0;};

    double b1 = 0.31938153; double b2 = -0.356563782;

    double b3 = 1.781477937;double b4 = -1.821255978;

    double b5 = 1.330274492;double p = 0.2316419;

    double c2 = 0.3989423;double a =fabs(x);

    double t = 1.0/(1.0+a*p);double b = c2*exp((-x)*(x/2.0));

    double n =((((b5*t+b4)*t+b3)*t+b2)*t+b1)*t;

    n=1.0-b*n;

    if(x<0.0)n=1.0-n;

    return n;

}
```

Program 2. European call option pricing

```
#include <math.h>

#include"normdist.h"

#include <iostream>

double option_price_call_black_scholes(const double &S,//underlying asset price

                                       const double &X,//exercise price

                                       const double &r,//risk free rate

                                       const double &sigma,//volatility

                                       const double &time)//period of exercise

{

    double time_sqrt = sqrt(time);

    double d1 = (log(S/X)+r*time)/(sigma*time_sqrt)+0.5*sigma*time_sqrt;//d1

    double d2 =d1-(sigma*time_sqrt);//d2
    double c = S*N(d1)-X*exp(-r*time)*N(d2);//call
    return c;
}
```

Program 3. European put option pricing

```
double option_price_put_black_scholes(const double &S,//underlying asset price
                                      const double &X,//exercise price
                                      const double &r,//risk free rate
                                      const double &sigma,//volatility
                                      const double &time)//period of exercise
{
    double time_sqrt = sqrt(time);
    double d1 = (log(S/X)+r*time)/(sigma*time_sqrt)+0.5*sigma*time_sqrt;//d1
    double d2 =d1-(sigma*time_sqrt);//d2
    double p = X*exp(-r*time)*N(-d2)-S*N(-d1);//put
    return p;
}
```

Program 4. Function of random number

```
#include <cstdlib>
#include <cmath>
using namespace std;
double random_normal()
{
    double result;double x;
    double y;double xysquare;
    do
{
        x = 2.0*rand()/static_cast<double>(RAND_MAX)-1;
        y = 2.0*rand()/static_cast<double>(RAND_MAX)-1;
        xysquare = x*x + y*y;
    }
    while
        (xysquare >= 1.0);
    result = x*sqrt(-2*log(xysquare)/xysquare);
    return result;
}
```

Program 5. Monte Carlo Method

```
#include <iostream>
#include <math.h>
#include "normdist.h"
#include "random.h"
double max(double a,double b)
{
    if(a>b) return a;
       else return b;
}
void option_price_european_simulated(const double &S,
                                     const double &X,
                                     const double &r,
                                     const double &sigma,
                                     const double &time,
                                     const double &no_sims,
                                     double &call_option,
                                     double &put_option)
{
    double R  = (r-0.5*pow(sigma,2))*time;
    double SD = sigma*sqrt(time);
    double sum_payoffs1 = 0.0;
    double sum_payoffs2 = 0.0;
    for (int n = 1;n<=no_sims;n++)
    {
        double S_T = S*exp(R+SD* random_normal());
        sum_payoffs1 += max(0.0,S_T-X);
        sum_payoffs2 += max(X-S_T,0.0);
    }
    call_option = exp(-r*time)*(sum_payoffs1/double(no_sims));
    put_option  = exp(-r*time)*(sum_payoffs2/double(no_sims));
}
```

Program 6. Binomial Tree method for European option pricing

```
#include <iostream>
#include <math.h>
#include <vector>
#include "normdist.h"
#include "random.h"
double max (double x,
            double y)
{
    if (x>y) return x;
        else return y;
}
double option_price_call_european_binomia(const double &S,
                                          const double &X,
                                          const double &r,
                                          const double &sigma,
                                          const double &time,
                                          const int &steps)
{
    double R      = exp(r*(time/steps));
    double Rinv   = 1.0/R;
    double u      = exp(sigma*sqrt(time/steps));
    double uu     = u*u;
    double d      = 1.0/u;
    double p_up   = (R-d)/(u-d);
    double p_down = 1.0-p_up;
    vector<double> prices(steps+1);
    prices[0] = S*pow(d,steps);
    for (int i = 1;i<=steps; ++i) prices[i] = uu*prices[i-1];
    vector<double> call_values(steps+1);
    for (int j = 0;j<=steps; ++j) call_values[j] = max(0.0,(prices[j]-X));
    for (int step = steps-1;step >=0; --step)
    {
        for (int i=0;i<= step; ++i)
        {
            call_values[i] = (p_up*call_values[i+1]+p_down*call_values[i])*Rinv;
        }
    }
    return call_values[0];
}
double option_price_put_european_binomia(const double &S,
                                          const double &X,
                                          const double &r,
                                          const double &sigma,
                                          const double &time,
                                          const int &steps)
{
    double R      = exp(r*(time/steps));
    double Rinv   = 1.0/R;
    double u      = exp(sigma*sqrt(time/steps));
    double uu     = u*u;
    double d      = 1.0/u;
    double p_up   = (R-d)/(u-d);
    double p_down = 1.0-p_up;
    vector<double> prices(steps+1);
    prices[0] = S*pow(d,steps);
    for (int i = 1;i<=steps; ++i) prices[i] = uu*prices[i-1];
    vector<double> put_values(steps+1);
    for (int j = 0;j<=steps; ++j) put_values[j] = max(0.0,(X-prices[j]));
    for (int step = steps-1;step >=0; --step)
    {
        for (int i=0;i<= step; ++i)
        {
            put_values[i] = (p_up*put_values[i+1]+p_down*put_values[i])*Rinv;
        }
    }
    return put_values[0];
}
```

Program 7. Binomial Tree method for American option pricing

```
#include <iostream>
#include "normdist.h"
#include <math.h>
#include "random.h"
#include <vector>

using namespace std;
double max(double x,double y)
{
    if(x>y) return x;
       else return y;
}
double option_price_call_american_binomial(const double &S,
                                           const double &X,
                                           const double &r,
                                           const double &sigma,
                                           const double &time,
                                           const int &steps)
{
    double R     = exp(r*(time/steps));
    double Rinv   = 1.0/R;
    double u     = exp(sigma*sqrt(time/steps));
    double uu    = u*u;
    double d      = 1.0/u;
    double p_pu   = (R-d)/(u-d);
    double p_down = 1.0-p_pu;
    vector<double> prices(steps+1);
    vector<double> call_values(steps+1);
    prices[0]     = S*pow(d,steps);
    for (int i = 1; i<=steps;i++) prices[i] = uu*prices[i-1];
    for (int i = 0; i<=steps;++i) call_values[i] = max(0.0,(prices[i]-X));
    for (int step = steps-1; step>=0;--step)
    {
        for  (int i = 0; i<=step; ++i)
        {
            call_values[i] = (p_pu*call_values[i+1]+p_down*call_values[i])*Rinv;
            prices[i]     = d*prices[i+1];
            call_values[i] = max(call_values[i],prices[i]-X);//checking exercise
        }
    }
    return call_values[0];
}
double option_price_put_american_binomial(const double &S,
                                          const double &X,
                                          const double &r,
                                          const double &sigma,
                                          const double &time,
                                          const int &steps)
{
    double R      = exp(r*(time/steps));
    double Rinv   = 1.0/R;
    double u      = exp(sigma*sqrt(time/steps));
    double uu     = u*u;
    double d      = 1.0/u;
    double p_pu   = (R-d)/(u-d);
    double p_down = 1.0-p_pu;
    vector<double> prices(steps+1);
    prices[0]     = S*pow(d,steps);
    for (int i = 1; i<=steps;i++) prices[i] = uu*prices[i-1];
    vector<double> put_values(steps+1);
    for (int i = 0; i<=steps;++i) put_values[i] = max(0.0,(X-prices[i]));
    for (int step = steps-1; step>=0;--step)
    {
        for  (int i = 0; i<=step; ++i)
        {
            put_values[i] = (p_pu*put_values[i+1]+p_down*put_values[i])*Rinv;
            prices[i]      = d*prices[i+1];
            put_values[i] = max(put_values[i],X-prices[i]); //checking exercise
        }
    }
    return put_values[0];
}
```

Program 8. Explicit finite-difference method on European option pricing

```
#include <iostream>
#include <vector>
#include <iostream>
using namespace std;
double max(double x,double y)
{
    if(x>y) return x;
       else return y;
}
double option_price_call_european_finite_different_explicit(const double &S,
                                                       const double &X,
                                                       const double &r,
                                                       const double &sigma,
                                                       const double &time,
                                                       const int &no_S_steps,
                                                               const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    unsigned int M;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;
    vector<double> S_values(M+1);
    for (unsigned m = 0;m<=M;m++){S_values[m] = m*delta_S;};
    int N = no_t_steps;
    double delta_t = time/N;

    vector<double> a(M);
    vector<double> b(M);
    vector<double> c(M);

    double r1 = 1.0/(1.0+r*delta_t);
    double r2 = delta_t/(1.0+r*delta_t);
    for (unsigned int j=1; j<M; j++)
    {
        a[j] = r2*0.5*j*(-r+sigma_sqr*j);
        b[j] = r1*(1.0-sigma_sqr*j*j*delta_t);
        c[j] = r2*0.5*j*(r+sigma_sqr*j);
    }
    vector<double> f_next(M+1);
    for (unsigned int n = 0; n<=M; ++n){f_next[n] = max(0.0,S_values[n]-X);};
    vector<double> f(M+1);
    for (int t=N-1; t>=0; --t)
    {
        f[0] = 0;
        for (unsigned m = 1; m<M; ++m)
        {
            f[m] = a[m]*f_next[m-1]+b[m]*f_next[m]+c[m]*f_next[m+1];
        }
        f[M] = 0;

        for (unsigned n = 0; n<=M; ++n){f_next[n] = f[n];};

    }

    double C = f[M/2];

    return C;


}

double option_price_put_european_finite_different_explicit(const double &S,

                                                           const double &X,

                                                           const double &r,

                                                           const double &sigma,

                                                           const double &time,

                                                           const int &no_S_steps,

                                                           const int &no_t_steps)

{

    double sigma_sqr = sigma*sigma;

    unsigned int M;

    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};

    double delta_S = 2.0*S/M;

    vector<double> S_values(M+1);

    for (unsigned m = 0;m<=M;m++){S_values[m] = m*delta_S;};

    int N = no_t_steps;

    double delta_t = time/N;


    vector<double> a(M);

    vector<double> b(M);

    vector<double> c(M);


    double r1 = 1.0/(1.0+r*delta_t);

    double r2 = delta_t/(1.0+r*delta_t);

    for (unsigned int j=1; j<M; j++)

    {

        a[j] = r2*0.5*j*(-r+sigma_sqr*j);
        b[j] = r1*(1.0-sigma_sqr*j*j*delta_t);

        c[j] = r2*0.5*j*(r+sigma_sqr*j);

    }

    vector<double> f_next(M+1);

    for (unsigned int n = 0; n<=M; ++n){f_next[n] = max(0.0,X-S_values[n]);};

    vector<double> f(M+1);

    for (int t=N-1; t>=0; --t)

    {

        f[0] = 0;

        for (unsigned m = 1; m<M; ++m)

        {

            f[m] = a[m]*f_next[m-1]+b[m]*f_next[m]+c[m]*f_next[m+1];

        }

        f[M] = 0;

        for (unsigned n = 0; n<=M; ++n){f_next[n] = f[n];};

    }

    double P = f[M/2];

    return P;

}
```

Program 9. Explicit finite-difference method on American option pricing

```
#include <iostream>

#include <vector>

#include <math.h>

#include <fstream>

using namespace std;

double max(double x,double y)

{

    if(x>y) return x;

    else return y;

}

double option_price_call_american_finite_different_explicit(const double &S,

                                                            const double &X,
                                                                    const double &r,
                                                                    const double &sigma,
                                                                    const double &time,
                                                                    const int &no_S_steps,
                                                                    const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    int M = no_S_steps;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;
    vector<double> S_values(M+1,0.0);
    for (int m = 0; m<=M; m++) {S_values[m] = m*delta_S;};
    int N = no_t_steps;
    double delta_t = time/N;

    vector<double> a(M,0.0);
    vector<double> b(M,0.0);
    vector<double> c(M,0.0);

    double r1 = 1.0/(1.0+r*delta_t);
    double r2 = delta_t/(1.0+r*delta_t);
    for ( int j=1; j<M; j++)
    {
        a[j] = r2*0.5*j*(-r+sigma_sqr*j);
        b[j] = r1*(1.0-sigma_sqr*j*j*delta_t);
        c[j] = r2*0.5*j*(r+sigma_sqr*j);
    }
    vector<double> f_next(M+1,0.0);
    for (int n = 0; n<=M; ++n){f_next[n] = max(0.0,S_values[n]-X);};
    vector<double> f(M+1,0.0);
    for (int t=N-1; t>=0; --t)
    {
        f[0] = 0;
        for (int m = 1; m<M; ++m)
        {
            f[m] = a[m]*f_next[m-1]+b[m]*f_next[m]+c[m]*f_next[m+1];
            f[m] = max(f[m],S_values[m]-X);
        }
        f[M] = S_values[M]-X;
        for (int n = 0; n<=M; ++n){f_next[n] = f[n];};
    }
    double C = f[M/2];
    return C;
}

double option_price_put_american_finite_different_explicit(const double &S,
                                                            const double &X,
                                                            const double &r,
                                                            const double &sigma,
                                                            const double &time,
                                                            const int &no_S_steps,
                                                            const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    int M = no_S_steps;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;
    vector<double> S_values(M+1);
    for (int m = 0; m<=M; m++) {S_values[m] = m*delta_S;};
    int N = no_t_steps;
    double delta_t = time/N;

    vector<double> a(M);
    vector<double> b(M);
    vector<double> c(M);
    double r1 = 1.0/(1.0+r*delta_t);
    double r2 = delta_t/(1.0+r*delta_t);
    for ( int j=1; j<M; j++)
    {
        a[j] = r2*0.5*j*(-r+sigma_sqr*j);
        b[j] = r1*(1.0-sigma_sqr*j*j*delta_t);
        c[j] = r2*0.5*j*(r+sigma_sqr*j);
    }
    vector<double> f_next(M+1);
    for (int n = 0; n<=M; ++n){f_next[n] = max(0.0,X-S_values[n]);};
    vector<double> f(M+1);
    for (int t=N-1; t>=0; --t)
    {
        f[0] = 0;
        for (int m = 1; m<M; ++m)
        {
            f[m] = a[m]*f_next[m-1]+b[m]*f_next[m]+c[m]*f_next[m+1];
            f[m] = max(f[m],X-S_values[m]);
        }
        f[M] = 0;
        for (int k = 0; k<=M; ++k){f_next[k] = f[k];};
    }
    double P = f[M/2];
    return P;
}
```

Program 10. Implicit finite-difference method on European option pricing

```
#include <math.h>
#include "newmat.h"
#include "normdist.h"
#include <vector>
#include <iostream>
using namespace std;
```

```
double max(double x,double y)
{
    if(x>y) return x;
    else return y;
}
double option_price_call_european_finite_different_implict(const double &S,
                                                           const double &X,
                                                           const double &r,
                                                           const double &sigma,
                                                           const double &time,
                                                           const int &no_S_steps,
                                                           const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    int M = no_S_steps;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;
    vector<double> S_values(M+1,0.0);
    for (int m = 0; m<=M; m++) {S_values[m] = m*delta_S;};
    int N = no_t_steps;
    double delta_t = time/N;
    BandMatrix A (M+1,1,1);A = 0.0;
    A.element(0,0) = 1.0;
    for (int j = 1; j<M; ++j)
    {
        A.element(j,j-1) = 0.5*j*delta_t*(r-sigma_sqr*j);
        A.element(j,j)   = 1.0+delta_t*(r-sigma_sqr*j);
        A.element(j,j+1) = 0.5*j*delta_t*(-r-sigma_sqr*j);
    }
    A.element(M,M) = 1.0;
    ColumnVector B(M+1);
    for (int m=0; m<=M; ++m){B.element(m) = max(0.0,S_values[m]-X);};
    ColumnVector F = A.i()*B;
    for (int t = N-1; t>0;--t)
    {
        B = F;
        F = A.i()*B;
    }
    return F.element(M/2);
}
double option_price_put_european_finite_different_implict(const double &S,
                                                          const double &X,
                                                          const double &r,
                                                          const double &sigma,
                                                          const double &time,
                                                          const int &no_S_steps,
                                                          const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    int M = no_S_steps;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;
    vector<double> S_values(M+1,0.0);
    for (int m = 0; m<=M; m++) {S_values[m] = m*delta_S;};
    int N = no_t_steps;
    double delta_t = time/N;
    BandMatrix A (M+1,1,1);A = 0.0;
    A.element(0,0) = 1.0;
    for (int j = 1; j<M; ++j)
    {
        A.element(j,j-1) = 0.5*j*delta_t*(r-sigma_sqr*j);
        A.element(j,j)   = 1.0+delta_t*(r-sigma_sqr*j);
        A.element(j,j+1) = 0.5*j*delta_t*(-r-sigma_sqr*j);
    }
    A.element(M,M) = 1.0;
    ColumnVector B(M+1);
    for (int n=0; n<=M; ++n){B.element(n) = max(0.0,X-S_values[n]);};
    ColumnVector F = A.i()*B;
    for (int t = N-1; t>0;--t)
    {
        B = F;
        F = A.i()*B;
    }
    return F.element(M/2);
}
```

Program 11. Implicit finite-difference method on American option pricing

```
#include <iostream>
#include "newmat10/newmat.h"
#include <math.h>
#include "normdist.h"
#include "vector"
using namespace std;
double max(double x,double y)
{
    if(x>y) return x;
    else return y;
}
double option_price_call_american_finite_different_implict(const double &S,
                                                           const double &X,
                                                           const double &r,
                                                           const double &sigma,
                                                           const double &time,
                                                           const int &no_S_steps,
                                                           const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    unsigned int M;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;

    vector<double> S_values(M+1);

    for (unsigned m = 0;m<=M;m++){S_values[m] = m*delta_S;};

    int N = no_t_steps;

    double delta_t = time/N;

    BandMatrix A (M+1,1,1);A = 0.0;

    A.element(0,0) = 1.0;

    for (int j = 1; j<M; ++j)

    {

        A.element(j,j-1) = 0.5*j*delta_t*(r-sigma_sqr*j);

        A.element(j,j)   = 1.0+delta_t*(r-sigma_sqr*j);

        A.element(j,j+1) = 0.5*j*delta_t*(-r-sigma_sqr*j);

    }

    A.element(M,M) = 1.0;

    ColumnVector B(M+1);

    for (unsigned n = 0; n<=M; ++n){B.element(n) = max(0.0,S_values[n]-X);};

    ColumnVector F = A.i()*B;

    for (int t = N-1; t>0; --t)

    {

        B = F;

        F = A.i()*B;

        for (unsigned m = 1; m<M; ++m)

        {

            F.element(m) = max(F.element(m),S_values[m]-X);

        }

    }

    return F.element(M/2);

}


double option_price_put_american_finite_different_implict(const double &S,

                                                          const double &X,

                                                          const double &r,

                                                          const double &sigma,
                                                              const double &time,
                                                              const int &no_S_steps,
                                                              const int &no_t_steps)
{
    double sigma_sqr = sigma*sigma;
    unsigned int M;
    if ((no_S_steps%2)==1){M=no_S_steps+1;}else{M=no_S_steps;};
    double delta_S = 2.0*S/M;
    vector<double> S_values(M+1,0.0);
    for (int m = 0;m<=M;m++){S_values[m] = m*delta_S;};
    int N = no_t_steps;
    double delta_t = time/N;
    BandMatrix A (M+1,1,1);A = 0.0;
    A.element(0,0) = 1.0;
    for (int j = 1; j<M; ++j)
    {
        A.element(j,j-1) = 0.5*j*delta_t*(r-sigma_sqr*j);
        A.element(j,j)   = 1.0+delta_t*(r-sigma_sqr*j);
        A.element(j,j+1) = 0.5*j*delta_t*(-r-sigma_sqr*j);
    }
    A.element(M,M) = 1.0;
    ColumnVector B(M+1);
    for (int n = 0; n<=M; ++n){B.element(n) = max(0.0,X-S_values[n]);};
    ColumnVector F = A.i()*B;
    for (int t = N-1; t>0; --t)
    {
        B = F;
        F = A.i()*B;
        for (unsigned m = 1; m<M; ++m)
        {
            F.element(m) = max(F.element(m),X-S_values[m]);
        }
    }
    return F.element(M/2);
}
```

Program 12. Loop of calculation

```
int main()
{
    int j;
    double  S[100],X[100],r[100],sigma[100],time[100];
         ifstream inf("/Users/ /data1.txt"); // Assume data is saved in c:\da.txt
file
    double data[100000];          // An array to hold the read out number
    int i=0;
    while (inf>>data[i])        // Read the number in the inf file into the data
array
        ++i;
    inf.close();              // After reading, close the file
                           /*for (int j=0; j<i; j++)
      {   // The number stored in the output data array (ie c:\da.txt file).
                cout<<data[j]<<'\t';
              }*/
    int s=0,x=0,p=0,sig=0,ti=0;
    for(j=0;j<i;j++)
    {
        if(j%5==0)
        {
            S[s]=data[j];//cout<<S[m]<<'\t';
            s++;
        }
        if(j%5==1)
        {
            X[x]=data[j];//cout<<X[m]<<'\t';
            x++;
        }
        if(j%5==2)
        {
            r[p]=data[j];//cout<<r[m]<<'\t';
            p++;
        }
        if(j%5==3)
        {
            sigma[sig]=data[j];//cout<<sigma[m]<<'\t';
            sig++;
        }
        if(j%5==4)
        {
            time[ti]=data[j];//cout<<time[m]<<'\t';
            ti++;
        }
    }
    //for( int s = 0; s <5; s++)
    //cout<<S[s]<<'\t'; //return 0;
```

Program 13. Least square Monte Carlo method

```
LSM <- function(n, d, S0, K, sigma, r, T) {
    s0 <- S0/K
    dt <- T/d
    z <- rnorm(n)
    s.t <- s0 * exp((r - 1/2 * sigma^2) * T + sigma * z * (T^0.5))
    s.t[(n + 1):(2 * n)] <- s0 * exp((r - 1/2 * sigma^2) * T -
                                        sigma * z * (T^0.5))
    CC <- pmax(1 - s.t, 0)
    payoffeu <- exp(-r * T) * (CC[1:n] + CC[(n + 1):(2 * n)])/2 * K
    euprice <- mean(payoffeu)
    for (k in (d - 1):1) {
        z <- rnorm(n)
       mean <- (log(s0) + k * log(s.t[1:n]))/(k + 1)
       vol <- (k * dt/(k + 1))^0.5 * z
```

```
            s.t.1 <- exp(mean + sigma * vol)
            mean <- (log(s0) + k * log(s.t[(n + 1):(2 * n)]))/(k +1)
            s.t.1[(n + 1):(2 * n)] <- exp(mean - sigma * vol)
            CE <- pmax(1 - s.t.1, 0)
            idx <- (1:(2 * n))[CE > 0]
            discountedCC <- CC[idx] * exp(-r * dt)
            basis1 <- exp(-s.t.1[idx]/2)
            basis2 <- basis1 * (1 - s.t.1[idx])
            basis3 <- basis1 * (1 - 2 * s.t.1[idx] + (s.t.1[idx]^2)/2)
            p <- lm(discountedCC ~ basis1 + basis2 + basis3)$coefficients
            estimatedCC <- p[1] + p[2] * basis1 + p[3] * basis2 +
                p[4] * basis3
            EF <- rep(0, 2 * n)
            EF[idx] <- (CE[idx] > estimatedCC)
            CC <- (EF == 0) * CC * exp(-r * dt) + (EF == 1) * CE
            s.t <- s.t.1
        }
        payoff <- exp(-r * dt) * (CC[1:n] + CC[(n + 1):(2 * n)])/2
        usprice <- mean(payoff * K)
        error <- 1.96 * sd(payoff * K)/sqrt(n)
        earlyex <- usprice - euprice
        data.frame(usprice, error, euprice)
  }
S0 <- 36
K <- 30
T <- 1
r <- 0.05
sigma <- 0.4
LSM(10000, 3, S0, K, sigma, r, T)
```

Table of calculation results

Table 1. Black-Scholes analytical solution

Call option

S	X	r	sigma	time	Black-Scholes
14.575	10	0.21	1.4094	0.02	4.64252
14.575	10.5	0.21	1.2383	0.02	4.14305
14.575	11	0.21	1.0838	0.02	3.64458
14.575	11.5	0.21	0.9506	0.02	3.14833
14.575	12	0.21	0.7994	0.02	2.649
14.575	12.5	0.21	0.6621	0.02	2.15129
14.575	13	0.21	0.613	0.02	1.67633
14.575	13.5	0.21	0.6165	0.02	1.24814
14.575	14	0.21	0.6601	0.02	0.906659
14.575	14.5	0.21	0.7313	0.02	0.668617
14.575	15	0.21	0.7526	0.02	0.461753
14.575	16	0.21	0.9005	0.02	0.2772
14.575	17	0.21	0.9621	0.02	0.146352
14.575	18	0.21	1.1134	0.02	0.112278
14.575	19	0.21	1.1457	0.02	0.0608679
14.575	20	0.21	1.1811	0.02	0.0339717
14.575	21	0.21	1.3255	0.02	0.0337002
14.575	22	0.21	1.4599	0.02	0.0334443
14.575	23	0.21	1.565	0.02	0.0304943

14.575	24	0.21	1.6845	0.02	0.0305528
14.575	25	0.21	1.797	0.02	0.030546
14.575	26	0.21	1.9035	0.02	0.0305165
14.575	27	0.21	2.006	0.02	0.0306195
14.575	28	0.21	2.1025	0.02	0.0305827
14.575	29	0.21	2.1961	0.02	0.0306898
14.575	30	0.21	2.2849	0.02	0.030706
14.575	32.5	0.21	2.4927	0.02	0.0308383
14.575	35	0.21	2.6817	0.02	0.0309451
14.575	37.5	0.21	2.8551	0.02	0.0310447
14.575	40	0.21	3.0152	0.02	0.0311349

Put option

S	X	r	sigma	time	Black-Scholes
14.575	10	0.21	2.2815	0.02	0.311095
14.575	10.5	0.21	1.8378	0.02	0.152188
14.575	11	0.21	1.5005	0.02	0.11019
14.575	11.5	0.21	1.2454	0.02	0.0890074
14.575	12	0.21	0.9886	0.02	0.0645449
14.575	12.5	0.21	0.7804	0.02	0.0509957
14.575	13	0.21	0.6906	0.02	0.0730808
14.575	13.5	0.21	0.6664	0.02	0.14302

14.575	14	0.21	0.6889	0.02	0.293736
14.575	14.5	0.21	0.7349	0.02	0.535775
14.575	15	0.21	0.7314	0.02	0.806737
14.575	16	0.21	0.8233	0.02	1.58444
14.575	17	0.21	0.8318	0.02	2.44278
14.575	18	0.21	0.9194	0.02	3.39937
14.575	19	0.21	0.9022	0.02	4.36127
14.575	20	0.21	0.8903	0.02	5.34574
14.575	21	0.21	0.9623	0.02	6.33992
14.575	22	0.21	1.024	0.02	7.33467
14.575	23	0.21	1.0685	0.02	8.32969
14.575	24	0.21	1.1158	0.02	9.3251
14.575	25	0.21	1.1573	0.02	10.3207
14.575	26	0.21	1.1947	0.02	11.3163
14.575	27	0.21	1.2274	0.02	12.312
14.575	28	0.21	1.2568	0.02	13.3078
14.575	29	0.21	1.2833	0.02	14.3035
14.575	30	0.21	1.3069	0.02	15.2993
14.575	32.5	0.21	1.358	0.02	17.7888
14.575	35	0.21	1.3977	0.02	20.2783
14.575	37.5	0.21	1.4309	0.02	22.7678
14.575	40	0.21	1.4583	0.02	25.2574

14.575	42.5	0.21	1.4803	0.02	27.7469
14.575	45	0.21	1.4984	0.02	30.2364

Table 2. Black-Scholes numerical solution in Monte Carlo

Call option

MCarlo50	MCarlo1000	MCarlo10000	MCarlo50000
4.65235	4.65235	4.65235	4.65235
4.18337	4.05996	4.1538	4.15782
3.56816	3.70136	3.6403	3.64479
3.08868	3.1357	3.14918	3.15287
2.72959	2.64618	2.64103	2.65809
2.15344	2.17803	2.15466	2.15032
1.7208	1.62676	1.68907	1.67487
1.192	1.27307	1.2569	1.25015
0.886319	0.895331	0.906307	0.90489
0.653438	0.713174	0.682706	0.671256
0.504526	0.406528	0.472554	0.462941
0.302568	0.297165	0.283866	0.282341
0.118608	0.134037	0.14344	0.149526
0.0929043	0.127157	0.119404	0.11407
0.0669014	0.0879925	0.061345	0.0617177
0.037268	0.0412407	0.0308473	0.0310582
0.0517506	0.0277222	0.0355566	0.0348945
0.0254658	0.0243404	0.0348823	0.0323416
0.0722633	0.0376004	0.0299929	0.0323629
0.0182184	0.0454621	0.0344832	0.0314231
0.0171045	0.00851379	0.0341984	0.0301797
0.0461339	0.019146	0.0311904	0.0296153
0.00158227	0.0450202	0.0335682	0.0320004
0.0690685	0.0291123	0.0342634	0.0287716
0.0238286	0.0449729	0.0306608	0.0307597

0.0325739	0.0154194	0.0360878	0.030258
0.0528498	0.0130879	0.0290923	0.0331503
0.00370223	0.0202905	0.0288721	0.0267708
0.0655619	0.0524024	0.0313402	0.0273229
0.039017	0.0221714	0.039819	0.0316094

put option

MonteCarlo50	MonteCarlo1000	MCarlo10000	MCarlo50000
0.2137	0.242663	0.219319	0.225429
0.277804	0.145098	0.15777	0.152949
0.134721	0.100421	0.112334	0.108977
0.119097	0.0903439	0.0902059	0.088398
0.0981901	0.0484329	0.0650068	0.0630125
0.0821978	0.0490903	0.052366	0.050724
0.0746992	0.0792378	0.0718467	0.072564
0.214533	0.157358	0.142136	0.142863
0.131578	0.281672	0.299963	0.292311
0.5239	0.507535	0.532672	0.535283
0.748091	0.820577	0.797485	0.813182
1.86999	1.61453	1.56132	1.58375
2.58932	2.47811	2.46404	2.44783
3.65502	3.33164	3.41722	3.40263
4.04825	4.39862	4.37456	4.35069
4.99796	5.30014	5.33896	5.36887
6.19633	6.24342	6.3221	6.34001
7.76876	7.35538	7.34277	7.33526
8.16181	8.35414	8.33572	8.31181
9.55775	9.25111	9.30802	9.31483
10.3593	10.4759	10.298	10.3065
11.4879	11.3099	11.3094	11.2996
12.2476	12.2967	12.2656	12.3154
13.3147	13.3003	13.3224	13.3173

14.818	14.2383	14.2931	14.3047
14.902	15.2696	15.3347	15.3142
17.7014	17.7177	17.7798	17.8032
20.5232	20.4266	20.2997	20.2786
22.7726	22.6855	22.7491	22.7745
25.7226	25.3068	25.2304	25.2697

Table 3. Black-Scholes numerical solution in binomial tree

Call option

BinTree50	BinTree1000	BinTree10000	BinTree50000
4.64235	4.64255	4.6426	4.6426
4.14293	4.14304	4.14305	4.14305
3.64457	3.64452	3.64458	3.64458
3.14831	3.1483	3.14833	3.14833
2.64896	2.64897	2.64899	2.649
2.15128	2.15125	2.15128	2.15129
1.67633	1.67631	1.67632	1.67633
1.24813	1.2481	1.24814	1.24814
0.90669	0.906676	0.906669	0.906659
0.668906	0.668667	0.668632	0.66862
0.461485	0.461836	0.461769	0.461754
0.277204	0.277314	0.27721	0.277198
0.146444	0.146281	0.146359	0.146354
0.111954	0.112236	0.112263	0.112277
0.0608242	0.060793	0.0608618	0.0608649
0.0338632	0.0338794	0.033968	0.0339701
0.0336088	0.0336481	0.0336978	0.0336985
0.0333677	0.033425	0.0334401	0.0334416
0.0302823	0.0304692	0.030484	0.0304935
0.030493	0.0304619	0.0305489	0.0305507
0.0304482	0.0305168	0.0305383	0.0305453
0.0302739	0.0303825	0.0305136	0.0305159

0.0305143	0.0305776	0.0306154	0.0306171
0.0305124	0.0305291	0.030576	0.0305824
0.0305586	0.0305527	0.0306837	0.0306893
0.0304247	0.0306519	0.030701	0.0307052
0.0307452	0.0306941	0.0308322	0.0308382
0.030755	0.0308982	0.0309416	0.0309449
0.030828	0.0308754	0.0310302	0.031045
0.0310227	0.031078	0.0311305	0.0311353
0.0310688	0.031132	0.0312331	0.031238

Put option

BinTree50	BinTree1000	BinTree10000	BinTree50000
0.222508	0.225031	0.224938	0.224932
0.153442	0.152029	0.152176	0.152185
0.110673	0.110048	0.110191	0.110189
0.0889809	0.0890055	0.0890071	0.0890078
0.063288	0.0644442	0.0645472	0.0645451
0.0499729	0.0509105	0.0509972	0.0509958
0.0723853	0.0731178	0.0730846	0.0730801
0.142535	0.143003	0.143026	0.14302
0.29607	0.293853	0.293745	0.293736
0.536222	0.535828	0.53579	0.535778
0.804097	0.806866	0.806734	0.806736
1.58426	1.58451	1.58443	1.58444
2.44332	2.4428	2.44279	2.44278
3.39906	3.39937	3.39937	3.39937
4.36085	4.36123	4.36127	4.36127
5.34531	5.34573	5.34574	5.34574
6.3397	6.33991	6.33991	6.33992
7.33434	7.33465	7.33467	7.33467
8.32955	8.32968	8.32969	8.32969
9.32496	9.3251	9.3251	9.3251
10.3206	10.3207	10.3207	10.3207

11.3162	11.3163	11.3163	11.3163
12.312	12.312	12.312	12.312
13.3077	13.3078	13.3078	13.3078
14.3035	14.3035	14.3035	14.3035
15.2993	15.2993	15.2993	15.2993
17.7888	17.7888	17.7888	17.7888
20.2783	20.2783	20.2783	20.2783
22.7678	22.7678	22.7678	22.7678
25.2574	25.2574	25.2574	25.2574

Table 4. Black-Scholes numerical solution in explicit finite difference method

Call option

explicit10 10	explicit20 20	explicit200 200	explicit2000 2000
4.6676	4.64012	1.73E+164	nan
4.16763	4.14979	1.06E+140	nan
3.66051	3.65428	2.85E+114	nan
3.15274	3.15095	2.61E+88	nan
2.68306	2.66202	1.13E+52	nan
2.20735	2.16505	2.20E+08	nan
1.73409	1.67317	1.67635	nan
1.27502	1.26589	1.24833	nan
0.842215	0.911424	3.54E+07	nan
0.44434	0.603048	1.09E+32	nan
0.343242	0.447423	3.73E+38	nan
0.302427	0.250262	1.62E+77	nan
0.167061	0.153346	4.85E+90	nan
0.111988	0.115123	2.98E+119	nan
0.0809906	0.0578896	8.85E+124	nan
0.0443477	0.0361063	5.38E+130	nan
0.0365465	0.0336728	2.85E+152	nan
0.0368831	0.028844	2.04E+170	nan
0.0254017	0.0240648	9.86E+182	nan
0.0203698	0.0195948	2.00E+196	nan

0.0175297	0.0135372	7.70E+207	nan
0.0101867	0.00897262	1.29E+218	nan
0.00710948	0.00609445	3.82E+227	nan
0.00480988	0.019408	-2.11E+236	inf
0.000765764	0.0347681	6.68E+243	nan

Put option

explicit 10 10	explicit 20 20	explicit 200 200	explicit 2000 2000
0.271463	0.224174	3.63E+188	nan
0.190018	0.157399	-1.38E+151	nan
0.125888	0.121602	-6.57E+115	nan
0.0807608	0.08757	1.99E+81	nan
0.0965918	0.0769244	3.18E+34	nan
0.113626	0.0658148	5.10E-02	nan
0.134609	0.0659368	7.31E-02	nan
0.168802	0.158105	1.43E-01	nan
0.227459	0.298464	2.93E-01	nan
0.311574	0.470643	5.36E-01	nan
0.689501	0.79181	8.07E-01	nan
1.61596	1.55921	-3.57E+16	nan
2.47429	2.45297	2.53E+24	nan
3.40953	3.40487	-2.43E+52	nan
4.38115	4.36247	2.15E+52	nan
5.35567	5.34838	-1.51E+52	nan
6.34492	6.34139	-1.70E+74	nan
7.33967	7.33518	5.87E+90	nan
8.33226	8.33016	-1.67E+102	nan
9.32662	9.32538	1.52E+113	nan
10.322	10.3208	-1.27E+122	nan
11.317	11.3164	1.32E+127	nan
12.3124	12.3121	2.00E+136	nan
13.308	13.3078	-7.95E+141	nan
14.3035	14.3035	4.64E+145	nan

15.2991	15.2992	6.75E+148	nan
17.7877	17.7883	3.85E+156	nan
20.2758	20.277	1.51E+162	nan
22.7634	22.7654	4.75E+166	nan
25.2506	25.2533	1.95E+170	nan

Table 5. Black-Scholes numerical solution in implicit finite difference method

Call option

implicit10 10	implicit20 20	implicit100 100	implicit200 200
4.65235	4.64235	4.64212	4.64235
4.13678	4.14098	4.14292	4.14291
3.64055	3.642	3.64424	3.64425
3.14701	3.14579	3.14809	3.14833
2.64873	2.64786	2.649	2.64885
2.15184	2.15139	2.15109	2.1513
1.6713	1.67377	1.67653	1.67637
1.25293	1.24446	1.24866	1.24817
0.915577	0.902191	0.906632	0.906251
0.662274	0.667004	0.669348	0.669229
0.475801	0.468597	0.462941	0.461472
0.273664	0.282572	0.278137	0.277166
0.147067	0.149644	0.146813	0.145517
0.101805	0.103628	0.112392	0.112473
0.061652	0.0608706	0.0602689	0.0608875
0.0229546	0.0314312	0.0338433	0.0338829
0.0237426	0.0320292	0.0333568	0.0334855
0.0246872	0.0322204	0.0323438	0.0330048
0.023904	0.0293373	0.0300723	0.0303773
0.0245548	0.0290223	0.0302828	0.0302574
0.0250395	0.0284206	0.0302396	0.0299109
0.0253966	0.0276008	0.0300025	0.030158

0.0257514	0.0267864	0.029767	0.0304037
0.0259181	0.0256459	0.0292433	0.0304087
0.0261206	0.024579	0.029482	0.0304613
0.0261832	0.0232717	0.0297621	0.030335
0.0261616	0.0241651	0.0302785	0.0302054
0.0258141	0.0250533	0.0304612	0.0306187
0.0252126	0.0257729	0.0303821	0.0307834
0.0243965	0.0263481	0.0300807	0.0307367

Put option

implicit 10， 10	implicit 20， 20	implicit 100 100	implicit 200 200
0.31234	0.29999	0.22543	0.22494
0.13253	0.13108	0.15237	0.15154
0.07891	0.09889	0.11055	0.11034
0.05985	0.08102	0.08907	0.08919
0.04548	0.06061	0.06372	0.06439
0.03596	0.04794	0.05032	0.05087
0.07420	0.07158	0.07284	0.07313
0.15547	0.14594	0.14235	0.14317
0.27607	0.30403	0.29303	0.29345
0.56559	0.52936	0.53650	0.53639
0.81386	0.82057	0.80801	0.80618
1.60101	1.58740	1.58287	1.58481
2.42173	2.43265	2.44311	2.44273
3.39272	3.40005	3.39869	3.39940
4.35661	4.35799	4.36110	4.36118
5.34118	5.34306	5.34563	5.34566
6.33698	6.33821	6.33980	6.33980
7.33279	7.33368	7.33454	7.33459
8.32860	8.32902	8.32958	8.32965
9.32441	9.32442	9.32504	9.32508
10.32020	10.32020	10.32060	10.32060
11.31600	11.31600	11.31630	11.31630

12.31180	12.31180	12.31200	12.31200
13.30760	13.30760	13.30770	13.30770
14.30350	14.30350	14.30350	14.30350
15.29930	15.29930	15.29930	15.29930
17.78880	17.78880	17.78880	17.78880
20.27830	20.27830	20.27830	20.27830
22.76780	22.76780	22.76780	22.76780
25.25740	25.25740	25.25740	25.25740

Table 6. Numerical solution in binomial tree method of American option

American call option

S	X	r	sigma	time	BinTree50000
14.575	10	0.21	1.4094	0.02	4.6426
14.575	10.5	0.21	1.2383	0.02	4.14305
14.575	11	0.21	1.0838	0.02	3.64458
14.575	11.5	0.21	0.9506	0.02	3.14833
14.575	12	0.21	0.7994	0.02	2.649
14.575	12.5	0.21	0.6621	0.02	2.15129
14.575	13	0.21	0.613	0.02	1.67633
14.575	13.5	0.21	0.6165	0.02	1.24814
14.575	14	0.21	0.6601	0.02	0.906659
14.575	14.5	0.21	0.7313	0.02	0.66862
14.575	15	0.21	0.7526	0.02	0.461754
14.575	16	0.21	0.9005	0.02	0.277198
14.575	17	0.21	0.9621	0.02	0.146354
14.575	18	0.21	1.1134	0.02	0.112277
14.575	19	0.21	1.1457	0.02	0.0608649
14.575	20	0.21	1.1811	0.02	0.0339701
14.575	21	0.21	1.3255	0.02	0.0336985
14.575	22	0.21	1.4599	0.02	0.0334416
14.575	23	0.21	1.565	0.02	0.0304935

14.575	24	0.21	1.6845	0.02	0.0305507
14.575	25	0.21	1.797	0.02	0.0305453
14.575	26	0.21	1.9035	0.02	0.0305159
14.575	27	0.21	2.006	0.02	0.0306171
14.575	28	0.21	2.1025	0.02	0.0305824
14.575	29	0.21	2.1961	0.02	0.0306893
14.575	30	0.21	2.2849	0.02	0.0307052
14.575	32.5	0.21	2.4927	0.02	0.0308382
14.575	35	0.21	2.6817	0.02	0.0309449
14.575	37.5	0.21	2.8551	0.02	0.031045
14.575	40	0.21	3.0152	0.02	0.0311353
14.575	42.5	0.21	3.1641	0.02	0.031238
14.575	45	0.21	3.3035	0.02	0.0313676
14.575	47.5	0.21	3.4322	0.02	0.0313401

American put option

S	X	r	sigma	time	BinTree50000
14.575	10	0.21	2.2815	0.02	0.225212
14.575	10.5	0.21	1.8378	0.02	0.152414
14.575	11	0.21	1.5005	0.02	0.11039
14.575	11.5	0.21	1.2454	0.02	0.0892056
14.575	12	0.21	0.9886	0.02	0.0647286
14.575	12.5	0.21	0.7804	0.02	0.0511868
14.575	13	0.21	0.6906	0.02	0.0734243
14.575	13.5	0.21	0.6664	0.02	0.14383
14.575	14	0.21	0.6889	0.02	0.295625
14.575	14.5	0.21	0.7349	0.02	0.539518
14.575	15	0.21	0.7314	0.02	0.813307
14.575	16	0.21	0.8233	0.02	1.59876
14.575	17	0.21	0.8318	0.02	2.47049
14.575	18	0.21	0.9194	0.02	3.43915
14.575	19	0.21	0.9022	0.02	4.425
14.575	20	0.21	0.8903	0.02	5.425

14.575	21	0.21	0.9623	0.02	6.425
14.575	22	0.21	1.024	0.02	7.425
14.575	23	0.21	1.0685	0.02	8.425
14.575	24	0.21	1.1158	0.02	9.425
14.575	25	0.21	1.1573	0.02	10.425
14.575	26	0.21	1.1947	0.02	11.425
14.575	27	0.21	1.2274	0.02	12.425
14.575	28	0.21	1.2568	0.02	13.425
14.575	29	0.21	1.2833	0.02	14.425
14.575	30	0.21	1.3069	0.02	15.425
14.575	32.5	0.21	1.358	0.02	17.925
14.575	35	0.21	1.3977	0.02	20.425
14.575	37.5	0.21	1.4309	0.02	22.925
14.575	40	0.21	1.4583	0.02	25.425
14.575	42.5	0.21	1.4803	0.02	27.925
14.575	45	0.21	1.4984	0.02	30.425
14.575	47.5	0.21	1.5142	0.02	32.925

Table 7. Numerical solution in explicit FDM method of American option

American put option

FDM EX 10	FDM EX 20	FDM EX 100	FDM EX200
0.271636	0.224332	1.71E+44	3.83E+148
0.190094	0.157585	-4.075	-4.075
0.125917	0.121694	-3.575	-3.575
0.0808651	0.0876486	0.272587	1.22E+55
0.0967486	0.0770379	0.0647022	1.98E+20
0.113696	0.0658942	0.0509992	0.0511809
0.134653	0.066039	0.074007	0.073433
0.168842	0.1586	0.144248	0.144016
0.227729	0.299188	0.293807	0.295412
0.313874	0.472194	0.539412	0.539894
0.692306	0.795294	0.813709	0.813232
1.61982	1.57009	1.59817	1.29E+07

2.47883	2.4711	2.47037	6.39E+11
3.4332	3.43597	3.43897	1.36E+30
4.425	4.425	4.425	1.98E+30
5.425	5.425	5.425	1.72E+30
6.425	6.425	6.425	1.60E+45
7.425	7.425	27.2226	4.17E+59
8.425	8.425	210876	8.425
9.425	9.425	1.03E+09	6.14E+78
10.425	10.425	1.02E+13	10.425
11.425	11.425	1.18E+16	5.14E+93
12.425	12.425	9.26E+18	1.67E+99
13.425	13.425	5.59E+20	13.425
14.425	14.425	2.57E+22	2.54E+108
15.425	15.425	15.425	15.425
17.925	17.925	17.925	17.925
20.425	20.425	20.425	20.425
22.925	22.925	22.925	22.925
25.425	25.425	25.425	25.425

Table 8. Numerical solution in implicit FDM method of American option

American put option			
FDM IN 10	FDM IN20	FDM IN 100	FDM IN 200
0.0246463	0.0177993	0.0242112	0.0256316
0.023181	0.0178893	0.022428	0.023972
0.0226994	0.0195644	0.0213696	0.0235232
0.0238995	0.0239306	0.0236072	0.0251829
0.0221137	0.0235546	0.0232184	0.0237446
0.0212015	0.0245754	0.0242078	0.0239952
0.0351479	0.0419567	0.0470163	0.0470936
0.129673	0.122095	0.119228	0.117277
0.257366	0.284698	0.279508	0.274882
0.567881	0.530434	0.542439	0.536901
0.838148	0.845976	0.83224	0.830166

1.66826	1.64916	1.65235	1.6481
2.52667	2.52384	2.5244	2.52227
3.47181	3.48634	3.48942	3.49026
4.448	4.44916	4.44756	4.4468
5.42521	5.425	5.42872	5.42906
6.425	6.425	6.42854	6.42868
7.425	7.425	7.42814	7.42829
8.425	8.425	8.42632	8.42683
9.425	9.425	9.42581	9.42666
10.425	10.425	10.4252	10.4264
11.425	11.425	11.425	11.4262
12.425	12.425	12.425	12.4261
13.425	13.425	13.425	13.4259
14.425	14.425	14.425	14.4257
15.425	15.425	15.425	15.4256
17.925	17.925	17.925	17.9253
20.425	20.425	20.425	20.4251
22.925	22.925	22.925	22.925
25.425	25.425	25.425	25.425