A Computational Analysis of the Black-Scholes Equations

by

Yifan Wang

Alessandro Veneziani

Advisor

Department of Mathematics and Computer Science

2013

A Computational Analysis of the Black-Scholes Equations

by

Yifan Wang

Alessandro Veneziani

Advisor

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Mathematics and Computer Science

2013

Abstract

A Computational Analysis of the Black-Scholes Equations

by Yifan Wang

This paper explores the most decorated option pricing model in recent history of the financial industry: the Black-Scholes Equation. We will first study the framework of the Black-Scholes Equation in detail by introducing its object of evaluation, distinguished assumptions, and deduction of the Black-Scholes partial differential equation. Although Black and Scholes(1973) has proposed the famous Black-Scholes formula to evaluate the European option, the PDE form has proposed struggle in finding the exact analytical solution, thus giving rise to the enormous interest in the numerical approach. In the second part of this paper, we will introduce three primary numerical and simulation methods including Finite Element Method(FEM), Finite Difference Method(FDM) and Monte Carlo Simulation(MC). We will discuss extensively about each method and present its advantages and shortcomings. In general, FEM are better founded mathematically on extensive theoretical analysis. Nevertheless, FDM and MC can have some advantages, in particular in terms of the easiness of implementation. We will consider some of these aspects in the present paper.

A Computational Analysis of the Black-Scholes Equations

by

Yifan Wang

Alessandro Veneziani

Advisor

a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Mathematics and Computer Science

2013

Acknowledgement

I would like to thank Dr. Alessandro Veneziani for being my advisor for this honor thesis. I am truly appreciative of his commitment and assistance along the whole process. I would also like to thank my committee member Dr. James Nagy and Dr. Kaiji Chen for their comments on the work.

Table of Contents

List of Tables and Graphs

Appendix

Outline of the work

I chose to write this paper because of my genuine interest in option pricing. The first chapter will cover the essential components in the Black-Scholes Pricing Model. It begins with the concept of an option, its category and characteristics. Then we introduce the Brownian Motion and Ito's Lemma to lay the foundation for our eventual deduction of the famous Black-Scholes Equation. Finally, we observe a specific case of plain vanilla European option and derive the analytical solution using the Black-Scholes Formula. The chapter will be followed by our discussion of preferable numerical methods to solve the Black-Scholes PDE. We show the discretization of the Black-Scholes equation using FEM and FDM and test them in specific given cases; meanwhile, we will analyze and compare the error behavior of both methods. In particular, we have created a code to test two proposed techniques--domain truncation and domain transformation-- in FEM in the Matlab environment. We will also show how the general Monte Carlo method can applied in the specific Black-Scholes framework and test various cases with our solver.

1.Introduction to Black-Scholes Pricing Model

The idea of options originated a long way back, but it was not until 1973 that options were officially traded in the financial market. Before the establishment of the Black-Scholes pricing model[2], there was little knowledge on how to price an option.

The risk and randomness in the derivative market were determined that it cannot be tamed until the appearance of Fischer Black and Myron Scholes's historical finding of the option pricing formula. They together developed the Black-Scholes model. Robert Merton, inspired by Japanese scientist Ito, confirmed the equation and added extra mathematical understanding to the celebrated Black-Scholes Option Pricing framework.

1.1 Basics about options

An option is a contract first and foremost, and is normally sold at a negotiated price. Investors, who has the right to an option, can buy or sell its underlying assets, which can be in varied forms. The option has a strike price, and the investor can exercise their rights to claim the assets or cash. Every option contract sets an expiration date, namely maturity. Generally speaking, there are two kinds of options, a call and a put. A *call option* gives the holder the right to buy the underlying asset by a certain date for a certain price. A *put option* grants the right to sell the underlying.

There are several basic elements of an option that we ought to know. Strike price, denoted as K, is the fixed price at which investor can buy or sell the underlying. Time to expiration, $T - t$, is tied to the notion that an option expires if not exercised. Exercising the option means using the right to buy or sell the underlying.

We now introduce two primary options categorized by their exercise styles. *European option*, which is our primary research object of the Black-Scholes model, is a contract that can only be exercised at pre-determined expiration date. *American* *option*, on the other hand, can be exercised on that date or any date before the maturity. Some more complicated derivatives like *Asian option* can have varied pre-specified strike price or final price.

Option price is also called *option premium*. The amount is paid when the option contract is established. The payoff value changes over time for American options, but both option types have the same payoff at time of expiration because they are equivalent at that point. The final price is denoted as S_T.

For European and American call, at expiration, its value is denoted as

$$C_t = Max(0, S_T - K) \tag{1.1}$$

For European and American put, at expiration, its value is denoted as

$$P_t = Max(0, K - S_T) \tag{1.2}$$

In the next chapters, we will see more precisely the mathematical foundation of the problem.

1.2 Black-Scholes Pricing

To study the theoretical background for Black-Scholes equation, we need to have

1) a methodology to measure the price movement of a claim on the asset as a function of the price movement of the asset,

2) a model for the price movement of the underlying asset.

1.2.1 Brownian Motion

Brownian Motion is our model for price movements. It is a stochastic model which gives probability for stock price being in a certain range at a certain time. The concept of random walk was used to explain the behavior of Brownian Motion. We are not going to details of discrete random walk, but the assumptions are the same[7]. We will have a continuous random walk if we move $\sqrt{h}$ per h units of time when we take $h \to 0$. $Z(t)$ is the symbol for Brownian Motion. The properties of Brownian Motion are listed below

1. $Z(0) = 0$
2. $Z(t+s)|Z(t) \sim N(Z(t), s)$

3 $Z(t+s_1) - Z(t)\ is\ independent\ of\ Z(t) - Z(t-s_2),$

$Z(t)\ has\ independent\ increments$

4. $Z(t) is\ continous\ in\ t$

For example, if the price of a stock follows Brownian Motion and is 58 at time 4, the probability that the price of the stock is at least 60 at time 8 is the following.

$$\Pr(Y > 60) = 1 - N\left(\frac{60-58}{\sqrt{8-4}}\right) = 1 - N(1) = 0.1587$$

We will further discuss *Geometric Brownian Motion(GBM)*. Geometric model denotes that the logarithm of the variable better defines the pattern of the data than the original variable does. We attempt to describe the behavior of the underlying stock, so we choose Geometric model to make sure that stock price cannot go negative and move proportionally along with the stock price.

We say $S(t)$ follows GBM if $lnS(t)$ follows Arithmetic Brownian Motion. $S(t)$ is a lognormal random variable, μ is the drift rate, σ is the volatility,

$$E[S(t)] = S_0 e^{\mu t + 0.5\sigma^2 t} \tag{1.3}$$

where $r = \mu + 0.5\sigma^2$ is continuously compounded interest rate.

1.2.2 Ito's Lemma

Ito's Lemma introduces a stochastic differential equation

$$dS = a(S(t), t)dt + b(S(t), t)dZ(t) \tag{1.4}$$

$a(S(t), t)$ and $b(S(t), t)$ can vary with S and t or be constant, $Z(t)$ follows Brownian Motion described by 1.2.1. The SDE is also called an Ito process, and describes a variable S that changes on a deterministic trend over time, but also entertains randomness going forward. In this equation, $a(S(t), t)$ and $b(S(t), t)$ can be either constant or variables that relies on $S(t)$, the price at time t and time t itself.

This equation is found very convenient in our evaluation of stock price, which follows a GBM. As investors would certainly expect a growth rate over a period of time, the randomness of the stock market is unpredictable and is fairly captured by the inclusion of $dZ(t)$. In this case, $a(S(t), t)$ would be the product of drift rate multiplied by the stock price, and $b(S(t), t)$ is the volatility times the underlying stock price.

$$a(S(t), t) = \mu S$$

$$b(S(t), t) = \sigma S$$

$$\frac{dS}{S} = \mu dt + \sigma dZ(t) \tag{1.5}$$

This differential equation is widely known as one of the more accurate model for the stock price behavior.

Derivation

Ito's Lemma states that a function $C(S,t)$of underlying stock price(S) and time(t) can be written in the form of

$$dC = C_s dS + C_t dt \tag{1.6}$$

Using Taylor expansion, we have

$$dC = \frac{\partial C}{\partial S} dS + \frac{\partial C}{\partial t} dt + 0.5 \frac{\partial^2 C}{\partial^2 S} dS^2 + 0.5 \frac{\partial^2 C}{\partial^2 t} dt^2 + \frac{\partial^2 C}{\partial S \partial t} dSdt + \cdots.$$

In ordinary calculus, the second order terms such as $dt^2 = 0, dSdt = 0$. Normally, We replace dS with the stochastic partial differential equation(1.4), but $dZ * dZ$ do not equal to 0 since dZ is a stochastic term. Surprisingly, according to one of the property of Brownian Motion, the expectation of variance of $Z \; is$

$$E(Z^2) = t$$

$$dZ * dZ = dt$$

Thus,

$$dC = \frac{\partial C}{\partial S} dS + \frac{\partial C}{\partial t} dt + 0.5 \frac{\partial^2 C}{\partial^2 S} (adt + bdZ)^2$$

Expand the parentheses, we will derive

$$dC = \frac{\partial C}{\partial S} dS + \frac{\partial C}{\partial t} dt + 0.5 * \frac{\partial^2 C}{\partial^2 S} (b^2 dt) \tag{1.7}$$

or we can write in the form of a Ito process(1.8)

$$dC = \frac{\partial C}{\partial S} (adt + bdZ) + \frac{\partial C}{\partial t} dt + 0.5 \frac{\partial^2 C}{\partial^2 S} (b^2 dt)$$

$$dC = (\frac{\partial C}{\partial S} a + \frac{\partial C}{\partial t} + 0.5 \frac{\partial^2 C}{\partial^2 S} b^2) dt + \frac{\partial C}{\partial S} bdZ \tag{1.8}$$

The universal property of the Ito's lemma depends on the fact that C is a function of its underlying asset, no matter it is one stock, a collection of stocks, or an investment portfolio.

With Ito's lemma, we prove that traditional derivatives like futures and forward options follows the GBM. Whenever we have a derivative whose underlying asset follows GBM, we can infer that the derivative follows Ito's lemma. One interesting phenomenon is that we can calculate the expected growth of the option if we can predict the growth rate of the underlying stock. This relationship is widely applied in derivative pricing, which we will further discuss in the next section.

1.2.3 Assumptions of Black-Scholes Equation

Before Black, Scholes and Merton proceeded to derive the actual Black-Scholes partial differential equation, they made important assumptions that validify the statement. Hull(2011) has listed seven assumptions before he derives the equation. However, this paper will integrate the assumptions with another approach.

We cannot compute the value of an option, namely C, if the eventual partial differential equation has dZ as a component. As a result, the idea is to construct a risk-free portfolio and cancel out the randomness of dZ. Techniques like delta hedging are applied in such portfolio construction. *Delta* is the sensitivity of the option price to the price movement of the underlying asset. *Delta hedging* is a specific technique that aims at reducing the risk of the price movement of the underlying asset, namely keeping the delta to zero or as close to zero as possible[7]. Assumptions will be put in the context of those steps.

1. Buy the option.

Assume $C(t)$ is the value of the option, and $S(t)$ is the value of the underlying asset. We mentioned the mostly known stock price model

$$\frac{dS}{S} = \mu dt + \sigma dZ \tag{1.9}$$

The stock price follows a geometric Ito process. The assumption we made here by using this model is that the expected rate of return μ and volatility σ are constant. In this step, we will buy positive asset $C(t)$.

2. Hedge the option by selling delta shares of the asset

We intend to erase the randomness by hedging certain amount of underlying assets. To eliminate the stochastic component dZ, we sell delta shares of stocks, which is $C_s S$. C_s is delta.

3. Net proceeds(Sum of step 1 and 2) are lent at the risk-free rate

Net proceeds of previous two steps is their net lending, which is $C_s S - C$ in this case. The risk-free loan will pay $(C_s S - C) r dt$ over the time difference.

The most essential assumption of Black-Scholes pricing model is built on the no-arbitrage principle. It essentially defines the portfolio that we constructed as a risk-free portfolio. Thus, no matter how the price of assets in the portfolios changes over time, the overall portfolio should still worth zero.

1.3 The Black-Scholes Equation

Assume the value of portfolio is *f*, we sum up step 1,2 and 3, for a risk-free portfolio,

$$df = dC - C_s dS + (C_s S - C) r dt = 0 \tag{1.10}$$

If the underlying asset S pays dividend at the rate of δ,

$$dC - C_s dS - C_s \delta dS + (C_s S - C) r dt = 0 \tag{1.11}$$

We apply Ito's Lemma on dC,

$$dC = C_s dS + 0.5 C_{ss} (dS)^2 + 0.5 S^2 \sigma^2 (dt)$$

Then we plug in (1.6) the equation of dC, now we have

$$C_t dt + 0.5 C_{ss} (dS)^2 + (C_s S - C) r dt = 0 \tag{1.12}$$

Since we can calculate second derivative $(dS)^2$ as we previously examined in our discussion of Brownian Motion,

$$(dS)^2 = S^2 \sigma^2 dt$$

We finally get the Black-Scholes Equation,

$$C_t + 0.5 S^2 \sigma^2 C_{ss} + (C_s S - C) r dt = 0 \tag{1.13}$$

or the more renown form

$$C_t + 0.5 S^2 \sigma^2 C_{ss} + C_s S r dt = rC \tag{1.14}$$

Adding the dividend , we will have

$$C_t + 0.5 S^2 \sigma^2 C_{ss} + C_s S (r - \delta) dt = rC \tag{1.15}$$

This is the model we will be utilizing for our computational analysis in the later chapters.

1.4 The Black-Scholes Formula

The Black-Scholes Formula can be observed as a specific solution of the general

Black-Scholes Equation that evaluates the plain vanilla European option. The general formulas for calls and puts are

$$C = S_0 N(d_1) - Ke^{-rT} N(d_2) \tag{1.16}$$

$$P = Ke^{-rT} N(-d_2) - S_0 N(-d_1) \tag{1.17}$$

where

$$d_1 = \frac{\ln\left(\frac{S_0}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \tag{1.18}$$

$$d_2 = \frac{\ln\left(\frac{S_0}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \tag{1.19}$$

In the solution, function $N(d_1)$ and $N(d_2)$ are Gaussian cumulative probability function[10]. Particularly, $N(d_2)$ is the probability of the call option being in the money at expiration. An interpretation of this formula can be derived from put-call parity, which has the below identity function at the end time,

$$C(S,t) - P(S,t) = S - Ke^{-rt} \tag{1.20}$$

At expiration, we exercise either call or put. Consider a call is in the money at expiration, the risk-neutral expectation of the option holder making the payment is $KN(d_2)$, and correspondingly, the risk neutral expectation of the asset price would be $S_0 e^{rt} N(d_1)$. Thus the call value at the end time would be

$$S_0 e^{rt} N(d_1) - KN(d_2)$$

Discounted back in time, we have

$$C(S,t) = S_0 N(d_1) - Ke^{-rt} N(d_2) \tag{1.21}$$

We remind readers that C and P are functions of the underlying asset price and time; thus, Ito's Lemma is applicable as we discussed previously.

Further details about the derivation of the Black-Scholes Formula can be found in Hull(2011, Ch14)[7] and Wilmott(2000)[11].

2. Finite Difference Method

Introduction to Numerical Methods

In real life, we always have problems that we cannot solve using an analytic approach, which means we cannot achieve analytical result. However, in the field of physics, applied mathematics and engineering, we can approximate results by applying numerical schemes. Partial differential equations, which are constructed for such problems, are primary objects to which we apply those numerical methods.

We have introduced the Black-Scholes equation and have derived the equation using analytical methods; as a result, we are focused on introducing the numerical approach to the Black-Scholes problem in the second part of the paper. In terms of the field of mathematical finance, the Finite Difference Method(FDM) and Finite Element Method(FEM) are well suited for the Black-Scholes option pricing model, and has proved to have better approximation than other numerical schemes[5]. In the next two chapters, we are going to discuss extensively about these two methods and later compare their results when we attempt to examine the test cases.

2.1 Basics of Finite Difference(basic schemes, errors).

The basic idea of FDM is to "collocate" the PDE only in some points of the region of interest and then to replace the derivatives with incremental quotients. This leads in general to an algebraic system of equations. The size of the system depends on the number of nodes selected. For example, we attempt to evaluate $u'(x)$, the first derivative of function $u(x)$ at point $\bar{x}$. The first order finite difference approximation of $u'(x)$ would be

$$\frac{u(\bar{x}+h)-u(\bar{x})}{h} \approx u'(\bar{x}) \tag{2.1}$$

or an alternate second-order approximation

$$\frac{u(\bar{x}+h)-u(\bar{x}-h)}{2h} \approx u'(\bar{x}) \tag{2.2}$$

In approximation(1), $\bar{x}$ and $\bar{x}+h$ are two adjacent nodes that we take. When h is small enough, we learn from basic calculus that the approximation would be almost accurate as the analytical solution of the derivative. h is the step size in our discussion of the numerical approximation, which has several properties and deductions that we ought to remember in convenience for our later discussion.

1. h is assumed to be very small

2. h can be either positive or negative

Finite difference (2.1),

When $h > 0$, we refer our approximation as *forward difference*

When $h < 0$, we refer the approximation as *backward difference*

Finite difference (2.2),

is *centered difference approximation*

For more details, see e.g. Quarteroni(2010)[9]

Numerical approximation results have errors, which is defined as the difference between the analytical solution and our approximations. The source of error can be divided in two major categories: round-off error and truncation error or discretization error.

Round-off error is strictly from the need of truncating numbers when string them in the computer[3]. Since most computers are 64 bits nowadays, rounding error is mostly considered insignificant. However, one of the consequences are that when discretization error decreases, it may be more apparent.

Discretization error, or truncation error results from the omission of Taylor's series approximation of the exact derivatives. Taylor expansion of forward difference is

$$f(x_0+h)=f(x_0)+\frac{f'(x_0)}{1!}h+\frac{f''(x_0)}{2!}h^2+\cdots\ldots\frac{f^n(x_0)}{n!}h^n+O(h^n) \qquad (2.3a)$$

or

$$f(x_0+h)=f(x_0)+\sum_{k=1}^{\infty}\frac{d^kf}{dx^k}|_{x=x_0^{}}h^k \qquad (2.3b)$$

For the approximation of first order accuracy, we ignored the remainder of the series after the second term.

$$f(x_0+h)=f(x_0)+\frac{f'(x_0)}{1!}h+\ O(h^2)$$

Last error term indicates that the error is proportionate to h^2; thus we say that this numerical approximation is first order accurate. If we apply centered difference scheme, we have Taylor expansion,

$$f(x_0+h)=f(x_0)+\frac{f'(x_0)}{1!}h+\frac{f''(x_0)}{2!}*h^2+\frac{f'''(x_0)}{3!}*h^3+\cdots.. \qquad (2.4)$$

$$f(x_0-h)=f(x_0)-\frac{f'(x_0)}{1!}*h+\frac{f''(x_0)}{2!}*h^2-\frac{f'''(x_0)}{3!}*h^3+\cdots.. \qquad (2.5)$$

By substituting (5) to (4), we have

$$\frac{f(x_0+h)-f(x_0-h)}{2h}=f'(x_0)+\frac{f'''(x_0)}{6}*h^2+\cdots.. \qquad (2.6)$$

$$\left|\frac{f(x_0+h)-f(x_0-h)}{2h}-f'(x_0)\right|\le\frac{f'''(x_0)}{6}*h^2=O(h^2) \qquad (2.7)$$

From (2.7), we know that centered difference scheme is second order consistent approximation of the first derivative of given function $f(x)$ at point x_0.

In principle, we can use the Taylor expansion to derive the accuracy of high order numerical approximation as long as the function is still differentiable at that point. That is

$$f \in C^n(\mathrm{R})$$

Detailed discussion can be found in e.g. [FSV][5]. In this paper, we will focus on how the Finite Difference Method is applied in the numerical approximation of the Black-Scholes Equation.

2.2 Finite Difference Method in Black-Scholes Equation

In this section, we will examine a specific case in European call option to further interpret the application of FDM in finance.

We construct the problem as an initial boundary value problem(IBVP). To solve such problem, we ought to have an initial condition, boundary condition and domain of our variables. In this case, we have a terminal condition at the time of expiration from (1.1), but we do not have an initial condition at the time 0. Since we know the final value of C at the time of expiration, we can solve the Black-Scholes equation by reversing the time variable, which essentially makes the final condition become our new initial condition. The boundary condition is 0 on the left, and on the right side, we have boundary condition of $S - Ke^{-rt}(1.20)$ according to put-call parity at the time of expiration. The second boundary condition also marks an exponential growth from the initial time to expiration. To sum up, we have the Black-Scholes equation as we previously discussed,

$$C_t + 0.5S^2\sigma^2 C_{ss} + (C_s S - C)rdt = 0 \tag{2.8}$$

Final condition(expiration time):

$$C_t = Max(0, S_T - X) \tag{2.9}$$

Boundary Condition:

$$C(0,t) = 0 \tag{2.10}$$

$$C(S,t) = S - Ke^{-rt} \tag{2.11}$$

Since we cannot solve the case of $S \to \infty$, we manually set up a domain $[0, S_{max}] * [0, T]$ to solve for values of S. The finite difference method essentially allow us to divide our domain space into several subspaces. For example, we take N_s nodes and N_t time steps,

$$\Delta S = \frac{S}{N_s} \quad , \Delta t = \frac{T}{N_t}$$

We let

$$C_i^j = C(\Delta S * i, \Delta t * j) = C(S_i, T_j),$$

$$i = 0,1, \ldots\ldots N_s; j = 0,1, \ldots\ldots N_t$$

We will see later that depending on the choice of time advancing scheme, we may have some restrictions on the time step Δt.

2.3 Explicit Euler Discretization for Time Advancing and Space Discretization

There are several way for implementing FDM. Explicit Euler, Implicit Euler and Crank-Nicholson schemes are obtained by forward, backward and centered difference.

In this section, we will focus on Explicit Euler Method for the time-advancing parabolic problems, which is the most common method in transforming the original Black-Scholes Equation.

We use forward difference at time t_j

$$\left.\frac{\partial C}{\partial t}\right|_{t=t_{j-1}} = \frac{C_i^j - C_i^{j-1}}{\Delta t} \tag{2.13}$$

The second part of the equation is the diffusion term, we use the centered difference of second order

$$\left.\frac{\partial^2 C}{\partial S^2}\right|_{s_i} = \frac{C_{i+1}^j - 2C_i^j + C_{i-1}^j}{\Delta S^2} \tag{2.14}$$

The third part of the equation is the advection term.

$$\left.\frac{\partial C}{\partial S}\right|_{s_i} \approx \frac{C_{i+1}^j - C_{i-1}^j}{2\Delta S} \tag{2.15}$$

Now, we rewrite the Black-Scholes Equation

$$\frac{C_i^j - C_i^{j-1}}{\Delta t} - 0.5\Delta S^2\sigma^2\frac{C_{i+1}^j + C_{i-1}^j - 2C_i^j}{\Delta S^2} - \frac{C_{i+1}^j - C_{i-1}^j}{2\Delta S}\Delta Sri + rC_i^j = 0 \tag{2.16}$$

or in the matrix form

$$C^j = \frac{1}{\Delta t}(I + K)C^{j-1}$$

for $i = 1,2,\ldots..N_S - 1, j = 1,\ldots.N_T - 1, N_T$ since we have inverted the time and consider the final condition as the new initial condition.

2.4. Possible Alternatives

Readers may wonder how effective the application of other numerical schemes in the Finite Difference are on the Black-Scholes equations.

The implicit method, the Backward Euler method is unconditionally stable. That means we are not restricted in the use of N_s and N_T, and we can take a large number of nodes so that we do not need as many time steps as we do in the explicit methods. The main drawback is computational efficiency. A linear system of partial differential equations need to be solved at each time step instead of simple iterations given the initial node value in the explicit scheme.

Crank-Nicholson method is numerically stable in solving heat equation. It is based on centered difference scheme in space, and the trapezoidal rule in time. The biggest advantage of Crank-Nicholson method minimized systematic discretization error: it is in fact second order accurate. Backward Euler or Forward Euler all exclusively use derivatives evaluated at a time point like $t - \Delta t$ or t to approximate the next node. Even though the general rule is that error tends to be 0 if we choose a very large number of time steps, we realize that a weighted average of the two methods may makes more sense in terms of accuracy because the backward and forward scheme produce errors in opposite directions.

3. Finite Element Method

3.1 Basics of the Finite Element Method

We discuss the Finite Element Method(FEM) because we believe it may be an alternative or better numerical approach to the Finite Difference Method(FDM) in solving the Black-Scholes equation. The idea of the Finite Element Method is to cut the entire space or structure into several pieces or elements, where we postulate the solution to have a manageable mathematical form(e.g. linear or polynomial). Comparable ideas are present everywhere in life; LEGO products are built upon simple building blocks that are small and manageable. The Finite Element Method(FEM) manages to divide the structure into elements with nodes, which is defined by local continuous polynomials that connects the nodes. The finite elements ought to cover the domain completely and are connected so that the entire domain is interpolated as a overall piecewise function. To simplify the notion, we take an example of a one-dimensional(1D) case; a straight line is divided into a certain number of subintervals. Those intervals are the finite elements that constructed the line.

There are two general steps in applying the FEM. We mentioned that the first is to create non-overlapping elements. Through selective cutting and choice of geometry, we can easily make the interpolation of local polynomial given its basic shape. More advantages of FEM will be discussed later in the chapter. A more mathematical analysis and foundational theory of FEM is well-documented in Chapter 2 of

[FSV][5].

Then we rewrite the approximate function $\tilde{u}$ as the following

$$\tilde{u} = \sum_{i=1}^{N+1} \tilde{u}_i \varphi_i \tag{3.1}$$

φ_i is the basis function or shape function. $\tilde{u}_i$ are the weights to be determined that satisfies (3.1). We choose φ_i in the hopes of accurately approximating the local polynomials that we choose to cut out. A possible choice of polynomial here would be the piecewise linear functions. The major success of FEM is due to the fact that we can approximate a smooth function locally using polynomials without losing much approximation property.

One important feature is that adjacent elements share the degree of freedom at the connecting nodes across the edge of the gird. In the particular case of Lagrangian finite elements, the degree of freedom are values of function v_h at a specific node N_i.[5]

3.2 Finite Element Method in Black-Scholes Equation

After time inversion, the Black-Scholes equation has the parabolic form

$$u_t - Lu = f, \qquad x \in \Omega, 0 < t < T \tag{3.2}$$

with the initial condition $u(x, 0) = \max(x - K, 0)$. f is a given function and L is a linear differential operator that is defined on $u = u(x, t)$.

We mentioned the similarity between the heat equation and the Black-Scholes equation. In the one-dimensional case, we can solve the Black-Scholes equation by applying similar approach. We introduce weak formulation. or variational from, which is more general to its strong counterpart in the purpose of reducing the regularity required for the unknown solution u (Quarteroni, 2011).

To derive the variational form, we multiply the entire differential equation by a test function $v(x)$, and integrate the function on the domain Ω. We denote $(u, v) = \int_\Omega u(x)v(x)dx$, Given that V is a Hilbert space and our initial condition is the final payoff of an European call option, we have a general form,

$$\frac{d}{dt}(u, v) + a(u, v) = 0 \tag{3.3}$$

$$u(x, 0) = \max(x - K, 0), t \in (0, T), \forall\, v \in V$$

or an alternative form in different notation,

$$\frac{d}{dt}(u(t), v) + a(u(t), v) = 0 \tag{3.4}$$

where $(u, v) = \int_\Omega uv$ and $a(u, v) = -(Lu, v)$, which is a bilinear form. V is a Hilbert space that contains the test functions $v(x)$.

In the particular case of Black-Scholes Equation,

$$a_t(u, v) = \left(\frac{\sigma^2 x^2}{2} u_x, v_x\right) + (xu_x, (\sigma^2 + x\sigma\sigma_x - r)v) + (ru, v) \tag{3.5}$$

We can see the weak formulation eliminates the term of second derivative through integration by parts.

Then we impose the finite element method, and perform the discretization. We can simply replace V with finite dimensional space V_h. In this case, we will consider simple linear finite elements. Thus, V_h is, for instance, a space of piecewise linear functions. The corresponding finite element problem is therefore,

$$\text{find } u_h \in V_h : \frac{d}{dt}(u_h, v_h) + a(u_h, v_h) = 0 \tag{3.6}$$

Now we define $\{\varphi_i, i = 1,2, \dots \dots N\}$ as basis of V_h. Every function in V_h can be denoted as a linear combination of φ_i. For instance, $u(x) = \varphi_1(x)u_1 + \varphi_2(x)u_2$. Since $u_h = \sum_1^N u_j\, \varphi_i$, we can thus rewrite the above problem:

$$\frac{d}{dt}\left(\sum_1^N u_j\, \varphi_i, \varphi_i\right) + a\left(\sum_1^N u_j\, \varphi_i, \varphi_i\right) = 0 \;\; \forall i = 1, \ldots . N \tag{3.7}$$

We can extract a system of equations, which is differential in time and algebraic in space. A and B are both $(N+1) * (N+1)$ matrix. A is stiffness matrix and B is the mass matrix[1].

$$B\frac{dU}{dt} + A(t)U = 0 \tag{3.8}$$

The next step is to discretize time. Therefore we slice the time $(0, T)$ to m intervals, as we did in FDM. We denote each time step as subinterval

$$[t_{m-1}, t_m], 1 \leq m \leq M, \text{ and } t_{m-1} - t_m = \Delta t_m$$

If we use the Euler implicit scheme, we have

$$B\frac{(U^{m+1} - U^m)}{\Delta t_m} + A^{m+1}U^{m+1} = 0 \tag{3.9}$$

If we use Crank-Nicholson scheme, we have

$$B\frac{(U^{m+1} - U^m)}{\Delta t_m} + \frac{1}{2}(A^{m+1}U^{m+1} + A^m U^m) = 0 \tag{3.10}$$

or explicit Euler scheme

$$B\frac{(U^{m+1} - U^m)}{\Delta t_m} + A^{m-1}U^{m-1} = 0 \tag{3.9}$$

A and B are tridiagonal matrices such that we can use the Thomas algorithm in the LU factorization to solve the linear system at each time step. Due to the length of the paper, I will leave the triagonal property of matrices for interested readers to further explore.

Another very important and interesting conclusion is: when the mesh is uniform, the finite element method with mass-lumping is equivalent to the finite difference centered scheme. Further numerical results will be presented later in this paper, and we will verify this conclusion.[1]

3.3 Advantages of the FEM & Comparison with FDM

The most obvious advantage for FEM over FDM is that the solution of FEM computes the entire domain space while FDM only computes solution at selected nodes. While the space domain is finite for both FEM and FDM, FEM allows us to choose between techniques like domain truncation and domain transformation, which will be discussed later in the numerical test cases.

Another shortcoming of FDM is that the boundary condition involves derivatives would be more complicated to treat. FEM can handle more complex boundary condition fairly easily. Furthermore, the domain of the partial differential equation can be of various shapes, sometimes even irregular. With FEM, we would have no problem incorporating nonuniform meshes by choosing appropriate finite elements and using local parameters. The mesh does not need to be cohesive as it is in finite difference analysis. Furthermore, FEM allows more room for mesh refinement. In 2D and 3D case, adaptive mesh and local refinement techniques can be employed to solve the Black-Scholes equation with more accuracy.

4. Numerical Results

In this chapter, we will focus on how we can handle the semi-infinite domain in option pricing problems, in particular the Black-Scholes equation, using techniques like domain truncation and domain transformation in FEM. Since the underlying stock can be priced from zero to infinity, we naturally consider truncating the unbounded domain to a bounded one so that we can apply standard numerical methods. Domain transformation is another intriguing technique that supposedly maintain the property of the boundary condition. As Duffy(2009) points out, domain truncation is quite simple, but it lacks certain theoretical support in the academics of numerical methods while domain transformation create a new space variable and represents possible improvement[4].

4.1 A Test Case

In this section, we will consider a test case of European call option to achieve numerical results for comparison purpose. We have all the information about this particular European option below:

$$\mathrm{S} = 70, \mathrm{K} = 65, \mathrm{r} = 0.08, \sigma = 0.3, \mathrm{T} = 2$$

and we attempt to solve the problem numerically. To compute errors, we calculate the exact solution using the Black-Scholes formula(1.16).

4.1.1 Domain Truncation

Techniques around how to replace semi-infinite domain with a bounded interval [a,b] has been developed for quite some time in the study of the FEM. Far field and near field boundary are defined to legitimize the choice of the artificial boundary. In our test case, we take examples of arbitrary boundary [0,70], [0,100] and [0,200] to compare and determine the trend of errors. Interested readers can explore Kangro (2000) for further verification and techniques of domain truncation[8]. In our test case, we will first compare the L2 error of the domain truncation between FEM and FDM, which has the same boundary. Then we observe the its error performance compared to that of domain transformation technique.

Error Expectation

The error here should come from two major sources: the boundary truncation error and the numerical error or so-called dicretization error.

$$Total\ Err = Boundary\ Truncation\ Err + Numerical\ Err$$

Refinement of time and space discretization should lower the numerical error, but such error may be worsened by extended boundary that we trialed in attempt to decrease the boundary error, the other component of the error equation. Thus, we expect mixed results with the change of variables: time intervals(Δt), space intervals(Δx), and boundary point($u(n)$). We will examine the error between FEM and FDM as well.

4.1.2 Domain Transformation

Besides domain truncation, domain transformation is the other viable option since we are interested in solving the Black-Scholes PDE. In this case, we transform the semi-infinite domain into a unit domain [0,1] by mapping technique. Duffy(2009) has found a mapping function

$$y = \frac{S}{S + \alpha} \tag{4.1}$$

α is a factor that can be chosen by us. For convenience, we choose $\alpha = 1$.

To rewrite the Black-Scholes equation, we need variables

$$S = \frac{y}{1 - y} \tag{4.2}$$

$$\frac{\partial C}{\partial S} = \frac{(1 - y)^2 \, \partial C}{\partial y} \tag{4.3}$$

$$\frac{\partial^2 C}{\partial S^2} = (1 - y)^2 \frac{\partial}{\partial y}\left((1 - y)^2 \frac{\partial C}{\partial y}\right) \tag{4.4}$$

The new transformed PDE is

$$C_t = 0.5\sigma^2 y^2 (1 - y)^2 C_{yy} + (y(1 - y) - \sigma^2 y^2 (1 - y))C_y - rC \tag{4.5}$$

$$0 < y < 1, 0 < t < T$$

The initial condition $u(x_0)$ is the payoff of the option at maturity, which is known to us. The trick lies in how we can define the boundary condition in order to solve the PDE.

We are faced with several layers of difficulties in dealing with the domain transformation approach. First, in (4.2), y cannot equal to 1. Ideally, we are not losing the entire domain space as the limit of shows

$$\lim_{y \to 1} \frac{y}{1 - y} = \infty \tag{4.6}$$

However, when we try to solve (4.5) numerically, we are taking linear uniform mesh on the 1-D domain[0,1], and we inevitably suffer from the loss of the very last part of the domain since we cannot generate the last part of the mesh. If we map our solution back to the supposedly real space domain, it translates into a large number(S_{max}) instead of infinity on the right bound. Before we find a better solution, we use a number that we consider large enough to replace infinity.

This defect has some knock-on effects, and we would apply some numeric operation in the hopes of refinement. The initial condition, which is the final payoff as we mentioned in 2.2, is $(S - \mathrm{K})^+$ for European call. The transformation (4.2) eliminates the chance of S to become infinity. Thus, we create a very small number $\epsilon > 0$, which is smaller than any number you can possibly take in the domain. and change (4.2) into

$$\mathrm{S}' = \frac{y}{1 - y + \epsilon} \tag{4.7}$$

The initial condition becomes

$$\left(\frac{y}{1 - y + \epsilon} - K\right)^+ \tag{4.8}$$

In terms of the exact solution, we replace the S with new S'(x in the new code) for the error comparison purpose. For the asymptotic boundary condition, which is $S_{max} - K * e^{-rt}(1.20)$, will be in the form of a large fixed number(S_{max}) minus the discounted payoff of the strike price. Since we have defined $\epsilon > 0$, we can use $1/\epsilon$, and we will get (4.9) for our new boundary condition,

$$\frac{1}{\epsilon} - 65 * e^{-rt} \tag{4.9}$$

With new initial and boundary condition, we can solve the Black-Scholes equation in the solver using FEM.

Error Expectation

Ideally, we expect the error of domain transformation to be better than that of domain truncation in the sense of y. The boundary error would almost be non-existent, and discretization error would be also small as we discretize in a small local domain like [0,1]. However, since we are converting back the solution to the semi-infinite domain, we should expect an additional error caused by the rounding error, which is a magnifying factor of the total error in the domain of y. One can argue the transformation is theoretically sound. Suppose

$$S = \frac{y}{1-y}, S_w = \frac{y+\epsilon}{1-y-\epsilon}$$

We take the limit of the quotient,

$$\frac{S_w}{S} = \frac{1+\frac{y}{\epsilon}}{1-\frac{\epsilon}{1-y}} \; and \; \lim_{y\to 0,1} \frac{S_w}{S} = 1$$

The transformation suffers numerically however. Since $\frac{y}{1-y}$ can be very large, we suffer from the amplification of numerical errors in computing S form y. Furthermore, more error may stem from the ill-conditioning of the matrices in associated linear system that we included in the solver. This kind of error is also well documented in section 4.2 of FSV[5]. Thus, we summarize,

$$Total\ Err = (Boundary\ Truncation\ Err + Numerical\ Err) * rounding\ error$$

Besides the magnifying factor, our conundrum can be further complicated by our struggle to generate an accurate mesh for the y domain. Thus, we are not surprised that we would some discouraging results.

4.2 Numerical Results and Error Analysis

Domain Truncation

First, we will exploit the code(see Appendix A) and test the error in domain truncation(see Table 3). There are several critical component to our analysis. In a bigger picture, we are trying to compare the pattern of errors between FEM and FDM. In our code, we implemented theta method, which allows us to further explore different numerical methods in Implicit Euler(θ=1), Crank-Nicholson ($\theta = 0.5$) and Forward Euler($\theta = 0$). Further division marks our use of exact solution as the boundary condition and asymptotic final payoff function as our right boundary.

In our test case, left boundary is always zero. The asymptotic solution is $S_{max} - K * e^{-rt}(1.20)$, and the exact solution is computed by the Black-Scholes Formula(1.16). ErrLil2 is the L2 Error Norm.

$$\text{ErrLil2} \equiv \mathrm{L}^{\infty}(\mathrm{L}^2) = \max_{0<t<T}\left(\int_0^{S_{max}} (\mathrm{u}_{\mathrm{ex}} - \mathrm{u})^2\right)^{1/2}.$$

The time domain is [0,2]. The first batch of test will apply FDM and implicit Euler scheme.

Trial	dt	dx	U(n)	X domain	ErrLil2
1	1/16	1/16	Exact solution	(0,70)	0.4604
2	1/16	1/16	Asymptotic solution	(0,70)	15.0578
3	1/32	1/32	Exact solution	(0,70)	0.2860
4	1/32	1/32	Asymptotic solution	(0,70)	15.0670
5	1/16	1/16	Exact solution	(0,100)	0.4828
6	1/16	1/16	Asymptotic solution	(0,100)	3.9007
7	1/16	1/16	Exact solution	(0,200)	0.4828
8	1/16	1/16	Asymptotic solution	(0,200)	0.4828

FDM, θ=1, Table 3

Comments:

With exact solution defined on the boundary, trials of odd numbers show very small error. Trial 3 shows the expected improvement of error over Trial 1 when we take more nodes and time steps. Trial 5 and 7 extend the boundary of the underlying asset, and the discretization error naturally increases. Thus, overall error grows despite the diminishing boundary truncation error. At this point, the major source of error is the numerical error. For asymptotic boundary condition, Trial 2 and 4 presents an interesting problem in our numerical approximation. As we take smaller time steps, the matrices tend to be singular and become ill-conditioned, thus the overall error does not improve. In Crank-Nicholson scheme for FDM, we did not achieve much improvement. We proceed and test the FEM in the domain truncation(See Table 4)

Trial	dt	dx	U(n)	X domain	ErrLil2
1	1/16	1/16	Exact solution	(0,70)	0.4603
2	1/16	1/16	Asymptotic solution	(0,70)	15.0577
3	1/32	1/32	Exact solution	(0,70)	0.2859
4	1/32	1/32	Asymptotic solution	(0,70)	15.0670
5	1/16	1/16	Exact solution	(0,100)	0.4827
6	1/16	1/16	Asymptotic solution	(0,100)	3.9007
7	1/16	1/16	Exact solution	(0,200)	0.4827
8	1/16	1/16	Asymptotic solution	(0,200)	0.4827

FEM θ=1, Table 4

As we expect, FEM barely presents an improvement on FDM in the case of implicit Euler scheme. The refinement is less than 0.1 percent, if not negligible. The pattern of error almost completely mirrors that of the FDM implicit scheme.

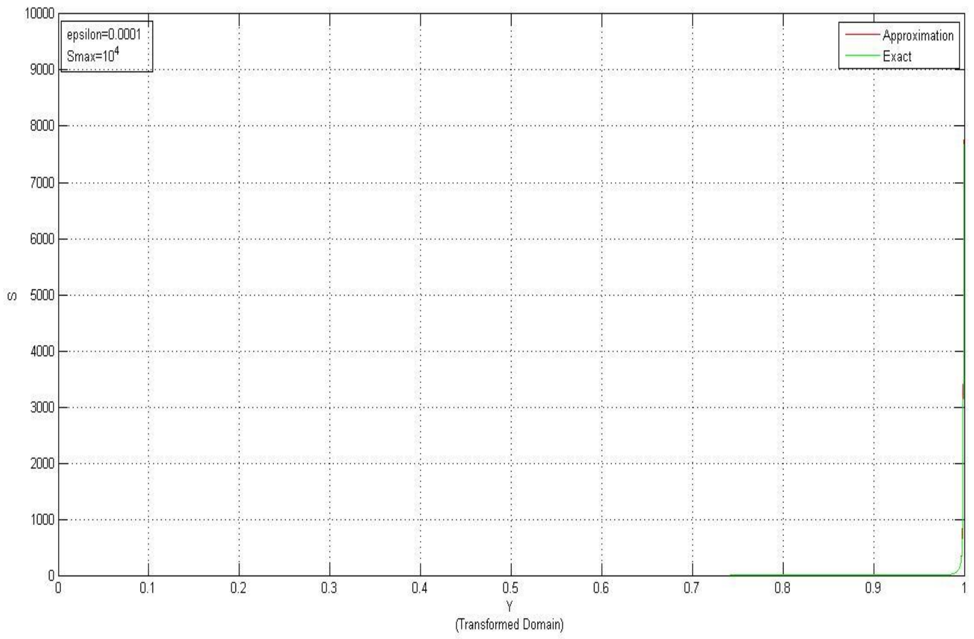
Domain Transformation

Now we will consider the error of domain transformation(See Table 5). If we consider absolute error, the error would be fairly large, but more discretization results in improvement as expected(See Table 6).

Trial	Dt	Dx	U(n)	ErrLil2
1	1/16	1/16	Asymptotic solution	104.0192
2	1/32	1/16	Asymptotic solution	104.0192
3	1/32	1/32	Asymptotic solution	76.6261
4	1/64	1/64	Asymptotic solution	54.1274
5	1/128	1/128	Asymptotic solution	37.4959
6	1/256	1/256	Asymptotic solution	25.0406
7	1/512	1/512	Asymptotic solution	15.5317
8	1/1028	1/1028	Asymptotic solution	8.0920

FEM, θ=1, Table 5

As we expected, error is large compared to domain truncation due to the reasons that we have listed in our error prediction. However, the interpolation(see Graph 1) shows an intuitive comparison that the numerical solution(red) we approximated is actually close to the exact solution.



Graph 1

To attain more appropriate result, we analyze normalized error, which is the percentage of our error with respect to the true value. The error percentage shows an asymptotic pattern of decrease(See Table 6).

Trial	dt	Dx	U(n)	ErrLil2
1	1/16	1/16	Asymptotic solution	0.0104
2	1/32	1/16	Asymptotic solution	0.0104
3	1/32	1/32	Asymptotic solution	0.0077
4	1/64	1/64	Asymptotic solution	0.0054
5	1/128	1/128	Asymptotic solution	0.0037
6	1/256	1/256	Asymptotic solution	0.0025
7	1/512	1/512	Asymptotic solution	0.0016
8	1/1028	1/1028	Asymptotic solution	8.0874e-004

FEM, θ=1, Table 6

Considering the flaw that we have in numerical approximation, the results of its normalized error is certainly desirable, and the results of normalized error in domain truncation technique is apparently not as good if we conduct some computation in Table 4.

While domain transformation does not draw the same accuracy as we expected, we can predict that the technique would accomplish better results if we can overcome the numerical difficulties in including the last subinterval in the domain towards the right boundary and extend physical boundary to virtually infinity.

5 . Monte Carlo methods for option pricing

Monte Carlo method(MC) is first applied to option pricing by Phelim Boyle(1976) and is a quiet different numerical approach in valuing the options. Its advantage is proven by the fact that it is a popular computational tool for almost all the financial options, especially complex options with no close-formed solution or entertaining various underlying assets. Simply put, it always provide some type of answer. Glassermann(2004) has further documented Monte Carlo method in exotic options like Asian options and American options. Monte Carlo method is not as efficient as the actual Black-Scholes equations in valuing the one-factor model for European option, but increasing number of simulations and time steps should improve the accuracy of the results since they both employ geometric Brownian motion as their random process. In this chapter, we will give a simple example of Monte Carlo simulation in valuing one-factor European put option using assumptions in the Black-Scholes equation. We will utilize an existent solver to simulate several test cases and compare the pattern of errors.

5.1 Basics of Monte Carlo Simulation

The core of a Monte Carlo simulation is the generation of a large number of random samples according to a chosen probability distribution. The methodology of generating random observations will be documented later. In the discipline of option pricing, the probability distribution also stands for its pattern of risk in the system. We have shown that in Ito's process(1.2.2) of a normal European vanilla option, term dZ is the lone risk factor. In fact, a given simulation can easily have multiple source of risks if we consider more complex derivatives. The law of large numbers is another foundational theory in MC simulations: the strong version ensures that our estimate of random sampling converges to the expected value as we conduct infinite number of simulations with the probability of 1.

$$\overline{X_n} \xrightarrow{a.s.} E(X) \qquad as\ n \to \infty$$

5.2 An illustrative example of Monte Carlo method on European option

We will profile a general case of European vanilla put option. From (1.2), we discount back the final payoff by factor e^{-rt} and derive the payoff function,

$$P = E\left(e^{-rT}\big(K - S(T)\big)^{+}\right) \tag{5.1}$$

Monte Carlo method is built on the foundation of risk-neutral pricing. The price of underlying asset in a risk-neutral world follows Ito process, same as (1.8)

$$dS = rSdt + \sigma SdZ(t) \tag{5.2}$$

r is the risk-neutral expected return on the option and Z is a standard Brownian motion and also the risk factor in the process. Thus, we compute the solution of stochastic differential equation (5.2), and the underlying asset follows

$$S(T+t) = S(T)\exp\left[\left(r - \frac{\sigma^2}{2}\right)t + \epsilon\sigma\sqrt{t}\right] \tag{5.3}$$

$$\epsilon \sim N(0,1)$$

We factor out random variable $Z \sim N(0, \Delta t)$; ϵ is a standard normal random variable. The stock price is thus lognormal as we expected. Obviously, we can use the

Black-Scholes formula (1.16) for put option to compute the expected payoff, but we will take advantage of this simple example and illustrate the Monte Carlo approach.

Ideally, we take a large number of trials of (5.3) to generate a sequence of simulated paths for future underlying assets for each $\epsilon_1, \epsilon_2, \epsilon_3 \dots\dots \epsilon_n$. Since we have known the $S(0), r, \sigma,$ and t. We can compute n results of underlying price by the time of expiration. Then we apply the payoff function (5.1) to achieve the option price. The last step is average the derived option price for each projected path of the underlying asset. We will sum up this simple process and show possible improvement using the Finite Difference Method in the next section.

5.3 Monte Carlo Simulation in Model, and Methodology:

Step 1

Generate a random process of stock based on the Brownian assumption

Usually, a random number generator of some sort would do the job. It produces a sequence of uniformly distributed random numbers in the range [0,1]. Some most distinguished algorithms are linear congruential generator, lagged Fibonacci generator and etc. In our simulation, lagged Fibonacci is used to generate such numbers.

Our purpose is to generate random variables $\epsilon_i (i = 1,2 \dots n)$ for simulations of (5.3). They are randomly drawn standard normal variable and are mutually independent. Methods like Box-Muller and inverse transformation are widely used[6]. Inverse transformation is used here and its idea is very simple. Suppose we have random discrete variable

$$X \sim (X_1, X_n, \dots\dots\dots\dots..X_n)$$

and its cumulative function $F(x)$, the uniformly generated number is within [0,1]. We take inverse function $F^{-1}(x)$ to get a random observation of variable X. Boyle(1973) searched extensively in the tricks of converting the random number to random variables. After all, we take advantage of (5.3) and plug in our random variable $\epsilon_i (i = 1,2 \dots n)$

Step 2

Get an expected payoff for the given derivative based on this walk

After calculation of (5.3), we have n results of $S(T)$. We then plug into (5.1), discount back and achieve n values of payoff.

In this simulation process, we notice that we are only considering two boundary value $S(0)$ and $S(T)$. For a long duration T, considerable time discretization error is highly likely besides the sizable numerical error. Naturally, we can take more time steps by partitioning the total time to expiration(T) to n discrete time intervals($\Delta t_n = t_n - t_{n-1}, n = 1,2 \dots.$), and approximate the option price for each time step to reduce the time discretization error. We recall the most common approximation method, FDM. We rewrite process (4.2) with Explicit Euler scheme of $\frac{dS}{dt} = \frac{S(t_n) - S(t_{n-1})}{\Delta t_n}$,

$$\frac{dS}{dt} = rS + \sigma S \frac{dZ(t)}{dt}$$

$$\frac{S(t_n) - S(t_{n-1})}{\Delta t_n} = rS(t_{n-1}) + \frac{\sigma S(t_{n-1})}{\Delta t_n} \sqrt{\Delta t_n} \epsilon_n$$

where $dZ(t) \sim N(0, \Delta t)$

$$S(t_n) = S(t_{n-1}) + rS(t_{n-1})\Delta t_n + \sigma S(t_{n-1})\sqrt{\Delta t_n}\epsilon_n \tag{5.4}$$

where $S(t_0) = S(0)$ is the initial condition (5.5)

Thus, we will compute $S(t_0)$, then $S(t_1)$ and finally $S(t_n)$after n iteration of (5.4).

3) Repeat steps 1 and 2 a large number of times

We pick our number of simulations(NSIM) and iterates through first two steps.

4) Achieve the present value for each results and average the cumulative results.

We use equation (5.1) to discount back the each simulated final price of our underlying asset. Then we average the simulation results,

$$C_i = \frac{C_1 + C_2 + \cdots C_n}{n} \tag{5.6}$$

5.4 Simulations and Expected Results:

As we run more and more simulations(NSIM), we expect that we can have more accurate results that are closer to the exact solution. Given the same time step, we expect error to be reduced a rate of $1/\sqrt{h}$ when we take h times of simulations. The convergence happens much slower than FEM and FDM. In order to simulate a path we shall consider the price of an asset on a finite set of m + 1 evenly-spaced dates 0 = $t_0,\ t_1, \ldots,\ t_m$ = T, where $t_j = j\Delta t = jT/m$ is the time of the *j*th observation.

As we take smaller time steps in each simulation(NT), we expect the results to track the path independent options more accurately. The confidence interval should be narrower when smaller steps are applied.

We consider Batch 1 of our data, T = 0.25, K = 65, sig = 0.30, r = 0.08, S = 60, Table 1 presents the value of this European call option given various time steps and number of simulations, The exact solution is 2.13293 by the Black-Scholes equation

NSIM \ NT	300	600	1200
2×10^6	2.13163(0.00130)	2.13286	2.13149
4×10^6	2.13124	2.13274	2.13247
8×10^6	2.13215	2.13318	2.13244(0.00049)

Table 1

We obviously see improvement from 300 time steps to 600 steps, but oddly enough, we see the simulated results shift away from exact solution when we take 1200 time steps. However, the overall accuracy of two places behind the decimal point is satisfying. We later found the same degree of accuracy in nearly every one of our test cases. We further explore the pattern of standard error in the table below. The pattern is very much discernible: the standard error decreases at the same rate as the increase of NSIM, which is expected.

NSIM \ NT	300	600	1200
2×10^6	2.212e-006	2.212e-006	2.212e-006
4×10^6	1.106e-006	1.106e-006	1.106e-006
8×10^6	5.529e-007	5.533e-007	5.529e-007

Table 2

Like Batch 1, We have gathered a total of eight sets of data of European call and put. Detailed results are listed in Appendix B. The result overall is fairly accurate, but it is not quite what we expected. The error between the simulation and exact solution does not display a steady pattern with the increase of time steps and number of simulations. While the standard deviation explicitly follows the expectation and is reduced by half when number of simulations is 4 times as many, some of the standard deviation are still pretty large(Batch 3 Call).

5.5 Advantages and Shortcoming of Monte Carlo Method in option pricing

We have mentioned that MC method applies to a wide range of problems. For instance, the Asian option has no closed-form solution, and its payoff relies on the intermediate price of underlying asset. The utilization of finite difference method can record the price for each intermediate time to help compute the average price of underlying asset. Its universal methodology is easy to understand and program. Even in cases of plain European option, we can consider MC method as a second opinion for other numerical methods or a method to check the exact solution. The most glaring disadvantage of the MC method is that it is slow to compute. The theory states that the time discretization error can be eliminated if we divide the time interval into smaller subintervals, but the cost is not proportional to the refinement of the results. A much longer time is taken in last several trials of each test cases, but results barely get better, if not worse.

6. Conclusion

In summary, we revisited the foundational work of the Black-Scholes framework and derived the Black-Scholes equation and its analytical solution of European options. We then examined the numerical approach to solve the model including the Finite Difference Method(FDM), Finite Element MethodFEM), and Monte Carlo Method(MC). MC method is a general process that can apply to a variety of derivatives given the probability distribution and can be modified to accommodate various processes for the underlying asset. The numerical results are fairly accurate and encouraging, but a large number of simulation is required to reach the desired results. More importantly, traditional MC Method does not track the Black-Scholes model and is not able to show dynamical information about the option price. If we apply FD time discretization, we are able to achieve intermediate data and find better final results. On the other hand, FDM proves to be a fairly fast and robust numerical scheme for pricing plain vanilla option, especially in the 1-D case. As we have observed in domain truncation cases(Table 1 and 2), FEM is barely an improvement over FDM in evaluating the one-dimensional Black-Scholes equation. However, we can speculate that FEM may represent an substantial upgrade over FDM if we consider nonlinear case in high dimensional space since FEM is able to cover the domain space significantly better than FDM. Domain transformation lacks theoretical foundation, but it presents an interesting attempt to include the entire semi-infinite domain space. Although this idea may be faced with multiple source of error, including the amplification effect of the rounding error, the normalized error is fairly desirable and shows improvement over the counterpart of the domain truncation. An obvious improvement can be made by applying a non-uniform mesh to the domain space[0,1] instead of a uniform mesh for our test cases. We may be able to obtain better approximation by employing FEM with a suitably graded mesh near infinity. However, a realistic issue with the domain transformation in the case of option pricing is that the participants in the derivatives market are using the physical price of the underlying asset, so the conversion of nodes requires a linear interpolation in the transformed domain, which may cause significant error and call for further error analysis.

Appendix A
Test Case Solver

```
clear;
%rs=70;%truncated domain
rs=1; %transformed domain
Tf=2;%expiration
ref=10;
%dx=1/2^ref;
dy=1/2^ref;
dt=1/2^10;
sigma=0.3;
r=0.08;

%x=[0:dx:rs];
y=[0:dy:rs];
n=length(y);
e=ones(n,1);
ze=zeros(n,1);

%g1=(-0.045*x.^2)'; %-0.5*sigma^2*x.^2/dx^2;
g1=(-0.045*(y.^2).*((1-y).^2));%tranformation coefficient
%g2=(-0.08*x)';
g2=(-0.08*y.*(1-y)+0.09*(y.^2).*(1-y))';%transformation coeff
G1=sparse(diag(g1));
G2=sparse(diag(g2));
e=ones(n,1);

%M=sparse(eye(n,n)); % FD
M=spdiags([1/6*e 2/3*e 1/6*e], -1:1,n,n); % Linear FEM
K=spdiags(1/dy^2*[e -2*e e], -1:1,n,n);%generate second derivative
K=G1*K;
% upwind
% C=spdiags(1/dy*[ze -e   e], -1:1,n,n);
C=spdiags(1/dy*[-0.5*e  ze  0.5*e], -1:1,n,n);%generate first
derivative
C=G2*C;

theta=1; %implicit Euler
%theta=0 %explicit Euler
%theta=0.5 %Crank-Nicholson

A=1/dt*M+theta*(K+C+r*M);%adding time-advancing term
B=1/dt*M-(1-theta)*(K+C+r*M);
```

```
u=zeros(n,1);

disp('Assembling done');


trun=10^4;
u=(y./(1+(trun)^(-1)-y)-65).*(y./(1+(trun)^(-1)-y)>=65);
x=y./(1-y+(trun)^(-1));

u=u';
bbc=zeros(n-2);
j=1;

disp('Time loop starting');
uey(:,j)=u;u(end-1,j), max_sol(j)=norm(uey(1:end-1,j))
sol(:,j)=u;

for t=dt:dt:Tf
        t
    u(1)=0; % left b.c.
    u(n) = trun-65*exp(-r*t);%asymptotic right bc.
    b=B*u;
    bbc=b(2:n-1);
    bbc=bbc-A(2:n-1,1)*u(1)-A(2:n-1,n)*u(n);
    Abc=A(2:n-1,2:n-1);
    ur=Abc\bbc;
    u=[u(1);ur;u(n)];
    j=j+1;
    uey(:,j)=
0.5*x.*(1+erf((log(x/65)+0.125*t)/(0.3*sqrt(2*t))))-0.5*65*exp(-0.08*
t)*(1+erf((log(x/65)+0.035*t)/(0.3*sqrt(2*t))));
    sol(:,j)=u;
    max_sol(j)=norm(uey(1:end-1,j));

end;
sol(:,j)=u;


rel_err=1;
end_err=n;%rel_err/dx+1;

for i=1:j,
 %plot(sol(2:end_err-1,i)-uey(2:end_err-1,i),'k'), pause(0.1)
   aux=(sol(1:end_err-1,i)-uey(1:end_err-1,i)).^2;
   err(i)=sqrt(0.5*dy*(aux(1)+aux(end))+dy*sum(aux(2:end-1))); %L2
error
end;
errLiL2=max(err)/rs %/max(max(max_sol))

for i=1:j, plot(y,sol(:,i),'r',y,uey(:,i),'g'),pause(0.1); end;
```

Appendix B
Monte Carlo simulation results

Batch 1
Call(2.13293)
1)NT =300, NSIM = 2000000; P = 2.13163, Std = 0.00312858, SE = 2.21224e-006
2)NT =300, NSIM = 4000000; P = 2.13124, Std = 0.00221191, SE = 1.10595e-006
3)NT =300, NSIM = 8000000; P = 2.13215, Std = 0.00156385, SE = 5.52903e-007
4)NT = 600,NSIM = 2000000; P = 2.13286, Std = 0.00312872, SE = 2.21234e-006
5)NT = 600,NSIM = 4000000; P = 2.13274, Std = 0.00221213, SE = 1.10607e-006
6)NT = 600,NSIM = 8000000; P = 2.13318, Std = 0.0015649, SE = 5.532976e-007
7)NT = 1200, NSIM = 2000000; P = 2.13149, Std = 0.00312852,SE= 2.2122e-006
8)NT = 1200, NSIM = 4000000; P = 2.13247, Std = 0.00221263, SE = 1.10631e-006
9)NT = 1200, NSIM = 8000000; P = 2.13244, Std = 0.00156385, SE = 5.52903e-007

Comment: Simulation 4 has the most accurate result compared to the exact solution, which is quite unexpected. The odd thing is that between simulation 8 and 9, the error grew bigger.

Put(5.84584)
1)NT = 300, NSIM = 2000000; P = 5.84439, Std = 0.00419261, SE = 2.96462e-006
2)NT = 300, NSIM = 4000000; P = 5.84427, Std = 0.00296357, SE = 1.48178e-006
3)NT = 300, NSIM = 8000000; P = 5.84553, Std = 0.00209586, SE = 7.40998e-007
4)NT = 600, NSIM = 2000000; P = 5.84565, Std = 0.00419022, SE = 2.96293e-006
5)NT = 600, NSIM = 4000000; P = 5.84619, Std = 0.00296349, SE = 1.48175e-006
6)NT = 600, NSIM = 8000000; P = 5.84419, Std = 0.00209495, SE = 7.406773-007
7)NT = 1200, NSIM = 2000000; P = 5.84528, Std = 0.00419027, SE = 2.96297e-006
8)NT = 1200, NSIM = 4000000; P = 5.84278, Std = 0.00296225, SE = 1.48113e-006
9)NT = 1200, NSIM = 8000000; P = 5.84616, Std = 0.0020957, SE = 7.40941e-007

Comments: Simulation 4 gives the most accurate result. Between 7, 8, 9, the results oscillates and does not have a clear pattern to show its improvement towards the exact solution.

Batch2
Call(7.96632)
1)NT = 300, NSIM = 2000000; P = 7.96431, Std = 0.00929421, SE = 6.572e-006
2)NT = 300, NSIM = 4000000; P = 7.96236 Std = 0.00657135, SE = 3.28568e-006
3)NT = 300, NSIM = 8000000; P = 7.96427, Std = 0.00464637, SE = 1.64274e-006
4)NT = 600, NSIM = 2000000; P = 7.96478, Std = 0.00929554, SE = 6.57294e-006
5)NT = 600, NSIM = 4000000; P = 7.9646, Std = 0.00657232, SE = 3.28616e-006
6)NT = 600, NSIM = 8000000; P = 7.96623 Std = 0.00464878, SE = 1.64359-006
7)NT = 1200, NSIM = 2000000; P = 7.96043, Std = 0.00929452, SE = 6.57221e-006
8)NT = 1200, NSIM = 4000000; P = 7.96469, Std = 0.00657322, SE = 3.28661e-006

9)NT = 1200, NSIM = 8000000; P = 7.96425, Std = 0.00464652, SE = 1.64279e-006

Comments: Simulation 6 is the most accurate. The others are not as accurate and shows no pattern.

Put(7.96632)
1)NT = 300, NSIM = 2000000; P = 7.96398, Std = 0.00735991, SE = 5.20424e-006
2)NT = 300, NSIM = 4000000; P = 7.96287 Std = 0.00520205, SE = 2.60103e-006
3)NT = 300, NSIM = 8000000; P = 7.96541, Std = 0.00367899, SE = 1.30072e-006
4)NT = 600, NSIM = 2000000; P = 7.96476, Std = 0.00735385, SE = 5.19996e-006
5)NT = 600, NSIM = 4000000; P = 7.96565 Std = 0.00520127, SE = 2.60063e-006
6)NT = 600, NSIM = 8000000; P = 7.96225 Std = 0.00367666, SE = 1.2999-006
7)NT = 1200, NSIM = 2000000; P = 7.96278, Std = 0.00735458, SE = 5.20047e-006
8)NT = 1200, NSIM = 4000000; P = 7.95944, Std = 0.00519834, SE = 2.59917e-006
9)NT = 1200, NSIM = 8000000; P = 7.96626, Std = 0.00367791, SE = 1.30034e-006

Comments: Simulation 9 is the most accurate. It is expected because it takes most simulations and subintervals. Other than this desired result, the others do not show a clear pattern.

Batch3
Call(92.1749)
1)NT = 300, NSIM = 2000000; P = 91.2047, Std = 0.0228614, SE = 1.61655e-005
2)NT = 300, NSIM = 4000000; P = 91.1992 Std = 0.0161788, SE = 8.08938e-006
3)NT = 300, NSIM = 8000000; P = 91.0868, Std = 0.0112461, SE = 3.97608e-006
4)NT = 600, NSIM = 2000000; P = 91.6321, Std = 0.02325, SE = 1.64402e-005
5)NT = 600, NSIM = 4000000; P = 91.5422, Std = 0.0162482, SE = 8.1241e-006
6)NT = 600, NSIM = 8000000; P = 91.6098, Std = 0.0114195, SE = 4.03741e-006
7)NT = 1200, NSIM = 2000000; P = 91.9614, Std = 0.0247095, SE = 1.74723e-005
8)NT = 1200, NSIM = 4000000; P = 92.0465, Std = 0.0172252, SE = 8.61258e-006
9)NT = 1200, NSIM = 8000000; P = 91.9474, Std = 0.0127421, SE = 4.505e-006

Comment: Simulation 8 is the most accurate, but doubling the simulation after that does not show improvement.

Put(1.24651)
1)NT = 300, NSIM = 2000000; P = 1.26174, Std = 0.000158334, SE = 1.11959e-007
2)NT = 300, NSIM = 4000000; P = 1.26069, Std = 0.000111916, SE = 5.59582e-008
3)NT = 300, NSIM = 8000000; P = 1.2611, Std = 7.91473e-005, SE = 2.79828e-008
4)NT = 600, NSIM = 2000000; P = 1.2537, Std = 0.000157689, SE = 1.11503e-007
5)NT = 600, NSIM = 4000000; P = 1.25438, Std = 0.000111516, SE = 5.5758e-008
6)NT = 600, NSIM = 8000000; P = 1.25333, Std = 7.88265e-005, SE = 2.78694e-008
7)NT = 1200, NSIM = 2000000; P = 1.25012, Std = 0.000157434, SE = 1.11323e-007

8)NT = 1200, NSIM = 4000000; P = 1.24901, Std = 0.000111291, SE = 5.56454e-008
9)NT = 1200, NSIM = 8000000; P = 1.25085, Std = 7.87431e-005, SE = 2.78399e-008

Comment: Simulation 8 is the most accurate, and overall 1200 subintervals are more accurate than the results we derive from 600 subinterval simulations.

Batch4
Call(0.204121)
1)NT = 300, NSIM = 2000000; P =0.203385, Std = 0.000639082, SE = 4.51899e-007
2)NT = 300, NSIM = 4000000; P = 0.203146, Std = 0.000451978, SE = 2.25989e-007
3)NT = 300, NSIM = 8000000; P = 0.203057, Std = 0.000318938, SE = 1.12762e-007
4)NT = 600, NSIM = 2000000; P = 0.20334, Std = 0.00063982, SE = 4.52421e-007
5)NT = 600, NSIM = 4000000; P = 0.20329, Std = 0.000451983, SE = 2.25991e-007
6)NT = 600, NSIM = 8000000; P = 0.203597, Std = 0.000319988, SE = 1.13133e-007
7)NT = 1200, NSIM = 2000000; P = 0.203514, Std = 0.000640914, SE = 4.53194e-007
8)NT = 1200, NSIM = 4000000; P = 0.203576, Std = 0.000453537, SE = 2.26769e-007
9)NT = 1200, NSIM = 8000000; P = 0.203299, Std = 0.000320095, SE = 1.13171e-007

Comment: Simulation 6 is the more accurate one. Simulations 7,8,9 shows overall improvement, but they are all very far from the actual results.

Put(4.0733)
1)NT = 300, NSIM = 2000000; P =4.07277, Std = 0.00131428, SE = 9.29339e-007
2)NT = 300, NSIM = 4000000; P = 4.07278, Std = 0.000929328, SE = 4.64664e-007
3)NT = 300, NSIM = 8000000; P = 4.07272, Std = 0.000657301, SE = 2.32391e-007
4)NT = 600, NSIM = 2000000; P = 4.07282, Std = 0.00131443, SE = 9.29445e-007
5)NT = 600, NSIM = 4000000; P = 4.07291, Std = 0.000929478, SE = 4.64739e-007
6)NT = 600, NSIM = 8000000; P = 4.07258, Std = 0.000657143, SE = 2.32335e-007
7)NT = 1200, NSIM = 2000000; P = 4.07334, Std = 0.00131387, SE = 9.29044e-007
8)NT = 1200, NSIM = 4000000; P = 4.07252, Std = 0.00092911, SE = 4.64555e-007
9)NT = 1200, NSIM = 8000000; P = 4.07301, Std = 0.000657165, SE = 2.32343e-007

Comments: Simulation 7 is the most accurate, but the error does not improve with the growing simulation numbers.

Reference

[1] Achdou, Yves, and Olivier Pironneau. *Computational Methods for Option Pricing*. Philadelphia: Society for Industrial and Applied Mathematics, 2005. Print.

[2] Black, Fischer, and Myron Scholes. "The Pricing of Options and Corporate Liabilities."*Journal of Political Economy* 81.3 (1973): 637. Print.

[3]Boyle, Phelim P. "Options, A Monte Carlo Approach." *Journal of Financial Economics* 4.3 (1977): 323-38. Print.

[4]Duffy, Daniel J. "Unconditionally Stable and Second-Order Accurate Explicit Finite Difference Schemes Using Domain Transformation Part I. One-Factor Equity Problems." *Datasimfinancial* (2009): 5-8. Print.

[5]Formaggia, L., Fausto Saleri, and A. Veneziani. *Solving Numerical PDEs: Problems, Applications, Exercises*. Milan: Springer, 2012. Print.

[6]Glasserman, Paul. *Monte Carlo Methods in Financial Engineering*. New York: Springer, 2004. Print.

[7]Hull, John. *Options, Futures, and Other Derivatives*. Upper Saddle River, NJ: Pearson Prentice Hall, 2011. Print.

[8]Kangro, Raul, and Roy Nicolaides. "Far Field Boundary Conditions for Black--Scholes Equations." *SIAM Journal on Numerical Analysis* 38.4 (2000): 1357. Print.

[9]Quarteroni, Alfio. *Numerical Models for Differential Problems (MS&A)*. 3rd ed. N.p.: Springer, 2012. Print.

[10]Salsa, Sandro. "An Application to Finance." *Partial Differential Equations in Action: From Modelling to Theory*. N.p.: n.p., n.d. 80-89. Print.

[11]Wilmott, Paul, Jeff Dewynne, and Sam Howison. *Option Pricing: Mathematical Models and Computation*. Oxford, UK: Oxford Financial, 2000. Print.