Applying the Monte Carlo Technique to Option Pricing

Sam Ansell[1,2†], Jack Burrows[1,2†], Zhiyuan Lei[1,2†], Junya Lui[1,2†], Abi Pibworth[1,2†], Josh Tilford[1,2†], Michael V Tretyakov[2,3*]

[1]School of Mathematical Sciences, University of Nottingham, Park, Nottingham, NG7 2RD, UK.

[2*]School of Mathematical Sciences, University of Nottingham, Park, Nottingham, NG7 2RD, UK.

*Corresponding author(s). E-mail(s): Michael.Tretyakov@nottingham.ac.uk;
Contributing authors: pmxzl2@nottingham.ac.uk;
†These authors contributed equally to this work.

Abstract

This report outlines an investigation into the use of Monte Carlo methods in financial option pricing and random number generators. Specifically, this report produces simulations for two types of call options, European call option and binary asset-or-nothing call option, using an analytical approach, a weak-Euler scheme and a Milstein scheme. A Monte Carlo method has been applied to the weak-Euler scheme to examine the option prices and delta values across various time points. The convergence rates of price and delta have been plotted and compared to the theoretical convergence rate. The results of numerical approximation

Keywords: Monte Carlo, Stochastic process, convergence

Contents

Chapter 1

Introduction

This report outlines an investigation into the use of Monte Carlo methods in financial option pricing and random number generators. Specifically, this report produces simulations for two types of call options, European call option and binary asset-or-nothing call option, using an analytical approach, a weak-Euler scheme and a Milstein scheme. A Monte Carlo method has been applied to the weak-Euler scheme to examine the option prices and delta values across various time points. The convergence rates of price and delta have been plotted and compared to the theoretical convergence rate. The results of numerical approximation and plots convergences can be found in Chapter 3.

More complex models of underliers than GBM are also considered in this report in Chapter 4. This chapter outlines SDE model for option pricing. Firstly, a PDE is extracted for an option price and the parameters that the formula needs are then estimated. A Monte Carlo approach is again applied to the problem and the PDE solution is found using the Crank-Nicolson method (see section 5.1).

The techniques used in this report are beneficial because simulations and the Monte Carlo method are often used in the financial industry to predict the behaviour of stock prices and to evaluate option prices. This is because analytical solutions are regularly difficult to compute, and a Monte Carlo approach estimates option pricing solutions well with low computational effort. Furthermore, it is beneficial to obtain an error analysis for the approximation method used, which can be done effectively using Monte Carlo techniques, as outlined in Chapter 2.

Chapter 2

Introduction to Monte Carlo Methods

The Monte Carlo method was first introduced by the Polish-American mathematician Stanislaw Ulam, who endeavored to calculate the probability of winning a game of solitaire but discovered that the combinatorial mathematics were too complex to find an analytical solution. He instead considered approximating this probability by playing the game multiple times and estimating the probability to be equal to the proportion of observed games that have been won. In order to obtain a sufficient number of games to make a reasonable approximation, Ulam decided to simulate solitaire games using a computer and estimate the probability of winning using the observed outcomes of these simulations [1]. Herein lies the purpose of Monte Carlo methods: when analytical solutions are too complicated to obtain, inference can be made from the simulations and observations. The more simulations or observations there are, the closer the estimated solution is to the true (analytical) solution.

2.1 Principles of the Monte Carlo Method

2.1.1 Estimations and Confidence Intervals

The above gives some intuition on the **law of large numbers**: the more observed events there are, the more accurate the estimations drawn from those events are likely to be. Mathematically, it can be seen that the sample mean of the outcome of a number n of events tends to the expected outcome of those events as n increases. That is:

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} X_i = E(X), \tag{2.1}$$

where the X_i's are observed, independent and identically distributed random variables with distribution X. Furthermore, it can be seen that the estimator for the variance of X,

$$\tilde{\sigma^2} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \tilde{\mu})^2, \tag{2.2}$$

where $\tilde{\mu}$ is the sample mean of the observations, is also unbiased and converges to the true variance of X, σ^2, for large n:

$$\lim_{n\to\infty} \tilde{\sigma^2} = \sigma^2 \tag{2.3}$$

By the **Central Limit Theorem**, the above analysis provides the ability to create an asymptotic normal distribution for X, allowing for the monitoring of error (given by the variance) produced by simulations from the distribution of X. An approximate 95% confidence interval for the outcome of X_i can be constructed as such:

$$\left[\mu \pm \frac{Z_{0.975}\sigma}{\sqrt{n}}\right] \text{ if } \sigma \text{ is known, } \quad \left[\tilde{\mu} \pm \frac{t_{0.975,n-1}\tilde{\sigma}}{\sqrt{n-1}}\right] \text{ if } \sigma \text{ is unknown,} \tag{2.4}$$

where $Z_{0.975}$ is the 97.5% quantile of the standard normal distribution and $t_{0.975,n-1}$ is the 97.5% quantile of the student t-distribution at $n-1$ degrees of freedom.

This provides a basic understanding of the underlying mathematics of Monte Carlo methods; producing n independent samples from computed simulations can allow for the reasonable estimate of an error in approximating $E(X)$ or other statistics using X_i through the use of a confidence interval as in equation (2.4) [2].

Additionally, it should be noted that, by the law of large numbers, the variance in outcomes decreases as n increases, and so the confidence intervals in equation (2.4) becomes narrower as more trials are produced. This means that the estimations drawn from Monte Carlo simulations become more certain as more simulations are run.

Now consider seeking to estimate some function $f(X)$. The Monte Carlo estimate of $f(X)$, $E(f(X))$ can be found similarly to the above:

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n} f(X_i) = E(f(X)), \tag{2.5}$$

and

$$\tilde{\sigma^2} = \frac{1}{n-1}\sum_{i=1}^{n}(f(X_i)-\tilde{\mu})^2, \tag{2.6}$$

where $\mu = \frac{1}{n}\sum_{i=1}^{n} f(X_i)$. This produces the same confidence interval as in equation (2.4).

The above shows that the simple Monte Carlo process is the same in principle for estimating $E(f(X))$ as for E(X).

2.1.2 Convergence Rates

A drawback of the Monte Carlo method stems from the relation of the size of the confidence interval equation (2.4) to the number n of trials run; the size of this confidence interval is proportional to the inverse square root of the number of simulations, which means that to reduce the size of the confidence interval by a factor of, say, c, the number of observed simulations must be increased by a factor of c^2, which may not be computationally feasible. This means that the Monte Carlo method does not always produce the high level of accuracy given by low variance and a small confidence interval. [3]

The concept of reducing variance by increasing number of data points (in this case, simulations) can be captured by the **convergence rate** of a method. Let $O(\cdot)$ denote the convergence rate of some estimation method to the true value the method is seeking to estimate. Then the convergence rate of the Monte Carlo method outlined above is $O(n^{-\frac{1}{2}})$ because the confidence interval decreases by a factor of $n^{-\frac{1}{2}}$ with n.

Consider, by way of example, that X follows a uniform distribution over $[0,1]$ such that $X_i = U_i$ for all i, where $U_i \overset{iid}{\sim} U(0,1)$. Then:

$$\int_0^1 f(x)\,dx = E(f(U)) \tag{2.7}$$

Therefore an estimate for $E(f(U))$ can be given by the trapezium rule:

$$\frac{f(0)-f(1)}{2n} + \frac{1}{n}\sum_{i=1}^{n-1} f(\frac{i}{n}), \tag{2.8}$$

which has a convergence rate of $O(n^{-2})$ [4]. Therefore, there exists another approximation method which converges more quickly than the Monte Carlo Method and can thus achieve more accuracy with given computational power. This begs the question as to why Monte Carlo methods would be ever used at all.

A key benefit of the Monte Carlo method is that it can be performed in higher dimensions with a variance still in the form $\frac{\sigma}{\sqrt{n}}$, so the convergence rate is still $O(n^{-\frac{1}{2}})$ for all dimensions d. Other methods, however, do not hold their convergence rate. The trapezium rule has a convergence rate of $O(n^{-\frac{2}{d}})$, which means that its convergence rate is lower than Monte Carlo for $d > 4$. Therefore, it can be seen Monte Carlo methods may perform well in estimating solutions in high dimensions [4].

2.2 Random Number Generators

The necessity for random number generation when dealing with a Monte Carlo simulation initiates it's own set of challenges to be addressed when implementing such a piece of computation. For example, computationally generated random numbers are generated within the execution of an algorithm and are therefore, by definition, deterministic. This is a unique problem as the use of an algorithm works from an initial condition (or set of conditions) defined in the code, and then follows a repeatable set of steps (algorithm execution) and produces a 'random' result. The intrinsic problem with this is that the repeatable process will be described by a pattern and the initial conditions, meaning that the numbers produced can never actually be random. This is first noticed when generating large sets of random numbers, and the biased distributions observed as a result of this.

2.2.1 PRNGs

The approach taken in the literature is to build either a hardware random number generator or program a pseudorandom number generator (PRNG). For the purposes of the completed project, a pseudorandom number generator was chosen. A PRNG is coded by setting an initial seed, and then approximates a sequence of random numbers. The choice of the seed value is imperative to the validity of the PRNG as certain seed values will produce more biased distributions than others. There are many types of PRNGs, all concerned with a unique statistical approach. For example, the original Middle-square method (J. von Neumann, 1946), and the significantly more sophisticated Linear congruential generator (W. E. Thomson; A. Rotenberg, 1958), leading all the way to the most recent development of the Squares RNG (B. Widynski, 2020). For a PRNG to be considered valid, it must pass a catalogue of statistical tests. For example, uniformity of the produced distribution, lack of correlation between successive values, and more general statistical checks such as the chi-squared test.

2.2.2 PRNGs and Monte Carlo Simulation

PRNGs are routinely used within the field of Monte Carlo simulation, due to the dependence on the source of randomness. However, it is noted in the literature that Monte Carlo simulations can remain valid with the use of a PRNG as long as the statistical check for the uniform distribution is passed and the resulting sequence of numbers 'appears random. Therefore, taking this into consideration, the random number generator chosen was the Ziggurat algorithm, MATLAB's default RNG.

The Ziggurat algorithm utilises the normal distributions probability density function [5]. This may be expressed as follows.

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{2.9}$$

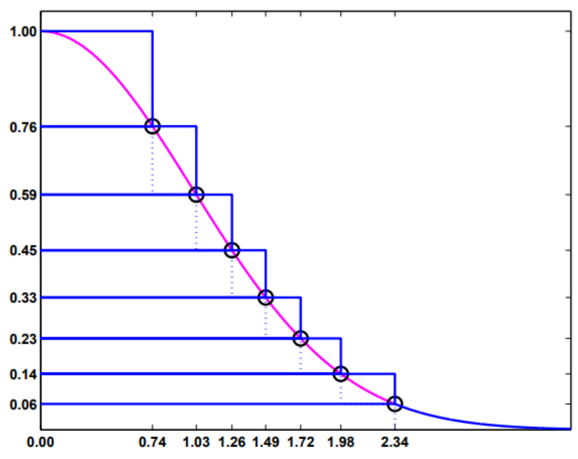


Figure 2.1: The ziggurat algorithm

Firstly, a series of rectangles are imposed upon the area underneath the function in the positive domain. These rectangles are of equal area and slightly overestimate the true area under the curve, graphically similar to that of an upper Riemann sum [6]. Each rectangle is then assigned a number and, using a uniform random number generator, a rectangle may then be selected [6]. A second number, $u \in [0, 1]$, is now generated which takes values within the rectangle. This number is compared to the x coordinate of the lower right hand corner of the rectangle above the one we have selected. If u is smaller than this value, we return u. This is the first validity check of the random number. If it is larger, we generate a second value and if this also fails, we begin rectangle selection again [6]. We can now complete this algorithm multiple times to create the desired number of normally distributed random numbers, before randomly assigning a sign to each u to account for the negative domain of the distribution.

It may be noted that as the number of rectangles are decreased, the random generated numbers do not become less normally distributed, which is a great strength of the RNG. This is because of the two validity checks in the algorithm. However, this does not suggest that we minimize the number of rectangles; as we reduce the number of rectangles, u begins to fail the first and second validity checks more frequently as the rectangles x-coordinates become more spread. This can be slow computationally. We have used 256 rectangles for our computations which, although requires lots of initial rectangle data, has a very fast computation time since u rarely fails validity checks ($< 3\%$ failure rate). We therefore have an algorithm which is both exceptionally fast computationally, and very well distributed as it depends only on the accuracy of the uniform RNG.

2.3 Monte Carlo Method: Applications in Security Pricing

The Monte Carlo method has many applications within finance, one of which is option pricing. Theoretically, the Black-Scholes equation allows for the derivation of a partial differential equation (PDE) which characterises an options pricing structure. However, in practice, the dynamics of the underlying assets from which an option is derived, can be so complex that solving a PDE may not be feasible [4, 7]. Furthermore, in many cases a suitable PDE may not exist, hence a more computationally appropriate method is required.

In 1977, Boyle [8], proposed the use of Monte Carlo methods to price complex derivatives. While this approximation technique may be computationally expensive, proved that Monte Carlo methods offer a good alternative to model complex derivatives in imperfect market conditions [7].

Firstly, to apply the Monte Carlo method to option pricing, the risk-free probability measure, Q is introduced. The Black-Scholes equation for an option price $V(0)$ can then be expressed relative to this measure:

$$V(0) = E_Q[e^{-rT}V(T)] \approx e^{-rT}\frac{1}{M}\sum_{m=1}^{M} f\left(\bar{X}^{(m)}_{t,s(0)}(T)\right) \tag{2.10}$$

and

$$V(t) = E_Q[e^{-r(T-t)}V(T)|\mathcal{F}_t] \approx e^{-r(T-t)}\frac{1}{M}\sum_{m=1}^{M} f\left(\bar{X}^{(m)}_{t,s(t)}(T)\right) \tag{2.11}$$

where s(t) is the stock price at node t and $X^{(m)}$ is the m^{th} path of the model.

The Monte Carlo method can then be used to evaluate the expectation, $E_Q[V(T)]$, over a given time interval [4]. An approximation of the options characteristics, including its price, can then be described using the mathematics explained in chapter 3.

Chapter 3

The GBM Model

Consider the geometric Brownian motion (GBM) model for a stock price given by $X(t)$:

$$dX(t) = \mu X(t)dt + \sigma X(t)dW(t) \tag{3.1}$$

where r, μ, σ are real constants representing the interest rate, rate of return and stock price volatility respectively. By considering the discounted stock price and Girsanov's theorem, under an EMM Q it is possible to obtain the martingale process:

$$dX(t) = rX(t)dt + \sigma X(t)dW^Q. \tag{3.2}$$

(See page 32-35 of the MATH4061 lecture notes for this derivation).
The price of a European call option $V(t)$ at time t with maturity time T is given by the Black-Scholes equation:

$$V(t) = u(t, X(t)) = E_Q[e^{-r(T-t)}f(X_T)|\mathcal{F}_t], \tag{3.3}$$

with:

$$f(X_T) = (X_T - K)_+,$$

where K is the strike price of the option and X(t) is $\mathcal{F}_t$-measurable. Note that $u(t,x)$ satisfies the Feymanm-Kac formula:

$$\frac{\partial u}{\partial t} + \frac{\sigma^2}{2}x^2\frac{\partial^2 u}{\partial x^2} + rx\frac{\partial u}{\partial x} - ru = 0, \quad t \in [0,T], x \in (0,\infty), \tag{3.4}$$

$$u(T,x) = f(x)$$

From (3.3):

$$V(0) = e^{-rT}E_Q[f(X_T)|\mathcal{F}_0] = e^{-rT}E_Q[f(X_T)] \tag{3.5}$$

There are two ways in which this report obtains solutions for $V(0)$: analytically and using Monte Carlo method to approximate V(0) from simulations. There were also two schemes used to simulate the stock prices: a weak-Euler scheme and a Milstein scheme. Chapter 3.2.1 examines the analytical solution while chapters 3.1.1 and 3.1.2 examine Monte Carlo estimates and their convergence to the analytical solution.

3.1 Schemes to the SDE Corresponding to GBM

3.1.1 Weak-Euler Scheme

Suppose a stock price follows (3.1). Then a weak-Euler approximation for the stock price $X(t)$ can be simulated as such:

1. Partition [0,T) into N equally-spaced intervals such that $\Delta t = \frac{T}{N}$ and let each interval be denoted $[t_i, t_{i+1}), i = 0, 1, 2, ..., N-1$.

2. Generate a Wiener process $\{W_i\}$ such that dW_i is randomly generated from a standard normal distribution scaled by $\sqrt{\Delta t}$ such that $dW_i \sim \sqrt{\Delta t} \cdot N(0,1)$. Then $W_{t_{i+1}} = dW_i + W_{t_i}$.

3. Generate a path $\{X_i\}$ such that $X_{i+1} = rX_i dt + \sigma X_i W_i$ for some pre-defined r, σ.

This method has been implemented for this report to a call option with payoff function $f(X_T) = (X_T - K)_+$, and for a binary asset-or-nothing call option with payoff function:

$$f(X_T) = \begin{cases} X_T, & \text{if } X_T > K \\ 0, & \text{otherwise} \end{cases}$$

The parameters used in producing the weak-Euler scheme were $r = 0.06,\ \sigma = 0.36,\ X_0 = 10,\ K = 10,$ and $T = 1$ for both call options. An example of a generated weak-Euler scheme path for each of these options is shown below:

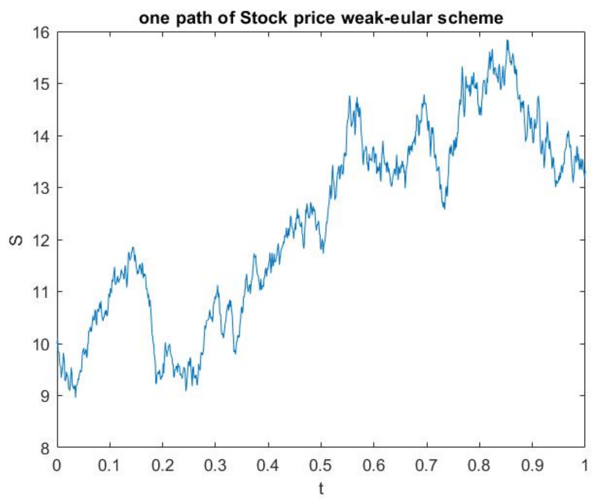


Figure 3.1: Stock Price Path Generated by weak-Euler Scheme

The Wiener process, as stated above, is generated from a standard normal distribution. This is done using a computerised random generator. For this project, the generator used was MATLAB's inbuilt randn function, which works as described in Section 2.2.

3.1.2 Milstein Scheme

The second method we have used to approximate the solution of the SDE (equation 3.1) is the Milstein scheme. To compute the approximation, we again partition our interval and generate a scaled normally distributed Wiener process denoted $\{W_i\}$. The method now differs from the weak-Euler scheme. For generality, consider an SDE of the form,

$$dX_t = p(X_t)dt + q(X_t)dW_t \tag{3.6}$$

We can approximate the solution $X(t)$ via the following iteration method [9].

$$X_{i+1} = X_i + p(X_i)\Delta t + q(X_i)\Delta W_i + \frac{1}{2}q(X_i)\frac{\partial q(X_i)}{\partial X_t}((\Delta W_i)^2 - \Delta t) \tag{3.7}$$

Of course, this approximation may now be applied to SDE (equation 3.1) by taking $p = \mu X(t)$ and $q = \sigma X(t)$. To recall, the parameters used for the weak-Euler method were $r = 0.06,\ \sigma = 0.36,\ X_0 = 10,\ K = 10,$ and $T = 1$. These same parameters may now be used in the Milstein scheme to compute an approximate set of prices X_t that solve the SDE. Simulating this path, we find the stock prices follow a very similar structure to those produced via the weak-Euler method in figure 3.1. This can be seen in figure 3.2.

3.1.3 Deriving a New SDE - Analytical Solution

Another approach to solving this SDE is to use Ito's formula. First we must perform the following calculations. We define the following functions [9].

$$a(t, X(t)) = r,\ b(t, X(t)) = \sigma,\ g(t, X(t)) = \ln(X(t))$$

Applying Itos formula we arrive at the following SDE problem.

$$dg(t, X(t)) = \left[0 + rX(t)\left(\frac{1}{X(t)}\right) + \frac{1}{2}\sigma^2 X(t)^2\left(\frac{1}{X(t)^2}\right)\right]dt + \sigma X(t)\left(\frac{1}{X(t)}\right)dW^Q(t)$$

Simplifying we receive the following SDE problem:

$$d\ln(X(t)) = (r - \frac{1}{2}\sigma^2)dt + \sigma dW^Q(t). \tag{3.8}$$

In integral form, we can rewrite this problem as follows:

$$\ln(X(t) = \ln(X(0)) + \int_0^t (r - \frac{1}{2}\sigma^2)ds + \int_0^t \sigma dW^Q$$

Therefore taking exponents of both sides and simplifying we receive the final problem:

$$X(t) = X(0)\exp\left(\int_0^t (r - \frac{1}{2}\sigma^2)ds + \int_0^t \sigma dW^Q\right) \tag{3.9}$$

The first term in the exponent can be easily integrated and evaluated given r and σ whereas the second term must be rewrote as a summation using properties of Ito stochastic integrals. We arrive at the final equation:

$$X(t) = X(0)\exp\left((r - \frac{1}{2}\sigma^2)t + \sum_{i=1}^{t}\sigma W(i)\right) \tag{3.10}$$

Given the same set of parameters used in section 3.1.1, this expression is entirely known. This provides us with our second approximation which we may evaluate this for M=1000 different times points, and receive the following figure below.

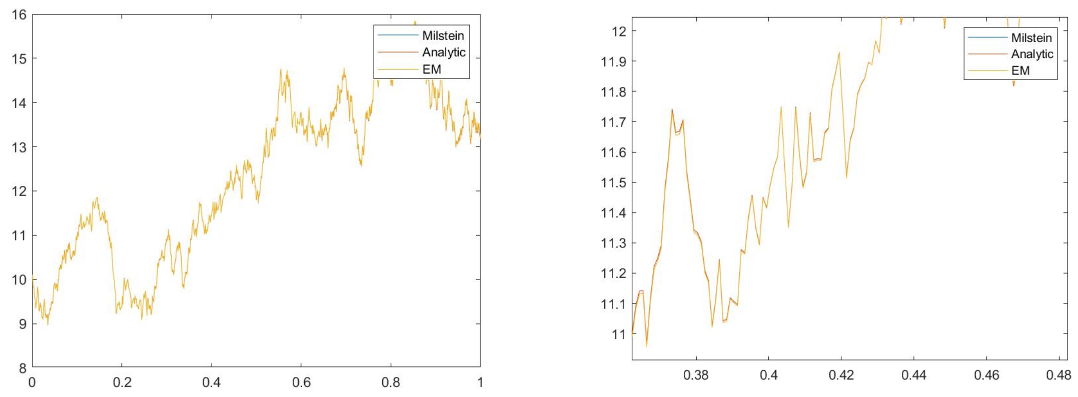


Figure 3.2: Result of 3 SDE schemes

3.1.4 Convergence of Schemes

Now we talk about the theoretical convergence of weak-Euler and Milstein. We can get their errors from stochastic Taylor expansion

$$\frac{d}{dt}X(t) = a[X(t)] \tag{3.11}$$

$$\frac{d}{dt}f[X(t)] = a[X(t)]\frac{\partial}{\partial X}f[X(t)] \tag{3.12}$$

Define $\mathcal{L} \equiv a(X)\frac{\partial}{\partial X}$, and the equation above can be written in form of integral:

$$f[X(t)] = f[X(t_0)] + \int_{t_0}^{t} \mathcal{L}f[X(\tau_1)]\,d\tau_1 \tag{3.13}$$

Expanding $f[X(t)]$ as in the above form:

$$\begin{aligned} f[X(t)] &= f\left[X\left(t_0\right)\right] + \int_{t_0}^{t} \mathcal{L}\left[f\left[X\left(t_0\right)\right] + \int_{t_0}^{\tau_1} \mathcal{L} f\left[X\left(\tau_2\right)\right] d\tau_2\right] d\tau_1 \\ &= f\left[X\left(t_0\right)\right] + \mathcal{L} f\left[X\left(t_0\right)\right] \int_{t_0}^{t} d\tau_1 + \int_{t_0}^{t} \int_{t_0}^{\tau_1} \mathcal{L}^2 f\left[X\left(\tau_2\right)\right] d\tau_2 d\tau_1 \\ &= f\left[X\left(t_0\right)\right] + \mathcal{L} f\left[X\left(t_0\right)\right]\left(t-t_0\right) + \int_{t_0}^{t} \int_{t_0}^{\tau_1} \mathcal{L}^2 f\left[X\left(\tau_2\right)\right] d\tau_2 d\tau_1 \end{aligned} \tag{3.14}$$

As

$$\begin{aligned} \mathcal{L}^2 f\left[X\left(\tau_2\right)\right] &= \mathcal{L}^2\left[f\left[X\left(t_0\right)\right] + \int_{t_0}^{\tau_2} \mathcal{L} f\left[X\left(\tau_3\right)\right] d\tau_3\right] \\ &= \mathcal{L}^2 f\left[X\left(t_0\right)\right] + \underbrace{\mathcal{L}^2 \int_{t_0}^{\tau_2} \mathcal{L} f\left[X\left(\tau_3\right)\right] d\tau_3}_{O(\mathcal{L}^3)} \end{aligned} \tag{3.15}$$

Finally we get:

$$f[X(t)] = f\left[X\left(t_0\right)\right] + \mathcal{L} f\left[X\left(t_0\right)\right]\left(t-t_0\right) + \frac{1}{2}\mathcal{L}^2 f\left[X\left(t_0\right)\right]\left(t-t_0\right)^2 + O\left(\mathcal{L}^3\right) \tag{3.16}$$

For the weak-Euler scheme, we have error $O(\mathcal{L}^2)$ for every step, and the total error will be $O(\mathcal{L})$.

$$\begin{aligned} f[X(t)] &= f\left[X\left(t_0\right)\right] + \mathcal{L} f\left[X\left(t_0\right)\right]\left(t-t_0\right) + \underbrace{\frac{1}{2}\mathcal{L}^2 f\left[X\left(t_0\right)\right]\left(t-t_0\right)^2 + O\left(\mathcal{L}^3\right)}_{O(\mathcal{L}^2)} \\ &= f\left[X\left(t_0\right)\right] + \mathcal{L} f\left[X\left(t_0\right)\right]\left(t-t_0\right) + O(\mathcal{L}^2) \end{aligned} \tag{3.17}$$

For Milstein scheme, we have error $O(\mathcal{L}^3)$ for every step, and the total error will be $O(\mathcal{L}^2)$.

$$f[X(t)] = f\left[X\left(t_0\right)\right] + \mathcal{L} f\left[X\left(t_0\right)\right]\left(t-t_0\right) + \frac{1}{2}\mathcal{L}^2 f\left[X\left(t_0\right)\right]\left(t-t_0\right)^2 + O\left(\mathcal{L}^3\right) \tag{3.18}$$

From equation (3.11) and the definition of $\mathcal{L}$:

$$\mathcal{L} = \frac{\partial}{\partial X}\frac{d}{\Delta t}X(t) = \Delta t$$

Since the error of the weak Euler scheme is $O(\mathcal{L}^2)$ at every time step over T, the total error of the weak Euler scheme is:

$$\begin{aligned} \frac{T}{\Delta t}O(\mathcal{L}^2) &= \frac{T}{\Delta t}O(\Delta t^2) \\ &= T\ O(\Delta t) \to O(\Delta t) \end{aligned} \tag{3.19}$$

Note that T is removed because it is a constant with no dimensions and O simply refers to the order of the error. Similarly, the error of the Milstein scheme can be shown to be $O(\Delta t^2)$. Recall equation (3.9). Taking expectations, we find:

$$E(X_T) = X_0 e^{rT} \tag{3.20}$$

$E(X_T)$ can be estimated by the three schemes we have previously defined. We may simulate these schemes multiple times to derive a series of approximations to $E(X_T)$. We can store these approximations in a vector and receive the following,

$$\underline{\hat{X}_T} = [\widehat{X}_T^{(1)}, \widehat{X}_T^{(2)}, ..., \widehat{X}_T^{(M-1)}, \widehat{X}_T^{(M)}], \tag{3.21}$$

where M is some large number, in our case 100,000 which is $\frac{1}{\Delta t^2}$. We may then use this vector to form one approximation for $\hat{X}_T$ by computing its mean. We therefore have that

$$E(\hat{X}_T) = \frac{1}{M}\sum_{i=1}^{M}\hat{X}_T^{(i)} \tag{3.22}$$

This approximation is of course not perfectly accurate. We may construct a confidence interval to demonstrate this. The confidence interval may be expressed as follows, where $\tilde{\sigma}$ is the estimated standard deviation of $\underline{\hat{X}_T}$ components:

$$E(\hat{X}_T) \pm 1.96\left(\frac{\tilde{\sigma}}{\sqrt{M}}\right) \tag{3.23}$$

The length of this interval may then be expressed as follows,

$$3.92\left(\frac{\tilde{\sigma}}{\sqrt{M}}\right) = 3.92\tilde{\sigma}\Delta t \tag{3.24}$$

We may now repeat this computation for a series of different values of Δt in order to study the relationship between our time step Δt and the corresponding size of the confidence interval for the approximation. We can plot this relationship and this can be seen in figure (3.3).

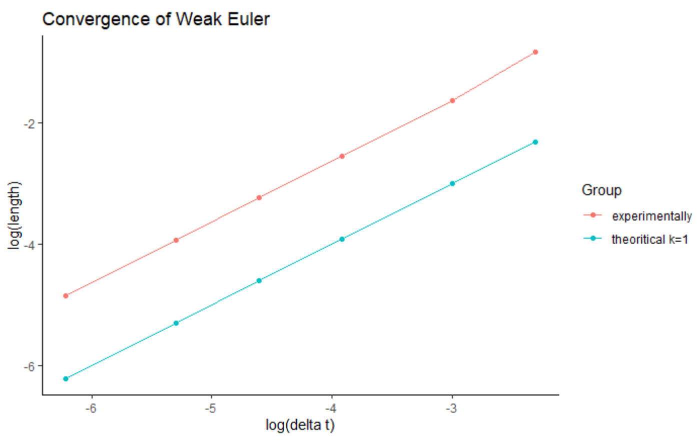


Figure 3.3: Convergence of weak-Euler scheme

Figure 3.3 demonstrates that log(Δt) against log(confidence interval length) for each of the errors produced by the Euler and Milstein schemes are linear, indicating that the confidence intervals for the stock price X_t do indeed shrink linearly with Δt.

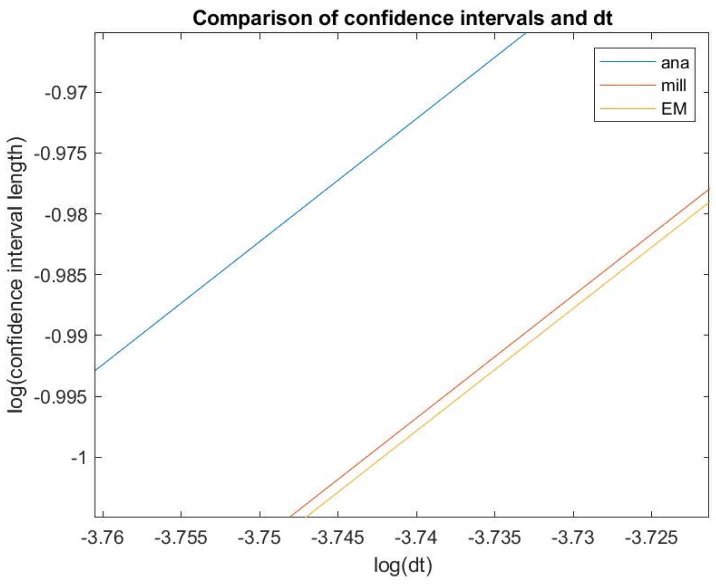


Figure 3.4: Result of 3 SDE schemes

Figure 3.4 shows the small differences between different schemes when generates GBM.

3.2 Option Price

3.2.1 Merton's Formula and Monte Carlo

Merton's formula finds an analytical solution for a European call option's price $V(t)$ at any time $0 \leq t \leq T$:

$$V(t) = x\Phi\left(\frac{\ln(\frac{x}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right) - Ke^{-r(T-t)}\Phi\left(\frac{\ln(\frac{x}{K} + (r - \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right) \quad (3.25)$$

and for the asset-or-nothing call option:

$$V(t) = x\Phi\left(\frac{\ln(\frac{x}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right), \quad (3.26)$$

where x is the stock price at t, K is the strike price, r is a constant interest rate, σ is the standard deviation of the stock price and T is the maturity time. As is common notation, Φ represents the cumulative normal distribution.

Since this solution is analytical, $V(t)$ does not depend on any stochastic element and therefore the graph of $V(t)$ against t is the same every time it is produced by a computer. Simulations produced by the weak-Euler scheme and the Milstein scheme can be compared to this analytical solution to assess their accuracy.

To implement a Monte Carlo method, stock prices must be simulated many times (e.g. using a weak-Euler or Milstein scheme) so that many different random paths for the stock price are produced. Thus, if there are M simulations, a set of M processes can be produced:

$$\{\{X_{t_i}^1\}, \{X_{t_i}^2\}, ..., \{X_{t_i}^M\}\}$$

The payoff function for the option at maturity time, $f(X_T)$, is calculated for each of these generated paths, then $E_Q[f(X_T)]$ is estimated by averaging the final stock price at maturity time across all simulations:

$$E_Q[f(X_T)] \approx \frac{1}{M}\sum_{j=1}^{M} f(X_T^j) \quad (3.27)$$

Finally, a Monte Carlo estimate $\hat{V}(0)$ for $V(0)$ can be obtained from (3.5) and (3.27) as such:

$$V(0) \approx e^{-rT}\frac{1}{M}\sum_{j=1}^{M} f(X_T^j) = \hat{V}(0) \quad (3.28)$$

For example, if a weak-Euler scheme were run 100,000 times using the same parameters that produced the graph in Figure 3.1, the Monte Carlo estimate $\hat{V}(0)$ to estimate $V(0)$ estimates of the option price at $t = 0$ are estimated to be:

$$\hat{V}_c(0) = 1.701437, \qquad \hat{V}_b(0) = 6.341225,$$

where $\hat{V}_c(0)$ is the estimated option price for the call option and $\hat{V}_b(0)$ is the estimated option price for the binary asset-or-nothing call option.

3.2.2 Convergence of the Monte Carlo Method Applied to the Weak-Euler Scheme

Comparing the estimations produced by weak-Euler's scheme to the values calculated by Merton's formula, the option prices estimated by a Monte Carlo method show convergence to a normal distribution with $O(M^{-\frac{1}{2}}) + O(\Delta t)$. This comes from the error of the weak Euler scheme as discussed in section 3.1.4 combined with the Monte Carlo error as discussed in section 2.1.2:

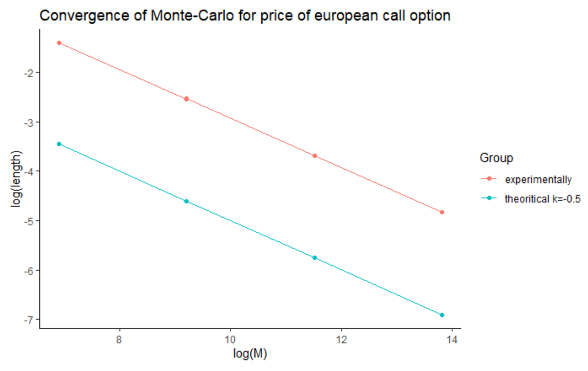
$$V(0) - \hat{V}(0) = \underbrace{V(0) - \bar{V}(0)}_{O(\Delta t)} + \underbrace{\bar{V}(0) - \hat{V}(0)}_{O(M^{-\frac{1}{2}})} \quad (3.29)$$

where $O(\Delta t)$ is caused by weak-Euler scheme and when we set $\Delta t = M^{-\frac{1}{2}}$, this leads the error for $\hat{V}(0)$ to be $O(M^{-\frac{1}{2}})$ and the 95% confidence interval of V(0):

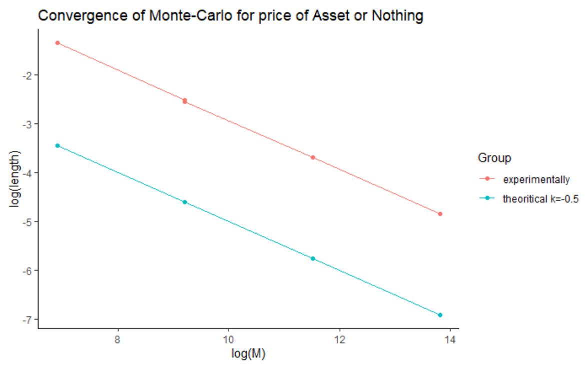
$$\left[\hat{V}(0) \pm 1.96\frac{\tilde{\sigma}}{\sqrt{M}}\right] \tag{3.30}$$

This error convergence has been demonstrated below for both the European call option and binary asset-or-nothing call option. For increasing values of M, $\hat{V}(0)$ converges to $V(0)$ as the confidence interval shrinks linearly by a factor of $\sqrt{M}$. We set $\Delta t = M^{-\frac{1}{2}}$, and result is shown in figure 3.5. The price does indeed converge to the true value of the delta at rate $M^{-0.5}$. (We set M $= (10^3, 10^4, 10^5, 10^6)$, and $\Delta t = \frac{1}{\sqrt{M}}$ and then compute the length of confidence interval. In simulating GBM, we use $r = 0.06,\ \sigma = 0.36,\ X_0 = 10,\ K = 10,$ and $T = 1$, which is the same as we used in section 3.1.1.

The below plots show convergence of the Monte Carlo method applied to each of the European call and asset or nothing call options.



(a) Call option



(b) Binary asset-or-nothing call option

Figure 3.5: Confidence interval lengths by number of trials

Table 3.1 shows the 95% confidence intervals for price of the two type of options. As we can see, the interval of price shrinks as Δt decreases or M increase (as we have set $\Delta t = \frac{1}{\sqrt{M}}$). Furthermore, $\hat{u}(0, X_0)$ is getting closer to the theoretical and exact value $u(0, X_0)$ (1.69706 for European option and 6.355791 for asset-or nothing option).

Δt	M	European call Option ($\hat{u}(0, X_0)$)	Asset-or-Nothing call Option ($\hat{u}(0, X_0)$)
0.1	1×10^2	1.7564 $\pm$0.3490	6.7350 $\pm$0.3026
0.05	4×10^2	1.6666 $\pm$0.1408	6.2592 $\pm$0.1316
0.01	1×10^4	1.6967 $\pm$0.0184	6.3824 $\pm$0.0183
0.005	4×10^4	1.6949 $\pm$0.0077	6.3481 $\pm$0.0077
0.001	1×10^6	1.6952 $\pm$0.0058	6.3564 $\pm$0.0058

Table 3.1: Confidence interval Price for the two options by Monte-Carlo

3.3 Portfolio of GBM

3.3.1 GBM Hedging Portfolio

When we consider the value of the self-financing portfolio at any given time t, we know it can be evaluated as the total value of stock held at the current price $X(t)$ and the cash in the savings account $\phi(t)$. In this section the delta variable is also the greek, as defined in equation (3.31).

$$\Pi(t) = \Delta(t)X(t) + \phi(t), \tag{3.31}$$

In the GBM model, the units of stock held at time t can be defined as in (3.32).

$$\Delta(t) = \frac{\partial V(t)}{\partial X(t)} \tag{3.32}$$

Theoretically, delta of European call option is given by

$$\Delta(t) = \Phi\left(\frac{\ln(\frac{x}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right) \tag{3.33}$$

and delta of Asset-or-nothing call option is given by

$$\begin{aligned}\Delta(t) &= \frac{\partial}{\partial x}(x\Phi\left(\frac{\ln(\frac{x}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right)) \\ &= \Phi\left(\frac{\ln(\frac{x}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right) + \frac{1}{\sigma\sqrt{T-t}}\Phi'\left(\frac{\ln(\frac{x}{K} + (r + \frac{\sigma^2}{2})(T-t)}{\sigma\sqrt{T-t}}\right)\end{aligned} \tag{3.34}$$

As we are dealing with a series of discrete values, we must approximate a numerical solution to the delta variable, as opposed to being able to calculate an exact analytical solution. Therefore, we must calculate the number of units of stock held at time t using the differences in option price $V(t)$, and stock price $X(t)$, between successive data points in time.

$$\Delta(t_i) \approx \frac{u(t, x+\Delta x) - u(t, x-\Delta x)}{2\Delta x} \tag{3.35}$$

Returning to the portfolio equation, note that the value of the portfolio at the successive time-step is now evaluated by adding the product of the number of units of stock at the previous time and the new value of the stock price, to the bank account that has accrued interest over the elapsed time.

$$\Pi(t_{i+1}) = \Delta(t_i)X(t_{i+1}) + (1 + r(t_i)\Delta t)\phi(t_i), \tag{3.36}$$

where,

$$\Delta t = t_{i+1} - t_i \tag{3.37}$$

However, by definition the value of the portfolio is calculated as:

$$\Pi(t_{i+1}) = \Delta(t_{i+1})X(t_{i+1}) + \phi(t_{i+1}) \tag{3.38}$$

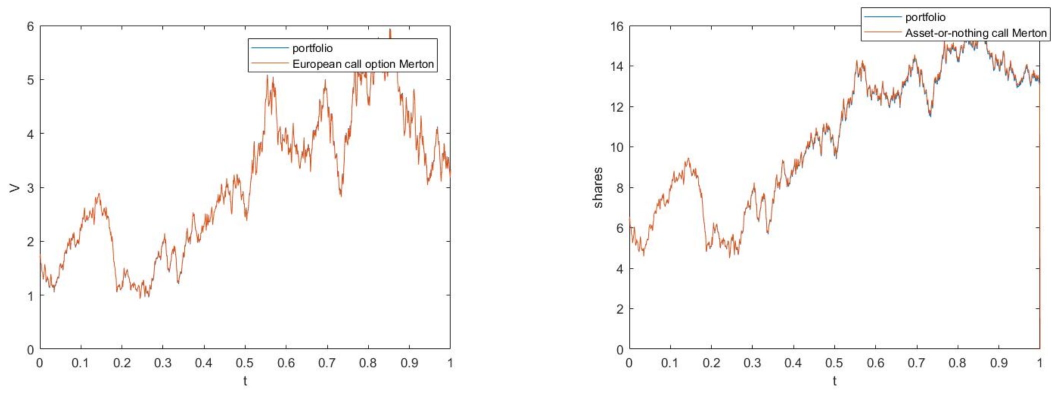
This gives,

$$\phi(t_{i+1}) = \phi(t_i) + (\Delta(t_i) - \Delta(t_{i+1}))X(t_{i+1}), \tag{3.39}$$

which leads us to the conclusion that our model is valid if and only if:

$$V(t_i) = \Pi(t_i), \forall t_i \tag{3.40}$$

By Merton's formula and self-financing condition, result of the portfolio and option price is shown in Figure 3.6.



(a) Portfolio for a call option (b) Portfolio for a asset-or-nothing call option

Figure 3.6: Portfolio compared to option price

3.3.2 Simulating the Delta by Monte Carlo

To simulate the delta, a Monte Carlo method must be derived to determine $V(t)$ values with $0 < t < T$, since thus far this report has only produced a Monte Carlo method to find $V(0)$.

The first step in estimating the delta for a given $\{X_t\}$ path is to generate this path using an weak-Euler scheme as described above. Then, m new paths can be generated from each time point up to time T and an expectation calculated on the payoffs at T to again provide a Monte Carlo approximation $\hat{V}(t)$. This is best seen using an exampleas outlined in the algorithm below for finding $\hat{V}(1)$:

1. Generate an approximate stock price path $\{\hat{X}_{t_i}\}$ using a weak-Euler scheme as in 3.1.1 and approximating as in 3.32.

2. Observe the value $\hat{X}_{t_1}$

3. Generate m new paths $\{\{A^1_{t_i}\}, ...\{A^m_{t_i}\}\}$ using the same weak-Euler scheme as for $\{\hat{X}_{t_i}\}$ but this time the paths start from from t_1 instead of t_0, with $A^j_{t_1} = \hat{X}_{t_1}$, continuing until maturity T, A^j_T.

4. Calculate $u(0, x + \Delta x) = e^{-r(T-1)} \frac{1}{m} \sum_{j=1}^m f(A^j_T), (A^j_0 = x + \Delta x)$
   and $u(0, x - \Delta x) = e^{-r(T-1)} \frac{1}{m} \sum_{j=1}^m f(A^j_T), (A^j_0 = x - \Delta x)$

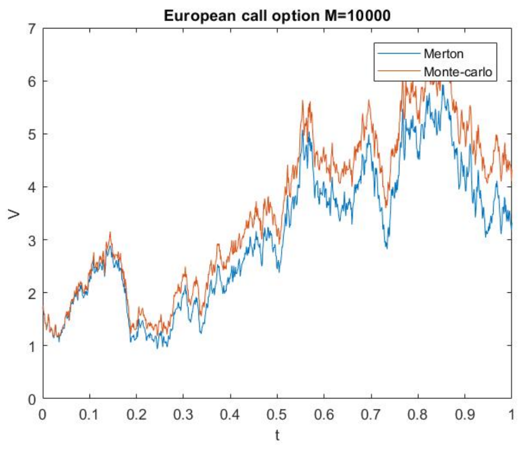
The above process can be repeated for $t_i \in \{t_2, t_3, ..., T\}$. Step 4 can be justified by recalling equation (3.3):

$$V(t) = e^{-r(T-t)} E_Q[f(X_T)|\mathcal{F}_t]$$

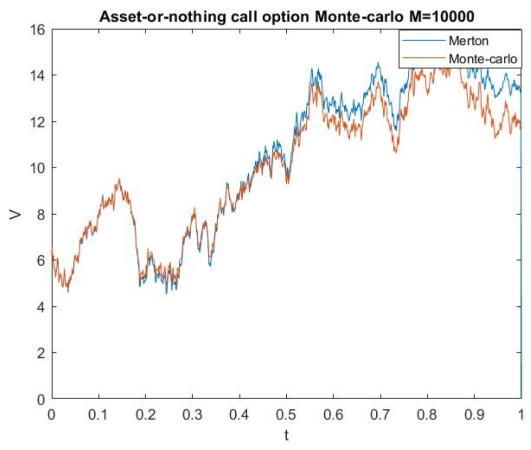
and therefore V(t) can be estimated by

$$\hat{V}(t) = e^{-r(T-t)} \frac{1}{m} \sum_{j=1}^m f(A^j_T)$$

The figure of $V(t)$ compared to $\hat{V}(t)$ are shown in 3.7 for each of the call options considered in this report.



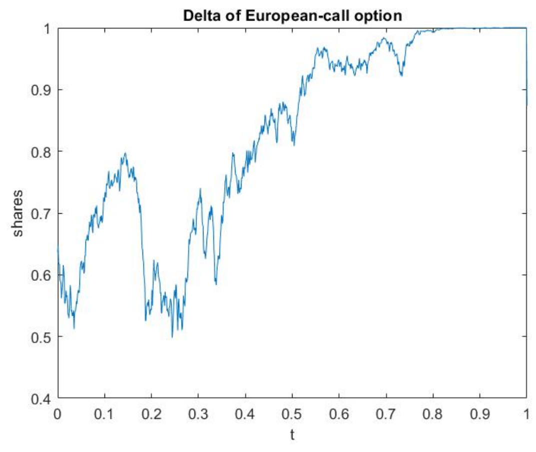
(a) $V(t)$,$\hat{V}(t)$ for a call option



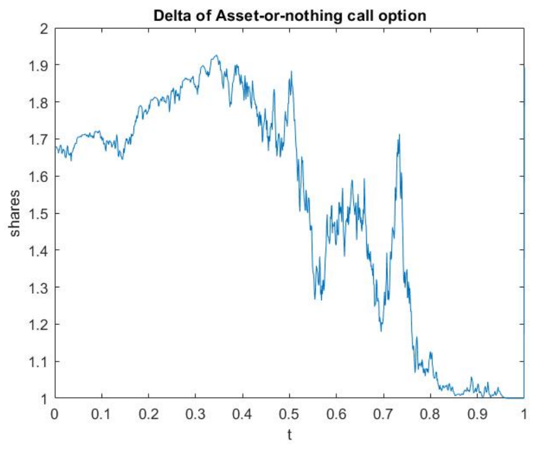
(b) $V(t)$,$\hat{V}(t)$ for a binary asset-or-nothing call option

Figure 3.7: Example: Simulations compared to analytical solution M=10,000

Substituting these Monte Carlo estimates for $\hat{V}(t)$ into the delta equation given in 3.35 gives an approximate delta path using the Monte Carlo method. The results of deltas produced from simulations with M=10,000 are shown below:



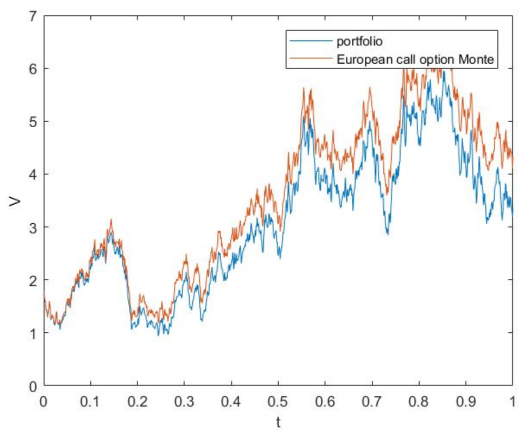
(a) Delta for a call option



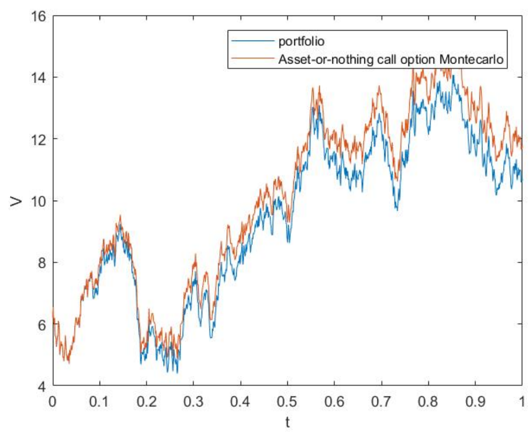
(b) Delta for an asset-or-nothing call option

Figure 3.8: Example: Delta by Monte Carlo, M=10,000

Applying self-financing into this delta problem produces the portfolio plots shown below:



(a) Portfolio for a call option



(b) Portfolio for an asset-or-nothing call option

Figure 3.9: Example: Portfolio By Monte Carlo, M=10000

3.3.3 Convergence of the Delta

Reference [10] should be used for the derivations involved in this section. The delta for an option can be estimated using the central finite difference and weak-Euler scheme:

$$\begin{aligned}\Delta(t) = \frac{\partial u}{\partial x} &= \frac{u(t,x+\Delta x)-u(t,x-\Delta x)}{2\Delta x} + O\left((\Delta x)^2\right) \\ &= \frac{\bar{u}(t,x+\Delta x)-\bar{u}(t,x-\Delta x)}{2\Delta x} + O(\Delta t + \frac{(\Delta t)^2}{2} + (\Delta x)^2)\end{aligned} \tag{3.41}$$

where O represents the error that comes from using the central finite difference to estimate $\Delta(t)$ and $\bar{u}$ is the mean estimate of u, $\bar{u}(t,x) = E(f(\hat{X}_T))$. Setting $\Delta x = \alpha\sqrt{\Delta t}$, where α is a real-valued constant, gives:

$$O\left(\Delta t + \frac{(\Delta t)^2}{2} + (\Delta x)^2\right) = O(\Delta t), \tag{3.42}$$

where $O(\Delta t)$ is the weak-Euler error. Now consider estimating u with $\hat{u}$ using a Monte Carlo method. Having discussed the error of the Monte Carlo error in section 3.2.2, it can be shown that:

$$\Delta(t) = \frac{\hat{u}(t,x+\Delta x)-\hat{u}(t,x-\Delta x)}{2\Delta x} + O(\Delta t) + O\left(\frac{1}{\sqrt{\Delta t}\sqrt{M}}\right), \tag{3.43}$$

where $O(\Delta t)$ contains the weak-Euler error and finite difference error as before and $O(\frac{1}{\sqrt{M}})$ is the Monte Carlo error.

To calculate the 95% confidence interval for $\frac{\partial u}{\partial x}$ for the simulated with Common Random Number and find:

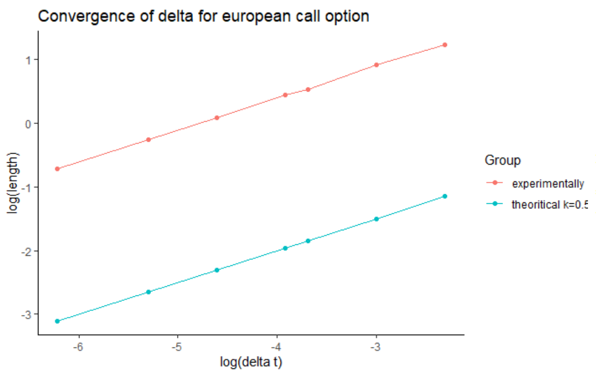
$$\left(\mu \pm 1.96\frac{\sqrt{\tilde{\sigma}_1^2+\tilde{\sigma}_2^2}}{2\sqrt{\Delta t}\sqrt{M}}\right) = \left(\mu \pm 1.96\frac{\sqrt{\tilde{\sigma}_1^2+\tilde{\sigma}_2^2}\sqrt{\Delta t}}{2}\right) \tag{3.44}$$

where

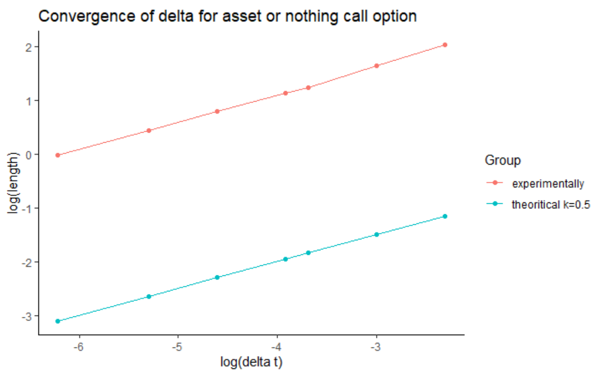
$$\mu = \frac{1}{2\alpha \Delta t^{1/2}}\frac{1}{M}\left[\sum_{m=1}^{M} f\left(\bar{X}^{(m)}_{t,x+\alpha,\Delta t^{1/2}}(T)\right) - \sum_{m=1}^{M} f\left(\bar{X}^{(m)}_{t,x-\alpha\Delta t^{1/2}}(T)\right)\right] \tag{3.45}$$

and $\tilde{\sigma}_1$ is the standard deviation of $\{f(X_T^1,...,f(X_T^M)\}|_{X(0)=10+\Delta x}$ where $\tilde{\sigma}_2$ is the standard deviation of $\{f(X_T^1,...,f(X_T^M)\}|_{X(0)=10-\Delta x}$, considering that $X(0)=10$ was the initial stock price used for all simulations outlined in this report thus far.

A plot of the log of the length of the 95% confidence interval against the log of Δt for the simulations have been plotted alongside the theoretical plots for each of the call option and the binary asset-or-nothing call option, as seen below: (we set $\Delta t = (0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001)$ and use $r = 0.06,\ \sigma = 0.36,\ X_0 = 10,\ K = 10,\ \text{and}\ T = 1$)



(a) Convergence plot for a European call delta



(b) Convergence lot for a binary asset or nothing delta

Figure 3.10: Convergence graphs for simulated deltas

Figure 3.10 shows that the simulated and theoretical plots have the same gradient, indicating that the delta does indeed converge to the true value of the delta at rate 0.5.

Table 3.1 shows the 95% confidence intervals for price of the two type of options. As we can see, the interval of price shrinks as Δt decreases, M increases or Δx decreases ($M = \frac{1}{\Delta t^2}, \Delta x = \sqrt{\Delta t}$). Furthermore, $\hat{\partial u}(0, X_0)$ is getting closer to the theoretical and exact value $\partial u(0, X_0)$ (0.635579 for European option and 1.679124 for asset-or nothing option).

Δt	M	Δx	European call Option ($\hat{\partial u}$)	Asset-or-Nothing call Option ($\hat{\partial u}$)
0.1	1×10^2	0.3162	0.5914 $\pm$0.4877	2.5026 $\pm$0.4680
0.05	4×10^2	0.2236	0.9061 $\pm$0.2016	1.2041 $\pm$0.1845
0.01	1×10^4	0.1000	0.7018 $\pm$0.0260	1.749 $\pm$0.0259
0.005	1×10^6	0.0707	0.6025 $\pm$0.0109	1.6003 $\pm$0.0108
0.001	4×10^6	0.0316	0.6362 $\pm$0.0082	1.6613 $\pm$0.0082

Table 3.2: Confidence interval Δ for the two options by Monte-Carlo

Chapter 4

SDE Model

4.1 Deriving the PDE

In order to establish our more complicated model of the option pricing process, we must begin with the derivatives of the stock price, risk free interest rate, and dividend equation respectively. These are defined as:

$$dX(t) = b(t, X(t))dt + b(t, X(t))dW(t) \tag{4.1}$$

$$dY(t) = -r(t, X(t))Y(t)dt = \frac{1}{B(t)} \tag{4.2}$$

$$dZ(t) = c(t, X(t))Y(t)dt, \tag{4.3}$$

where c in equation (4.3) is the fraction of the total percentage dividend payout, at the current time step.

The solution to equations (4.1), (4.2), and (4.3) can be obtained by integrating each side of the equation to give the following:

$$Y(t) = Y(0)\exp\left(\int_0^t r(t, X(t))dt\right) \tag{4.4}$$

$$Z(t) = \int_0^t cX(s)Y(s)ds, \tag{4.5}$$

where the risk free interest rate r is a Cox-Ingersoll-Ross (CIR) process. A CIR process is defined by:

$$dr(t) = a(b - r(t))dt + \sigma\sqrt{r(t)}dW^Q \tag{4.6}$$

The option price of our more complicated model is given by equation (4.11) below.

$$V(t) = u(t, X(t)) = E_Q[f(X_T)Y_{T-t} + Z_{T-t}|\mathcal{F}_t] \tag{4.7}$$

By Ito's formula considered under an EMM Q, the derivative of the stock price is given by:

$$dX(t) = r(t)X(t)dt + \sigma(t, X(t))dW^Q, \tag{4.8}$$

where dW is defined in terms of the risk neutral price as:

$$dW^Q(t) = -\theta(t) + dW(t) \tag{4.9}$$

and the risk neutral price is defined as:

$$\theta(t) = \frac{b(t, X(t)) - r(t)X(t) + c(t, X(t))}{\sigma(t, X(t))} \tag{4.10}$$

In order to reach the PDE to simulate the option price process, we need to deal with a martingale in order to make use of the integral form for the derivative. By using the famous Feynman-Kac formula,

$$u(t_0, x) = E(f(X_{t_0,x}(T))) \tag{4.11}$$

The derivative of $u(t, X(t))Y(t) + Z(t)$ then takes the Ito formula representation, allowing us to finally reach our PDE.

$$\begin{aligned} d[u(t,X(t))Y(t)] = Y(-r(t)u(t,X(t)))dt + Y\frac{\partial u(t,X(t))}{\partial t}dt + Yr(t)X(t)\frac{\partial u(t,X(t))}{\partial X}dt \\ + Y\frac{\sigma_z^2(t,X(t))}{2}\frac{\partial^2 u(t,X(t))}{\partial X^2}dt + Y\sigma(t,X(t))dW^Q(t) \end{aligned} \tag{4.12}$$

As the term in dt goes to 0, we extract the PDE given below.

$$(-r(t)u(t,X(t))) + \frac{\partial u(t,X(t))}{\partial t} + r(t)X(t)\frac{\partial u(t,X(t))}{\partial X} + \frac{\sigma_z^2(t,X(t))}{2}\frac{\partial^2 u}{\partial X^2} = 0, \tag{4.13}$$

and:

$$\begin{aligned} V(t,X(t) &= E_Q(f(X_{t_0,x}(T)Y(T-t) + Z(T-t))|F_t) \\ &= u(t,X(t) + E_Q(Z(T-t)|F_t) \end{aligned} \tag{4.14}$$

4.1.1 Parameter Estimation

For our model considered in this report, the derivative of the risk free interest rate is seen in the equation below.

$$dr(t) = (\theta_1 - \theta_2 r(t))dt + \sigma_z\sqrt{r(t)}dW^Q(t) \tag{4.15}$$

In order to determine the values for the θ parameters in equation, certain parameter estimation techniques were used. First, the successive $r(t)$ term is defined.

$$r(t+1) = -\beta_1 r(t) + \beta_0 \tag{4.16}$$

Next, an ordinary least squares regression (OLS) process was implemented, resulting in a response vector to build our version of equation (4.5) to use in the parameter estimation, and solve for θ and σ. The OLS scheme gave the following:

$$Y = \beta_1 X + \beta_0 + \epsilon \tag{4.17}$$

Note that here $Y = (r_1, \ldots, r_n)$ is the response vector, and $X = (r_0, \ldots, r_{n-1})$ is the initial column vector.

Using equation (4.6), the θ parameters were finally estimated as follows:

$$\theta_1 = \frac{\beta_0}{\Delta t} = ab, \theta_2 = \frac{1-\beta_1}{\Delta t} = a, \sigma_z = \frac{sd(\frac{\epsilon}{\sqrt{X}})}{\Delta t} \tag{4.18}$$

where $sd(\frac{\epsilon}{\sqrt{X}})$ is the standard deviation of vector$\frac{\epsilon}{\sqrt{X}}$ and ϵ is the residual of the OLS estimation.

This estimates the values of our model to be:

$$a = 0.24044417, b = 0.04181835, \sigma_z = 0.45655528 \tag{4.19}$$

4.1.2 Monte Carlo and PDE: the Value of Option

Similar to GBM model, the estimation of the option price by Monte Carlo is given by

$$\hat{u}(t,x) = \frac{1}{M}\sum_{m=1}^{M}\left[f\left(\bar{X}_{t,x}^{(m)}(T)\right)\bar{Y}_{t,x,1}^{(m)}(T) + \bar{Z}_{t,x,1,0}^{(m)}(T)\right] \tag{4.20}$$

By the Feymann-Kac formula, we get:

$$(-r(t)u(t,X)) + \frac{\partial u}{\partial t} + (r(t)X)\frac{\partial u}{\partial X} + \frac{\sigma^2(t,X)}{2}\frac{\partial^2 u}{\partial X^2} = 0 \tag{4.21}$$

We use the Crank-Nicolson scheme to solve the PDE problem (see appendix 5.1). A tridiagonal solver is used to calculate each u_i^j component within the linear system described in (5.1). These components represent the option value at each discretized time and spatial coordinate in figure 5.1. Then we get the European put option price.

Applying the call-put parity of this situation to get call option price, which gives:

$$\begin{aligned} call(t) &= u(t,X(t)) + E_Q[X(T)Y(T-t) - KY(T-t)|F_t] \\ &= u(t,X(t)) + X(t) + KP(t,T) \end{aligned} \tag{4.22}$$

where

$$P(t,T) = \exp(A(t,T) - C(t,T)r(t)) \tag{4.23}$$

and

$$\begin{aligned} C(t) &= \frac{2\left(e^{\gamma t}-1\right)}{(\gamma+a)\left(e^{\gamma t}-1\right)+2\gamma} \\ A(t) &= \frac{2ab}{\sigma^2}\ln\left(\frac{2\gamma e^{(\gamma+a)t/2}}{(\gamma+a)\left(e^{\gamma t}-1\right)+2\gamma}\right) \end{aligned} \tag{4.24}$$

where $\gamma = \sqrt{a^2+2\sigma^2}$, r(t) is driven by CIR model. The final option value is characterised using:

$$V(t,X) = call(t) + E_Q[Z(T-t)|\mathcal{F}_t] \tag{4.25}$$

The expectation of $Z(t)$ is given by:

$$\begin{aligned} E_Q[Z(T-t)|\mathcal{F}_t] &= E_Q[\int_t^T cX(s)Y(s)ds|\mathcal{F}_t] \\ &= (T-t)cX(t)Y(t) \end{aligned} \tag{4.26}$$

Here, the first step substitutes the definition of $Z(t)$ 4.5, before utilising the fact that $Z(t)$ is a martingale under probability measure Q [See MATH4061 lecture notes, chapter 6]. Therefore, its expectation is itself and (4.25) becomes:

$$V(t,X) = call(t) + (T-t)X(t)Y(t) \tag{4.27}$$

In order to implement the iteration scheme described in (5.16), boundary conditions are applied. Furthermore, a backward induction method is used meaning the boundary conditions are only imposed at $t = T$, $x = 0$ and $x = A$. On figure 5.1 these are represented as the top, left and right boundaries respectively, where the bottom, $t = 0$, gives the options characteristics at the present day.

$$\text{Boundary Conditions } = \begin{cases} \dfrac{KY(T)}{Y(t)} & \text{if } x = 0 = u(t,0), \\ 0 & \text{if } x = A = u(t,A), \\ (K - X(T))_+ & \text{if } t = T = u(T,X) \end{cases} \tag{4.28}$$

Boundary: $x = 0$

According to (4.21)

$$u(t,0) = r(t)\frac{\partial u}{\partial t}(t,0) = u(0,0)\exp\left(\int r(t)dt\right) \tag{4.29}$$

and $u(T,0) = K$, then we get $u(0,0) = KY(T)$, then

$$u(t,0) = \frac{KY(T)}{Y(t)} \tag{4.30}$$

Boundary: $x = A$

At the boundary, the value of the option is characterised by:

$$u(t, A) = E_Q[(K - A)_+ Y(T - t)|\mathcal{F}_t], \tag{4.31}$$

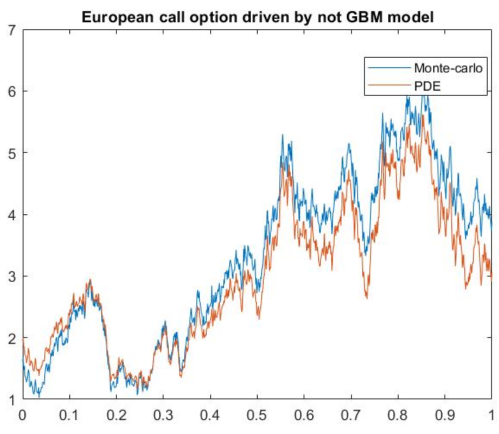
where A is the maximum spatial distance from the origin (see figure 5.1). Since $A > K$, the first term in (4.31) goes to zero, hence the value at the boundary is zero.

Boundary: $t = T$

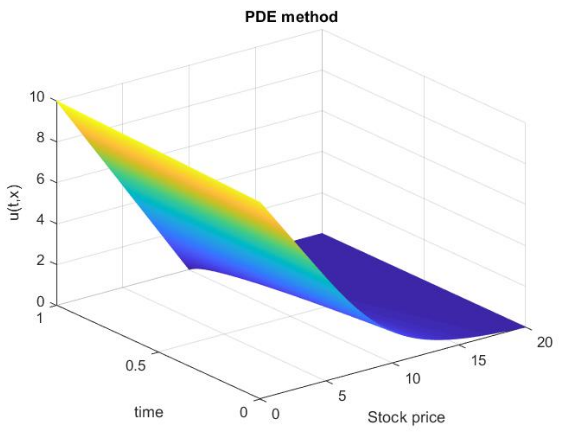
Given that a backward induction method is used, the boundary conditions at $t = T$ are simply the value of the option at T, which is described by:

$$\begin{aligned} u(T, X) &= E_Q[(K - X(T))_+ Y(T) + Z(T)|\mathcal{F}_t] & (4.32)\\ &= (K - x)_+ Y(T) + Z(T) & (4.33)\end{aligned}$$

For example, we set M=10,000 in Monte Carlo and solve the PDE by using Crank-Nicolson. The results of PDE and European call option are shown below



(a) European-call-option



(b) Approximation to PDE

Figure 4.1: Example: Monte-carlo compared with PDE, M=10000

4.2 PDE Model Hedging Portfolio

As we move to the more complicated SDE model as defined in this chapter, it remains sufficient to calculate the units of stock for the delta variable in the same way as described in section 3.3. This is because the numerical approximation is still calculated with the difference between successive points in the discretised series of points as a consequence of the difficulty associated with attaining the exact analytical solution.

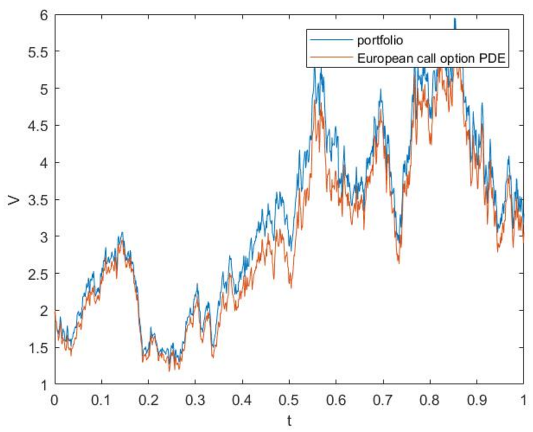


Figure 4.2: Hedging Portfolio For The PDE Model

Chapter 5

Appendix

5.1 Crank-Nicolson

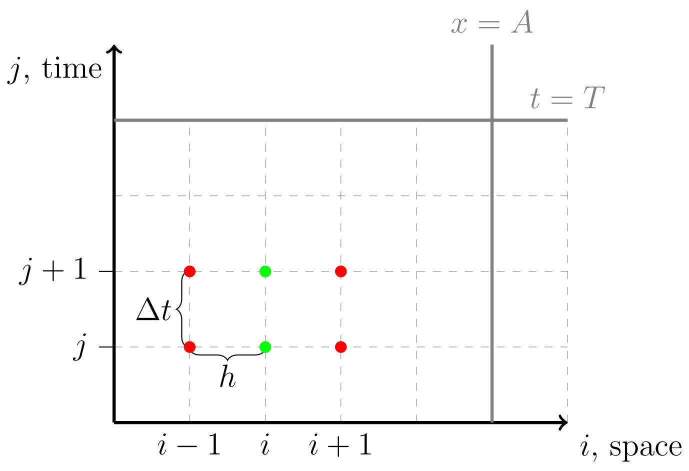


Figure 5.1: The Crank-Nicolson uniform mesh with coloured nodes representing points used each iteration.

By Feymann-kac formula, we get:

$$(-r(t)u(t,X)) + \frac{\partial u}{\partial t} + (r(t)X)\frac{\partial u}{\partial X} + \frac{\sigma^2(t,X)}{2}\frac{\partial^2 u}{\partial X^2} = 0 \tag{5.1}$$

To solve this PDE both the time and spatial components are discretize creating a uniform mesh as shown in Figure 5.1. The Crank-Nicolson schema is then applied, resulting in the following derivatives [Epperson 2013]:

$$\frac{\partial u}{\partial t} = \frac{u_j^{j+1} - u_j^i}{\Delta t} \tag{5.2}$$

$$\frac{\partial u}{\partial X} = \frac{1}{2}(\frac{u_{j+1}^{i+1} - u_j^{i+1}}{h} + \frac{u_{j+1}^i - u_j^i}{h}) \tag{5.3}$$

$$\frac{\partial^2 u}{\partial X^2} = \frac{1}{2}(\frac{u_{j+1}^{i+1} - 2u_j^{i+1} + u_{j-1}^{i+1}}{h^2} + \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2}) \tag{5.4}$$

Furthermore, the other components of equation (4.1) were chosen as follows:

$$r(t) = \frac{r^i + r^{i+1}}{2}, \quad r(t)X = \frac{r^i X_j^i + r^{i+1}X_j^{i+1}}{2} \tag{5.5}$$

$$r(t)u(t,X) = \frac{r^i u_j^i + r^{i+1}u_j^{i+1}}{2}, \quad \sigma^2(t,X) = \frac{(\sigma_j^i x_j^i)^2 + (\sigma_j^{i+1}x_j^{i+1})^2}{2} \tag{5.6}$$

Where,

$$r(t_i) = r^i \tag{5.7}$$

$$\sigma(t_i, X_j) = \sigma_j^i \tag{5.8}$$

Note, equations (4.5-4.8) were derived using a finite difference forward weak-Euler approximation. Inputting these into equation (4.1):

$$\begin{aligned} 0 \;=\; & \frac{-(r^i u_j^i + r^{i+1}u_j^{i+1})}{2} + \frac{u_j^{j+1} - u_j^i}{\Delta t} \\ & +(\frac{r^i X_j^i + r^{i+1}X_j^{i+1}}{2})\frac{1}{2}(\frac{u_{j+1}^{i+1} - u_j^{i+1}}{h} + \frac{u_{j+1}^i - u_j^i}{h}) \\ & +\frac{(\sigma_j^i x_j^i)^2 + (\sigma_j^{i+1}x_j^{i+1})^2}{4}(\frac{1}{2}(\frac{u_{j+1}^{i+1} - 2u_j^{i+1} + u_{j-1}^{i+1}}{h^2} + \frac{u_{j+1}^i - 2u_j^i + u_{j-1}^i}{h^2})) \end{aligned} \quad (5.9)$$

To solve equation (4.13) using the Crank-Nicolson schema, a tridiagonal matrix is formed at each iteration. To make this algebraically more interpretable, four components are introduced:

$$\alpha_j^i = \frac{r^i + r^{i+1}}{2},\; \theta_j^i = \frac{(\sigma_j^i X_j^i)^2 + (\sigma_j^{i+1}X_j^{i+1})^2}{2},\; \eta_j^i = \frac{r^i X_j^i + r^{i+1}X_j^{i+1}}{2} \quad (5.10)$$

Substituting these values into equation (4.13) and rearranging to isolate individual u_j^i components, an implicit iteration schema is created:

$$\begin{aligned} & (\frac{\theta_j^i}{2h^2})u_{j-1}^{i+1} + (\frac{1}{\Delta t} - \frac{\theta_j^i}{h^2} + \frac{\eta_j^i}{2h} - \frac{\alpha_j^i}{2})u_j^{i+1} + (-\frac{\eta_j^i}{2h} + \frac{\theta_j^i}{2h^2})u_{j+1}^{i+1} \\ & = (-\frac{\theta_j^i}{2h^2})u_{j-1}^i + (\frac{1}{\Delta t} + \frac{\theta_j^i}{h^2} - \frac{\eta_j^i}{2h} + \frac{\alpha_j^i}{2})u_j^i + (\frac{\eta_j^i}{2h} - \frac{\theta_j^i}{2h^2})u_{j+1}^i \end{aligned} \quad (5.11)$$

Again, to improve interpretability, new matrix components are introduced where:

$$\mathbf{A_j^i} = \frac{1}{\Delta t} - \frac{\theta_j^i}{h^2} + \frac{\eta_j^i}{2h} - \frac{\alpha_j^i}{2} \quad (5.12)$$

$$\mathbf{B_j^i} = -\frac{\eta_j^i}{2h} + \frac{\theta_j^i}{2h^2} \quad (5.13)$$

$$\mathbf{C_j^i} = \frac{1}{\Delta t} + \frac{\theta_j^i}{h^2} - \frac{\eta_j^i}{2h} + \frac{\alpha_j^i}{2} \quad (5.14)$$

$$\mathbf{D_j^i} = \frac{\eta_j^i}{2h} - \frac{\theta_j^i}{2h^2} \quad (5.15)$$

Hence, equation (4.18) becomes:

$$(\frac{\theta_j^i}{2h^2})u_{j-1}^{i+1} + \mathbf{A_j^i}u_j^{i+1} + \mathbf{B_j^i}u_{j+1}^{i+1} = (-\frac{\theta_j^i}{2h^2})u_{j-1}^i + \mathbf{C_j^i}u_j^i + \mathbf{D_j^i}u_{j+1}^i \quad (5.16)$$

This equation can be written as a linear system, whereby:

$$\begin{aligned} & \begin{bmatrix} C_1 & D_1 & & & \\ -\frac{\theta_2^i}{2h^2} & C_2 & D_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\theta_{N-2}^i}{2h^2} & C_{N-2} & D_{N-2} \\ & & & -\frac{\theta_{N-1}^i}{2h^2} & C_{N-1} \end{bmatrix} \cdot \begin{bmatrix} u_1^i \\ u_2^i \\ \vdots \\ u_{N-2}^i \\ u_{N-1}^i \end{bmatrix} \\ = & \begin{bmatrix} A_1 & B_1 & & & \\ \frac{\theta_2^i}{2h^2} & A_2 & B_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{\theta_{N-2}^i}{2h^2} & A_{N-2} & B_{N-2} \\ & & & \frac{\theta_{N-1}^i}{2h^2} & A_{N-1} \end{bmatrix} \cdot \begin{bmatrix} u_1^{i+1} \\ u_2^{i+1} \\ \vdots \\ u_{N-2}^{i+1} \\ u_{N-1}^{i+1} \end{bmatrix} \\ & + \begin{bmatrix} \frac{\theta_1^i}{2h^2}(u_0^{i+1} + u_0^i) \\ 0 \\ \vdots \\ 0 \\ \frac{-1}{2h} + B_{N-1}^i u_N^{i+1} + D_{N-1}^i u_N^i \end{bmatrix} \end{aligned} \quad (5.17)$$

5.2 Code Guide

5.2.1 GBM

Schemes for GBM

In 'code/GBM and its portfolio/Schemes for SDE(GBM)/' folder

Run function **'eular.m'**, **'MilsteinApprox.m'**, **'analyticalApprox.m'** to add them to the path of MATLAB compiler.

Then run the **'eularmain.m'**, you get one path of GBM driven by weak-Euler scheme. Run **'Milsteinmain.m'**, you get one path of GBM driven by Milstein scheme. Run **'analyticalApprox.m'**, you directly get one path of GBM.

Example of Option Price and Delta for GBM

In 'code/GBM and its portfolio/option and delta by Merton/' folder, Run function **'weak-Euler.m'** to add it to path of MATLAB.

Then run **'eucall.m'**, you get the value of European call option, European put option and Asset-or-nothing call option. In the workspace, **call** is European call option, **put** is European put option and **assetno** is Asset-or-nothing call option. They are all come from Merton's formula.

Close the picture but do not clear the workspace, run the **'portfolioforGBM.m'** you will see the portfolio, European-call-option by Merton and European-call-option by Monte-carlo where we set M=10000 in Monte-carlo.

Close the picture but do not clear the workspace, run the **'portfolioforGBM2.m'** you will see the portfolio, Asset-or-nothing call option by Merton and Asset-or-nothing call option where we set M=10000 in Monte-carlo.

In workspace, **delta** is delta of European-call-option, **deltaasset** is delta of Asset-or-nothing call option. **portfoli** is the portfolio value of the corresponding option.

Monte Carlo Option and Delta for GBM Changing dt

In 'code/GBM and its portfolio/option and delta by Montecarlo/' folder

Firstly, you can change the value of dt. We initially set dt=0.005, Run the **'delta-monte.m'**, you get $\hat{u}(0, X_0)$ and $\hat{\partial u}(0, X_0)$ of European call option. In workspave, callmonte is the option price at t=0, lenu is its confidence interval. delta0 is the delta at t=0 and lendelta is its confidence interval.

Run the **'assetdelta.m'**, you get $\hat{u}(0, X_0)$ and $\hat{\partial u}(0, X_0)$ of Asset-or-nothing call option. In workspave, callmonte is the option price at t=0, lenu is its confidence interval. delta0 is the delta at t=0 and lendelta is its confidence interval. Similarly, you can change the value of dt.

Convergence of schemes, option and delta for GBM

In 'code/GBM and its portfolio/option and delta by Montecarlo/ covergence of Weak schemes/Task4.rmd' We use R to do this part. There 7 chunks in file Task4.rmd.

The first chunk is to approximate price of European option at time 0. Run the first chunk and you will get two number, first number is the simulating price and the second is the exact price. (You can choose to use function 'St_GMB'(analytical solution) or function 'St_weakEuler' (weak-Euler scheme by signing one of the statements with " and using the other. So as the following chunks). The second chunk is to plot the convergence rate of weak-Euler scheme. Run it and you will get the graph.

The third chunk is to plot the convergence rate of Monte-Carlo for European option. Run it and you will get the graph.

The fourth chunk is to approximate delta of European option and plot the convergence.

The fifth chunk is to approximate the price of asset-or-nothing call option.

The sixth chunk is to plot convergence rate for asset-or-nothing call option by Monte-Carlo.

The last chunk is to approximate delta of asset-or-nothing call option and plot the convergence.

5.2.2 Other SDE model

Short interest rate

In code/Other model and PDE/short rate/ folder, run the **'CIR.m'** to add the function to path of MATLAB.

Then run **'shortrate.m'**, you can see one path of CIR compared with UK's short interest rate from 1975. **'shortrate.mat'** is the data of short rate from 1975.

Weak-euler scheme

In 'code/Other model and PDE/Weak euler for not GBM/' folder, run the **'notGBM.m'** to add the function to path of MATLAB.

Run the **'notGBMx.m'**, you will get paths of this stock model.

Boundary condition

In 'code/Other model and PDE/PDE solution/initializing boundary/' folder, run the **'integrade.m'** to add the function to path of MATLAB.

Run the **'Yt.m'**, then run **'integradey.m'**, at last run **'Zt.m'** and save the data of Yt, Zt in order to intialzing boundary condition in PDE solution.

PDE solution

In code/Other model and PDE/PDE solution/ folder, run the **'tridisolve.m'** to add it to the path of MATLAB.

Then run the **'option.m'** to get the matrix of PDE, save the data of V as V4.mat which is used to do the plot in **'optionplot.m'**.

Then run the **'optionplot.m'**, you will see the European call option solved by PDE.

Close the figure but do not clear the workspace, run 'portfolio.m', you will see the comparison between self-financing portfolio and option price.

PDE and Monte-carlo

In 'code/Other model and PDE/Monte-carlo solution/' folder, run the **'montenotGBM.m'** to add it to path of MATLAB.

Then run **'monteoptionnotGBM.m'**, you will get European call option solved by Monte-carlo. It takes a very long time(almost 10-20 mins) to run this code, its result is saved as **callnotgbmmonte.mat** in the same folder.

Run **'monteandpde.m'**, you will see comparison between Monte-carlo and PDE solution.

Bibliography

[1] John Guttag. 6. monte carlo simulation - youtube. https://www.youtube.com/watch?v=OgO1gpXSUzUt=2641s, May 2019.

[2] Bernard Lapeyre Damien Lamburton. *Introduction to Stochastic Calculus Applied to Finance.* Taylor Francis Group, LLC, 2008.

[3] Desmond J. Higham. *Introduction to Financial Option Valuation.* Cambridge University Press, 2004.

[4] Paul Glasserman. *Monte Carlo Methods in Financial Engineering.* Springer, 2003.

[5] Cleve Moler. *Numerical Computing with MATLAB.* 2004.

[6] The ziggurat algorithm for random gaussian sampling, 2011.

[7] JlAN-QlANG HU MICHAEL C. FU. Sensitivity analysis for monte carlo simulation of option pricing. pages 417–446, 1995.

[8] P. Boyle. Options: A monte carlo approach. pages 323–338, 1977.

[9] Michael J Panik. *Stochastic differential equations : an introduction with applications in population dynamics modeling.* Hoboken, New Jersey : Wile, 2017.

[10] M.V. Tretyakov G.N. Milstein. Numerical analysis of monte carlo evaluation of greeks by Önite di§erences. page 11, 2005.