Numerical and Analytic Methods in Option Pricing

D. Edwards

Student Number: 20006274

Department of Mathematics and Statistics
University of Reading

June 25, 2015

Abstract

In this paper we discuss how to price American, European and Asian options using a geometric Brownian motion model for stock price. We investigate the analytic solution for Black-Scholes differential equation for European options and consider numerical methods for approximating the price of other types of options. These numerical methods include Monte Carlo, binomial trees, trinomial trees and finite difference methods. We conclude our discussion with an investigation of how these methods perform with respect to the changes in different Greeks. Further analysing how the value of a certain Greeks affect the price of a given option.

Contents

List of Figures

List of Tables

1 Introduction

1.1 The History and Origin of Options

Options, in one form or another, have been around for at least centuries. There are reports that they have been around for millennia, being used as early as the sixth century B.C. in Ancient Greece (Sinclair, 2010). Further there are many other instances where we see options crop up in history, clauses in marine cargo contracts, such as those of the Romans and Phoenicians, would now be considered options.

The reason for their wide spread use is, in part, due to the guarantees that they afford to those who buy them. If we were to be traveling to a foreign country to buy something to trade, it is helpful to know when we get back that we are going to be able to sell our product for a price agreed upon prior to the trip. This allows us insurance on the trip, the knowledge of how much we will make when we return. Alternatively, if someone else was to go on a voyage to acquire something we wanted, and we believed that the price of that commodity was to go up, then making sure we could buy it at the current price would allow us to make profit.

The interest then became how to price these contracts. If you want to sell grain for twice its current price in one months time, it is obvious that the price of grain is unlikely to double in one month, thus the price of this contract would likely be very high. Conversely, if we were wanting to buy grain at twice the price in one months time, the cost of this contract would be very low. Owing again to the unlikely event that the grain does double in price. Thus someone would happily take on the contract cheaply knowing they can sell their grain to you for what is likely to be much more than it would be worth in a months time.

1.2 Option Basics

Before we may discuss the pricing of *options* we need to introduce a few terms. The most key of these being the definition of *options* and, we introduce this and some other key ideas here following the structure of Hull (2005, Chap. 8).

Definition 1.1 (Option)**.** *An **option** is a contract that gives the buyer the right, but not the obligation, to buy or sell an underlying asset at a specific price (known as the **strike price**) on or before a certain date.*

By an *underlying asset* we refer to the financial instrument on which the options price is based. This could be futures, stocks, commodities or currency, noting a change in the price of the underlying asset causes a change in the price of the option.

In the introduction we discussed the two types of options in an imprecise way, the buy side and sell side of an option. These concepts are extended within the next definition, to give rise to the two types of options that are used today, which we define here.

Definition 1.2 (Call)**.** *A **call** option is a contract that gives the buyer the right, but not the obligation, to buy an underlying asset at a specific price on or before a certain date (known as the **expiry date**).*

Definition 1.3 (Put)**.** *A **put** option is a contract that gives the buyer the right, but not the obligation, to sell an underlying asset at a specific price on or before a certain date (known as the **expiry date**).*

There are many different variations of puts and calls, the most popular being *European, American* and *Asian* option types. Other types of options are often referred to as *exotic options*, these include; basket options, barrier options, binary options, and down-and-out options. Each of these types has a different structure in the way the pay-out is calculated or when it may be *exercised*. This term is often used, and before we can differentiate between the types of options we must introduce it.

Definition 1.4 (Exercise)**.** *We say that we* ***exercise*** *an option when we decide to use that option. That is, to buy or sell the underlying asset the option corresponds to.*

The main difference between European and American options is when you can exercise them. The term **European** is given to those options that may be exercised **only** at the date of expiration. Alternatively **American** options may be exercised **at any time** before, or on, the date of expiration.

For both European and American options, as call options allow us to buy the stock at a specific price, the payoff would be given by $\max(S-K,0)$ and for a put $\max(K-S,0)$, where the stock price at the time the option is exercised (sometimes termed when it reaches maturity) is S and the strike price is K.

Asian options are very different from the European and American options. This is because the pay-out of this type of option is not dependent on a single value on or before the option expires.

Definition 1.5 (Asian Options)**.** *An option is said to be* ***Asian*** *or sometimes an* ***average option*** *if the pay-out depends on the average price of the underlying asset over a period of time.*

Insofar, we have discussed many different terms relating to options. We seek to give substance to these ideas with two brief examples, one for a put and one for a call. These are similar example as to the ones found in Hull (2005, P. 193) but have been adapted to better suit our discussion.

Example 1.1. Assume that a stock is trading at \$50 per share. We could buy a European call option, the right to buy the stock, with a strike price of \$55. This is essentially "betting the price goes up"; here we are hoping that the price rises to above \$55, and if this happens we make profit. Lets say that this option entitles us to buy 100 shares of the stock and cost us \$11. For a European option the pay-out, that is the amount of money we make on this transaction, is dependent on the price at the date of expiration. So we have three cases,

1. The stock ends below \$55. In this case we have lost money, that is the cost of the option \$11. This is because we are able to buy the stock for \$55, but it is trading at less than that, so exercising this option is worthless. So we have made no money here, yet only lost the cost of the option.

2. The stock rises to or above \$55.12. Here we have made a profit, we are able to buy 100 shares of the stock at \$55.12 or above. Hence, we buy these for our strike price \$55 and sell them for the price it is trading at. As the stock is on or above \$55.12 per share we make $\$55.12-\$55=\$0.12$, so for all 100 shares we make \$12. Finally, subtracting the cost of the option \$11 we have made at least a dollar. It is worth noting that as the price of the stock could increase up to any amount, our theoretical maximum profit here is infinite.

3. The stock ends between \$55 and \$55.11. Here we have made a profit. As with the profitable case, we buy the stock for \$55 and sell it for the price it is trading at. We see the maximum profit we make is $(\$55-\$55.11)\times 100=\$11$. However we have already paid \$11 for the option,

so we have lost money, yet we still exercise the option. This is because we make some money on the cost of the option back, i.e. if the stock was at \$55.07, then we only lose \$4 due to the \$7 dollar we made from the option.

Example 1.2. As our second example we will consider an American put on a stock. Here we have the right to sell a stock for a specified strike price, lets assume that this is \$45. Much like with the call this is a kind of bet that we are placing, here we are "betting that the stock will decrease in price". So lets say this option cost us \$15 then if the stock is trading at \$50, as with the previous example we either make nothing, reduce our losses or make profit. However this is an American option and as such can be exercised at any time between purchase and expiry dates. Hence, the only way we are guaranteed to lose the cost of the option is if the stock does not ever drop below \$44.85.

If it only drops below this value once and we do not exercise we would loose out. This shows the difficulty when choosing to exercise American options.

1.3 Preliminaries

Lastly before we begin our full discussion of the methods used to price options we must introduce a few basic mathematical results and notations that will be used throughout. These are mainly either results from stochastic calculus (Gardiner, 1985) or probability (Durrett, 2010).

Definition 1.6 (Stochastic variable)**.** *Given a probability space, with events x, we can introduce a* ***stochastic (or random) variable*** *as a function of x, denoted $f(x)$. In particular the identity function $I(x) = x$ is one such random variable. Continuing on we will use capital letters to denote random variables and lower case letters to denote their values.*

Note that the random variable can be either continuous or discrete.

Definition 1.7 (Stochastic process)**.** *If a random variable X is dependent on time, so that it is defined at different instances of time $t_1, t_2, \ldots, t_n$, then $X(t)$ is called a* ***stochastic process****.*

Definition 1.8 (Gaussian process)**.** *A process is said to be* ***Gaussian*** *if all possible distributions X_t are Gaussian. This means that a Gaussian process is characterised fully by the mean and variance of X_t.*

Definition 1.9 (Characteristic function)**.** *If X is a stochastic variable taking a continuous range of real numbers, its* ***characteristic function****, $G(s)$, is defined as,*

$$G(s) = \left\langle \mathrm{e}^{\mathrm{i}sX} \right\rangle = \int P(x)\mathrm{e}^{\mathrm{i}sx}\mathrm{d}x$$

where the integral varies over the range of x.

Example 1.3 (Characteristic function of a Gaussian)**.** The **Gaussian** or **normal distribution** is defined as,

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \qquad x \in (-\infty, \infty),$$

where μ is the mean and σ^2 the variance. Hence, from the definition of a characteristic function we have that,

$$\begin{aligned} G(s) &= \int P(x)\mathrm{e}^{\mathrm{i}sx}\mathrm{d}x \\ &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2} + \mathrm{i}sx\right) \\ &= \exp\left(-\frac{\sigma^2 s^2}{2} + \mathrm{i}s\mu\right) \end{aligned}$$

Definition 1.10 (The Itô integral)**.** *The **Itô integral** between two points t and t_0 is defined similarly to the Riemann integral. We first begin by discretizing time, a typical type would be $t_i = t_0 + \frac{i}{n}(t - t_0)$ for $i = 0, 1, \ldots, n$. Then the Itô integral of a function $g(X_t, t)$ will be defined as,*

$$\int_{t_0}^{t} g(X_t)\mathrm{d}W_t = \lim_{n\to\infty} \sum_{i=1}^{n} g(X_{t_i}, t)(W_{t_i} - W_{t_{i-1}})$$

where W_t is the Wiener process which we will introduce later. We will use the notation, $\Delta W_i = W_{t_i} - W_{t_{i-1}}$ and $\Delta t = t_i - t_{i-1}$

Definition 1.11 (Moments and their properties)**.** *The **moment** of order m is defined as,*

$$\mu_m = \begin{cases} \int x^m P(x)\mathrm{d}x & \text{for continuous variables} \\ \sum_i x_i^m P(x_i) & \text{for discreet variables.} \end{cases}$$

*The **average** of any function of a stochastic variable X, denoted $\mathbb{E}[f(X)]$, and is defined as,*

$$\mathbb{E}[f(X)] = \begin{cases} \int f(x)P(x)\mathrm{d}x & \text{for continuous variables} \\ \sum_i f(x_i)P(x_i) & \text{for discreet variables} \end{cases}$$

Then we may now write the definition of a moment as,

$$\mu_m = \mathbb{E}[x^m]$$

Lastly from the our definition, due to the linearity of sums and integrals as well as the fact the measure of a probability space is one, the following hold, for random variables X and Y and $a \in \mathbb{R}$:

1. $\mathbb{E}[aX] = a\mathbb{E}[X]$,
2. $\mathbb{E}[X + a = \mathbb{E}[X] + a$,
3. $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$,
4. $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ *if X and Y are independent.*

*The last property is derived from the fact that if they are independent then their **covariance** is zero, which we define below.*

Definition 1.12 (Variance and Covariance)**.** *The **variance** of a stochastic variable X is given by,*

$$\begin{aligned}\mathrm{Var}[X] &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2] - (\mathbb{E}[X])^2.\end{aligned}$$

*Furthermore we have the **covariance** of two stochastic variables X and Y is defined by,*

$$\begin{aligned}\mathrm{Covar} &= \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \\ &= \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]\end{aligned}$$

Definition 1.13 (Lognormal Distribution)**.** *A variable, X, is said to be **lognormally distributed** if its logarithm is normally distributed. Thus, $Y = \ln(X)$ is normally distributed. Further, if Y is normally distributed with mean a, and standard deviation b, we have that,*

$$\begin{aligned}\mathbb{E}[X] &= \mathrm{e}^{a+\frac{b^2}{2}}, \\ \mathrm{Var}[X] &= (\mathrm{e}^{\sigma^2} - 1)\mathrm{e}^{2a+b^2}.\end{aligned}$$

Definition 1.14 (Cumulative distribution function)**.** *The **cumulative distribution function**, $C_X(x)$, of a continuous random variable X is given by,*

$$C_X(x) = P(X \leq x),$$

Where $P(X \leq x)$ is the probability of $X < x$. The cumulative distribution function of the standard normal distribution is given by,

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-t^2/2} \mathrm{d}t.$$

Definition 1.15 (Probability density function)**.** *The **probability density function** (P.D.F.) of a continuous distribution is the derivative of the cumulative distribution function. This satisfies that if X is a random variables and has P.D.F. f then the expected value of X is given by,*

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x) \mathrm{d}x$$

2 The Black-Scholes Model for Stocks

The trouble with pricing options is without knowing the path the stock is likely to travel, it is difficult to price the option, as clearly the price of an option must be dependent on the stock price. Here we will develop a continuous time stochastic model for stocks and through this derive an equation for the price of options. We may then solve this explicitly for European options to gain the formulae for the pricing of European options; this model was first developed by Black and Scholes (1974) and further improved by Merton (1973). Before we develop this model, there are a number of key ideas and concepts we will need to introduce. We will follow much the same derivation as Hull (2005, Chap.9,12-13).

2.1 Factors Affecting the Price of Stock Options

As we expect the price of the option to reflect the price of the underlying asset, it seems logical that a factor that affects the price of the underlying asset would also affect the price of the option (Hull, 2005, Chap. 9). So for stock options we consider the factors that affect stock prices. There are six major factors. These are;

1. The current price the stock is trading at, S_0,
2. The strike price, K,
3. The time until expiration, T,
4. The volatility of the price of the stock, σ,
5. The risk-free interest rate, r,
6. The dividends that are expected to be paid, q.

Here the risk-free interest rate is the theoretical rate of return on a completely risk free investment. The volatility represents how the price varies over time, which we will give a more rigorous definition of this later. How each of these factors affects the price of European and American puts and calls is given in Table 1. We will discuss how each of these factors affects the option. Note that in Table 1 we use + to mean the factor increases the price of the option, a − for a decrease and a ? when the relationship is unknown.

Variable	European Call	European Put	American Call	American Put
Current stock price	+	-	+	-
Strike Price	-	+	-	+
Time to expiration	?	?	+	+
Volatility	+	+	+	+
Risk-free rate	+	-	+	-
Amount of future dividends	-	+	-	+

Table 1: The effect different factors have on the price of different options

Most of the ways each factor affects each type of option is intuitivee. If the price goes up, the price of a call goes up as we are likely to see greater increases in the price of the stock and the price of a put goes down as the decrease will not be as fast as the increase. This is due to the fact that, as we will see later, the path the stock takes is the initial price multiplied by some variables. Hence increases tend to happen faster than decreases.

The risk free interest rate is a more complex idea. Within the economy, as interest rates increase, investors expect more return from the option, however the value of any money earnt in the future decreases, due to these interest rates, termed inflation. This increases the price of stocks slightly resulting in a higher chance of calls paying off and less of puts paying off.

With relation to the dividends, as the ex-dividend date approaches, that is the date at which entitlement to dividends changes, the stock price decreases. Hence, due to the relationships for calls and puts, the price of calls decreases and the price of a put increases.

The most anomalous observations are those of the time to expiration and the volatility. The time to expiration for American options increases for both puts and calls. This is due to that, for two options, if the only difference is that one has a longer time to expiration, then the owner of the option with a longer time has all of the same opportunities to exercise and more. This increases the value of the option. For European options, this is not necessarily the case. If we have two options that straddle the ex-dividend date, the one with the shorter life would be worth more.

Finally we consider volatility, we have not defined volatility until this point, we will discuss definition in later sections. For now we merely remark that volatility is a measure of how uncertain we are of the stocks future price. If volatility increases, the chance the stock will do very well or very poorly increases. As our maximum loss from an option contract is the price but the profit is either infinite or large this benefits the owner hugely.

2.2 Markov and Wiener Processes

The next concept we will need to introduce is that of a *Markov process*. This is defined as a stochastic process with the *Markov property*; where the future value of the variable is dependent only on its current value and not its history. We often say that a Markov process is memoryless due too this property. We will examine a very specific type of Markov process, known as the *Wiener process* (Gardiner, 1985, Chap. 3).

Definition 2.1 (Wiener process)**.** *Let $W = (W_t)_{t\in[0,\infty)}$ be a continuous process. We say that this process is a* ***Wiener process*** *if the following properties hold:*

Property 1. *We require that if Δt is a small period of time,*

$$W_{t+\Delta t} - W_t = \epsilon\sqrt{\Delta t},$$

where ϵ has a standardised normal distribution (i.e. a normal distribution with a mean of zero and standard deviation of one).

Property 2. *We require that each increment is independent, so that W_t and W_s are independent for $0 \leq s < t$,*

Property 3. *The function defined by $t \to W_t$ is almost surely everywhere continuous (i.e. the probability that W_t is everywhere continuous is one).*

We see this is indeed a Markov process as the attribute described in Property 2 is precisely the Markov property. Also note that the first property shows that the Wiener process is normally distributed. An interesting implication of the Wiener process is found when we consider these normally distributed Markovian variables.

Proposition 2.1. *Let X and Y be two independent normally distributed variables with means μ_X and μ_Y respectively and variances σ_X^2 and σ_Y^2 respectively. Then the variable defined as $Z = X + Y$ is normally distributed with mean $\mu_X + \mu_Y$ and variance $\sigma_X^2 + \sigma_Y^2$. We will use the notation $A \sim N(\mu, \sigma^2)$ to mean "A variable A is normally distributed with mean μ and variance σ^2".*

Proof. The characteristic function of X and Y are by definition,

$$G_X(s) = \mathbb{E}\left[\mathrm{e}^{isX}\right] \text{ and } G_Y(s) = \mathbb{E}\left[\mathrm{e}^{isY}\right].$$

We sum these two variables to generate a new variable $Z = X + Y$, which has characteristic function given by (Durrett, 2010),

$$G_Z(s) = \mathbb{E}\left[\mathrm{e}^{\mathrm{i}sZ}\right] = \mathbb{E}\left[\mathrm{e}^{\mathrm{i}sX+\mathrm{i}sY)}\right] = \mathbb{E}\left[\mathrm{e}^{\mathrm{i}sY}\right]\mathbb{E}\left[\mathrm{e}^{\mathrm{i}sX}\right] = G_X(s)G_Y(s),$$

as these variables are independent. Now using the general formula for the characteristic function of a normal distribution from section 1.3 we have that,

$$\begin{aligned} G_z(s) = G_X(s)G_Y(s) &= \exp\left(\mathrm{i}t\mu_x - \frac{\sigma_X^2 s^2}{2}\right)\exp\left(\mathrm{i}t\mu_Y - \frac{\sigma_Y^2 s^2}{2}\right) \\ &= \exp\left(\mathrm{i}t(\mu_X + \mu_Y) - \frac{(\sigma_x^2 + \sigma_Y^2)s^2}{2}\right). \end{aligned}$$

Which is precisely the characteristic function of a normally distributed variable with mean $\mu_X + \mu_Y$ and variance $\sigma_X^2 + \sigma_Y^2$. Hence, $W \sim (\mu_X + \mu_Y, \sigma_X^2 + \sigma_Y^2)$ mean W has normal distribution with mean $\mu_X + \mu_Y$ and variance $\sigma_X^2 + \sigma_Y^2$. □

Consider a variable X that follows a Markov process. If we know that the change in value of a single day is a normal distribution with mean zero and variance one, then we may find the distribution for two days. Due to the Markov property the above proposition applies, and we have that the change in the variable over two days is normally distributed with mean zero and variance two.

Note that we may apply this as many times as we please to find the change in the variable over any period but the variance increases showing the uncertainty of these predictions.

It follows that we may consider the change in the variable W over a relatively large period of time W. We denote this $W_T - W_0$ and it can be thought of as the sum of changes in W in N small time intervals of length $\Delta t = T/N$. Hence we have that,

$$W_T - W_0 = \sum_{i=1}^{N} \epsilon_i \sqrt{\Delta t}, \tag{1}$$

where each of the ϵ_i for $i = 1, 2, \ldots, N$ are normally distributed with mean zero and variance one.

It follows from (1) that, the mean of $W_T - W_0$ is zero and has variance $N\Delta t = T$. In normal calculus we consider the limit of a discrete process as the changes head toward zero. Similar notations and conventions exist in stochastic calculus, here we will use the notation $\mathrm{d}W$ to refer to the Wiener process $W_{\Delta t+t} - W_t$ in the limit $\Delta t \to 0$.

2.3 Generalisation of the Wiener Process and Itôs Process

Insofar we have discussed the Wiener process, however there is a problem with this. We can see this by defining the *drift rate* and *variance rate*, these are given as the mean change per unit time and variance change per unit time. In our prior discussion of the Wiener process it is obvious that these have been zero and one respectively. This leads to a small issue, the expected value of W at any given time t, is equal to its current value. This is clearly an issue as stocks trend upward or downward. We may capture this aspect of stocks by generalising the Wiener process. The *generalised Wiener process* for a variable, here denoted as x is defined as,

$$\mathrm{d}x = a\mathrm{d}t + b\mathrm{d}W, \tag{2}$$

where a and b are constants and $\mathrm{d}W$ is the Wiener process. Here if the $\mathrm{d}W$ term where removed from (2), then it would have solution $x = x_0 + at$ for some x_0 specified by an initial condition. This is the way we assume the stock will grow, moving at a constant rate a. The $\mathrm{d}W$ term adds "noise" to this. As stock movement is assumed to be random, this term introduces the randomness in the form of a Wiener process.

Expressing this in discrete terms we have that $\Delta x = a\Delta t + b\epsilon\sqrt{\Delta t}$, with ϵ as before. Hence we have that the mean is now $a\Delta t$ and variance is $b^2\Delta t$. Again following similar arguments as with the Wiener process, we see that for the continuous process that the mean and variance are now aT and b^2T respectively.

It is easy to see that even this model which allows us to drift the stock either up or down in a given direction is not particularly favourable. It assumes that the drift is constant. This is, in practice, is not the case, a stock may rise at a constant rate but crash the next day. Here we introduce an *Itô process* to compensate for this.

Definition 2.2 (Itô processes)**.** *An Itô process is a is a type of generalised Wiener process where the constants a and b are now dependent on the value of the underlying variable x and time t. In mathematical terms,*

$$\mathrm{d}x = a(x,t)\mathrm{d}t + b(x,t)\mathrm{d}W.$$

Itô processes address the issues discussed. Following the same structure as previous arguments this has drift rate $a(x,t)$ and variance rate $b(x,t)^2$.

2.4 Itôs Lemma

Before we may prove Itôs lemma we must first prove a very important result, often regarded as the corner stone of Itô calculus, $\mathrm{d}W_t^2 = \mathrm{d}t$. This form of expressing this is merely short hand, the exact statement is given and proven below (Gardiner, 1985, P. 87-88).

Theorem 2.1 ($\mathrm{d}W_t^2 = \mathrm{d}t$)**.** *Given a function $g(X_t)$ we have that,*

$$\int_{t_0}^{t} g(X_t)\mathrm{d}W_t^2 = \int_{t_0}^{t} g(X_t)\mathrm{d}t.$$

Proof. To begin first let us define a new variable Y as,

$$\begin{aligned} Y &= \int_{t_0}^{t} g(X_t)\mathrm{d}W_t - \int_{t_0}^{t} g(X_t)\mathrm{d}t \\ &= \sum g(X_{i-1})\Delta W_{t_i}^2 - \sum g(X_{i-1})\Delta t_i \\ &= \sum g(X_{i-1})(\Delta W_{t_i}^2 - \Delta t_i) \end{aligned}$$

We may then calculate the moments of this new variable.

$$\begin{aligned} \mathbb{E}[Y] &= \lim_{n\to\infty} \sum_{i=1}^{n} \mathbb{E}\left[g(X_{i-1})(\Delta W_{t_i}^2 - \Delta t_i)\right] \\ &= \lim_{n\to\infty} \sum_{i=1}^{n} \mathbb{E}\left[g(X_{i-1})\right] \mathbb{E}\left[(\Delta W_{t_i}^2 - \Delta t_i)\right], \end{aligned}$$

as $g(X_{i-1})$ and $(\Delta W_{t_i}^2 - \Delta t_i)$ are independent, so the mean of the product is the product of the mean. Furthermore, as Δt is deterministic and merely a constant, from the properties discussed in the preliminaries,

$$\mathbb{E}[Y] = \lim_{n\to\infty} \sum_{i=1}^{n} \mathbb{E}[g(X_{i-1})] \left(\mathbb{E}\left[\Delta W_{t_i}^2\right] - \Delta t_i\right).$$

Now note that, as $\mathbb{E}\left[W_{t_i} W_{t_{i-1}}\right] = \mathbb{E}\left[(W_{t_i} - W_{t_{i-1}})W_{t_{i-1}} + W_{t_{i-1}}^2\right]$. Splitting this up we have that,

$$\mathbb{E}\left[(W_{t_i} - W_{t_{i-1}})W_{t_{i-1}} + W_{t_{i-1}}^2\right] = \mathbb{E}\left[(W_{t_i} - W_{t_{i-1}})W_{t_{i-1}}\right] + \mathbb{E}\left[W_{t_{i-1}}^2\right]$$

The first term consists of two independent variables, thus,

$$\mathbb{E}\left[(W_{t_i} - W_{t_{i-1}})W_{t_{i-1}}\right] = \mathbb{E}\left[(W_{t_i} - W_{t_{i-1}})\right]\mathbb{E}\left[W_{t_{i-1}}\right] = 0,$$

as the average of the Wiener process is zero. We then notice the second term may be expressed as,

$$\begin{aligned}\mathbb{E}\left[W_{t_{i-1}}^2\right] &= \mathbb{E}\left[W_{t_{i-1}}\right]^2 + \mathbb{E}\left[W_{t_{i-1}}^2\right] - 2\mathbb{E}\left[W_{t_{i-1}}\right]^2 + \mathbb{E}\left[W_{t_{i-1}}\right]^2 \\ &= \mathrm{Var}[W_{t_{i-1}}] + \mathbb{E}\left[W_{t_{i-1}}\right]^2.\end{aligned}$$

The average and variance of the Wiener process are known. Hence, we know that $\mathbb{E}\left[W_{t_{i-1}}^2\right] = t_{i-1}$. Then we have,

$$\mathbb{E}\left[\Delta W_{t_i}^2\right] = \mathbb{E}\left[(W_{t_i} - W_{t_{i-1}})^2\right] = \mathbb{E}\left[W_{t_i}^2\right] + \mathbb{E}\left[W_{t_{i-1}}^2\right] - 2\mathbb{E}\left[W_{t_i} W_{t_{i-1}}\right] = |t_i - t_{i-1}| = \Delta t_i,$$

hence the average of our variable Y is zero. Now note that,

$$\mathrm{Var}[Y] = \mathbb{E}\left[Y^2\right] - \mathbb{E}[Y]^2 = \mathbb{E}\left[Y^2\right].$$

So the second moment is precisely the variance. Considering the second moment we have that,

$$\begin{aligned}\mathbb{E}\left[Y^2\right] &= \lim_{n\to\infty} \mathbb{E}\left[\left(\sum_{i=1}^{n} g(X_{t_{i-1}})(\Delta W_{t_i}^2 - \Delta t)\right)^2\right] \\ &= \lim_{n\to\infty} \sum_{i=1}^{n} \mathbb{E}\left[g(X_{t_{i-1}})^2\right]\mathbb{E}\left[(\Delta W_{t_i} - \Delta t)^2\right] \\ &\quad + 2\sum_{i=1}^{n}\sum_{j<i} \mathbb{E}\left[\Delta W_{t_i}^2 - \Delta t\right]\mathbb{E}\left[g(X_{t_{i-1}})g(X_{t_{j-1}})(\Delta W_{t_j}^2 - \Delta t)\right].\end{aligned}$$

Note that in the second term in the above we have that $\mathbb{E}\left[\Delta W_{t_i}^2 - \Delta t\right]$ is independent of all the other terms, so we may split up the angular brackets. However $\mathbb{E}\left[\Delta W_{t_i}^2 - \Delta t\right] = 0$ hence there is no contribution from the second term. Now consider $\mathbb{E}\left[(\Delta W_{t_i} - \Delta t)^2\right]$,

$$\begin{aligned}\mathbb{E}\left[(\Delta W_{t_i} - \Delta t)^2\right] &= \mathbb{E}\left[W_{t_i}^4\right] - 2\Delta t\mathbb{E}\left[W_{t_i}^2\right] + \Delta t^2 \\ &= 3\Delta t^2 - 2\Delta t^2 + \Delta t^2 = 2\Delta t^2.\end{aligned}$$

Here we used that as ΔW is a Gaussian variable with zero mean this means that $\mathbb{E}\left[W^4\right] = 3\mathbb{E}\left[W^2\right]$, this can be seen by direct integration of the Gaussian distribution and is quoted but not derived here. Hence, we now have,

$$\begin{aligned}\mathbb{E}\left[Y^2\right] &= \lim_{n\to\infty}\sum_{i=1}^{n}\mathbb{E}\left[g(X_{t_{i-1}})^2 2\Delta t^2\right] \\ &= \lim_{n\to\infty} 2\Delta t\sum_{i=1}^{n}\mathbb{E}\left[g(X_{t_{i-1}})^2 2\Delta t\right].\end{aligned}$$

Note that taking the limit $n \to \infty$ is equivalent to taking $\Delta t \to 0$, as $\Delta t = (t - t_0)/n$. Hence, $\mathbb{E}\left[Y^2\right] = 0$. We have shown Y has mean and variance that are zero. The argument also holds for higher moments as $\Delta t \to 0$ and these can be shown to be zero as well. We now have a variable with all moments equal to zero and hence $Y \equiv 0$. This proves our result. □

With this corner stone of stochastic calculus we may now prove Itô's lemma, sometimes referred to as Itô's equation or Itô's differentiation rule. Proving this will then allow us to derive the Black-Scholes formula. We follow the method of Gardiner (1985, P. 95).

Lemma 2.1 (Itôs Lemma)**.** *Consider an Itô process described by,*

$$\mathrm{d}X_t = A(t, X_t)\mathrm{d}t + B(t, X_t)\mathrm{d}W_t, \tag{3}$$

where W_t is the Wiener process. Then, if $g(t, x)$ is a twice-differentiable scalar function of two variables $x, t \in \mathbb{R}$, then,

$$\mathrm{d}g(t, X_t) = \left(\frac{\partial g}{\partial t} + \frac{\partial g}{\partial X}A + \frac{1}{2}\frac{\partial^2 g}{\partial X^2}B^2\right)\mathrm{d}t + \frac{\partial g}{\partial X}B\mathrm{d}W.$$

Proof. We do not give a full rigorous proof here as it is beyond the scope of this project, however we may derive this result using results from Riemann calculus. Consider $g(t, X_t)$, then from Taylor's Theorem we know that an approximation for the derivative of $g$of order $\mathcal{O}(dt)$ is given by,

$$\mathrm{d}g(t, X_t) = \frac{\partial g}{\partial t}\mathrm{d}t + \frac{\partial g}{\partial X}\mathrm{d}X_t + \frac{1}{2}\frac{\partial^2 g}{\partial X^2}\mathrm{d}X_t^2.$$

We include the final term as when we substitute in (3) we will obtain a $\mathrm{d}W^2$ term which we know to be dt. Substituting (3) into the above we obtain,

$$\begin{aligned}\mathrm{d}g(t, X_t) = \frac{\partial g}{\partial t}\mathrm{d}t + \frac{\partial g}{\partial X}(A(t, X_t)\mathrm{d}t + B(t, X_t)\mathrm{d}W_t) \\ + \frac{1}{2}\frac{\partial^2 g}{\partial X^2}(A(t, X_t)\mathrm{d}t + B(t, X_t)\mathrm{d}W_t)^2.\end{aligned}$$

Recall that $\left\langle \Delta W_t^2 \right\rangle = \Delta t$. Now taking limits we see that $\left\langle \mathrm{d}W_t^2 \right\rangle = \mathrm{d}t$, so $\mathrm{d}W_t$ can be thought of as $\mathcal{O}(\mathrm{d}t)$. Expanding and removing terms using this rule of order greater than $\mathrm{d}t$ we have that,

$$\begin{aligned}\mathrm{d}g(t, X_t) &= \frac{\partial g}{\partial t}\mathrm{d}t + \frac{\partial g}{\partial X}(A(t, X_t)\mathrm{d}t + B(t, X_t)\mathrm{d}W_t) + \frac{1}{2}\frac{\partial^2 g}{\partial X^2}\left[A(t, X_t)^2\mathrm{d}t^2\right. \\ &\quad \left. + B(t, X_t)^2\mathrm{d}W_t^2 + 2A(t, X_t)B(t, X_t)\mathrm{d}t\mathrm{d}W_t^2\right] \\ &= \frac{\partial g}{\partial t}\mathrm{d}t + \frac{\partial g}{\partial X}(A(t, X_t)\mathrm{d}t + B(t, X_t)\mathrm{d}W_t) + \frac{1}{2}\frac{\partial^2 g}{\partial X^2}\left[B(t, X_t)^2\mathrm{d}W_t^2\right].\end{aligned}$$

Replacing $\mathrm{d}W_t^2$ with $\mathrm{d}t$ we have,

$$\mathrm{d}g(t, X_t) = \left(\frac{\partial g}{\partial t} + \frac{\partial g}{\partial X}A(t, X_t) + \frac{1}{2}\frac{\partial^2 g}{\partial X^2}B(t, X_t)^2\right)\mathrm{d}t + \frac{\partial g}{\partial X}B(t, X_t)\mathrm{d}W_t,$$

as required. □

2.5 Black-Scholes Model

As discussed before it is incorrect to assume that the stock follows a generalized Wiener process. We then introduced an Itô process to counter this. It remains to be determined what the functions $a(x, t)$ and $b(x, t)$ need to be in our Itô process. The appropriate assumption is that the expected return (the drift over the stock price) is constant. Thus we must have that the expected return is μS for some $\mu \in \mathbb{R}$ (Hull, 2005, Chap. 12).

Furthermore, we assume that the percentage return's variability over a small time interval, Δt, is constant and independent of stock price. This means that a buyer is as uncertain of the return (as a percentage) when the stock costs \$1 as when the stock costs \$1000. This leads to the fact that the the stock price should be proportional to the standard deviation over a small period of time Δt. This leads to the following final model,

$$\mathrm{d}S = \mu S\mathrm{d}t + \sigma S\mathrm{d}W, \tag{4}$$

where the variable σ is the volatility of the stock per year and μ is the expected rate of return on the stock per year. This is the most widely used model for stock behavior. Now using (4) and applying Itô's lemma we obtain, for a function $G(S, t)$ we have the process G follows is given by,

$$\mathrm{d}G = \left(\frac{\partial G}{\partial S} + \frac{\partial G}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 G}{\partial S^2}\right)\mathrm{d}t + \frac{\partial G}{\partial S}\sigma S\mathrm{d}W. \tag{5}$$

We see from (4) that the volatility is a measure of how unsure we are about the path the stock will take. This is because it is multiplying the random component. It can also be viewed as the standard deviation of the lognormal distribution of S_T, as we will see in the next section.

2.6 The Lognormal Property

Consider the equation as described in (4). We may use Itôs lemma to derive the process followed by $\log(S)$ (Hull, 2005, Chap. 13). Let $G = \log(S)$ by applying Itôs lemma, (2.1), with $X_t = S$ and $g(x, t) = \log(S)$ we obtain,

$$\mathrm{d}G = \left(\mu - \frac{\sigma^2}{2}\right)\mathrm{d}t + \sigma\mathrm{d}W.$$

With μ as, the expected rate of return, and σ being the volatility of the stock, these are constant. Thus G follows a generalised Wiener process, with drift rate $\mu - \frac{\sigma^2}{2}$ and variance rate σ. Therefore, by Proposition 2.1, between time 0 and T we see that G has mean $(\mu - \frac{\sigma^2}{2})T$ and variance $\sigma^2 T$. Hence,

$$\log(S_T) - \log(S_0) \sim \phi\left[\left(\mu - \frac{\sigma^2}{2}\right)T, \sigma\sqrt{T}\right]$$
$$\Rightarrow \log(S_T) \sim \phi\left[\log(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)T, \sigma\sqrt{T}\right].$$

Where S_t is the stock price at time t. Then by Definition 1.13 we have that,

$$\mathbb{E}(S_T) = S_0 \mathrm{e}^{\mu T}$$
$$\mathrm{Var}(S_T) = S_0^2 \mathrm{e}^{2\mu T} \left(\mathrm{e}^{\sigma^2 T} - 1 \right).$$

2.7 The Black-Scholes Differential Equation

Using this model for stock prices, we may derive the Blakc-Scholes differential equation (Hull, 2005, Chap. 13). Given that f is an option subject to S, then f must be some function of S and t. Hence, from (5),

$$\mathrm{d}f = \left(\frac{\partial f}{\partial S} + \frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} \right) \mathrm{d}t + \frac{\partial f}{\partial S} \sigma S \mathrm{d}W.$$

The equations (4) and the above have discretized versions,

$$\Delta S = \mu S \Delta t + \sigma S \Delta W. \tag{6}$$
$$\Delta f = \left(\frac{\partial f}{\partial S} + \frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} \right) \Delta t + \frac{\partial f}{\partial S} \sigma S \Delta W, \tag{7}$$

over a time interval Δt. As the Wiener processes contained in Δf and ΔS are the same, it follows that we may construct a portfolio to eliminate this. Such a portfolio should sell an option and buy $\frac{\partial f}{\partial S}$ shares. Then by definition our portfolio, Π, is,

$$\Pi = -f + \frac{\partial f}{\partial S} S. \tag{8}$$

The change in this over Δt is,

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

By substituting in (6) and (7) we obtain,

$$\Delta \Pi = \left(\frac{\partial f}{\partial t} - \frac{1}{2} \frac{\partial^2 f}{\partial S^2} \sigma^2 S^2 \right) \Delta t \tag{9}$$

Over the time period Δt we have eliminated ΔW, so the portfolio must be riskless in this time period and must therefore make the riskfree interest rate. Thus,

$$\Delta \Pi = r \Pi \Delta t$$

substituting (8) and (9) into the above, we yield,

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

so that,

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

The above is known as the Black-Scholes differential equation. It is solvable for some boundary conditions and unsolvable analytically for others. In particular this is solvable for European options, which we investigate in the next section.

2.8 Black-Scholes Formula for European Options

Theorem 2.2 (Black-Scholes Pricing Formula)**.** *The value of a call option, f_c, and a put option, f_p, are given by the following formulae;*

$$f_c = S_0\Phi(d_1) - K\mathrm{e}^{-rT}\Phi(d_2),$$
$$f_p = K\mathrm{e}^{-rT}\Phi(-d_2) - S_0\Phi(-d_1),$$

where

$$d_1 = \frac{\log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \tag{10}$$
$$d_2 = \frac{\log(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma.\sqrt{T}. \tag{11}$$

Before proving this we prove the following claim (Hull, 2005, P. 310-312).

Claim:. If V is lognormally distributed and the standard deviation of $\log(V)$ is ω, then,

$$\mathbb{E}(\max(V - K, 0)) = \mathbb{E}(V)N(d_1) - KN(d_2),$$

where $\mathbb{E}$ denotes the expected value, and we have that,

$$d_1 = \frac{\log\left(\frac{\mathbb{E}(V)}{K}\right) + \frac{\omega^2}{2}}{\omega}$$
$$d_2 = \frac{\log\left(\frac{\mathbb{E}(V)}{K}\right) - \frac{\omega^2}{2}}{\omega}.$$

Proof of claim. Define $h(V)$ to be the probability density function of V. Then we must have that,

$$\mathbb{E}(\max(V - K, 0)) = \int_0^\infty \max(V - K, 0)h(V)\mathrm{d}V \tag{12}$$
$$= \int_K^\infty (V - K)h(V)\mathrm{d}V. \tag{13}$$

By assumption the variable $\log(V)$ is normally distributed with standard deviation ω. Then from Definition 1.13 we have that the mean, m, is given by,

$$m = \log(\mathbb{E}(V)) - \frac{\omega^2}{2}.$$

We further define a new variable, W, by the following,

$$W = \frac{\log(V) - m}{\omega}. \tag{14}$$

This is the transformation that turns the distribution of $\log(V)$ to the standard normal distribution. Let the probability distribution function of W be $g(W)$, so that,

$$g(W) = \frac{1}{\sqrt{2\pi}}\mathrm{e}^{\frac{-W^2}{2}}.$$

Using (14) as a change of variable for (13) we obtain that,

$$\mathbb{E}(\max(V-K,0)) = \int_{\frac{\log(K)-m}{\omega}}^{\infty} \left(\mathrm{e}^{Q\omega+m} - K\right) h(Q)\mathrm{d}Q \tag{15}$$

$$= \int_{\frac{\log(K)-m}{\omega}}^{\infty} \mathrm{e}^{Q\omega+m} h(Q)\mathrm{d}x - K\int_{\frac{\log(K)-m}{\omega}}^{\infty} h(Q)\mathrm{d}Q. \tag{16}$$

To solve this first consider,

$$\begin{aligned}\mathrm{e}^{Q\omega+m}h(Q) &= \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(Q-\omega)^2+2m+\omega^2}{2}\right)\\ &= \mathrm{e}^{m+\frac{\omega^2}{2}}h(Q-\omega),\end{aligned}$$

thus we now have that (16) is,

$$\mathbb{E}(\max(V-K,0)) = \mathrm{e}^{m+\frac{\omega^2}{2}}\int_{\frac{\log(K)-m}{\omega}}^{\infty} h(Q-\omega)\mathrm{d}Q - K\int_{\frac{\log(K)-m}{\omega}}^{\infty} h(Q)\mathrm{d}Q. \tag{17}$$

In (17) the first integral, the integrand is a normal distribution with a shifted mean. Thus it must be a function of cumulative normal distribution as we are summing the area under the probability distribution function, giving that,

$$\begin{aligned}\int_{\frac{\log(K)-m}{\omega}}^{\infty} h(Q-\omega)\mathrm{d}Q &= \left[\Phi(Q-\omega)\right]_{\frac{\log(K)-m}{\omega}}^{\infty}\\ &= 1-\Phi\left(\frac{\log(K)-m}{\omega}-\omega\right)\\ &= \Phi\left(\frac{-\log(K)+m}{\omega}+\omega\right).\end{aligned}$$

Substituting for m we obtain,

$$\Phi\left(\frac{-\log(K)+m}{\omega}+\omega\right) = \Phi\left(\frac{\log\left(\frac{\mathbb{E}(V)}{K}\right)+\frac{\omega^2}{2}}{\omega}\right) = \Phi(d_1).$$

Similarly we obtain that the second integral in (17) is $\Phi(d_2)$. Thus we have that,

$$\mathbb{E}(\max(V-K,0)) = \mathrm{e}^{m+\frac{\omega^2}{2}}\Phi(d_1) - K\Phi(d_2).$$

Substituting for m and we obtain our required result. □

Proof of Black-Scholes Formula. We will prove this for a call option. The proof for a put follows similarly using a similarly proved claim and a similar strategy for this proof.

Consider a call option on a non-dividend paying stock with time of expiry T (initial time $t=0$), strike price K, risk-free interest rate r, current stock price S_0 and volatility σ. The value of such a call at T in a risk-neutral world would be,

$$\mathbb{E}(\max(S_T-K,0)). \tag{18}$$

Note here this is the expected value in a risk neutral world and not necessarily the real world. For the rest of this proof all expected values will be as such.

Then as this is its value at time T, its value, c, would be (18) discounted to the initial time,

$$c = \mathrm{e}^{-rT}\mathbb{E}(\max(S_T - K, 0)).$$

Then as we have shown the stochastic process underlying the stock is log normally distributed. Then at time $t = T$, we have S_T is log normally distributed, and from Section 2.6 $\mathbb{E}(S_T) = S_0\mathrm{e}^{rS_T}$, the standard deviation of $\log(S_T)$ is $\sigma\sqrt{T}$. Using our claim,

$$\begin{aligned} c &= \mathrm{e}^{-rT}\left(S_0\mathrm{e}^{rT}\Phi(d_1) - \Phi(d_2)\right) \\ &= S_0\Phi(d_1) - K\mathrm{e}^{-rT}\Phi(d_2). \end{aligned}$$

Further in this case,

$$\begin{aligned} d_1 &= \frac{\log\left(\frac{\mathbb{E}(S_T)}{K}\right) + \frac{\sigma^2 T}{2}}{\sigma\sqrt{T}} = \frac{\log(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}} \\ d_2 &= \frac{\log\left(\frac{\mathbb{E}(S_T)}{K}\right) - \frac{\sigma^2 T}{2}}{\sigma\sqrt{T}} = \frac{\log(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}}. \end{aligned}$$

□

2.9 Black-Scholes Model for Other Options

In the previous section we found an analytic solution to the Black-Scholes differential equation for European options. Due to the inequality constraints for American and averaging for Asian options, it is not possible to solve the Black-Scholes equation analytically for these in general. There does exist an analytic formula for American options if there is only one dividend to pay known as Roll-Geske-Whaley model and for some types of Asian options however these are specialist and shall not be discussed here.

2.10 Black-Scholes for Dividend Paying Stocks

On the date a dividend is paid the stock declines by the amount of the dividend (Hull, 2005, Chap. 13). Therefore we have that our expected rate of return is now $\mu = r - q$ and the model in (4) still holds. The derivation follows similarly and we yield the following equation,

$$\frac{\partial f}{\partial t} + (r - q)S\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 f}{\partial S^2} = rf. \tag{19}$$

Again this may be solved for European options and the previous statements for Asian and American options are still true, so we may not solve this analytically for these. To solve this notice that the previous probabilistic analysis as seen in Section 2.8 still holds if we take $\mu = r - q$ instead of $\mu = r$. So we see that as all of our formulae still hold, we merely replace r with $r - q$. Thus we have the following theorem,

Theorem 2.3 (Black-Scholes Pricing Formula with Dividends)**.** *The value of a call option, f_c, and a put option, f_p, are given by the following formulae.*

$$f_c = S_0 \mathrm{e}^{-qT}\Phi(d_1) - K\mathrm{e}^{-rT}\Phi(d_2),$$
$$f_p = K\mathrm{e}^{-rT}\Phi(-d_2) - S_0\mathrm{e}^{-qt}\Phi(-d_1),$$

where

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

3 Monte Carlo Simulation

3.1 Monte Carlo Concept

Monte Carlo is the first numerical method we will look at for finding the price of options. The method is incredibly simple and relies on some of the ideas we built in the previous section. We shall develop these ideas following the same ideas as Glasserman (2003, Chap. 1). The concept is as follows;

1. Attempt to predict the path of the price of the underlying stock from $t = 0$ to $t = T$ where T is the expiry date of the option,
2. Evaluate the stock price at $t = T$,
3. Using the previous step, calculate the option price at maturity ($t = T$),
4. Repeat (1) - (3) a statistically significant number of times,
5. Calculate the average option price at maturity,
6. Discount the average option price by the interest rate to obtain the option price at $t = 0$.

For the first step we use the model for stock behavior we developed in (4). We may solve this with an application of Itôs lemma along with the $\mathrm{d}W_t^2 = \mathrm{d}t$ formula. Firstly note that by Itôs lemma we have that,

$$\mathrm{d}(\ln(S_t)) = \frac{1}{S_t}\mathrm{d}S_t - \frac{1}{2S_t^2}\mathrm{d}S_t^2. \tag{20}$$

Then by using (4) with (20) and substituting in for $\mathrm{d}S_t$ we have that,

$$\begin{aligned}\mathrm{d}(\ln(S_t)) &= \frac{1}{S_t}S_t(\mu\mathrm{d}t + \sigma W_t) - \frac{1}{2S_t^2}S_t^2(\sigma^2\mathrm{d}W_t^2)\\ &= \mu\mathrm{d}t + \sigma\mathrm{d}W_t - \frac{1}{2}\sigma^2\mathrm{d}t.\end{aligned}$$

Then through exponentiation we have that,

$$S_t = S_0\left[\exp\left((\mu - \frac{1}{2}\sigma^2)t + \sigma W_t\right)\right], \tag{21}$$

where S_0 is our initial stock price as before. Now note that as the Wiener process is normally distributed with mean zero and standard deviation $\sqrt{T}$ we may rewrite (21) as

$$S_t = S_0 \left[\exp\left((\mu - \frac{1}{2}\sigma^2)t + \sigma\sqrt{T}N \right) \right], \tag{22}$$

where N is a standardized normal random variable. Note here that by taking logarithms of the above we may again see that $\ln(S_t)$ is normally distributed.

Using (22) we may perform our first step by sampling for N to generate our path. This works well for European options however for Asian options as we are considering the average this may not generate a realistic average.

To compensate for this we split our interval into many different time points $0 = t_0 < t_1 < \cdots < t_{n-1} < t_n = T$. We sample at each of these time points to generate a path between $[t_i, t_{i+1}]$ for $i = 0, \ldots, n-1$. This allows us to generate more realistic paths.

The next step in the algorithm that was described above is very simple. However the third step changes and varies depending on what kind of option we are considering so we shall see how we adapt these methods for European, American and Asian options.

3.2 Applications of Monte Carlo

3.2.1 Monte Carlo for European Options

Monte Carlo is not a particularly useful method for European options. In light of the analytic formulas that we have there is no need to use Monte Carlo for European option. Further as Glasserman (2003) said "Monte Carlo is not a competitive method for computing one dimensional integrals" so a poor convergence rate is expected of Monte Carlo.

For European options the method is fairly simple. For the third step in our algorithm we need only use the payoff function to calculate the value of the option at the termination of each path. We then calculate the average price at the terminal date and discount to the initial time $t = 0$. Using the methods we have discussed insofar our algorithm for European options is as follows;

1. Simulate a path for the interval by taking a random sample from N,
2. Calculate the payoff at maturity $t = T$, using the payofff function for European options Payoff$(S_{j,T})$,
3. Repeat (1) - (2) a statically significant number of times to generate a large number, M, of different terminal points indexed by $j = 1, \ldots, M$,
4. Calculate the average payoff, $\overline{S_T}$, at maturity,

$$\overline{S_T} = \frac{1}{M} \sum_{j=1}^{M} \text{Payoff}(S_{j,T}),$$

5. Discount by the risk-free interest rate to the initial time $t = 0$ to find the value of the option V such that

$$V = \mathrm{e}^{-rT}\overline{S_T}.$$

3.2.2 Monte Carlo for American Options

It is possible to adapt Monte Carlo to American options however it is very difficult. The problem arises from the possibility of early exercise, therefore we would need to find an optimal exercise rule(Glasserman, 2003, Chap. 8). As we will develop other methods to price American options we will not consider this here.

3.2.3 Monte Carlo for Asian Options

Monte Carlo is a method we may employ to price Asian options (Glasserman, 2003, Chap. 1). The difficulty with Asian options is in payoff function for these. For Asian options we have the following payoff function; let $\overline{S}$ be the average price of the option between the initial time and terminal time $[0, T]$. Then,

$$\text{Payoff} = \begin{cases} \max(\overline{S} - K, 0) & \text{Call} \\ \max(K - \overline{S}, 0) & \text{Put.} \end{cases} \tag{23}$$

The way that the average is calculated yields several different types of Asian options. We will focus on the discrete case where the average is calculated in the following way. Given a set of discrete dates where the price will be monitored at times $t_1, \ldots, t_n$, and a strike price K, the average of the stock, $\overline{S}$ will be given by,

$$\overline{S} = \frac{1}{n} \sum_{i=1}^{n} S_{t_i}. \tag{24}$$

Note that $\overline{S}$ is dependent upon the value of n we choose. We shall just use the notation $\overline{S}$ to mean $\overline{S}_n$. There are other ways such as a continuous average, priced through the methods developed by Geman and Yor (1993). We will mainly focus on the above.

Assuming the same model for stock movement as in the European case and partitioning the interval $[0, T]$ into n sub intervals as before we have that the discrete version of (22) is given by,

$$S_{t_{i+1}} = S_{t_i} \exp\left((\mu - \frac{1}{2}\sigma^2)(t_i - t_{i-1}) + \sigma\sqrt{t_i - t_{i-1}} N_i \right) \tag{25}$$

where $i = 1, \ldots, n$ and N_i denotes the ith sampling of N.

Using this our algorithm becomes;

1. For $i = 1, \ldots, n$ generate a N_i as our random component,
2. Use (25) to calculate the stock price at the next point where the stock price is monitored,
3. Calculate the average using (24),
4. Calculate the payoff, C_j, using (23),
5. Repeat the first three steps for $j = 1, \ldots, m$ to generate m different paths,
6. Find the average payoff from the option,

$$C = \frac{1}{m} \sum_{j=1}^{m} \text{Payoff}(S_i).$$

7. Discount this average to find the initial price of the option, so that option price is given by $\mathrm{e}^{-rT}C$.

Using this method we may price Asian options (Glasserman, 2003, P. 99). There is another type of option known as the *Geometric average option*. The only difference between the Asian and geometric average options is the method in which the average is calculated. For the geometric option, under all the previous assumptions, the average is given by,

$$\overline{S} = \left(\prod_{i=1}^{n} S(t_i)\right)^{\frac{1}{n}}.$$

3.3 Optimization, Bias and Variance

The natural questions that arise for these methods are that of efficiency. Here we will analyze two different methods of reducing our error. To begin we note that two different factors will cause our answer to be different from the true answer; these are obviously variance and bias. As such we need a "measure" to balance these. The appropriate tool for balancing these factors is the *mean square error* (Glasserman, 2003, P. 16).

Definition 3.1. *Let $\hat{x}$ be an estimator of x, the* mean square error *of $\hat{x}$ is then,*

$$\begin{aligned} \mathrm{MSE}(\hat{x}) &= \mathbb{E}[(\hat{x}-x)^2] && (26)\\ &= \mathbb{E}[(\hat{x}-\mathbb{E}[\hat{x}])^2] + (\mathbb{E}[\hat{x}]-x)^2 && (27)\\ &= \mathrm{Var}(\hat{x}) + \mathrm{Bias}^2(\hat{x}). && (28) \end{aligned}$$

In (27) the right hand term on the right hand side is $\mathrm{Bias}^2(\hat{x})$*, and the variance is given by the leftmost term on the right hand side.*

As Glasserman (2003) said "In applications of Monte Carlo to financial engineering, estimator variance is typically larger than the bias", we will focus on variance reduction techniques here on in to reduce the mean square error.

3.3.1 Control Variate Techniques

Here the idea is to consider a similar option, with similar price, with a closed from formula for the price available (Glasserman, 2003, P. 185-186). Then we simulate the paths for both options using the same set of random variables. We compute the price from our Monte Carlo method and the analytic solution to calculate the error. Then the correct simulation for our option, without a closed-form pricing formula, would be our simulated value minus some weighting times the error involved.

Mathematically speaking, let P_A be the price of an option and P_B be the price of a different option with closed form solution. We simulate with random variables, $N_1, N_2, \ldots, N_n$, to get two prices for the options from simulation say P_A^{sim} and P_B^{sim}. Then we have our error is given by $E_B = P_B - P_B^{\mathrm{sim}}$, where P_B is found from our closed form formula. Then we have that the correct price for P_A under this simulation, P_A^* would be $P_A^* = P_A^{\mathrm{sim}} - bE_B$.

So we seek the optimal b for this. Consider $Y_1, \ldots Y_n$ as n replications of simulation trying to determine $\mathbb{E}[Y_i]$ with each Y_i independent and identically distributed (i.i.d.). Suppose now at each i

we compute some X_i with the same random variable, then from the method as above define Y_i^* and $\overline{Y}^*$ to be,

$$Y_i^* = Y_i - b(X_i - \mathbb{E}[X]),$$
$$\overline{Y}^* = \frac{1}{n}\sum_{i=1}^{n} Y_i - b(X_i - \mathbb{E}[X].$$

Then the control variate estimator above is unbiased as,

$$\mathbb{E}[\overline{Y}^*] = \mathbb{E}[Y_i - b(X_i - \mathbb{E}[X])] = \mathbb{E}[\overline{Y}] = \mathbb{E}[Y].$$

This is consistent as it holds under the limit $n \to \infty$ in the definition of $\overline{Y}$. Now we may calculate the variance of each of the Y_i^*,

$$\mathrm{Var}[Y_i^*] = \mathrm{Var}[Y_i - b(X_i - \mathbb{E}[X])] \tag{29}$$
$$= \sigma_Y^2 - 2b\sigma_X\sigma_Y\rho_{X,Y} + b^2\sigma_X^2, \tag{30}$$

where σ_X^2 is the variance of X and the analogous definition for σ_Y^2. It is easy to see that this technique reduces the variance if $2b\sigma_X\sigma_Y\rho_{X,Y} > b^2\sigma_X^2$. To optimize b we use (30) and upon solving this we see that b is optimized when,

$$b = \frac{\sigma_Y^2}{\sigma_X^2}\rho_{X,Y} = \frac{\mathrm{Covar[X,Y]}}{\mathrm{Var}[X]}.$$

In practice it is not guaranteed that all the values required to calculate b are present. As such we use our simulations to yield the estimate, b^*,

$$b^* = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2}. \tag{31}$$

In the case of Asian options we have a closed form solution for the geometric average case but not the arithmetic average case. As such we may apply the above using Y as our Asian option and X as our geometric option. The close correlation of the price of these two types of assets was shown by Broadie and Glasserman (2005).

To see this formula for geometric average options (Glasserman, 2003, P. 99-100) note that the product of lognormal variables is itself lognormal. Furthermore, the geometric average of lognormal variables is itself lognormal. It follows then from (22) that,

$$\left(\prod_{i=1}^{n} S(t_i)\right)^{1/n} = S_0 \exp\left((r - \frac{1}{\sigma^2})\frac{1}{n}\sum_{i=1}^{n} t_i + \frac{\sigma}{n}\sum_{i=1}^{n} W_{t_i}\right).$$

Following similar logic to Proposition 2.1 we can see that,

$$\sum_{i=1}^{n} W_{t_i} \sim N(0, \sum_{i=1}^{n}(2i-1)t_{n+1-})$$

as each W_{t_i} is independent of the next due to the Markov property.

Thus it follows that at time $t = T$ the geometric average follows a process described by geometric Brownian motion with,

$$\mu = r - \frac{1}{2}\sigma^2 + \frac{1}{2}\hat{\sigma}^2,$$
$$\sigma^2 = \hat{\sigma}^2,$$

where,

$$\hat{\sigma}^2 = \frac{\sigma^2}{n^2 T}\sum_{i=1}^{n}(2i-1)t_{n+1-i}.$$

Thus we may price these options using the Black-Scholes formula with these values of μ and σ. Now we may see how we can apply these for Asian options. We may use a geometric option as our option with a closed form solution to help reduce the variance when finding the price of an Asian option and we have the following algorithm to price the option,

1. Calculate the simulated price of the Asian and geometric option using the usual technique as given in Section 3.1, using the same random variables,
2. Calculate the error in the geometric option by using the closed form solution we have for it,
3. Calculate b^* as given in (31),
4. Calculate the new option price using by subtracting the error multiplied by b^* from the value of the Asian option.

This will become useful and is how we will implement this method when investigating performance later on.

3.3.2 Antithetic Variates

We will here explore another variance reduction technique, antithetic variates (Glasserman, 2003, P. 205-207). In this method for each Y_i generated we create a corresponding $\hat{Y}_i$. Each pair $(Y_i, \hat{Y}_i)$ must be i.i.d.. Define the antithetic estimator as the average of these $2n$ replications, we then have the value of our estimator is of this is given by,

$$\begin{aligned}\overline{Y}^* &= \frac{1}{2n}\left(\sum_{i=1}^{n} Y_i + \sum_{i=1}^{n}\hat{Y}_i\right)\\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_i + \hat{Y}_i}{2}\right).\end{aligned}$$

There are several tricks we may employ here to reduce computing time, one such trick is choosing $\hat{N}_i = -N_i$ for Gaussian variables (note that this is non-zero as we apply the payoff to obtain Y_i). These tricks are not always readily available thus in the worst case scenario to do this method we will require $2n$ replications. Thus we need to compare the new variance after the variance reduction technique to the variance of if we used a normal method with $2n$ replications. It follows that this method is reduces variance if,

$$\mathrm{Var}[\overline{Y}^*] < \mathrm{Var}\left[\frac{1}{2n}\sum_{i=1}^{2n} Y_i\right]$$

thus the above implies that,

$$\text{Var}[Y_i + \hat{Y}_i] < 2\text{Var}[Y_i]. \tag{32}$$

We note that the left hand side can be written as,

$$\begin{aligned} \text{Var}[Y_i + \hat{Y}_i] &= \text{Var}[Y_i] + \text{Var}[\hat{Y}_i] + 2\text{Covar}[Y_i, \hat{Y}_i] &(33)\\ &= 2\text{Var}[Y_i] + 2\text{Covar}[Y_i, \hat{Y}_i], &(34)\end{aligned}$$

as both Y_i and $\hat{Y}_i$ have the same distribution and therefore must have the same variance. So using (32) with (34) we see that the condition for antithetic variance reduction to be effective is,

$$\text{Covar}[Y_i, \hat{Y}_i] < 0,$$

for each path.

Now we may apply this to the field of option pricing. To do this we will use the trick we discussed earlier. The algorithm is as follows,

1. Simulate the price to generate the paths in the usual way as given in Section 3.1 and calculate their payoff,
2. Use the same random components and choose, for the new paths, that $N_i = -N_i$ where N_i is the ith random variable we simulated in step 1 and calculate the pay off of these new paths,
3. For each path generated in step 1 average it with the new path generated in step 2,
4. Take the average of these paths and divide by n to find the approximate value of the option.

This is how the method will be implemented later on and we will explore its comparison to the control variate technique.

4 Tree-based Models

Tree based models are another way we may price options. Here we assume that the underlying asset follows a *random walk*, i.e. in each time step we assume it may move in a number of directions, each with their own probability of occurring. We begin with the simplest model ,the binomial tree, we develop this to attain some pricing formulae for European and American options (Glasserman, 2003, Chap. 11).

4.1 Binomial Trees

The development of this model begins by assuming that there are only two directions in which the asset can go; up or down. This model was first introduced by Cox *et al.* (1979) and has the advantages of being easy to understand and yielding accurate prices, leading to its wide spread use. We will follow much the same derivation and discussion.

4.1.1 One-step Model

This model is best introduced through an example over a single time step. Consider a stock that is currently trading at \$20, assume that over the period of a month we know that the price will be either \$18 or \$24. If we sell a European call option that expires at this date with strike price \$21 then the value of the option will be \$3 if the end price is \$24 or \$0 if it is \$18.

Then we may set up a riskless portfolio of stock. Consider if we sell one option an buy x shares of stock, then the value of the stock will be $24x$ if the stock ends at \$24 and the value of the portfolio will be $24x - 3$. If the stock drops to \$18 then the value of the portfolio is $18x$. We choose x so that the portfolio is riskless, i.e. so that in the event of both outcomes the value of the portfolio is the same. This gives $24x - 3 = 18x \Rightarrow x = 0.5$, and we have that our riskless portfolio is selling one option and 0.5 shares, with a value of $24 * 0.5 - 3 = 9$.

A riskless portfolio must earn the riskless interest rate, so if this is value the of the portfolio at the end of the month, we may discount this and obtain the value of the portfolio now. So assuming the risk free interest rate is 12%,

$$9 * \mathrm{e}^{-\frac{1}{12} \times 0.12} = 8.910$$

It follows that, with the price of the stock today being \$20, that the value of the portfolio today is $20 \times 0.5 - f$ where f is the price of the option, and hence the price of the option is \$11.09. We see that we may use this concept to price options.

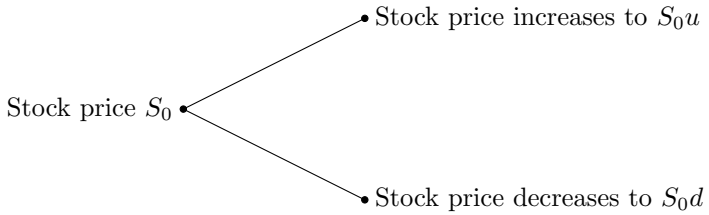


Figure 1: A one step binomial tree

We may generalise this algebraically. Given that the underlying may increase or decrease by values u and d respectively, where $u - 1$ and $1 - d$ are the percentage increase and decrease in the value respectively we assign probabilities to these of q and $1 - q$. Hence, if the current stock price is S_0, at the end of the period the value of the stock will be either S_0u or S_0d with corresponding option payoff value f_u and f_d respectively. This may be seen from Figure 1. We also assume that the interest rate is constant. Then we have that the value of our portfolio is,

$$S_0ux - f_u \quad \text{or} \quad S_0dx - f_d,$$

for up and down movements respectively. Equating and solving for x we obtain that $x = \frac{f_u - f_d}{S_0u - S_0d}$. Furthermore we know that the value of the portfolio now is $(S_0ux - f_u)\mathrm{e}^{-rT}$, where $T/1 = \Delta t$ is our time period and the cost of setting up the portfolio now is $S_0x - f$ with f as the true option price. These must be equal at the initial time so equating and $f = S_0x(1 - u\mathrm{e}^{-rT}) + f_u\mathrm{e}^{-rT}$. Substituting

in the value for our x we obtain,

$$f = S_0\left(\frac{f_u - f_d}{S_0u - S_0d}\right)(1 - u\mathrm{e}^{-rT}) + f_u\mathrm{e}^{-rT} \tag{35}$$

$$= \frac{f_u(1 - d\mathrm{e}^{-rT} + f_d(u\mathrm{e}^{-rT} - 1))}{u - d} \tag{36}$$

$$= \mathrm{e}^{-rT}(pf_u + (1 - p)f_d), \tag{37}$$

where $p = \frac{\mathrm{e}^{rT}-d}{u-d}$.

Here we see that p is our probability q. Consider the expected value of S at time T, denoted $\mathbb{E}[S_T]$ this is given by $\mathbb{E}[S_T] = qS_0u + (1-q)S_0d$. We know that the investment is constructed to be riskless, hence the expected value of the stock at time T would be given by $\mathbb{E}[S_T] = S_0\mathrm{e}^{rT}$. Thus,

$$\Rightarrow S_0\mathrm{e}^{rT} = qS_0u + (1 - q)S_0d \tag{38}$$

$$\Rightarrow \mathrm{e}^{rT} = (u - d)q + d \tag{39}$$

$$\Rightarrow q = \frac{\mathrm{e}^{rT} - d}{u - d}. \tag{40}$$

So we see that p is indeed our probability, referred to and denoted as such herein.

4.1.2 Two-step Model

Here we consider a two step binomial tree this can bee seen in Figure 2. Here our time step is now $\Delta t = \frac{T}{2}$, and thus we now have that equations (37) and (40) are now,

$$f = \mathrm{e}^{-r\Delta t}(pf_u + (1 - p)f_d) \tag{41}$$

$$p = \frac{\mathrm{e}^{r\Delta t} - d}{u - d}. \tag{42}$$

Repeated application of (41) yields,

$$f_u = \mathrm{e}^{-r\Delta t}(pf_{uu} + (1 - p)f_{ud}), \tag{43}$$

$$f_d = \mathrm{e}^{-r\Delta t}(pf_{ud} + (1 - p)f_{dd}). \tag{44}$$

Then substituting equations (41), (43) and (44) we obtain,

$$f = \mathrm{e}^{-r2\Delta t}(p^2f_{uu} + 2p(1 - p)f_{ud} + (1 - p)^2f_{dd}). \tag{45}$$

So we have again calculated the expected value of the tree, discounting to move from our finishing time T to the initial time. Note that this formula only works for European options. This is as the above does not allow for early exercise. We here give an example to give some substance to this idea, this example is biased on one from Glasserman (2003, P. 250).

Example 4.1. Given that the stock price is at \$30 and it may either increase or decrease by 10%, then for two, one month periods our tree can be seen in Figure 3. Here we may combine the two middle nodes as $S_0ud = S_0du$. Consider if we bought a European call option with a strike price of \$31. Then we see at all the nodes the option is out of the money, except the ones with values \$33 and

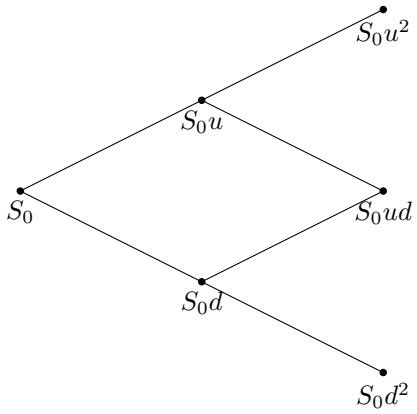


Figure 2: A two step binomial tree

\$36.3, here it has value \$2 and \$5.3 respectively. We seek to discount our option price to the initial node. We can either do this in two stages using equation (37) or much more efficiently using equation (45). Noting that here $u = 1.1, d = 0.9$ and assume that $r = 12\%$ we may solve this using either of these methods which we give below,

1. Using (37) to discount to the node with value \$33 we have that, $p = \frac{e^{-0.12\times\frac{1}{12}}-0.9}{1.1-0.9} = 0.45$. Hence,

$$e^{-\frac{1}{12}\times 0.12}(0.45 \times 5.3 + 0.55 \times 0) = 2.36.$$

   Then applying this again to discount back to the initial node we have, that the option has value \$2.36 at this node and value \$0 at the one below it. Hence,

$$e^{-\frac{1}{12}\times 0.12}(0.45 \times 2.36 + 0.55 \times 0) = 1.053.$$

   Thus we have found the value of our option.

2. Now using the alternate formula given in equation (45) we have,

$$f = e^{-2\times 0.12\times\frac{1}{12}}(0.45^2 \times 5.3 + 2 \times 0.45 \times 0.55 \times 0 + 0.55^2 \times 0) = 1.053.$$

   As before.

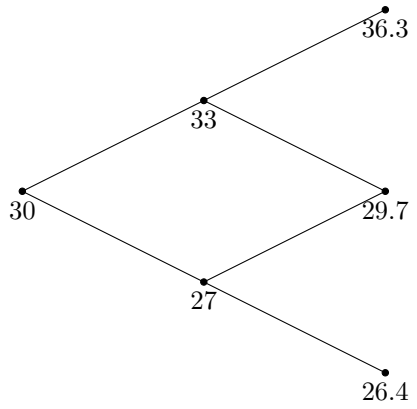


Figure 3: A two step binomial tree for a stock starting at \$30

We see that both of these methods are effective, however the general two step equation is in general easier to use. We now wish to generalise this method for $n \in \mathbb{N}$ time steps.

4.1.3 Generalization

We may further generalize this model to $n \in \mathbb{N}$ time steps, for European options (Hull, 2005, Chap. 11). In Section 4.1.2 we found the formula for a two step tree from a one step tree by repeatedly applying (41). It follows from this that the formula for the value of the option after j upward moves and i downward moves is given by,

$$f_{u^j,d^i} = \mathrm{e}^{-r\Delta t}(pf_{u^{j+1},d^i} + (1-p)f_{u^j,d^{i+1}}).$$

Noting that our formula for the two step model has coefficients given by pascals triangle, and the powers of p and $(1-p)$ follow this as well we make the following claim.

Claim:. The general price of a European option for an $n \in \mathbb{N}$ step tree is given by,

$$f = \mathrm{e}^{-rn\Delta t}\left[\sum_{j=0}^{n}\binom{n}{j}p^j(1-p)^{n-j}f_{u^j,d^{n-j}}\right]. \tag{46}$$

Proof. We prove this by induction. Clearly for $n = 1$ we have,

$$\begin{aligned} f &= \mathrm{e}^{-r\Delta t}\left[\sum_{j=0}^{1}\binom{1}{j}p^j(1-p)^{1-j}f_{u^j,d^{1-j}}\right] \\ &= \mathrm{e}^{-r\Delta t}\left[\binom{1}{0}p^0(1-p)^1 f_{u^0,d^1} + \binom{1}{1}p^1(1-p)^0 f_{u^1,d^0}\right] \\ &= \mathrm{e}^{-r\Delta t}[pf_u + (1-p)f_d]. \end{aligned}$$

Which is exactly what we found in Section 4.1.1 and is equation (37). We can also check this for $n = 2$ against equation (45). We now perform our inductive step. Assume that (46) hold for some $k \in \mathbb{N}$. Then note that,

$$\begin{aligned} f &= \mathrm{e}^{-rk\Delta t}\left[\sum_{j=0}^{k}\binom{n}{j}p^j(1-p)^{k-j}f_{u^j,d^{k-j}}\right] \\ &= \mathrm{e}^{-rk\Delta t}\left[\binom{k}{0}(1-p)^k f_{d^k} + \binom{k}{1}p(1-p)^{k-1}f_{u,d^{k-1}} + \dots\right. \\ &\quad \left. + \binom{k}{k-1}p^{k-1}(1-p)f_{u,d^{k-1}} + \binom{k}{k}p^k f_{u^k}\right]. \end{aligned}$$

Now using the formula from (46) we may expand this giving,

$$\begin{aligned} f = \mathrm{e}^{-rk\Delta t}\Bigg[&\binom{k}{0}(1-p)^k[pf_{u,d^k} + (1-p)f_{d^{k+1}}]\mathrm{e}^{-r\Delta t} \\ &+ \binom{k}{1}p(1-p)^{k-1}[pf_{u^2,d^{k-1}} + (1-p)f_{u,d^k}]\mathrm{e}^{-r\Delta t} + \dots \\ &+ \binom{k}{k}p^k[pf_{u^{k+1}} + (1-p)f_{u^k,d}]\mathrm{e}^{-r\Delta t}\Bigg]. \end{aligned}$$

Here we note that for a term $f_{u^j,d^{k-j+1}}$ for $j = 1, 2, \ldots, k+1$ the only contributing factors from are $f_{u^j,d^{k-j+1}}$ and $f_{u^{j+1},d^{k-j}}$. These contributing factors are $\binom{k}{j-1}p^j(1-p)^{k-j+1}$ and $\binom{k}{j}p^{j-1}p(1-p)^{k-j}(p-1)$ respectively. Hence we have that the coefficient of $f_{u^j,d^{k-j+1}}$ is given by,

$$\begin{aligned}(1-p)^{k-j+1}p^j\binom{k}{j-1} + \binom{k}{j}(1-p)^{k-j+1}p^j &= (1-p)^{k-j+1}p^j\left(\binom{k}{j} + \binom{k}{j-1}\right)\\ &= (1-p)^{k-j+1}p^j\binom{k+1}{j}.\end{aligned}$$

Hence, as this is the coefficient of our next step we have that,

$$\begin{aligned}f &= \mathrm{e}^{-rk\Delta t}\sum_{j=1}^{k+1}\mathrm{e}^{-r\Delta t}\binom{k+1}{j}(1-p)^{k-j+1}p^j f_{u^j,d^{k-j+1}}\\ &= \mathrm{e}^{-r(k+1)\Delta t}\sum_{j=1}^{k+1}\binom{k}{j}(1-p)^{k-j}p^j f_{u^j,d^{k-j}}.\end{aligned}$$

We have shown that if it is true for k then it is true for $k+1$. This completes the proof. □

4.1.4 Finding u and d

We may find these parameters for both European and American options by considering variance (Hull, 2005, Chap. 11). From Section (2.3) and (2.5) we see that for a single step Δt the variance of the discrete version of (4) is $\sigma^2\Delta t$. Furthermore from a one step tree we see that the variance is clearly, $pu^2 + (1-p)d^2 - [pu + (1-d)]^2$. Equating these two we have that,

$$pu^2 + (1-p)d^2 - [pu + (1-d)]^2 = \sigma^2\Delta t.$$

Using the values for p that we found in Section 4.1.1 we have,

$$\mathrm{e}^{r\Delta t}(u+d) - ud - \mathrm{e}^{2r\Delta t} = \sigma^2\Delta t.$$

Using the series expansion for e^x and ignoring terms of higher order than Δt we have that,

$$\begin{aligned}u &= \mathrm{e}^{\sigma\sqrt{\Delta t}},\\ d &= \mathrm{e}^{-\sigma\sqrt{\Delta t}}.\end{aligned}$$

This is the Cox, Ross, and Rubenstien model.

4.1.5 Moment Matching and Other Probabilities

Above we have used a risk neutral argument to choose the parameters u and d (Hull, 2005, Chap. 11). There are many methods to do this. These include the; Cox, Ross and Rubinstein model, the equal probability model, Cox, Ross and Rubinstein with a drift model and the Titan model.

The problem with the Cox, Ross, Rubinstein model is that it does not match the moments of Black-Scholes stochastic differential equation (S.D.E.) (Jarrow and Rudd, 1983). So we seek to find out how we would choose the parameters u and d to match these moments. This is due to the approximation we used in the last section ignoring some terms.

As we have three unknowns to find we need three equations to define these. One idea came from Titan this is that we require that the first three moments of the binomial trees and the Black-Scholes S.D.E. This generates a new model not considered here.

Another model is the Cox, Ross, Rubinstein model with a drift. This introduces a new parameter η so that the stock may drift. This allows us to model stocks more effectively as stocks tend to drift up or down. In this case we have that,

$$u = \mathrm{e}^{\eta\Delta t+\sigma\sqrt{\Delta t}}$$
$$d = \mathrm{e}^{\eta\Delta t-\sigma\sqrt{\Delta t}}$$

with p as before. Alternatively as Jarrow and Rudd (1986) said we may derive these in an alternative way. Consider (4) this implies which,

$$\log\left(\frac{S_T}{S_t}\right) = \mu t + \sigma\sqrt{t}W_t.$$

We may rewrite the above with $W_t = N$ where N is a standardized normal distribution. Further from our discussion on binomial trees for an n step tree we have that, if $u = \mathrm{e}^{u'}$ and $d = \mathrm{e}^{d'}$,

$$S_T = S_t\mathrm{e}^{ju'+(n-j)d'} \tag{47}$$
$$\Rightarrow \log\left(\frac{S_T}{S_t}\right) = nd' + (u'-d')j. \tag{48}$$

for a tree with j upward movements, with probability,

$$\binom{n}{j}p^j(1-p)^{n-j}.$$

It follows that this is binomialy distributed with mean np and variance $np(1-p)$. Then we wish that for large n that these two distributions are equal, causing their moments to be the same. This is a known result in probability Chung (1974). Thus we have that, if p is a constant independent of n, as $n \to \infty$,

$$\frac{j-np}{\sqrt{np(1-p)}} \sim N \tag{49}$$

where $\sim$ denotes convergence in the distribution. Now here we may choose $p = \frac{1}{2}$ as the above guarantees the first two moments are equivalent. We require a third equation to solve for our three unknowns. Then we have that (49) becomes,

$$j \sim \frac{\sqrt{n}}{2}N + \frac{n}{2},$$

and substituting this into (48) we have,

$$\log\left(\frac{S_T}{S_t}\right) \approx nd' + (u'-d')\left(\frac{\sqrt{n}}{2}N + \frac{n}{2}\right).$$

Equating this with (47) and equating coefficient of N and constants we yield,

$$\left.\begin{array}{l}\hat{\mu}t = nd' + (u'-d')\frac{n}{2} \\ \sigma\sqrt{t} = (u'-d')\frac{\sqrt{n}}{2}\end{array}\right\} \Rightarrow \left\{\begin{array}{ll} u' & = \frac{\hat{\mu}t}{n} + \sigma\sqrt{\frac{t}{n}} \\ d' & = \frac{\hat{\mu}t}{n} - \sigma\sqrt{\frac{t}{n}}\end{array}\right.$$

where $\mu = r - \sigma^2/2$ thus we have that,

$$u = \mathrm{e}^{u'} = \mathrm{e}^{(r-\sigma^2/2)t/n+\sigma\sqrt{t/n}}$$
$$d = \mathrm{e}^{d'} = \mathrm{e}^{(r-\sigma^2/2)t/n-\sigma\sqrt{t/n}}$$
$$p = \frac{1}{2}.$$

This is known as the *Equal probability model* or *Jarrow-Rudd* model.

4.2 Trinomial Trees

Trinomial trees have a near identical concept to that of binomial trees (Hull, 2005, Chap. 11). The key difference is that instead of having two paths the stock can take at any given node we now have three. This now assumes that the stock can rise or fall by specified amounts or stay the same in any given time step. This can be seen from Figure 4. The reason that we write S_0m for the middle branch is as for models such as the Boyles model with a drift we may have that $m \neq 1$.

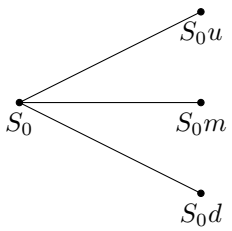


Figure 4: A one step trinomial tree

4.2.1 One-step Model

Following exactly the same logic as before we have that for a one step tree our option price would be given by,

$$f = \mathrm{e}^{-rT}(p_u f_u + p_m f_m + p_d f_d),$$

where p_u is the probability of going up and the analogous definitions for p_m and p_d.

4.2.2 Generalization

Here the generalization is very akin to that of the binomial tree case. Here we may work backwards from the end nodes to obtain the price at the initial node as before. The concepts and algorithms for price calculation are similar to the binomial case with the obvious changes for the middle branch.

Here we do not have the availability of a general formula for the European or American case. Thus we must work backwards through the tree to calculate our price.

4.2.3 Moment Matching and Other Probabilities

Here we now have six unknowns to solve for. Assuming that we match the first two moments as in Jarrow and Rudd (1983) or the alternative derivation of Cox, Ross, and Rubinstein, we still need four more equations. Using that $p_u + p_d + p_m = 1$, $ud = 1$ and $m = 1$ we still require another equation.

The approach proposed by Boyle (1988) is to introduce a stretch parameter, λ, so that $u = \mathrm{e}^{\lambda\sigma\sqrt{t}}$. This can take many values but we shall only consider $\lambda = \sqrt{2}$ due to the equivalence of the trinomial trees method and the explicit finite difference method. Taking this value of λ we obtain that,

$$\begin{aligned} u &= \mathrm{e}^{\lambda\sqrt{2\Delta t}}, \\ d &= \mathrm{e}^{-\lambda\sqrt{2\Delta t}}, \\ m &= 1, \end{aligned}$$

with probabilities,

$$\begin{aligned} p_u &= \left(\frac{\mathrm{e}^{\mu\Delta t/2-} - \mathrm{e}^{-\sigma\sqrt{\Delta t/2}}}{\mathrm{e}^{\sigma\sqrt{\Delta t/2}} - \mathrm{e}^{-\sigma\sqrt{\Delta t/2}}} \right)^2 \\ p_d &= \left(\frac{\mathrm{e}^{\mu\Delta t/2-} - \mathrm{e}^{-\sigma\sqrt{\Delta t/2}}}{\mathrm{e}^{\sigma\sqrt{\Delta t/2}} - \mathrm{e}^{\sigma\sqrt{\Delta t/2}}} \right)^2 \\ p_m &= 1 - p_d - p_m. \end{aligned}$$

There are other methods including equal probabilities or four moment matching trees that we shall not consider here.

5 Finite Difference Methods

We may use finite difference methods to solve the Black-Scholes equation and therefore price options. We first recall a few basic results about Taylor series and finite difference methods. Let $f(x)$ be a function that is twice differentiable, we know using Taylors theorem that,

$$f''(x) = \frac{1}{(\Delta x)^2}(f(x+\Delta x) + f(x-\Delta x) - 2f(x)) + \mathcal{O}((\Delta x)^2), \tag{50}$$

$$f'(x) = \frac{1}{2\Delta x}(f(x+\Delta x) + f(x-\Delta x)) + \mathcal{O}((\Delta x)^2). \tag{51}$$

These are well known approximation to us, for the single variable case. We know that Equation (51) is known as the *central difference approximation*, we are also familiar with the *forward difference approximation* and *backward difference approximation*, which respectively are,

$$\begin{aligned} f'(x) &= \frac{1}{\Delta x}(f(x+\Delta x) - f(x)), \\ f'(x) &= \frac{1}{\Delta x}(f(x) - f(x-\Delta x)). \end{aligned}$$

Now we attempt to extend these methods to functions of two variables, we will follow the same approach as Smith (1965).

5.1 Concept

We begin by extending our notion of how to approximate a single variable function to two variables (Hull, 2005, Chap. 17). Let f be a function of two variables x and y as we may not divide the whole infinite plane we must truncate, choosing some x_{min} and x_{max} and similarly for y. We then subdivide

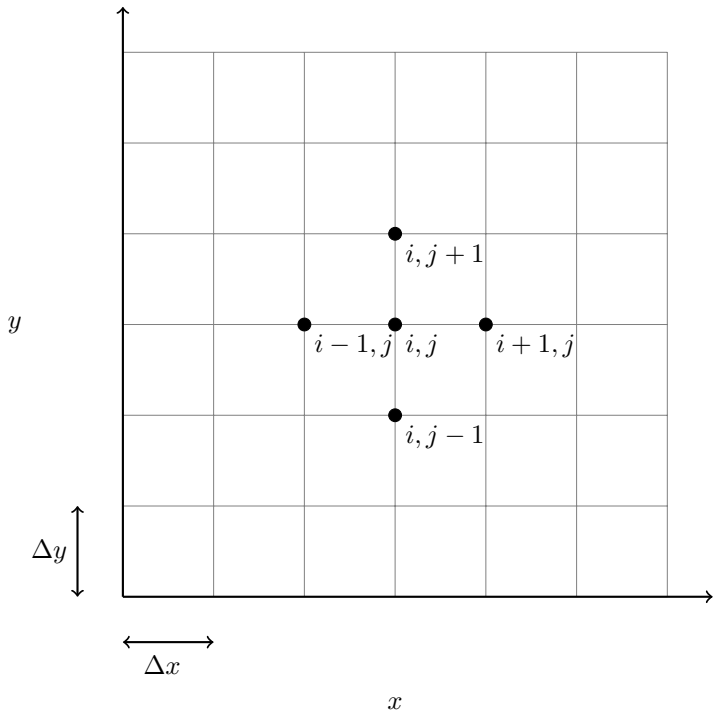


Figure 5: The partitioning of the x, y plane for a two dimensional finite difference method

the truncated x, y plane into rectangles of width $\Delta x = h$ and height $\Delta y = k$, this may be seen in Figure 5. Then we may represent a point (x, y) on the corner of any rectangle as $x = ih$ and $y = jk$, for appropriate $i, j \in \mathbb{N}$. Then we denote the value of f at this point as, $f(x, y) = f(ih, ik) = f_{i,j}$. The by (50) we have that,

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{f((i+1)h, jk) - 2f(ih, jk) + f((i-1)h, jk)}{h^2} \\ &= \frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{h^2},\end{aligned}$$

with leading error of order h^2. Similarly, we may find the second derivative with respect to y and redefine the forward difference approximation and backward difference approximation in terms of our new notation, for two variables.

$$\begin{aligned}\frac{\partial^2 f}{\partial y^2} &= \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{k^2} + \mathcal{O}(k^2), \\ \frac{\partial f}{\partial y} &= \frac{f_{i,j+1} - f_{i,j}}{k} + \mathcal{O}(k), \\ \frac{\partial f}{\partial y} &= \frac{f_{i,j} - f_{i,j-1}}{k} + \mathcal{O}(k).\end{aligned}$$

For options we are attempting to approximate the derivatives of the price of the option with respect to time and the underlyings' price, as well as the second derivative with respect to the underlyings' price. Following the method we described we first need to partition the plane, here the (t, S) plane. We choose divide T into N equally spaced intervals, so $h = \Delta t = \frac{T}{N}$. Similarly we choose to partition S into M equally spaced intervals so $k = \Delta S = \frac{S}{M}$, indexing N by i and M by j. It is important that

M be chosen so that S_0 is one of the points considered ($i = 0$), so we may calculate the value of the option at this point.

5.2 Terminal and Boundary Conditions

We have the framework required to approximate the derivatives, however to solve this we need terminal and boundary conditions (Gilli, 2011, P. 78). As we are required to truncate the plane the actual boundary conditions at ∞ and 0 are of little use to us. We must therefore examine the boundary conditions for the truncated plane. These will differ for both calls and puts, and for European and American options.

Say that we truncate the plane so that $S \in [S_{\min}, S_{\max}]$, we still consider $t \in [0, T]$. Then we have the boundary and terminal conditions for a European option, are given in Table 2, where $V(S, t)$ is the value of the option at stock price S and at time t. These boundary and side conditions are found

Boundary	European Call	European Put
$t = T$	$V(S,T) = \max(S_{\min} + j\Delta S - K, 0)$	$V(S,T) = \max(K - S_{\min} - j\Delta S, 0)$
$S = S_{\min}$	$V(S_{\min}, t) = \max(S_{\min} - K, 0)$	$V(S_{\min}, t) = K\mathrm{e}^{-r(T-t)}$
$S = S_{\max}$	$V(S_{\max}, t) = \max(S_{\max} - K, 0)$	$V(S_{\max}, t) = \max(K - S_{\max}, 0)$

Table 2: The conditions at each boundary of the gird for a European option

by using the payoff function at the necessary points. The points of interest are at the termination date and the sides as this is where we have truncated our plane to, hence we arrive at the above. We may then extend this idea to American options.

The main difference between the American and European case is the lack of need to discount, as we are able to exercise early. This means that our boundary and terminal conditions for an American option are given by Table 3.

Boundary	American Call	American Put
$t = T$	$V(S,T) = \max(S_{\min} + j\Delta S - K, 0)$	$V(S,T) = \max(K - S_{\min} - j\Delta S, 0)$
$S = S_{\min}$	$V(S_{\min}, t) = \max(S_{\min} - K, 0)$	$V(S_{\min}, t) = K$
$S = S_{\max}$	$V(S_{\max}, t) = \max(S_{\max} - K, 0)$	$V(S_{\max}, t) = \max(K - S_{\max}, 0)$

Table 3: The conditions at each boundary of the gird for an American option

5.3 Implicit Method

Hence partitioning the plane as we have discussed, we now have that $h = \Delta S$, $k = \Delta t$ and $x = S$. When approximating the Black-Scholes differential equation we use a backwards difference approximation for the $\frac{\partial f}{\partial S}$ derivative, a forward difference approximation for the $\frac{\partial f}{\partial t}$ and a central difference approximation for the $\frac{\partial^2 f}{\partial S^2}$ derivative (Hull, 2005, Chap, 17). Hence, the Black-Scholes equation becomes,

$$\frac{f_{i+1,j} - f_{i,j}}{\Delta t} + rj\Delta S\frac{f_{i,j+1} - f_{i,j-1}}{2\Delta S} + \frac{1}{2}\sigma^2(j\Delta S)^2\frac{f_{i,j+1} + f_{i,j-1} - 2f_{i,j}}{\Delta S^2} = rf_{i,j}. \tag{52}$$

We may then rearrange this to find coefficients for the $f_{i,j-1}$, $f_{i,j}$ and $f_{i,j+1}$. This yields that (52) is now,

$$a_j f_{i,j-1} + b_j f_{i,j} + c_j f_{i,j+1} = f_{i+1,j} \tag{53}$$

where,

$$\begin{aligned} a_j &= \frac{1}{2}(r-q)j\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t, \\ b_j &= 1 + \sigma^2 j^2 \Delta t + r\Delta t, \\ c_j &= -\frac{1}{2}(r-q)j\Delta t - \frac{1}{2}\sigma^2 j^2 \Delta t. \end{aligned}$$

Note that these constants are independent of S. This is because the ΔS and ΔS^2 introduced by the derivatives are the same ΔS that we are considering the option over in any given triangle. Thus once we discretize the plane we have cancellation resulting in the above relationships.

Now we need to solve this so that we can calculate the value of our option at $t = 0$ i.e. $i = 0$. For a European option we begin by considering $i = N - 1$ in (53), for $j = 1, \ldots, M - 1$. Thus we have $M - 1$ simultaneous equations to solve. If we write out a few of these,

$$\begin{aligned} a_1 f_{N-1,0} + b_1 f_{N-1,1} + c_1 f_{N-1,2} &= f_{N,1} \\ a_2 f_{N-1,1} + b_2 f_{N-1,2} + c_2 f_{N-1,3} &= f_{N,2} \\ &\vdots \\ a_{M-2} f_{N-1,M-3} + b_{M-2} f_{N-1,M-2} + c_{M-2} fN-1, M-1 &= f_{N,M-1} \\ a_{M-1} f_{N-1,M-2} + b_{M-1} f_{N-1,M-1} + c_{M-1} fN-1, M &= f_{N,M}. \end{aligned}$$

Note that all the $f_{i,N}$ are known from the boundary conditions. Furthermore the terms $a_1 f_{N-1,0}$ and $c_{M-1} f_{N-1,M}$ are known from the boundary conditions. We may then express our system with unknowns on the left and knowns on the right.

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & a_{M-2} & b_{M-2} & c_{M-2} \\ & & & & a_{M-1} & b_{M-1} \end{pmatrix} \begin{pmatrix} f_{N-1,1} \\ f_{N-1,2} \\ f_{N-1,3} \\ \vdots \\ f_{N-1,M-2} \\ f_{N-1,M-1} \end{pmatrix} = \begin{pmatrix} f_{N,1} \\ f_{N,2} \\ f_{N,3} \\ \vdots \\ f_{N,M-2} \\ f_{N,M-1} \end{pmatrix} + \begin{pmatrix} -a_1 f_{N-1,0} \\ 0 \\ 0 \\ \vdots \\ 0 \\ -c_{M-1} f_{N-1,M} \end{pmatrix}.$$

Allowing A to be the leftmost matrix and F_i to be the leftmost vector so that F_{i+1} is the second leftmost vector and B_i to be the rightmost vector we have that,

$$AF_i = F_{i+1} + B_i. \tag{54}$$

In the matrix form above we have taken $i = N - 1$ however we note that this is true for arbitrary i.

We may solve these using the tridiagonal matrix algorithm for $N-1$, then the F_{N-1} we have found is used in (54) with $i = N - 2$ to find F_{N-2}. We may continue doing this until we reach $i = 0$ and we may find the value of our option at this point.

For an American option we apply the same method however after each iteration of the method we compare the F_i we found to a vector, P the payoff at each j,

$$P_j = \begin{cases} j\Delta \max(S-K,0) & \text{if call} \\ \max(K-j\Delta S,0) & \text{if put} \end{cases}$$

we compare the each of the $j = 1, \ldots, M-1$ vector and take the maximum. After performing our method once, we obtain

$$F_{i,j} = \max(P_j, F_{i,j})$$

where $F_{i,j}$ is the jth component of F_i.

So we have discussed how to use the implicit method to find the value of an option. There is also the explicit method which has a very different solution method.

5.4 Explicit Method

The main difference between the implicit and explicit methods is that we assume that the values of $\frac{\partial f}{\partial S}$ and $\frac{\partial^2 f}{\partial S^2}$ at a given point (i,j) is the same as at the point $(i+1,j)$ for the explicit method (Hull, 2005, Chap. 17). We then have that,

$$\frac{f_{i+1,j}-f_{i,j}}{\Delta t} + rj\Delta S\frac{f_{i+1,j+1}-f_{i+1,j-1}}{2\Delta S} + \frac{1}{2}\sigma^2(j\Delta S)^2\frac{f_{i+1,j+1}+f_{i+1,j-1}-2f_{i+1,j}}{(\Delta S)^2} = rf_{i,j}.$$

Again we rearrange the above to obtain,

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

where,

$$\begin{aligned} a_j^* &= \frac{1}{1+r\Delta t}\left(-\frac{1}{2}(r-q)j\Delta t + \frac{1}{2}\sigma^2 j^2\Delta t\right), \\ b_j^* &= \frac{1}{1+r\Delta t}\left(1-\sigma^2 j^2\Delta t\right), \\ c_j^* &= \frac{1}{1+r\Delta t}\left(\frac{1}{2}(r-q)j\Delta t + \frac{1}{2}\sigma^2 j^2\Delta t\right). \end{aligned}$$

Note here that this is a much simpler method to implement. To go back a time step it is merely a linear combination of the nodes that came before it. Thus, as the value of $f_{N,j}$ are known, we may start with these and work back to $i = 0$ for the price of our option.

5.5 Change of Variables

We discussed before in Section 2.6 that the Black-Scholes equation is lognomally distributed. This gives us motivation to perform the change of variables, $z = \ln(x)$ (Hull, 2005, Chap. 17).

We begin this by defining $G = \ln(S)$. Then by computing derivatives we see that the Black-Scholes differential equation becomes,

$$\frac{\partial f}{\partial t} + \left(r - q - \frac{\sigma^2}{2}\right)\frac{\partial f}{\partial G} + \frac{1}{2}\sigma^2\frac{\partial^2 f}{\partial G^2} = rf.$$

We now apply the finite difference methods again. Note that we are now considering equally spaced values of G and not S. Then we have that the implicit methods' equation becomes,

$$\frac{f_{i+1,j}-f_{i,j}}{\Delta t}+(r-q-\frac{\sigma^2}{2})\frac{f_{i,j+1}-f_{i,j-1}}{2\Delta G}+\frac{1}{2}\sigma^2\frac{f_{i,j+1}+f_{i,j-1}-2f_{i,j}}{\Delta G^2}=rf_{i,j}.$$

So that now, we have

$$\alpha_j f_{i,j-1}+\beta_j f_{i,j}+\gamma_j f_{i,j+1}=f_{i+1,j},$$

where,

$$\begin{aligned}\alpha_j&=\frac{\Delta t}{2\Delta G}(r-q-\frac{\sigma^2}{2})-\frac{\Delta t}{2\Delta G^2}\sigma^2,\\ \beta_j&=1+\frac{\Delta t}{\Delta G^2}\sigma^2+r\Delta t,\\ \gamma_j&=-\frac{\Delta t}{2\Delta G}(r-q-\frac{\sigma^2}{2})-\frac{\Delta t}{2\Delta G^2}\sigma^2.\end{aligned}$$

Furthermore the explicit methods' equation becomes,

$$\frac{f_{i+1,j}-f_{i,j}}{\Delta t}+(r-q-\frac{\sigma^2}{2})\frac{f_{i+1,j+1}-f_{i+1,j-1}}{2\Delta G}+\frac{1}{2}\sigma^2\frac{f_{i+1,j+1}+f_{i+1,j-1}-2f_{i+1,j}}{\Delta G^2}=rf_{i,j}.$$

Again we now have that,

$$\alpha_j^* f_{i+1,j-1}+b_j^* fi+1,j+c_j^* fi+1,j+1=fi,j,$$

where,

$$\begin{aligned}\alpha_j^*&=\frac{1}{1+r\Delta t}\left(-\frac{\Delta t}{2\Delta G}(r-q-\frac{\sigma^2}{2})+\frac{\Delta t}{2\Delta G^2}\sigma^2\right),\\ \beta_j^*&=\frac{1}{1+r\Delta t}\left(1-\frac{\Delta t}{2\Delta G^2}\sigma^2\right),\\ \gamma_j^*&=\frac{1}{1+r\Delta t}\left(\frac{\Delta t}{2\Delta G}(r-q-\frac{\sigma^2}{2})+\frac{\Delta t}{2\Delta G^2}\sigma^2\right).\end{aligned}$$

Note that this change of variables method has eliminated dependency on j for all our constants. Thus these are constant for each j as well as all i. This means that they need only be calculated once, a helpful trait.

We now need to consider the boundary conditions. If we transform the variable it follows we should transform the boundary conditions. Hence these are now given in Tables 4 and 5.

Boundary	European Call	European Put
$t=T$	$V(S,T)=\max(\mathrm{e}^{S_{\min}}+\mathrm{e}^{j\Delta S}-\mathrm{e}^{K},1)$	$V(S,T)=\max(\mathrm{e}^{K}-\mathrm{e}^{S_{\min}}-\mathrm{e}^{j\Delta S},1)$
$S=S_{\min}$	$V(S_{\min},t)=\max(\mathrm{e}^{S_{\min}}-\mathrm{e}^{K},1)$	$V(S_{\min},t)=\mathrm{e}^{-r(T-t)}\mathrm{e}^{K}$
$S=S_{\max}$	$V(S_{\max},t)=\max(\mathrm{e}^{S_{\max}}-\mathrm{e}^{K},1)$	$V(S_{\max},t)=\max(\mathrm{e}^{K}-\mathrm{e}^{S_{\max}},1)$

Table 4: The conditions at each boundary of the transformed gird for European option

Boundary	American Call	American Put
$t = T$	$V(S,T) = \max(\mathrm{e}^{S_{\min}} + \mathrm{e}^{j\Delta S} - \mathrm{e}^{K}, 1)$	$V(S,T) = \max(\mathrm{e}^{K} - \mathrm{e}^{S_{\min}} - \mathrm{e}^{j\Delta S}, 1)$
$S = S_{\min}$	$V(S_{\min},t) = \max(\mathrm{e}^{S_{\min}} - \mathrm{e}^{K}, 1)$	$V(S_{\min},t) = \mathrm{e}^{K}$
$S = S_{\max}$	$V(S_{\max},t) = \max(\mathrm{e}^{S_{\max}} - \mathrm{e}^{K}, 1)$	$V(S_{\max},t) = \max(\mathrm{e}^{K} - \mathrm{e}^{S_{\max}}, 1)$

Table 5: The conditions at each boundary of the transformed gird for an American option

5.6 Comparison of Explicit Method to Trinomial Trees

The explicit method is equivalent to the trinomial tree approach. This is due to the mechanics of the method. The explicit method as seen in (5.4) gives a relationship between the three values at the next time step $(i+1)\Delta t$ and the value current time step, $i, \Delta t$ (Hull, 2005, P. 425-427). This can be seen in Figure 6. Through this graphic it is easily seen how this can be equivalent to trinomial tree

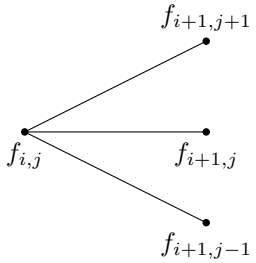


Figure 6: A one step trinomial tree

approaches. We can see that the terms a_j^*, b_j^* and c_j^* may be seen as follows:

$\left(-\frac{1}{2}rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right)$ Probability that the stock price decreases in time Δt from $j\Delta S$ to $(j-1)\Delta S$,

$\left(1 - \sigma^2 j^2 \Delta t\right)$ Probability that the stock price is constant in time Δt remaining at $j\Delta S$,

$\left(\frac{1}{2}rj\Delta t + \frac{1}{2}\sigma^2 j^2 \Delta t\right)$ Probability that the stock price increases in time Δt from $j\Delta S$ to $(j+1)\Delta S$.

One may be inclined to say that this would not be possible, as the value of S_0 increase by exponentiation of either u or d. Thus the partitioning grid would have to be non-linear. We deal with this by performing the change of variables as seen in Section 5.5. Then we may use a linearly increasing grid.

6 Sensitivities of Financial Derivatives and their Numerical Estimation

There are a number of derivatives that are of interest to us. These pertain to how the price of the option, f, changes with respect to the different factors that affect it (Hull, 2005, Chap. 15). These

changes are termed "Greeks" due to the Greek letters that refer to them, the three most common are Delta, Theta and Gamma. These are defined as,

6.1 Delta, Theta and Gamma

Definition 6.1 (Delta, Theta and Gamma)**.** ***Delta****, Δ, is a measure of the rate of change of the options calculated value, f, with respect to the change of the underlying assets price, S. I.e.*

$$\Delta = \frac{\partial f}{\partial S}.$$

Theta*, Θ, is a measure of f with respect to the passage of time, t. This is sometimes referred to as the* ***time decay*** *of the value of the option. I.e.*

$$\Theta = -\frac{\partial f}{\partial t}.$$

Gamma*, Γ, is a measure of the rate of change of Δ with respect to the price of the underlying asset. It could be thought of as the Delta of Delta. I.e.*

$$\Gamma = \frac{\partial \Delta}{\partial S} = \frac{\partial^2 f}{\partial S^2}.$$

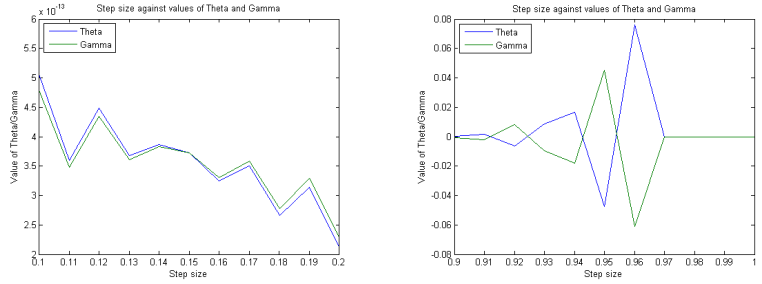
6.1.1 Relationship Between Delta, Theta and Gamma

From the Black-Scholes differential equation we can see a useful relationship between Delta, Theta and Gamma (Hull, 2005, Chap. 359). Delta, Theta, and Gamma are defined as derivatives consider the Black-Scholes differential equation,

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

Note that every derivative is a Greek. Thus by direct substitution we have,

$$\Theta + rS\Delta + \frac{1}{2}\sigma^2 S^2 \Gamma = rf.. \tag{55}$$



(a) The relationship between Theta and Gamma when their values are small

(b) The relationship between Theta and Gamma when their values are large

Figure 7: The relationship between Gamma and Theta

Alternatively if we use f as the value of a portfolio, Π and have the analogous definitions Theta, Delta and Gamma the above still holds for the whole portfolio.

When we constructed the ideas for a binomial tree for one step we stipulated that the portfolio should be riskless. Stating that the up and down values of the portfolio should be equal, this is also known as creating a delta-neutral portfolio. That is we have constructed a portfolio where $\Delta = 0$. If this is the case then (55) yields,

$$\Theta + \frac{1}{2}\sigma^2 S^2 \Gamma = r\Pi.$$

Notice that if Theta or Gamma is large then the contribution from the right hand side becomes negligible. It follows that if Theta is large and positive then Gamma must be large and negative. This relationship for Theta and Gamma can be seen in Figure 7b; furthermore we may see the lack of this correlation in Figure 7a. This will become useful to us as we investigate the performance of these methods in later sections.

6.2 Vega

The other most used Greek is Vega. This is defined as below.

Definition 6.2 (Vega)**.** ***Vega****, ν, is the derivative of the value of the option with respect to its volatility, σ. I.e.*

$$\nu = \frac{\partial f}{\partial \sigma}.$$

It is easy to see why these are not as popular as some of the other Greeks. Knowing the change of option price with respect to the stock price is obviously more likely to be useful than that of the risk-free interest rate.

These Greeks are seldom exactly calculable, as a general formula for f is not often readily available. As such it is often the case that we must estimate these using numerical methods just as we must solve the Black-Scholes equation using numerical methods. The next section details how we may estimate the Greeks through the methods we have discussed insofar.

6.3 Estimation of Greeks

6.3.1 Estimation of Greeks Through Black-Scholes

Due to the existence of the solution of the Black-Scholes differential equation for European options it is simple to differentiate our solution to obtain exact formulae for the calculation of the Greeks (Chriss, 1997, P. 180-181). These can be seen in Table 6. Note here that $\Phi'(x)$ is the derivative of $\Phi(x)$ with respect to x,

For other types of options, such as American and Asian, as the Black-Scholes equation does not admit an analytic solution calculation of the Greeks through analytic formulae is not possible.

6.3.2 Estimation of Greeks Through Monte Carlo

Estimation of Greeks through Monte Carlo must follow a very different tact as we are numerically finding the value of the option and as such we must needs calculate the Greeks in a different way (Glasserman, 2003, Chap. 7).

As we shall, see finite difference methods will serve us well and it is through these, that we will calculate the Greeks for Monte Carlo. The parameters that are required to use Monte Carlo are

Greek	European Call	European Put
Delta	$\mathrm{e}^{-qT}\Phi(d_1)$	$\mathrm{e}^{-qT}(\Phi(d_1)-1)$
Theta	$-\frac{S_0\Phi'(d_1)\sigma\mathrm{e}^{-qT}}{2\sqrt{T}}$	$-\frac{S_0\Phi'(d_1)\sigma\mathrm{e}^{-qT}}{2\sqrt{T}}$
	$-rK\mathrm{e}^{-rT}\Phi(d_2)+qS_0\Phi(d_1)\mathrm{e}^{-qT}$	$+rK\mathrm{e}^{-rT}\Phi(-d_2)-qS_0\Phi(-d_1)\mathrm{e}^{-qT}$
Gamma	$\frac{\Phi'(d_1)\mathrm{e}^{-qT}}{S_0\sigma\sqrt{T}}$	$\frac{\Phi'(d_1)\mathrm{e}^{-qT}}{S_0\sigma\sqrt{T}}$
Vega	$S_0\sqrt{T}\Phi'(d_1)\mathrm{e}^{-qT}$	$S_0\sqrt{T}\Phi'(d_1)\mathrm{e}^{-qT}$

Table 6: The exact formula for the Greeks for European options

σ, r, S_0, T from (3.2.1), along with the number of replications. Then we may calculate the Greek Delta in the following method,

1. Simulate N option prices $C(\sigma, r, S_0, T)$,
2. Simulate a further N option prices slightly incremented $C(\sigma, r, S_0 + \Delta S, T) = C^i$,
3. Calculate the average of these in the usual way $\overline{C}^i$ and $\overline{C}$,
4. Gain a finite difference approximation using these values.

Obviously we may choose to increment σ, S_0 or T to gain our approximations for ν, Δ or Γ, and Θ respectively. For Delta, Theta and Vega we may use a central difference approximation, forward difference or backward difference. Each of these has its own advantages and disadvantages but notice that for Gamma, as in Section 5, we need to use a central difference approximation. So we have that Gamma is given by,

$$\Gamma \approx \frac{\hat{C}(\sigma, r, S_0 + \Delta S, T) - 2\hat{C}(\sigma, r, S_0, T) + \hat{C}(\sigma, r, S_0 - \Delta S, T)}{\Delta S^2}.$$

For the other Greeks depending on the difference approximation used we obtain different formulae. Consider a forward difference approximation of Delta denoted Δ_F, then we have that,

$$\Delta_F = \frac{\overline{C}(S_0 + \Delta S) - \overline{C}(S_0)}{\Delta S},$$

removing the other parameters as we need not consider them. Then it follows that, if $\alpha(S) = \mathbb{E}[\Delta_F]$,

$$\mathbb{E}[\Delta_F] = \Delta S_0^{-1}(\alpha(S_0 + \Delta S) - \alpha).$$

Then if $\alpha(S_0)$ is twice differentiable at S_0, then using a Taylor expansion of order $o(\Delta S^2)$ we have that,

$$\mathrm{Bias}(\Delta_F) = \mathbb{E}[\Delta_F - \alpha'(S_0)] = \frac{1}{2}\alpha''(S_0)h + o(h).$$

Following an similar argument we may show that,

$$\mathrm{Bias}[\Delta_C] = \frac{1}{6}\alpha'''(S_0)\Delta S^2 + o(\Delta S^2).$$

Then it follows that the variance of a forward difference estimator is,

$$\mathrm{Var}[\Delta_F] = \Delta S^2 \mathrm{Var}[\overline{C}(S_0 + \Delta S_0) - \overline{C}(S_0)].$$

Then $\text{Var}[\overline{C}(S_0 + \Delta S_0) - \overline{C}(S_0)] = \frac{1}{n}\text{Var}[C(S_0 + \Delta S_0) - C(S_0)]$ is either $\mathcal{O}(1)$ if we sample using different random numbers. Then,

$$\text{Var}(\Delta_\beta) = \frac{\sigma^2}{n\Delta S^2} + o(\Delta S^{-2})$$
$$\text{Bias}(\Delta_\beta) = b\Delta S^\beta + o(\Delta S^\beta),$$

for some non-zero b and taking $\beta = 1$ for the forward difference case and $\beta = 2$ for the central difference case. Then consider a ΔS of the form $\Delta_n S = \Delta S_* n^{-\gamma}$ for some positive ΔS_* and where Δ_n is the estimated value of Delta from a Monte Carlo simulation using n paths. It can be shown that the optimal γ is $\frac{1}{2\beta+2}$. Then we have our mean square error for a general Delta is given by,

$$\text{MSE}(\Delta) = b^2\Delta S_n^{2\beta} + \frac{\sigma^2}{n\Delta S_n^2}.$$

Taking the square root of this yields the *root mean square error*, RMSE. This is a good measure of convergence, so taking using $\Delta_n S = \Delta S_* n^{-\gamma}$ with our value ofγ we see that,

$$\text{RMSE}(\Delta) = \mathcal{O}\left(n^{-\frac{\beta}{2\beta+2}}\right).$$

So the rates of convergence for the forward and central difference methods are $\mathcal{O}(n^{-1/4})$ and $\mathcal{O}(n^{-1/3})$. Furthermore it can be shown with a little more analysis that the optimal ΔS in for each of these methods is $\Delta S^{-4} \sim n$ for the forward difference case and $\Delta S^{-6} \sim n$ for the central difference case, from the optimal γ and the appropriate values of β.

6.3.3 Estimation of Greeks Through Trees

We may estimate the Greeks from binomial trees using finite difference methods (Hull, 2005, P. 397-398). This allows us to evaluate the Greeks at each node. Consider a tree constructed as in Figure 8. Then at the node denoted $S_0u^id^j$ let $f_{i,j}$ be the payoff of the option at that point. Thus at time step $i + j$, and the stock price be $S_{i,j}$. Then we further denote $\Delta_{i,j}$ to be the value of Delta at the node with the stock value $S_0u^id^j$. Then we can estimate Delta at that point using the difference between the two nodes that emanate from our point. Thus we would have,

$$\Delta_{i,j} = \frac{f_{i+1,j} - f_{i,j+1}}{S_{i+1,j} - S_{i,j+1}}. \tag{56}$$

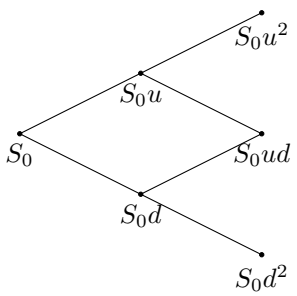


Figure 8: A two step binomial tree with stock prices given at each node

Using this method we may also calculate some of the other Greeks. As Gamma is the second derivative of with respect to stock price, we need to go two steps forward to calculate Delta at a given step before the node. Then we may calculate Gamma at the node we are considering. Thus it follows that,

$$\Gamma_{i,j} = \frac{[(f_{i+2,j} - f_{i+1,j+1})/(S_{i+2,j} - S_{i+1,j+1})] + [(f_{i+1,j+1} - f_{i,j+2})/(S_{i+1,j+1} - S_{i,j+2})]}{\frac{1}{2}(S_{i+2,j} - S_{i,j+2})}.$$

For Theta if $ud = 1$ is fulfilled then we have that the option price is the same at nodes $f_{i,j}$ and $f_{i+1,j+1}$ the only factor that would have affected the price of the option is time. Thus we may use the finite difference approximation and the definition of Theta to see that,

$$\begin{aligned}\Theta_{i,j} &= -\frac{f_{i,j} - f_{i+1,j+1}}{2\Delta t} \\ &= \frac{f_{i+1,j+1} - f_{i,j}}{2\Delta t}\end{aligned}$$

where Δt is the time between steps.

If $ud = 1$ is not fulfilled then Rubinstein (1994) found using the relationship between Δ, Θ and Γ as is (55) we must have,

$$\Theta_{i,j} = rf_{i,j} - (r-q)S_{i,j}\Delta_{i,j} - \frac{1}{2}\sigma^2 S_{i,j}^2 \Gamma_{i,j}. \tag{57}$$

For Vega we must assume a small change in σ. Let $f_{i,j}(\sigma) = S_{i,j}$ with u and d as defined by our model. Then $f_{i,j}(\sigma + 0.01) = S_{i,j}$ would be the same with u and d calculated with a very small change. Hence using another finite difference approximation we may calculate Vega in the following way,

$$\nu_{i,j} = \frac{f_{i,j}(\sigma + 0.01) - f_{i,j}(\sigma)}{0.01}.$$

Here we have used 0.01 as a small change however any small value may be taken.

6.3.4 Estimation of Greeks Through Finite Difference Methods

The idea for approximating the Greeks can be extended from the way they are estimated from trees to finite difference methods in a simple way (Hull, 2005, P. 430). Once we have calculated all of the values then we may employ exactly the same method as from the tree estimation. For a point (i, j) in the partition, the same formulae hold.

Obviously here we have never stipulated that $ud = 1$ need be a condition. As such (57) does not hold. Thus the equations for our Greeks are the same as trees. Though now for us to estimate ν, as u and d are not readily available we must instead calculate the values for a_j, b_j and c_j for a slightly different σ, say $\sigma + 0.01$. Then our formula for ν as before holds.

7 Implementation of Numerical Methods and Investigation of their Performance

In this section we will discuss the implementation of the methods previously described and their performance under certain conditions. To implement these methods MatLab was chosen. This is for

a number of reasons; MatLab has many functions built in that we may use, such as the cumulative normal distribution function. Furthermore unlike some programing languages, such as Visual Basic, it allows us to view arrays as vectors and perform vector operations with them. As we will see later on in this section this will be invaluable for calculating option prices using binomial trees or finite difference methods.

7.1 European Options

7.1.1 Black-Scholes

We have developed an analytic formula for the price of a European option through the solution of the Black-Scholes differential equation. Implementing this is easy as we merely have to calculate d_1 and d_2, test whether the option is a call or put so that we may use the appropriate formula and finally implement the formula using the command "normcdf(d_1)" to calculate $\Phi(d_1)$ or the required variable we need to take the cumulative normal distribution function we are considering. The code for this may be found in Appendix A.1. As this formula is analytic it is helpful for us to compare the performance of our other methods. This is how this will be used herein.

7.1.2 Binomial Trees

Before we discuss the implementation of this method it is useful for us to make the following observations.

To begin note that there are some tricks we may employ to make things easier. Notice that for a binomial tree once we have calculated u and d appropriately regardless of the model or parameters entered the method remains the same. As such we, for European options, we use Appendix C.1.3 as a hub program that calculates the appropriate u and d and then merely calls Appendix C.1.1 which is our method using the general formula. Alternatively we could calculate this by using our back stepping method as in Appendix C.1.2.

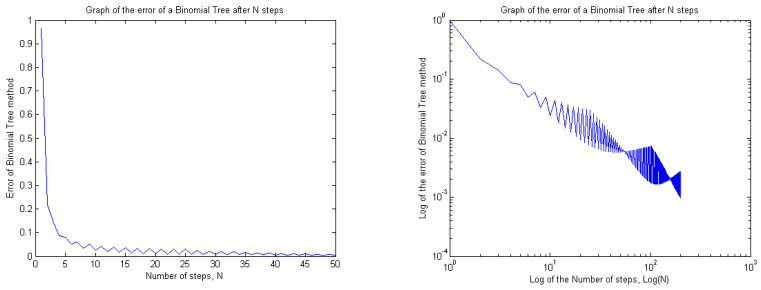
Also note that to calculate Greeks we must use vectors. We generate the vector of stock prices at the terminal time, T, then shift the vectors so that the values that are needed to calculate the previous nodes' value are aligned. Then we may use our formula and loop back through to obtain our initial option price. Then we can use the values to calculate the value of the Greeks at each node.

We have talked about how Binomial Trees converge to the Black-Scholes formula in the case of European options. We may see this in Figure 9a. We see that as the number of steps increase the error decreases quickly.

We now investigate the performance of these methods. In Section 6 we discussed the Greek, which are measures of how the price changes with respect to different factors. It therefore makes sense for us to investigate how the price of the option changes with respect to these. This will allow us to evaluate the robustness of our models.

To begin with we consider Delta. We have already discussed how these are estimated in Section 6.3.3. The best way to evaluate the performance of binomial trees is to choose a $\varepsilon > 0$ then evaluate the number of steps required to find the price of the option to this level of accuracy. We may do this by comparing the value obtained by a Binomial Tree approach to the value found from the Black-Scholes formula. This allows us to see how the value of Delta affects the efficiency of the method.

For us to be able to do this we need to be able to change Delta. Notice that as u increases, Delta increases. As the numerator increases more than the denominator does. So we affect Delta

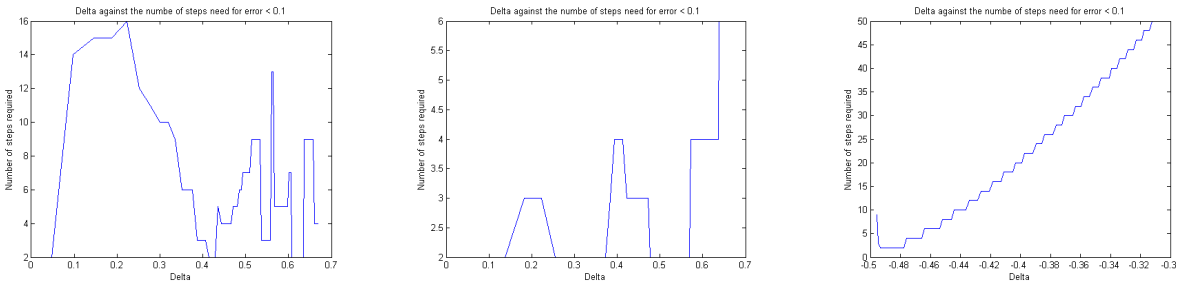


(a) A normal plot of the error when using a binomial tree approximation

(b) A Log-Log plot of the error when using a binomial tree approximation

Figure 9: The error of a binomial tree approximation

by increasing σ then use the method we just developed to see how Delta affects the number of steps needed. Using this method we generate the graphs as seen in Figures 10a-10c.



(a) $r = 0.12$, $K = 48$, $S_0 = 40$ (b) $r = 0.04$, $K = 22$, $S_0 = 20$ (c) $r = 0.02$, $K = 102$, $S_0 = 100$

Figure 10: The number of steps required to achieve an error tolerance as Delta varies

Each of these graphs is generated with different r, S_0 and K. We see that there is no correlation between these and therefore no correlation in general. It is important to understand that we do not affect directly. To cause our change in Delta we use the formula for Delta from Table 6. Notice that from (10) as σ decreases d_1 increases meaning that $\Phi(d_1)$ increases. This causes Delta to increase. So by changing σ is how we affect Delta in Figure 10a - 10c.

We see from these graphs that there is no correlation between the value of Delta and the number of steps it requires to calculate the price accurate to a certain value. This shows that the convergence of the method is unaffected by Delta. Computationally speaking this is beneficial. We are able to price an option with the same amount of steps as an equivalent option with a higher Delta. Note that this does not mean that there is no correlation between the price and Delta. Furthermore we may see that as there is no correlation here it follows that there is no correlation between the convergence and Gamma, as Gamma is the derivative of Delta with respect to Theta.

Finally notice that in Figure 10c all the Deltas are negative. This is because this was done with a put. Notice that from our formula for the estimation of Delta from (56) for a call the numerator is always positive and for a put it is always negative, whilst the denominator is always positive.

7.1.3 Trinomial Trees

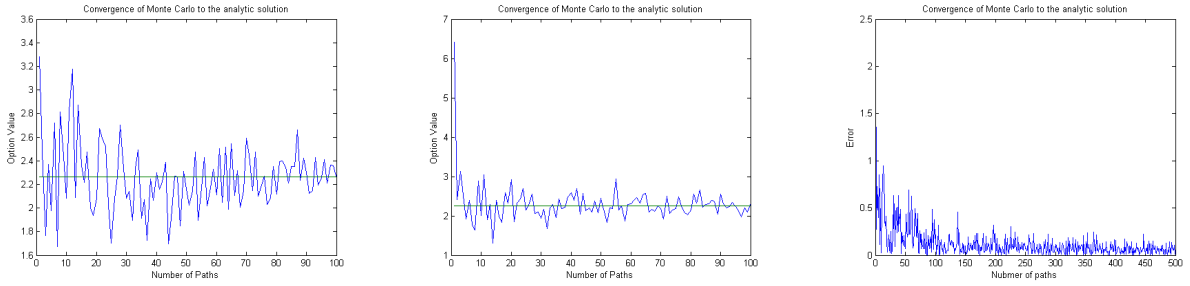
We shall not consider the performance of these, as they are redundant. This is due to their equivalence to the explicit finite difference method, so all conclusions we draw for the explicit finite difference method apply to trinomial trees. However note that they may be implemented in a very similar way to binomial trees, using vectors and looping to backdate to our initial time, to find the value of the option. The code for this method for European options can be seen in Appendix D.1.1. Notice that again once u and d are calculated the algorithm is identical, so we may use a source program. This can be seen in Appendix D.1.2.

Finally notice that the algorithm for a European and American option is similar, the only difference being the need to see if early exercise is optimal after we backdate a step. Thus these algorithms are near identical as can be seen in Appendices D.1.1 and D.2.1. Again, we may use a source program given in Appendix D.2.2.

7.1.4 Monte Carlo

For the Monte Carlo methods in Section 3.1 we have essentially already formed our algorithm. It is merely a case of encoding these using the appropriate functions. This can be seen in Appendix B.1.1.

The first question is of convergence. We have remarked that Monte Carlo converges to the analytic solution given by the Black-Scholes formula. In Figure 11a-11c we see this behavior. Notice that in Figure 11a and 11b the convergence is different. This is due to the random component, as the random numbers were different for these sets of simulations, it generates different graphs.



(a) Convergence to the value given by the analytic solution for European options

(b) Convergence to the value given by Black-Scholes with different random numbers

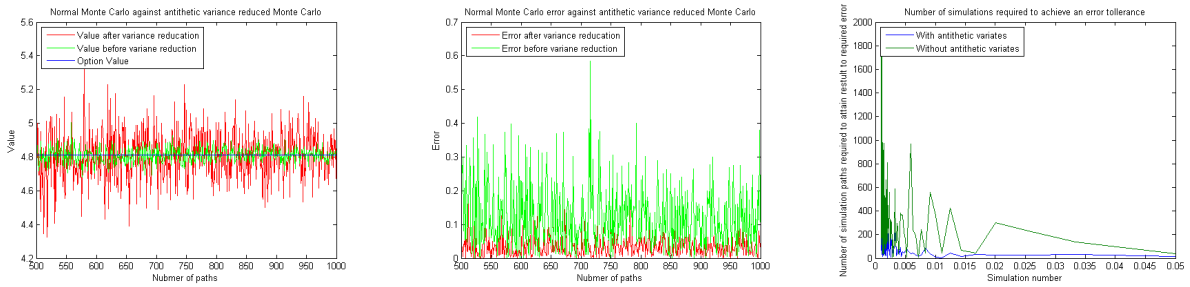
(c) The error between the Black-Scholes value and Monte Carlo estimate

Figure 11: Convergence of Monte Carlo to the Analytic solution for European options

Now consider antithetic variates. When implementing this we use a small trick. As described in Section 3.3.2 we may generate secondary paths by choosing the random numbers to be the negative of the random numbers for the usual set of paths. This saves us a lot computational time and further relaxes the condition that $\text{Cov}[Y_i, \hat{Y}_i] < 0$.

In Figure 12a we see the convergence of Monte Carlo to the option price. Note that it is erratic, due to the random numbers; this is why we sample a large amount of times. Figure 12b shows the error of each of the two methods as the number of replications increases. We see here how much the antithetic variance reduction technique reduces our error, this is even more greatly seen in Figure 12c. Here we see how many replications are required to gain a certain error tolerance. Notice that the normal method requires a thousand or more replications to find the correct price at the lower

error estimates. Helpfully the antithetic variate technique does not require as many paths, in one case requiring one ninth of the paths of the normal method.



(a) $r = 0.04$, $K = 22$ and $S_0 = 20$ (b) $r = 0.12$, $K = 48$ and $S_0 = 40$ (c) $r = 0.04$, $K = 22$ and $S_0 = 20$

Figure 12: Convergence of Monte Carlo methods to the Analytic solution for European options

As stated earlier the requirement for the covariance being less than zero is more relaxed as we do not simulate $2n$ paths for the variance reduced method due to our trick. This is seen as the covariance in the graph above is only slightly smaller than zero with the covariance being negative and of order 10^{-3} in general. This shows how beneficial our method can be.

As we mentioned earlier Monte Carlo is not particularly competitive for European options. This can be seen as the error in Figure 12b once below 0.1 is erratic and in general much higher than in the binomial tree approximation given in Figure 9a. The key difference here is that we have need to simulate $500 - 1000$ paths to achieve this in Monte Carlo, and then only the variance reduced method achieved this error tolerance and not for the normal method, however we achieve better than this with a 50 step binomial tree. This shows us how bad this method is as both of these programs are $\mathcal{O}(n)$ as can be seen in their respective codes, Appendices B.1.1, B.1.2 and C.1.4. The reason is we needed the Greeks when we generated our binomial tree graph hence the use of this code. We now see how this method compares to finite difference methods.

7.1.5 Finite Difference Methods

For the finite difference method we use different approaches for the implicit and explicit methods. For the explicit method, due to the nature of it being equivalent to the trinomial tree approach we may implement using a similar method for the trinomial trees; vector operations. We need only add the appropriate boundary conditions in.

For the implicit method we need another approach. We may use the Thomas Algorithm to help us implement this problem. This has the effect of solving the equations in $\mathcal{O}(MN)$ instead of $\mathcal{O}(M^2N^2)$ required by Gaussian elimination. To see how we implement this consider the Thomas Algorithm. This is a method for solving tridiagonal matrices.

Given the tridiagonal matrix as A in Section 5.3 multiplied by $X = (x_1, x_2, \ldots, x_n)^T$ and a vector

of knowns, $D = (d_1, \ldots, d_n)^T$, we perform the following transformation,

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \ldots, n-1 \end{cases}$$

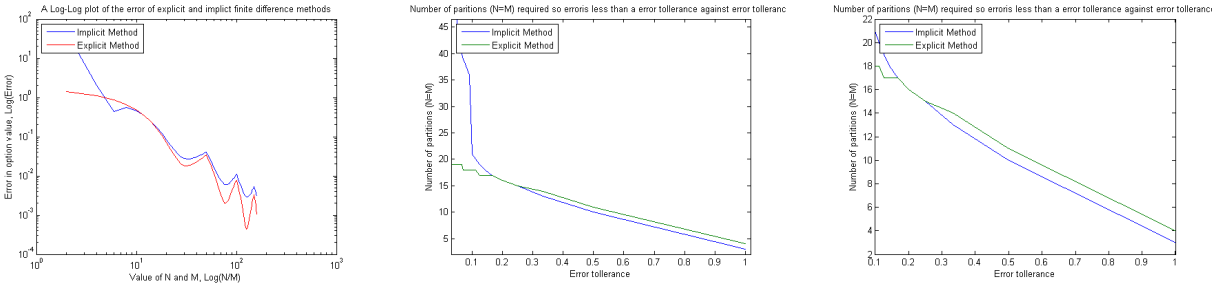
$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \ldots, n. \end{cases}$$

This has the bonus of eliminating the a_i so that $a_i = 0 \; \forall i$ and transforming the b_i so that $b_i = 1 \; \forall i$. Thus we may then solve this system of equations through backwards substitution.

$$\begin{aligned} x_n &= d'_n \\ x_i &= d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \ldots, 1 \end{aligned}$$

We apply this to our system as seen in Section 5.3 by saying $F_i = X$ and $D = F_{i+1} + B_i$. We may now implement this method again using vector operations to reduce computing time. We do this for each iteration step, going backwards until the initial value found at the start is found.

We need to compare the performance of these two methods separately. This is because the explicit method, as seen in Section 5.6, is equivalent to trinomial trees. The implicit method is very different to trinomial tress using the Thomas Algorithm to solve this. This means that due to the large amount of work required for the implicit method, that is not used in the explicit method, we expect this to require more time to achieve the same error tolerance.



(a) Convergence to the value given by the analytic solution for European options

(b) Number of partitions for a given error tolerance with large M and n and with $M = N$

(c) Number of partitions for a given error tolerance with small M and n and with $M = N$

Figure 13: Convergence of finite difference methods to Convergence of Monte Carlo to the Analytic solution for European options

We can see this as if we time each method in Appendices E.1.1 and E.1.2 for say $N = M = 50$ the implicit method requires twice as much time as the explicit method to achieve the same error tolerance level. We also see that in Figures 13b and 13c the number of rectangles we partition into (choosing $M = N$ so that the x-axis is the square root of the number of rectangles used) to gain a given error tolerance is vastly more for lower error tolerances than the explicit method. This is due to the implicit method being $\mathcal{O}(\Delta t, \Delta S^2)$ while the explicit method is $\mathcal{O}(\Delta t^2, \Delta S^2)$.

This shows the behavior we see in Figures 13b and 13c. In the second of these we see for Δt and ΔS larger these have similar convergence, however as $\Delta t \to 0$ the number of rectangles needed increases greatly. As in the graphs we have taken $M = N$ this causes the vast increase as our error tolerance decreases.

Lastly see that these orders of convergence show that error of the explicit method is less than the error of the implicit method for the same given M and N. This can be seen in Figure 13a, showing that in general as we increases the number of rectangles the error of the explicit method has a lower error than the implicit method. It is important to see that all of these result hold for American options which can be seen in the next section.

7.2 American Options

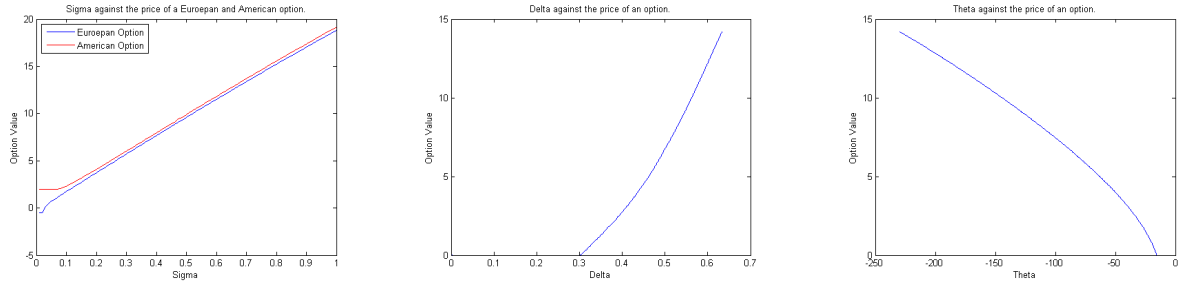
We shall only investigate the performance of the method we have discuses for American options. While some that we have discussed may be adapted to American options, we have not looked at how these may be adapted and as such shall not consider them here. We only considering finite difference methods and binomial trees for which we have developed models for American options.

7.2.1 Binomial Trees

For an American option our implementation is near identical to the vector implementation of European options seen in Appendix C.1.2. As can be seen in AppendixC.2.1 the only minor alteration is having to, after calculating the value at the previous nodes, take the maximum of these and the possible payoff for early exercise. Further again we may use a source program to allow for more user input as seen in Appendix C.2.2. The Greek calculation is again near identical with the only change being taking the maximum of payoff at the current node and the backward calculated value as seen in Appendix C.2.3.

The model is developed in the same way, meaning that all of the conclusions we discussed in Section 7.1.2 still hold. We shall therefore investigate how this method performs when different Greeks change.

As when we explored Delta in Section 7.1.2 we may not change Theta directly. Note that for European options from Table 6 we see that as σ increases Theta decreases. Thus we shall take the same approach changing σ slowly to observe the behavior of the price of the option as Theta changes. We first examine the price of American options against the price of European options.



(a) Option price of equivalent American and European options

(b) Option price compared to the value of Delta

(c) Option Price compared to the value of Theta

Figure 14: American options compared to the Greeks and European options

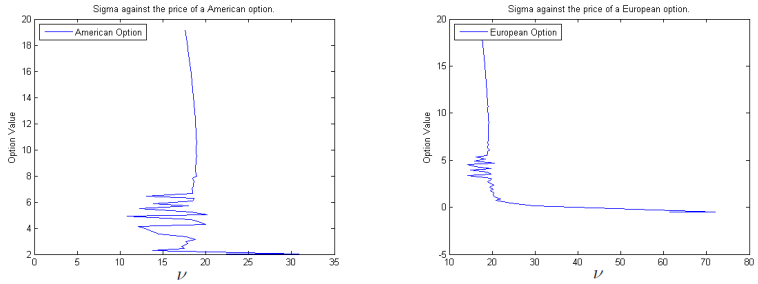
From Figure 14a we see that the price of an American option is always greater than the equivalent European option. The above is obviously only for a given r, S_0, T and K, however this is a trend we

expect to see. The reason for this is that an American option is always more valuable than a European option. This is because the owner of an American option has every chance the owner of a European option has to exercise and more. This fairly obviously makes the option more valuable.

We see that there is a definite correlation between Theta and Delta and the price of the option. In Figure we observe that 14b the higher Delta is the higher the price of the option. This from the same line of reasoning as to why we expect the volatility to increase the value of an option as discussed in Section 2.1. This is because as the rate the stock price changes increases if the stock price goes up we are likely to make a profit. However if it decreases severely our loses are bounded by the price of the option. Hence we want this to be as big as possible. Further note that the gradient of the line is increasing. This means that as Gamma increases the stock price increases as well. The reason for this follows similar logic to that of Gamma.

Furthermore we see in Figure 14c that as Theta increases the price of the option decreases. This is because Theta is a measure of the time decay of the option price. As such we wish for this to be as small as possible; the faster the price decays the less the option is worth.

Again note that from our formula for the approximation of these Greeks we see that the above is for calls. As for puts both Theta and Delta would be negative.



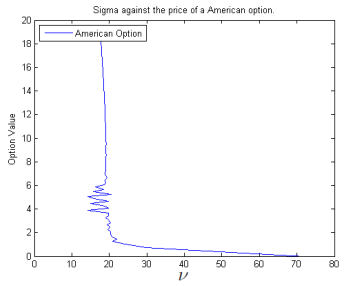
(a) The price of an American option compared to the value of Vega

(b) The price of a European option compared to the value of Vega

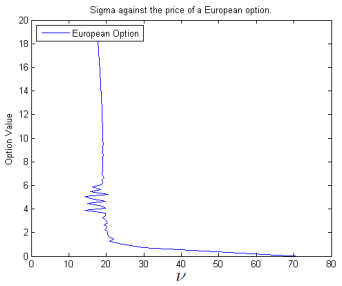
Figure 15: American and European option as Vega varies, if early exercise is optimal

In Figures 15a and 15b we see that there is a general correlation between the European option and Vega. In general we have that as Vega increases the price of the option decreases. This is because if σ is changing rapidly it does not guarantee that it will be larger, which will increase our price.

The lack of correlation seen in Figure 15a is not necessarily a trend as seen in Figures 16a and 16b. Here the relationship between Vega and the option price is identical for European and American options. The lack of correlation seen in Figure 15a is due to early exercise, as it is optimal here, however in Figure 16a it is not leading to the presence of correlation. As when we exercise early σ plays a less significant role, due to the option acting over a shorter time. This is as when we exercise early for a binomial tree we multiply by u and d less, meaning σ has less impact when we exercise early. Thus we expect that Vega would have less impact when we exercise early, resulting in our lack of correlation. Alternatively when we do not exercise early, the relationship is near identical to European options.



(a) American option compared to Vega when early exercise is never optimal



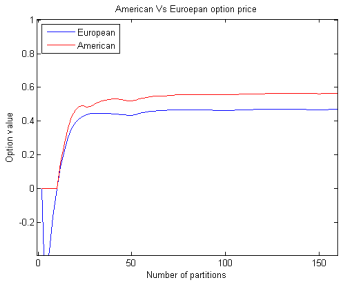
(b) European option compared to Vega where early exercise is no possible

Figure 16: American and European option as Vega varies, if early exercise is never optimal

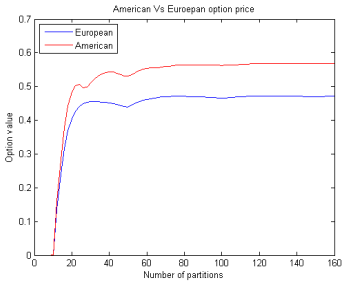
7.2.2 Finite Difference Methods

This is identical in both cases to the European option implementation. Again the only difference between the American and European case is we need to consider early exercise, this can be seen in Appendices E.2.1 and E.2.2. In the case of the implicit method we need compare the values of F_i we have found to the values that would be given by early exercise at this point.

In the case of the explicit method this is essentially the same. Though in the formation of the problem we never defined a F_i it is helpful to do so in implementing this. As explained in the European case it is advantageous to do this using vectors. The method is identical for the American case however we now only do the comparison as above. All of the conclusions we drew from Section 7.1.5 still hold.



(a) The value of an American option compared to a European option approximated using the implicit method



(b) The value of an American option compared to a European option approximated using the explicit method

Figure 17: Price of an American option compared to a European option using different finite difference approximations

We may see in Figure 17a for the implicit method we still have that American options are more valuable than European options. This is again echoed in Figure 17b for the explicit method.

Furthermore all of our analysis for European options hold. Note here that we would observe the same behavior w.r.t. Vega here as before due to early exercise.

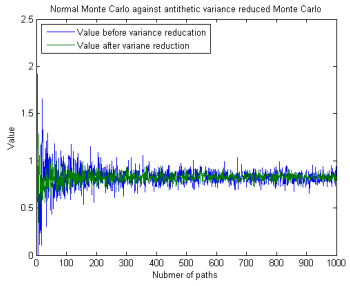
As these options are so similar to European options, we see that there are not many new conclusions we can draw from these. We now consider a very different type of option; Asian options.

7.3 Asian Options

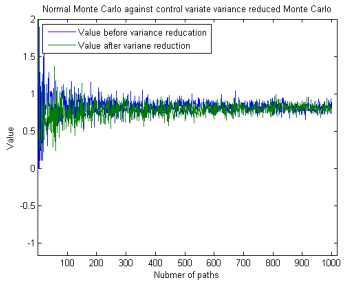
As with American options we only consider here the numerical method that we have applied to Asian options in our development in the theory. Thus we only consider Monte Carlo, so we shall see how the different types of Monte Carlo we developed perform.

7.3.1 Monte Carlo

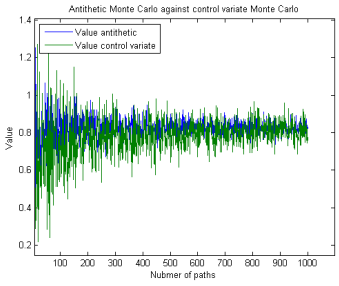
As with the European case the algorithms we are using are known from Section 3.2.3. The implementation the normal, antithetic and control variate techniques can be seen in Appendices B.2.1, B.2.2 and B.2.3 respectively. Thus we need only consider how these perform.



(a) Option price of equivalent American and European options



(b) Option price compared to the value of Delta



(c) Option Price compared to the value of Theta

Figure 18: Different Variance reduction techniques and their affect on a Monte Carlo approximation

We see in Figures 18a and 18b that both of these variance reduction techniques improve on the normal Monte Carlo estimates. So the natural question is that of which is best. If one is not always optimal, under what circumstances is each better in.

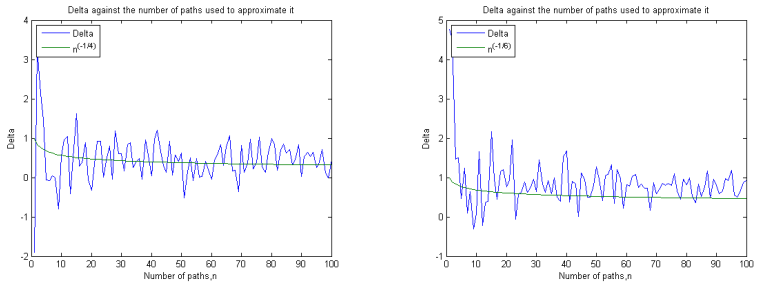
In Figure 18c we see that the antithetic variance reduction techniques is much better in the given case. The way in which the antithetic values technique reduces variance is always the same and always improves the precision. However, consider if the price of an Asian option is very close to that of a geometric average option. In this case the control variate will bring the price much closer much quicker as the error correction is nearly exactly accurate. Thus if this was the case we would expect to see the control variate technique out perform the antithetic values technique.

Now we seek to examine the performance of the estimation of Greeks. In Section 6.3.2 we considered that convergence of two different methods of estimating Delta.

We see in Figure 19a the value of Delta compared to n, the number of paths. We showed in Section 6.3.2 that as the number of paths increases Delta should converge approximately like $n^{-1/4}$. We see this Figure 19a as the estimations of Delta seem to follow roughly the same behavior.

Furthermore we stipulated that for the central difference method the convergence should behave like $n^{-1/6}$. This is demonstrated in Figure 19b where we see the estimation seems to follow the same path roughly. Note that in Figures 19a and 19b are generated with antithetic values. This is as for the option this graph was generated from, the method had reduced variance more showing the correlation more clearly.

Here ends our discussion of numerical and analytic methods in option pricing. Having developed our key ideas such as; the Black-Scholes differential equation, Black-Scholes formula for European options, Mont Carlo methods, binomial trees, finite difference methods and the Greeks and their



(a) Convergence of Delta using a forward difference approximation

(b) Convergence of Delta using a central difference approximation

Figure 19: Convergence of Delta using as estimated from a Monte Carlo approximation

numerical estimation we have implemented these and discussed how they perform under certain circumstances. This has led us to some interesting insights as to which methods are best in given scenarios and how we may improve them. We have further seen the convergence of many methods and convergence of approximations of Greeks with these methods. While our study or investigation has by no means be exhaustive it has given us an interesting insight into the field of option pricing

8 Bibliography

Black, F. and Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy*. 81 (May-June., 1973), p.637-654.

Boyle, P.P. (1988). A Lattice Framework for Option Pricing with Two State Variables. *The Journal of Financial and Quantitative Analysis*. 23 (March 1988), p. 1-12.

Chriss A. N. (1997) *Black-Scholes and Beyond*. 2nd Ed. McGraw-Hill Professional.

Chung, K., (1974) *Elementary probability theory with stochastic processes*. Springer-Verlag.

Cox, J., Ross, S. and Rubinstein, M. (1979) Option Pricing: A Simplified Approach. *Journal of Financial Economics*. 7 (September 1979). p.229-263

Durrett, R. 2010., *Probability.* [online]. Cambridge University Press. Available from:<http://www.myilibrary.com?ID= 281866 > 6 November 2014

Gardiner, C.W., *Handbook of stochastic methods* 2nd Ed. Springer

German H. and Yor, M., Bessel processes, Asian options and perpetuities, *Mathematical Fiance*. 3. (December 1993) p.349-375.

Gilli M.,*Numerical Methods and Optimization in Finance.* 1 Edition. Academic Press.

Glasserman, P., (2003) *Monte Carlo Methods in Financial Engineering.* 3rd Ed. Springer.

Hull J. C. (2005) *Options, Futures, and Other Derivatives*. 6th Ed. Pearson.

Broadie, M., and Glasserman, P. (1997) *Handbook of Risk Management and Analysis*. Wiley, Chichester, England

Jarrow R., and Rudd A., (1983) *Option pricing*. 1st Ed. Richard D. Irwin, Inc.

Jarrow, R., and Rudd, A., (1986). Option Pricing *The Journal of Banking and Finance.* 10 (March 1986), p. 157-161.

Merton, R., Theory of Rational Option Pricing, *Bell Journal of Economics & Management*. 4. (June 1973) p.141-183.

Rubinstein, M.,(1994). Implied Binomial Trees. *Journal of Finance* . 49, pp. 771-818.

Sinclair E. (2010) *Option Trading: Pricing and Volatility Strategies and Techniques*. 1st Ed. John Wiley & Sons.

Smith G. D. (1965) *Numerical solutions of partial differential equations*. 3rd Ed. Oxford university press.

Appendix A Black-Scholes

A.1 Black-Scholes Model MatLab Code

```
function Value=BlackScholesE(s_0, K, r, T, sigma, type,q)

%% This is a function to calculate the value of a European option
% using the closed for solution for the value.

%% Inputs:
% s_0 = intial stock price
% K = strike price
% r = risk free intrest rate
% T = time period the option is over
% sigma = volaility of the stock
% type = either call ('C') or put ('P')
% q = amount of dividends to be paid during options lifetime

%% Calculate d_1 an d_2,
a = log(s_0/K);
b = (r-q + (sigma^2)/2);
c= sigma*sqrt(T);
d_1 = (a + b*T)/(c);
d_2 = d_1 - sigma*sqrt(T);

%% Determine weather the option is call or put

 if  strcmp(type, 'C') == 1;
        %% Calculate N(d_1) and N(d_2)
         n_1 = normcdf(d_1);
         n_2 = normcdf(d_2);
        %calculate the value of the option
         Value = exp(-q*T)*s_0*n_1 - K*exp(-r*T)*n_2;
 else
        %% Calculate N(d_1) and N(d_2)
         n_2 = normcdf(-d_2);
         n_1 = normcdf(-d_1);
        %calculate the value of the option
         Value = K*exp(-r*T)*n_2 - s_0*n_1*exp(-q*T);
 end
```

Appendix B Monte Carlo

B.1 European

B.1.1 Monte Carlo MatLab Code

```
function Value = MonteCarloE(S_0,r,sigma,T,K,N, type)

%% This is a function to use a monte carlo method to approximate the
   ↪ value
% of a European option.

%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% T - the time of the option to exirary
% sigma - the volatility of the stock
% N - number of times we simulate
% type - either "C" for call or "P" for put

%% Generate our random numbers

X = zeros(N,1);
Y= ones(N,1);
Z = normrnd(X,Y);

%% We then forcast to the end of the time period
S_T = zeros(N,1);
for i = 1:N
    S_T(i,1) = S_0*(exp((r - 0.5*sigma^2)*T + sigma*sqrt(T)*Z(i,1)));
end

%% Now we calculate the payoff at the end point

if  strcmp(type, 'C') == 1;
    S_T = max(S_T-K,0);
else
    S_T = max(K-S_T,0);
end

%% Calculate the average
S = 0;
for i = 1:N;
   S = S + S_T(i,1);
```

```
end
Sbar= (1/N)*S;

%% Discount to the inital time and we have our price
Value = exp(-r*T)*Sbar;
```

B.1.2 Monte Carlo Antithetic MatLab Code

```
function Value = MonteCarloEAnithetic(S_0,r,sigma,T,K,N, type)

%% This is a function to use a monte carlo method to approximate the
   ↪ value
% of a European option, with a antithetic variance reduction technique.

%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% T - the time of the option to exirary
% sigma - the volatility of the stock
% N - number of times we simulate
% type - either "C" for call or "P" for put

%% Generate our random numbers

X = zeros(N,1);
Y= ones(N,1);
Z = normrnd(X,Y);
%% We now generate our other random values using the trick
Zhat = -Z;

%% We then forcast to the end of the time period
S_T = zeros(N,1);
S_That = zeros(N,1);
for i = 1:N
    S_T(i,1) = S_0*(exp((r - 0.5*sigma^2)*T + sigma*sqrt(T)*Z(i,1)));
    S_That(i,1) = S_0*(exp((r - 0.5*sigma^2)*T + sigma*sqrt(T)*Zhat(i,1))
       ↪ );
end

%% Now we calculate the payoffs at the end point

if  strcmp(type, 'C') == 1;
    S_T = max(S_T-K,0);
```

```
    S_That = max(S_That-K,0);
else
    S_T = max(K-S_T,0);
    S_That = max(K-S_That,0);
end

%% Calculate the average of each variable and the comination of the two.
S = 0;
Shat = 0;
for i = 1:N;
   S = S + S_T(i,1);
   Shat = Shat + S_That(i,1);
end
Sbar= (1/(2*N))*(S + Shat);
S= S/N;
Shat = Shat/N;

%% Discount to the inital time and we have our price
Value = exp(-r*T)*Sbar;
```

B.2 Asian

B.2.1 Monte Carlo MatLab Code

```
function Value = MonteCarloA(S_0,r,sigma,T,K,N,type,M)

%% This is a function to use a monte carlo method to approximate the
   ↪ value
% of a Asian option.


%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% T - the time of the option to exirary
% sigma - the volatility of the stock
% N - number of times we simulate
% type - either "C" for call or "P" for put
% M- number of points the average is being sampled at


%% Generate random numbers
```

```
X = zeros(N,M);
Y= ones(N,M);
Z = normrnd(X,Y);
dt = T/M;

%% We then forcast to the end of the time period
S_T = zeros(N,M);
for i = 1:N
        S_T(i,1) = S_0*(exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*Z(i,1))
           ↪ );
end
for j = 2:M
    for i = 1:N
        S_T(i,j) = S_T(i,j-1)*(exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*
           ↪ Z(i,j)));
    end
end
% Calculate the average for each path
AvS=zeros(N,1);
for i = 1:N;
    for j = 1:M;
        AvS(i,1) = S_T(i,j) + AvS(i,1);
    end
end
AvS = AvS + S_0;
AvS = AvS./(M+1);
%% Calculate the payoff for each path at the end point

if  strcmp(type, 'C') == 1;
    AvS = max(AvS-K,0);
else
    AvS = max(K-AvS,0);
end

%% Calculate the average payoff
AvSS = 0;
for i = 1:N;
   AvSS = AvSS + AvS(i,1);
end
Sbar= (1/N)*AvSS;
%% Discount to the inital time and we have our price
Value = exp(-r*T)*Sbar;
```

B.2.2 Monte Carlo Antithetic MatLab Code

```
function Value = MonteCarloAAntithetic(S_0,r,sigma,T,K,N,type,M)

%% This is a function to use a monte carlo method to approximate the
    ↪ value
% of a Asian option, with a antithetic variance reduction technique.

%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% T - the time of the option to exirary
% sigma - the volatility of the stock
% N - number of times we simulate
% type - either "C" for call or "P" for put
% M- number of points the average is being sampled at


%% Generate random numbers

X = zeros(N,M);
Y= ones(N,M);
Z = normrnd(X,Y);
dt = T/M;
%% Generate other random numbers using our trick
Zhat = -Z;

%% We then forcast to the end of the time period
S_T = zeros(N,M);
S_That = zeros(N,M);
for i = 1:N
        S_T(i,1) = S_0*(exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*Z(i,1))
            ↪ );
        S_That(i,1) = S_0*(exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*Zhat
            ↪ (i,1)));
end
for j = 2:M
    for i = 1:N
        S_T(i,j) = S_T(i,j-1)*(exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*
            ↪ Z(i,j)));
        S_That(i,j) = S_That(i,j-1)*(exp((r - 0.5*sigma^2)*dt + sigma*
            ↪ sqrt(dt)*Zhat(i,j)));
    end
```

```
end
%% Calculate the average for each path
AvS=zeros(N,1);
AvShat=zeros(N,1);
for i = 1:N;
    for j = 1:M;
        AvS(i,1) = S_T(i,j) + AvS(i,1);
        AvShat(i,1) = S_That(i,j) + AvShat(i,1);
    end
end
AvS = AvS + S_0;
AvShat = AvShat + S_0;
AvS = AvS./(M+1);
AvShat = AvShat./(M+1);
%% Calculate the payoff for each path

if  strcmp(type, 'C') == 1;
    AvS = max(AvS-K,0);
    AvShat = max(AvShat-K,0);
else
    AvS = max(K-AvS,0);
    AvShat = max(K-AvShat,0);
end

%% Calculate the average payoff
AvSS = 0;
AvSShat=0;
for i = 1:N;
   AvSS = AvSS + AvS(i,1);
   AvSShat = AvSShat + AvShat(i,1);
end
Sbar= (1/(2*N))*(AvSS+AvSShat);
%% Discount to the inital time and we have our price
Value = exp(-r*T)*Sbar;
```

B.2.3 Monte Carlo Control Variate MatLab Code

```
function Value = MonteCarloAControlVariate(S_0,r,sigma,T,K,N,type,M)

%% This is a function to use a monte carlo method to approximate the
   ↪ value
% of a Asian option, with a control variate variance reduction technique.
```

```
%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% T - the time of the option to exirary
% sigma - the volatility of the stock
% N - number of times we simulate
% type - either "C" for call or "P" for put
% M- number of points the average is being sampled at
% (not including the start point)

%% Generate random numbers

X = zeros(N,M);
Y= ones(N,M);
Z = normrnd(X,Y);
dt = T/M;

%% We then forcast to the end of the time period
S_T = zeros(N,M);
for i = 1:N
        S_T(i,1) = S_0*(exp((r - 0.5*sigma^2)*dt + sigma*sqrt(dt)*Z(i,1))
           ↪ );
end
for j = 2:M
    for i = 1:N
        S_T(i,j) = S_T(i,j-1)*(exp((r - 0.5*sigma^2)*dt +...
            sigma*sqrt(dt)*Z(i,j)));
    end
end

%% Calculate mu and sigmahat^2
sigmahat2=0;
for i = 1:M
    sigmahat2 = sigmahat2+(2*i-1)*(M+1-i)*dt;
end
sigmahat2 = sigmahat2*((sigma^2)/(M^2*T));

d = 0.5*sigma^2 - 0.5*sigmahat2;
mu = r-d;

%% Now continue to forcast using mu and sigmahat^2.
S_TG = zeros(N,M);
for i = 1:N
```

```
        S_TG(i,1) = S_0*(exp((mu - 0.5*sigmahat2)*dt + sqrt(sigmahat2)...
            *sqrt(dt)*Z(i,1)));
end
for j = 2:M
    for i = 1:N
        S_TG(i,j) = S_TG(i,j-1)*(exp((mu - 0.5*sigmahat2)*dt + ...
            sqrt(sigmahat2)*sqrt(dt)*Z(i,1)));
    end
end

% We now calculate the average for each path
AvS=zeros(N,1);
AvSG=zeros(N,1);
for i = 1:N;
    for j = 1:M;
        AvS(i,1) = S_T(i,j) + AvS(i,1);
        AvSG(i,1) = S_TG(i,j) + AvSG(i,1);
    end
end
AvS = AvS + S_0;
AvS = AvS./(M+1);
AvSG = AvSG + S_0;
AvSG = AvSG./(M+1);
%% Now we calculate the payoff for each path

if  strcmp(type, 'C') == 1;
    AvS = max(AvS-K,0);
    AvSG = max(AvSG-K,0);
else
    AvS = max(K-AvS,0);
    AvSG = max(K-AvSG,0);
end

%% Calculate the average payoff
AvSS = 0;
AvSSG = 0;
for i = 1:N;
   AvSS = AvSS + AvS(i,1);
   AvSSG = AvSSG + AvSG(i,1);
end
SbarG= (1/N)*AvSSG;
Sbar= (1/N)*AvSS;

%% Calculate the value of the geometric average option from the BS
```

```
AnSol=BlackScholesE(S_0, K, mu, T, sqrt(sigmahat2), type);

%% Calculate the error
error =  SbarG - AnSol;

%% Calculate optimal b approimation b*

numb = (AvS-Sbar).*(AvSG-SbarG);
denb = (AvSG-SbarG).^2;
num =0;
den =0;
for i = 1:N;
    num = num + numb(i,1);
    den = den + denb(i,1);
end
bstar = num/den;

%% Calculate our adjusted payoff

P = Sbar - bstar*error;

%% Discount to the inital time and we have our price
Value = exp(-r*T)*P;
```

Appendix C Binomial Trees

C.1 European

C.1.1 Binomial Trees MatLab Code Using General Formula MatLab Code

```
function OptionValue = BinomialTreesCRRE(u,d,K,s_0,dt,r, steps, type)
%% Function to approximate the value of a European option using the
    ↪ general
% formula

%%Inputs:
% u = amount stock goes up by
% d= amount stock goes down by
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
% steps = number of steps that are to be taken
% type = type of option being considered, either C or P for call or put
% First we need to calculate proability, for CRR this is givn by,
```

```
%Find probability
R = exp(r*dt);
p = (R - d)/(u - d);
%%Define end nodes tree size
BTree = zeros( steps+1,1);
%%Generate the end nodes of the tree
for j = 0:steps
        BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
end
%%Define the payofftree
PayOffTree = zeros(size(BTree));
%%Calcultae payoff at end nodes
if  strcmp(type, 'C') == 1;
   for j = 1:steps+1
        PayOffTree(j,1) = max(BTree(j,1)-K,0);
   end
else
   for j = 1:steps+1
        PayOffTree(j,1) = max(K-BTree(j,1),0);
   end
end
Value =0;
%%Use the formula to calculate the vaue of the option.
for j = 0:steps
    V = nchoosek(steps,j)*p^j*(1-p)^((steps)-j);
    Value = Value + V*PayOffTree(steps-j+1,1);
end
Value = exp(-r*dt*steps)*Value;
OptionValue = Value;
```

C.1.2 Binomial Trees MatLab Code Using Vectors MatLab Code

```
function Value = BinomialTreesEVectors(u,d,K,s_0,dt,r, steps, type)
%% Function calculates the value of a European option using the backward
% stepping method

%% Inputs:
% u = amount stock goes up by
% d= amount stock goes down by
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
```

```
% steps = number of steps that are to be taken
% type = type of option being considered, either C or P for call or put
% First we need to calculate proability, for CRR this is givn by,
% Currently only works for puts not calls

%% Calculate probability
 R = exp(r*dt);
 p = (R - d)/(u - d);
  BTree = zeros( steps+1,1);
%% Generate Binomial tree at end step.
 for j = 0:steps
        BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
 end
 EPayOffTree = zeros(size(BTree));
 %% Find the Payoff for the option at the end point
 if  strcmp(type, 'C') == 1;
    for j = 0:steps
        EPayOffTree(j+1,1) = max(BTree(j+1,1)-K,0);
    end
 else
    for j = 1:steps+1
        EPayOffTree(j,1) = max(K-BTree(j,1),0);
    end
 end
 %% Generate the binomial tree as a vector at step (steps-1)
 for i = 0:steps-1
        BTree(i+1,1) = s_0*(u^((steps-1)-i))*d^(i);
 end
    APayOffTree = zeros(size(BTree));
    %% At step (steps-1) the last entry of the vector is now zero we
    % set this.
    BTree(steps+1, 1) = 0;
    %% Shifts the payoff tree up one to allow vector operations
    EPayOffTree2 = circshift(EPayOffTree,[-1 1]);
    %% Calculates the value of a option at previous step
    DiscountedPayOffTree = exp(-r*dt)*(p*EPayOffTree + (1-p)*EPayOffTree2
       ↪ );
    %% As we recive a contribution from the penultimate term of the
       ↪ vector
    % to give an extra term. Set this to zero.
    DiscountedPayOffTree(steps+1, 1) = 0;
    APayOffTree = DiscountedPayOffTree;
for j = 2:steps
    BTree= zeros(size(APayOffTree));
```

```
    %% Calulate the binomial tree at step (steps-j)
    for i = 0:steps-j
        BTree(i+1,1) = s_0*(u^((steps-j)-i))*d^(i);
    end
    %% Shifts the payoff tree up one to allow vector operations
    APayOffTree2 = circshift(APayOffTree,[-1 1]);
    % Calculate the value of the option at each node at step steps-j
    DiscountedPayOffTree = exp(-r*dt)*(p*APayOffTree + (1-p)*APayOffTree2
        ↪ );
    %% As we recive a contribution from the penultimate term of the
        ↪ vector
    % to give an extra term. Set this to zero.
    DiscountedPayOffTree(steps+2-j, 1) = 0;
    % Calculate the payoff tree of an american option
    APayOffTree = DiscountedPayOffTree;
    APayOffTree(steps+2-j:steps+1, 1) = 0;
end
Value = APayOffTree(1,1);
```

C.1.3 Source Program Binomial Trees MatLab Code

```
function Value = BinomialTrees(k, s_0, dt, steps, type, r, vars, o1,o2)

%% Function to approximate the price of a Euroepan option using Binomial
% trees

%% Inputs:
% K - strike price
% s_0 - stock price
% dt - steps size
% steps - number of steps
% type - 'C' for call 'P' for put
% r - risk free intrest rate
% vars- describes optional parameter o2 and implicitly defines o2. This
    ↪ has
% a few defined values vars=1 \Rightarrow o1=u o2=d. vars=2 \Rightarrow
% o1=sigma o2 need not be specified. vars=3 \Rightarrow o1=sigma, o2 =eta
    ↪ .

%% Test how many arguments there are to define inputs
if nargin==9
    %% if o1 and o2 are given we need to dscern between the vars=1 and
    % vars =3 cases.
    if vars == 1
```

```
        %% Call the oproriate function to find these value
        Value = BinomialTreesCRRE(o1,o2,k,s_0,dt,r, steps, type);
    elseif vars == 3
        %% Calculate u and d first then call the opropriate function.
        u = exp(o1*sqrt(dt) + o2*dt);
        d = exp(-o1*sqrt(dt) + o2*dt);
        Value = BinomialTreesCRRE(u,d,k,s_0,dt,r, steps, type);
    else
        disp('The number of variables given disagrees with the case given
           ↪ in vars')
    end
elseif nargin == 8
    if vars == 2
        %% Calculate u and d first then call the opropriate function.
        u = exp(o1*sqrt(dt));
        d = exp(-o1*sqrt(dt));
        Value = BinomialTreesCRRE(u,d,k,s_0,dt,r, steps, type);
    else
        disp('The number of variables given disagrees with the case given
           ↪ in vars')
        Value = -1;
    end
else
    disp('The number of variables given are not sifficent to solve the
       ↪ probelm')
    Value = -1;
end
```

C.1.4 Numerical Estimation of Greeks from a Binomial Tree MatLab Code

```
function Value=BinomialTreesCRREGreeks(u,d,K,s_0,dt,r, steps, type, greek
   ↪ )

%% Approximates the Greeks for a European option from a Binomial Tree.

%% Inputs:
% u = amount stock increases by
% d= amount stock decreases by
% m = amount stock increases/decrease if it stays the same
%(Drift modles this will be not always equal to one)
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
% steps = number of steps that are to be taken
```

```
% type = type of option being considered, either C or P for call or put
% greek = the greek we wish to calculate (1-Delta, 2-Gamma,3-Theta).

%% Find probability
 R = exp(r*dt);
 p = (R - d)/(u - d);
%% Define end nodes tree size
 BTree = zeros( steps+1,1);
%% Generate the end nodes of the tree
 for j = 0:steps
         BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
 end

%% Calculate the payoff from the end nodes
PayOffTree = zeros(size(BTree));
 if  strcmp(type, 'C') == 1;
         PayOffTree = max(BTree-K,0);
 else
         PayOffTree = max(K-BTree,0);
 end


 %% We quickly define a matrix here that will store our values our Greeks

 Deltas = zeros(steps, steps);
 Theta = zeros(steps,steps);
 Gamma = zeros(steps-1,steps-1);

 %% Now we must backdate to the previous nodes.
 % Define 2 new vectors to allow vector operations to be performed.
 for i = 0:steps-1
 ShiftTree1 = circshift(PayOffTree,[-1 1]);

 %% In ShiftTree1 the nth entry is nonzero and will give us an
 % incorect vector awnser.

ShiftTree1(steps+1-i,1) = 0;

%% Now we backdate to the previous step

PayOffTree1 = exp(-r*dt)*(p*PayOffTree + (1-p)*ShiftTree1);
PayOffTree2 = PayOffTree;
ShiftTree3 = circshift(PayOffTree,[-1 1]);
ShiftTree3(steps-i+1,1) = 0;
```

```
%% We now have all we require to calculate delta at this step.
% We need our stock prices at this step.
BTree = 0;
for j = 0:steps-i
        BTree(j+1,1) = s_0*(u^((steps-i)-j))*d^(j);
end
%% Define two vectors to calculate Greeks
ShiftTree2 = circshift(BTree,[-1 1]);
ShiftTree2(steps-i+1,1) = 0;
%% Now we calculate delta at our steps and store it.
A = (PayOffTree2- ShiftTree3);
B= BTree - ShiftTree2;
D= A./B;
        D = D(1:steps-i,1);
        A = A(1:steps-i,1);
Deltas(1:steps-i,steps-i) = D;

%% Calculate the other greeks. Theta is the given by,
Theta(1:steps-i,steps-i) = A./(2*dt);
%% Test to ensure we are not at last step where Gamma has no value.
if i ~= steps-1
    %% Use Delta and calculate approximate derivative.
    D_1 = circshift(D,[-1 1]);
    D_1 = D_1(1:steps-i-1,1);
    D = D(1:steps-i-1,1);
    %% Now we may calculate the difference.
    DeltaDifference = D-D_1;

%% This relies on the prices a node backwards we need to calculate these.
    BTree = 0;
    for j = 0:steps-i-1
            BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
    end
    ShiftTree2 = circshift(BTree,[-1 1]);
    ShiftTree2(steps-i) = 0;
    ShiftTree2 = ShiftTree2(1:steps-i-1,1);
    BTree= BTree(1:steps-i-1,1);
    B= BTree - ShiftTree2;
    Gamma(1:steps-i-1,steps-i-1) = DeltaDifference./B;

end
%% We use then do this the apropriate number of times to lead us back to
%our first step.
```

```
PayOffTree = PayOffTree1(1:steps-i,1);
 end

if greek == 1;
    Value = Deltas;
elseif greek ==2;
    Value = Gamma;
else
    Value = Theta;
end
```

C.2 American

C.2.1 Binomial Trees MatLab Code MatLab Code

```
function Value = BinomialTreesACrr(u,d,K,s_0,dt,r, steps, type)

%% Approxmates the value of an American option using a Binomail Tree
% approximation

%% Inputs:
% u = amount stock goes up by
% d= amount stock goes down by
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
% steps = number of steps that are to be taken
% type = type of option being considered, either C or P for call or put
% First we need to calculate proability, for CRR this is givn by,
% Currently only works for puts not calls

%% Calulate probability
 R = exp(r*dt);
 p = (R - d)/(u - d);
  BTree = zeros( steps+1,1);
%%Generate Binomial tree as a vector for the final step.
 for j = 0:steps
        BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
 end
 EPayOffTree = zeros(size(BTree));
 %% Find the Payoff function for the option at the end point
 if  strcmp(type, 'C') == 1;
    for j = 0:steps
        EPayOffTree(j+1,1) = max(BTree(j+1,1)-K,0);
```

```
    end
 else
    for j = 1:steps+1
        EPayOffTree(j,1) = max(K-BTree(j,1),0);
    end
 end
%% Generate the binomial tree as a vector at step (steps-1)
 for i = 0:steps-1
        BTree(i+1,1) = s_0*(u^((steps-1)-i))*d^(i);
 end
    APayOffTree = zeros(size(BTree));

   %% At step (steps-1) the last entry of the vector is now zero we
   % set this.
    BTree(steps+1, 1) = 0;
   %% Shifts the payoff tree up one to allow vector operations
    EPayOffTree2 = circshift(EPayOffTree,[-1 1]);
   %% Calculates the value of a option at previous step
    DiscountedPayOffTree = exp(-r*dt)*(p*EPayOffTree + (1-p)*EPayOffTree2
       ↪ );
   %% As we recive a contribution from the penultimate term of the
       ↪ vector
   % to give an extra term. Set this to zero.
    DiscountedPayOffTree(steps+2-1, 1) = 0;
   %% Calulate the payoff function for a american option
    if  strcmp(type, 'C') == 1;
        APayOffTree = max(DiscountedPayOffTree, BTree-K );
    else
        APayOffTree = max(DiscountedPayOffTree, K - BTree);
    end
   %% As the final value of the vector is zero this will return a payoff
       ↪  of
   % K at the final entry for a put. Hence we set this to zero.
    APayOffTree(steps+2-1, 1) = 0;
for j = 2:steps
    BTree= zeros(size(APayOffTree));
   %% Calulate the binomial tree at step steps-j
    for i = 0:steps-j
        BTree(i+1,1) = s_0*(u^((steps-j)-i))*d^(i);
    end
   %% Shifts the payoff tree up one to allow vector operations
    APayOffTree2 = circshift(APayOffTree,[-1 1]);
   %% Calculates the value of a option at previous step
```

```
    DiscountedPayOffTree = exp(-r*dt)*(p*APayOffTree + (1-p)*APayOffTree2
        ↪ );
     %% As we recive a contribution from the penultimate term of the
        ↪ vector
    % to give an extra term. Set this to zero.
    DiscountedPayOffTree(steps+2-j, 1) = 0;
    %% Calculate the payoff tree of an american option
    if  strcmp(type, 'C') == 1;
        APayOffTree = max(DiscountedPayOffTree, BTree-K );
    else
        APayOffTree = max(DiscountedPayOffTree, K - BTree);
    end
    % Again as for a put we would recive a payoff of K at the zero terms
        ↪ we
    % set all of these to zero.
    APayOffTree(steps+2-j:steps+1, 1) = 0;
end
Value = APayOffTree(1,1);
```

C.2.2 Source Program Binomial Trees MatLab Code

```
function Value = BinomialTreesA(k, s_0, dt, steps, type, r, vars, o1,o2)
%% Fucntion to approximate the price of a European option using a
    ↪ Binomail
% tree method.


%% Inputs:
% K - strike price
% s_0 - stock price
% dt - steps size
% steps - number of steps
% type - 'C' for call 'P' for put
% r - risk free intrest rate
% vars- describes optional parameter o2 and implicitly defines o2. This
    ↪ has
% a few defined values vars=1 \Rightarrow o1=u o2=d. vars=2 \Rightarrow
% o1=sigma o2 need not be specified. vars=3 \Rightarrow o1=sigma, o2 =eta
    ↪ .

%% Test how many arguments there are to define inputs
if nargin==9
    %% if o1 and o2 are given we need to dscern between the vars=1 and
    % vars =3 cases.
```

```
    if vars == 1
        %% Call the oproriate function to find these value and greeks.
       Value = BinomialTreesACrr(o1,o2,k,s_0,dt,r, steps, type);
    elseif vars == 3
        %% Calculate u and d first then call the opropriate function.
        u = exp(o1*sqrt(dt) + o2*dt);
        d = exp(-o1*sqrt(dt) + o2*dt);
        Value = BinomialTreesACrr(u,d,k,s_0,dt,r, steps, type);
    else
        disp('The number of variables given disagrees with the case given
           ↪ in vars')
    end
elseif nargin == 8
    if vars == 2
        %% Calculate u and d first then call the opropriate function.
        u = exp(o1*sqrt(dt));
        d = exp(-o1*sqrt(dt));
        Value = BinomialTreesACrr(u,d,k,s_0,dt,r, steps, type);
    else
        disp('The number of variables given disagrees with the case given
           ↪ in vars')
    end
else
    disp('The number of variables given are not sifficent to solve the
       ↪ probelm')
end
```

C.2.3 Numerical Estimation of Greeks from a Binomial Tree MatLab Code

```
function Value = BinomialTreesAGreeks(u,d,K,s_0,dt,r, steps, type, greek)
% Function calculates inital price of a American option, either put or
% call, using CRR model No dirft.

% u = amount stock goes up by
% d= amount stock goes down by
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
% steps = number of steps that are to be taken
% type = type of option being considered, either C or P for call or put
% First we need to calculate proability, for CRR this is givn by,
% Currently only works for puts not calls
% greek = the greek we wish to calculate (1-Delta, 2-Gamma,3-Theta).
```

```
%% Calculate probability
 R = exp(r*dt);
 p = (R - d)/(u - d);
  BTree = zeros( steps+1,1);
%% Generate Binomial tree as a vector for the final step.
 for j = 0:steps
        BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
 end
 EPayOffTree = zeros(size(BTree));
%% Finds the Payoff function for the option at the end point
 if  strcmp(type, 'C') == 1;
    for j = 0:steps
        EPayOffTree(j+1,1) = max(BTree(j+1,1)-K,0);
    end
 else
    for j = 1:steps+1
        EPayOffTree(j,1) = max(K-BTree(j,1),0);
    end
 end
 %% Generates the binomial tree as a vector at step (steps-1)
 for i = 0:steps-1
        BTree(i+1,1) = s_0*(u^((steps-1)-i))*d^(i);
 end
    APayOffTree = zeros(size(BTree));
    %% At step steps-1 the last entry of the vector is now zero we
    % set this.
    BTree(steps+2-1, 1) = 0;
    %% Shifts the payoff tree up one to allow vector operations
    EPayOffTree2 = circshift(EPayOffTree,[-1 1]);
    %% Calculates the value of a european option at each of the nodes at
    % step steps-1
    DiscountedPayOffTree = exp(-r*dt)*(p*EPayOffTree + (1-p)*EPayOffTree2
       ↪ );
    %% As we recive a contribution from the penultimate term of the
       ↪ vector
    % to give an extra term we set this to zero.
    DiscountedPayOffTree(steps+2-1, 1) = 0;
    %% Calulate the payoff function for a american option
    if  strcmp(type, 'C') == 1;
        APayOffTree = max(DiscountedPayOffTree, BTree-K );
    else
        APayOffTree = max(DiscountedPayOffTree, K - BTree);
    end
```

```
    %% As the final value of the vector is zero this will return a payoff
       ↪ of
    % K at the final entry for a put. Hence we set this to zero.
    APayOffTree(steps+2-1, 1) = 0;

    %% Define matricies to store vetores

    Deltas = zeros(steps, steps);
    Theta = zeros(steps,steps);
    Gamma = zeros(steps-1,steps-1);

    %% Now we may calculate them.
    PayOffTree2 = APayOffTree;
    %% Shift vector to allow vecotr operations
    ShiftTree3 = circshift(APayOffTree,[-1 1]);
    ShiftTree3(steps+1,1) = 0;

    ShiftTree2 = circshift(BTree,[-1 1]);
    ShiftTree2(steps+1,1) = 0;
    %% Calculate Delta
    A = (PayOffTree2- ShiftTree3);
    B= BTree - ShiftTree2;
    D= A./B;
    D = D(1:steps,1);
    A = A(1:steps,1);
    %% Calculate Theta
    Deltas(1:steps,steps) = D;
    Theta(1:steps,steps) = A./(2*dt);

    %% Gamma is Delta of Delta so we need to change the vectors as before
       ↪ .
    D_1 = circshift(D,[-1 1]);
    D_1 = D_1(1:steps-1,1);
    D = D(1:steps-1,1);
    %% Now we may calculate the difference.
    DeltaDifference = D-D_1;

%% As this relies on the prices a node backwards we need to calculate
   ↪ these.
    BTree = 0;
    for j = 0:steps-1
            BTree(j+1,1) = s_0*(u^((steps)-j))*d^(j);
    end
    ShiftTree2 = circshift(BTree,[-1 1]);
```

```
    ShiftTree2(steps) = 0;
    ShiftTree2 = ShiftTree2(1:steps-1,1);
    BTree= BTree(1:steps-1,1);
    B= BTree - ShiftTree2;
    Gamma(1:steps-1,steps-1) = DeltaDifference./B;
for j = 2:steps
    %% Calulate the binomial tree at step steps-j
    BTree = 0;
    for i = 0:steps-j+1
         BTree(i+1,1) = s_0*(u^((steps-j)-i))*d^(i);
    end
    %% Shifts the payoff tree up one to allow vector operations
    APayOffTree2 = circshift(APayOffTree,[-1 1]);
    % Calculates the value of The option at each node at step steps-j
    DiscountedPayOffTree = exp(-r*dt)*(p*APayOffTree + (1-p)*APayOffTree2
       ↪ );
    %% As we recive a contribution from the penultimate term of the
       ↪ vector
    % to give an extra term we set this to zero.
    DiscountedPayOffTree(steps+3-j, 1) = 0;
    DiscountedPayOffTree = DiscountedPayOffTree(1:steps-j+2,1);
    %% Calculate the payoff of an american option
    if  strcmp(type, 'C') == 1;
        APayOffTree = max(DiscountedPayOffTree, BTree-K );
    else
        APayOffTree = max(DiscountedPayOffTree, K - BTree);
    end
    %% Again as for a put we would recive a payoff of K at the zero terms
       ↪  we
    % set all of these to zero.

    PayOffTree2 = APayOffTree;
ShiftTree3 = circshift(APayOffTree,[-1 1]);
ShiftTree3(steps-j+2,1) = 0;
%% We construct 2 vectors so we may peform vector operations to find
   ↪ Delta
ShiftTree2 = circshift(BTree,[-1 1]);
ShiftTree2(steps-j+2,1) = 0;
%% Now we calculate delta at our steps and store it in a vector.
A = (PayOffTree2- ShiftTree3);
B= BTree - ShiftTree2;
D= A./B;
D = D(1:steps-j+1,1);
A = A(1:steps-j+1,1);
```

```
Deltas(1:steps-j+1,steps-j+1) = D;

%% Calculate Theta
Theta(1:steps-j+1,steps-j+1) = A./(2*dt);
%% Gamma is Delta of Delta so we need to change the vectors as before.
    D_1 = circshift(D,[-1 1]);
    D_1 = D_1(1:steps-j,1);
    D = D(1:steps-j,1);
    %% Now we may calculate the difference.
    DeltaDifference = D-D_1;

%% As this relies on the prices a node backwards we need to calculate
   ↪ these.
if j~= steps
    BTree = 0;
    for k = 0:steps-j
            BTree(k+1,1) = s_0*(u^((steps)-k))*d^(k);
    end
    ShiftTree2 = circshift(BTree,[-1 1]);
    ShiftTree2(steps-j+1) = 0;
    ShiftTree2 = ShiftTree2(1:steps-j,1);
    BTree= BTree(1:steps-j,1);
    B= BTree - ShiftTree2;
    Gamma(1:steps-j,steps-j) = DeltaDifference./B;

end
end
if greek == 1;
    Value = Deltas;
elseif greek ==2;
    Value = Gamma;
else
    Value = Theta;
end
```

Appendix D Trinomial Trees

D.1 European

D.1.1 Trinomial Trees MatLab Code (Boyle)

```
function Value=TrinomialTreesEBoyle(u,d,m,K,s_0,dt,r, steps, type)
```

```
%% Approximates the value of a European option using Boyles values for u&
    ↪ d

%% Inputs
% u = amount stock increases by
% d=  amount stock decreases by
% m = amount stock increases/decrease if it stays the same
%(Drift modles this will be not always equal to one)
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
% steps = number of steps that are to be taken
% type = type of option being considered, either C or P for call or put



%% We must first calculate the probabilities of each branch of the tree.

R = exp(r*dt/2);
p_u = ((R - d^(1/2))/(u^(1/2)-d^(1/2)))^2;
p_d = ((u^(1/2) - R)/(u^(1/2)-d^(1/2)))^2;
p_m = 1-p_u-p_d;

%% Generate our tree up to the nth step.
n = 2*steps + 1;
BTree = zeros(n,1);
for j = 0:steps-1
    BTree(j+1,1) = s_0*u^(steps-j)*m^(j);
    BTree(n-j,1) = s_0*d^(steps-j)*m^(j);
end
    BTree(((n-1)/2)+1,1) = s_0*m^steps;

%% Calculate the payoff from the end nodes
PayOffTree = zeros(size(BTree));
 if  strcmp(type, 'C') == 1;
        PayOffTree = max(BTree-K,0);
 else
        PayOffTree = max(K-BTree,0);
 end

 %% Now we must backdate to the previous nodes.
 % Define 2 new vectors to allow vector operations to be performed.
 % Loop through the number of steps taken t reach the first step
 for i = 0:steps-1
```

```
 ShiftTree1_m = circshift(PayOffTree,[-1 1]);
 ShiftTree2_d = circshift(PayOffTree,[-2 1]);

 %% In ShiftTree1 the nth entry is nonzero and should be.
 %In ShiftTree2 this is also true and also for the n-1th entry.
 %We must needs make these zero.

ShiftTree1_m(n,1) = 0;
ShiftTree2_d(n,1) = 0;
ShiftTree2_d(n-1,1) =0;

%% Now we backdate to the previous step using the formula

PayOffTree = exp(-r*dt)*(p_u*PayOffTree + p_m*ShiftTree1_m + p_d*
    ↪ ShiftTree2_d);
n= 2*(steps-i-1) + 1;
PayOffTree = PayOffTree(1:n,1);
 end
 Value = PayOffTree(1,1);
```

D.1.2 Trinomial Trees Source Program (Boyle)

```
function TrinomialTreesE(k, s_0, dt, steps, type, r, vars, o1,o2)

%% Approximates the value of a European option using a trinomial tree
% with Boyles u,d and p

%% Inputs:
% K - strike price
% s_0 - stock price
% dt - steps size
% steps - number of steps
% type - 'C' for call 'P' for put
% r - risk free intrest rate
% vars- describes optional parameter o2 and implicitly defines o2. This
    ↪ has
% a few defined values vars=1 \Rightarrow o1=u o2=d. vars=2 \Rightarrow
% o1=sigma o2 need not be specified.

%% First we need to test how many arguments there are.
if nargin==9
    %% If o1 and o2 are given we need to dscern between the vars=1
    % and vars =3 cases.
     if vars == 1
```

```
        %% Call the oproriate function to find these value
        TrinomialTreesEBoyle(o1,o2,1,k,s_0,dt,r, steps, type)
    else
        disp('The number of variables given disagrees with the case given
            ↪ in vars')
    end
elseif nargin == 8
    if vars == 2
        %% Calculate u and d first then call the opropriate function.
        u= exp(sigma*sqrt(2*dt));
        d = 1/u;
        m=1;
        TrinomialTreesEBoyle(u,d,m,k,s_0,dt,r, steps, type)
    else
        disp('The number of variables given disagrees with the case given
            ↪ in vars')
    end
else
    disp('The number of variables given are not sifficent to solve the
        ↪ probelm')
end
```

D.2 American

D.2.1 Trinomial Trees MatLab Code MatLab Code (Boyle)

```
function Value = TrinomialTreesABoyle(u,d,m,K,s_0,dt,r, steps, type)

%% Approximates the value of a American option using Boyles values for u&
    ↪ d

%% Inputs
% u = amount stock increases by
% d= amount stock decreases by
% m = amount stock increases/decrease if it stays the same
%(Drift modles this will be not always equal to one)
% K = strike price
% s_0 current stock price
% r = risk-free intrest rate
% steps = number of steps that are to be taken
% type = type of option being considered, either 'C' or 'P' for call or
    ↪ put

%% We must first calculate the probabilities of each branch of the tree.
```

```
R = exp(r*dt/2);
p_u = ((R - d^(1/2))/(u^(1/2)-d^(1/2)))^2;
p_d = ((u^(1/2) - R)/(u^(1/2)-d^(1/2)))^2;
p_m = 1-p_u-p_d;

%% Generate our tree up to the nth step.
n = 2*steps + 1;
BTree = zeros(n,1);
for j = 0:steps-1
    BTree(j+1,1) = s_0*u^(steps-j)*m^(j);
    BTree(n-j,1) = s_0*d^(steps-j)*m^(j);
end
    BTree(((n-1)/2)+1,1) = s_0*m^steps;

%% Calculate the payoff from the end nodes
PayOffTree = zeros(size(BTree));
 if  strcmp(type, 'C') == 1;
        PayOffTree = max(BTree-K,0);
 else
        PayOffTree = max(K-BTree,0);
 end

%% We must now backdate to the previous nodes
for i = 0:steps-1
 ShiftTree1_m = circshift(PayOffTree,[-1 1]);
 ShiftTree2_d = circshift(PayOffTree,[-2 1]);

 %% In ShiftTree1 the nth entry is nonzero and should be.
 %In ShiftTree2 this is also true and also for the n-1th entry.
 %We must needs make these zero.

ShiftTree1_m(n,1) = 0;
ShiftTree2_d(n,1) = 0;
ShiftTree2_d(n-1,1) =0;

%% Calcultate the backdated value of the option.

PayOffTree = exp(-r*dt)*(p_u*PayOffTree + p_m*ShiftTree1_m + p_d*
    ↪ ShiftTree2_d);
PayOffTree2 = PayOffTree(1:2*(steps-(i+1)) + 1,1);
%% Now we must evaluate the value of the option if exercised at this node
    ↪ .
n = 2*(steps-(1+i)) + 1;
```

```
BTree = zeros(n,1);
for j = 0:steps-(i+1)
    BTree(j+1,1) = s_0*u^(steps-j)*m^(j);
    BTree(n-j,1) = s_0*d^(steps-j)*m^(j);
end
    BTree(((n-1)/2)+1,1) = s_0*m^steps;

 if  strcmp(type, 'C') == 1;
        PayOffTree1 = max(BTree-K,0);
 else
        PayOffTree = max(K-BTree,0);
 end

 %% We tke the maximum of these two values as our new payoff tree.

 PayOffTree = max(PayOffTree1,PayOffTree2);
end
Value = PayOffTree(1,1);
```

D.2.2 Trinomial Trees Source Code (Boyle)

```
function TrinomialTreesA(k, s_0, dt, steps, type, r, vars, o1,o2)

%% Approximates the value of a American option using a trinomial tree
% with Boyles u,d and p

%% Inputs:
% K - strike price
% s_0 - stock price
% dt - steps size
% steps - number of steps
% type - 'C' for call 'P' for put
% r - risk free intrest rate
% vars- describes optional parameter o2 and implicitly defines o2. This
    ↪ has
% a few defined values vars=1 \Rightarrow o1=u o2=d. vars=2 \Rightarrow
% o1=sigma o2 need not be specified.

%% First we need to test how many arguments there are
if nargin==9
    %% if o1 and o2 are given we need to dscern between the vars=1
    % and vars =3 cases.

    if vars == 1
```

```
        %% Call the oproriate function to find these value
        TrinomialTreesABoyle(o1,o2,1,k,s_0,dt,r, steps, type)
    else
        disp('The number of variables given disagrees with the case given
            ↪ in vars')
    end
elseif nargin == 8
    if vars == 2
        %% Calculate u and d first then call the opropriate function.
        u= exp(sigma*sqrt(2*dt));
        d = 1/u;
        m=1;
        TrinomialTreesABoyle(u,d,m,k,s_0,dt,r, steps, type)
    else
        disp('The number of variables given disagrees with the case given
            ↪ in vars')
    end
else
    disp('The number of variables given are not sifficent to solve the
        ↪ probelm')
end
```

Appendix E Finite Difference Methods

E.1 European

E.1.1 Implicit Finite Difference Method MatLab Code

```
function Value = FiniteDiffereceMethodsEI(k,s_0,r,q,T,sigma,s_min, s_max,
    ↪ M,N, type)

%% This is a function to use an implicit finite difference methods
% to approximate the value of a European option




%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% q- dividends during the lifetime of the option
% T - the time of the option to exirary
% sigma - the volatility of the stock
```

```
% s_min - the minimum value of the stock conisdered
% s_max - the maximum value of the stock considered
% M - the number of intervals of stock price considered. (M+1 points
    ↪ considered)
% N - the numer of intervals of time considered (N+1 points considerd)
% type - takes value 'C' for calls and 'P' for puts.

% s_min should be chosen s.t. the value for a call is zero here.
% s_max should be chosen s.t. the valye for a put is zero here.


%% Calculate dt and dS from T, s_min, s_max, N and M.
dt = T/N;
dS = (s_max - s_min)/M;

%% Construct vectors to store our values for the a_j, b_j and c_j
A=zeros(M-1,1);
B=zeros(M-1,1);
C=zeros(M-1,1);
for j = 1:M-1
   A(j,1) =  0.5*(r-q)*j*dt - 0.5*(sigma^2)*(j^2)*dt;
   B(j,1) = 1 + (sigma^2)*(j^2)*dt + r*dt;
   C(j,1) = -0.5*(r-q)*j*dt - 0.5*(sigma^2)*(j^2)*dt;
end

% Calculate a vector containing the knowns
if  strcmp(type, 'C') == 1;
    %% These are from Boundry contitions
    D(1,1) = max(0,s_min + dS - k);  %%a_1*f_N-1,0 = 0.
    D(M-1, 1) = max(0,s_max - k) - C(M-1,1)*max(0,s_max - k);
    for j = 2:M-2
        D(j,1) = max(s_min + j*dS-k, 0);
    end
else
    %% these are from Boundry conditions
    D(1,1) = max(k - s_min - dS, 0) - A(1,1)*k*exp(-r*dt); %%
    D(M-1, 1) = 0; %%Both terms are zero.
    for j = 2:M-2
        D(j,1) = max(k - s_min - j*dS, 0);
    end
end

%% Now that we have our vectors we may calculate our new coefficeints
Cprime = zeros(size(C));
```

```
Cprime(1,1) = C(1,1)/B(1,1);
for i = 2:M-1
Cprime(i,1) = C(i,1)./(B(i,1)-A(i,1)*Cprime(i-1,1));
end
Dprime = zeros(size(D));
Dprime(1,1) = D(1,1)/B(1,1);
for i = 2:M-1
    Dprime(i,1) = (D(i,1)-A(i,1)*Dprime(i-1,1))./(B(i,1)-A(i,1)*Cprime(i
       ↪ -1,1));
end
%% Calculate our solution through backward subsitution.

F = zeros(size(Dprime));
F(M-1,1) = Dprime(M-1,1);
for i = M-2:-1:1
   F(i,1) = Dprime(i,1) - Cprime(i,1)*F(i+1,1);
end


%% We may now constuct a Vector to store our solutions for these M-1
% simultaeous equations.

f= zeros(M-1,N);
f(1:M-1,N) = F(1:M-1,1);

%% We loop through the N-1 steps remaining to time t=0
for i = 0:N-2.

%% The main difference per step is that our d coefficents are now the
% solution to the previous step (j = 2,..,M-2).
% Define these as such.
for j = 2:M-2
    D(j,1) = f(j,N-i);
end


if  strcmp(type, 'C') == 1;
    %% Using our boundary conditions from earlier.
    D(1,1) = f(1,N-i); %% f_{N-i-1,0} = 0 for all i.
    D(M-1, 1) = max(0,s_max - k) - C(M-1,1)*max(0,s_max - k);
else
    D(1,1) = f(1,N-i) - A(1,1)*k*exp(-r*(T-(T-1-i)*dt));
    D(M-1, 1) = 0; %% f_{N-1-i,M} and f_f_{N-i-2,M} = 0 for all i
end
```

```
%% Now that we have our vectors we may calculate our new coefficeints
% using Thomas algorithm.
Cprime = zeros(size(C));
Cprime(1,1) = C(1,1)/B(1,1);
for k = 2:M-1
Cprime(k,1) = C(k,1)./(B(k,1)-A(k,1)*Cprime(k-1,1));
end
Dprime = zeros(size(D));
Dprime(1,1) = D(1,1)/B(1,1);
for k = 2:M-1
    Dprime(k,1) = (D(k,1)-A(k,1)*Dprime(k-1,1))./(B(k,1)-A(k,1)*Cprime(k
        ↪ -1,1));
end
%% Calculate our solution through backward subsitution.

F = zeros(size(Dprime));
F(M-1,1) = Dprime(M-1,1);
for k = M-2:-1:1
   F(k,1) = Dprime(k,1) - Cprime(k,1)*F(k+1,1);
end

f(1:M-1,N-i-1) = F(1:M-1,1);

end

%% Find the S_0 for the right solution.
for j = 0:M
        if s_0 - j*dS + s_min <= 0.0001 ;
     Value = f(j+1,1);
        jfound=1;
    end
end
if jfound ~= 1
    disp('Your stock price is not one of the paritions of the plane. Try
        ↪ again and make sure s_0 = s_min+j*dS for some j')
Value = 0;
end
```

E.1.2 Explicit Finite Difference Method MatLab Code

```
function Value = FiniteDifferenceMethodsEE(k,s_0,r,q,T,sigma,s_min, s_max
    ↪ , M ,N, type)
```

```
%% This is a function to use an explicit finite difference methods to
% approximate the value of a European option

%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% q- dividends during the lifetime of the option
% T - the time of the option to exirary
% sigma - the volatility of the stock
% s_min - the minimum value of the stock conisdered
% s_max - the maximum value of the stock considered
% M - the number of intervals of stock price considered. (M+1 points
    ↪ considered)
% N - the numer of intervals of time considered (N+1 points considerd)
% type - takes value 'C' for calls and 'P' for puts.

% s_min should be chosen s.t. the value for a call is zero here.
% s_max should be chosen s.t. the valye for a put is zero here.

%% Calculate dt and dS from T, s_min, s_max, N and M.
dt = T/N;
dS = (s_max - s_min)/M;

%% Impose the terminal conditions.

if  strcmp(type, 'C') == 1;
    %% Store boundary conditions
    for j = 1:M+1
        D(j,1) = max(s_min + (j-1)*dS-k, 0);
    end
else
    %% Store Boundry conditions
    for j = 1:M+1
        D(j,1) = max(k - s_min - (j-1)*dS, 0);
    end
end

%% We now calculate our coefficents a*_j, b*_j and c*_j.

for j = 1:M+1
```

```
    A(j,1) = (1/(1+r*dt))*(-0.5*(r-q)*(j-1)*dt + 0.5*(sigma^2)*((j-1)^2)*
        ↪ dt);
    B(j,1) = (1/(1+r*dt))*(1-(sigma^2)*((j-1)^2)*dt);
    C(j,1) = (1/(1+r*dt))*(0.5*(r-q)*(j-1)*dt + 0.5*(sigma^2)*((j-1)^2)*
        ↪ dt);
end

%% We will perfrom vector operations to calculate at the N-1th step.
% We shift the vecotr to do this.

ShiftD = circshift(D,[-1 1]);
ShiftD2 = circshift(D,[1 1]);
%% These shifts leave non-zero values where there should be. We zero
    ↪ these now.
ShiftD(M+1,1) = 0;
ShiftD2(1,1) = 0;

%% Calculate the values one time inverval backwards

F(1:M+1,1) = A(1:M+1,1).*ShiftD2(1:M+1,1) + B(1:M+1,1).*D(1:M+1,1) + C(1:
    ↪ M+1,1).*ShiftD(1:M+1,1);

%% We have calculated the terms at the boundary of S (j=1 and j=M+1).
% We must set these to the boundary conditions.
if strcmp(type, 'C') == 1;
        F(1,1) = 0;
        F(M+1,1) = max(s_max-k, 0);
else
        F(1,1) = k*exp(-r*dt);
        F(M+1,1) = 0;
end

%% These are our values at time step N-1. We store these in a matrix.
f = zeros(M+1,N);
f(1:M+1,N) = F(1:M+1,1);

%% We must now loop through back to time step 0.

for i = 0:N-2

D(1:M+1,1) = f(1:M+1,N-i);

%% We shift this again.
```

```
ShiftD = circshift(D,[-1 1]);
ShiftD2 = circshift(D,[1 1]);
%% These shifts leave non-zero values where there should be. We zero
    ↪ these now.
ShiftD(M+1,1) = 0;
ShiftD2(1,1) = 0;
%% Calculate the values one time step backwards
F(1:M+1,1) = A(1:M+1,1).*ShiftD2(1:M+1,1) + B(1:M+1,1).*D(1:M+1,1) + C(1:
    ↪ M+1,1).*ShiftD(1:M+1,1);

%% Here we have calculated the terms at the boundary of S (j=1 and j=M+1)
    ↪ .
% We must set these to the boundary conditions.
if  strcmp(type, 'C') == 1;
        F(1,1) = 0;
        F(M+1,1) = max(s_max-k, 0);
else
        F(1,1) = k*exp(-r*(T-i*dt));
        F(M+1,1) = 0;
end
F = max(0,F);
%% These are our values at time step N-1. We store these in a matrix.
f(1:M+1,N-1-i) = F(1:M+1,1);
end
%% Find the S_0 for the right solution.
for j = 0:M
        if s_0 - j*dS + s_min <= 0.0001 ;
     Value = f(j+1,1);
        jfound=1;
    end
end
if jfound ~= 1
    disp('Your stock price is not one of the paritions of the plane. Try
       ↪ again and make sure s_0 = s_min+j*dS for some j')
Value=0;
end
```

E.2 American

E.2.1 Implicit Finite Difference Method MatLab Code

```
function Value = FiniteDifferenceMethodsAI(K,s_0,r,q,T,sigma,s_min, s_max,
    ↪ M,N, type)
```

```
%% This is a function to use an implicit finite difference methods
% to approximate the value of a European option


%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
% q- dividends during the lifetime of the option
% T - the time of the option to exirary
% sigma - the volatility of the stock
% s_min - the minimum value of the stock conisdered
% s_max - the maximum value of the stock considered
% M - the number of intervals of stock price considered. (M+1 points
    ↪ considered)
% N - the numer of intervals of time considered (N+1 points considerd)
% type - takes value 'C' for calls and 'P' for puts.

% s_min should be chosen s.t. the value for a call is zero here.
% s_max should be chosen s.t. the valye for a put is zero here.


%% Calculate dt and dS from T, s_min, s_max, N and M.
dt = T/N;
dS = (s_max - s_min)/M;

%% We construct vectors to store our values for the a_j, b_j and c_j

for j = 1:M-1
   A(j,1) =  0.5*(r-q)*j*dt - 0.5*(sigma^2)*(j^2)*dt;
   B(j,1) = 1 + (sigma^2)*(j^2)*dt + r*dt;
   C(j,1) = -0.5*(r-q)*j*dt - 0.5*(sigma^2)*(j^2)*dt;
end

% Define a vecotr sotring the knowns
if  strcmp(type, 'C') == 1;
    %% From the boundary Boundry contitions
    D(1,1) = max(0,s_min + dS - K);  %%a_1*f_N-1,0 = 0.
    D(M-1, 1) = max(0,s_max - K) - C(M-1,1)*max(0,s_max - K);
     for j = 2:M-2
         D(j,1) = max(s_min + j*dS-K, 0);
     end
else
    %% From the Boundry conditions
```

```
    D(1,1) = max(K - s_min - dS, 0) - A(1,1)*K; %% JChull.
    D(M-1, 1) = 0; %%Both terms are zero.
    for j = 2:M-2
        D(j,1) = max(K - s_min - j*dS, 0);
    end
end

%% Calculate our new coefficeints
Cprime = zeros(size(C));
Cprime(1,1) = C(1,1)/B(1,1);
for i = 2:M-1
Cprime(i,1) = C(i,1)./(B(i,1)-A(i,1)*Cprime(i-1,1));
end
Dprime = zeros(size(D));
Dprime(1,1) = D(1,1)/B(1,1);
for i = 2:M-1
    Dprime(i,1) = (D(i,1)-A(i,1)*Dprime(i-1,1))./(B(i,1)-A(i,1)*Cprime(i
        ↪ -1,1));
end
%% Calculate our solution through backward subsitution.

F = zeros(size(Dprime));
F(M-1,1) = Dprime(M-1,1);
for i = M-2:-1:1
   F(i,1) = Dprime(i,1) - Cprime(i,1)*F(i+1,1);
end


%% Calculate the payoff at these nodes.
if  strcmp(type, 'C') == 1;
    for j = 1: M-1
    PayOff(j,1) = max(0,j*dS + s_min -K );
    end
else
    for j = 1: M-1
    PayOff(j,1) = max(0, K-j*dS - s_min );
    end
end

%% Take the maximum to determine if early exercise is optimal.

F = max(PayOff, F);

%% We may now constuct a Vector to store our solutions for these M-1
```

```
% simultaeous equations.
f= zeros(M-1,N);
f(1:M-1,N) = F(1:M-1,1);

%% We loop through the N-1 steps remaining to time t=0
for i = 0:N-2

%% The main difference per step is that our d coefficents are now the
% solution to the previous step (j = 2,..,M-2).
% Define these as such.
for j = 2:M-2
    D(j,1) = f(j,N-i);
end

if  strcmp(type, 'C') == 1;
    %% Using our boundary conditions from earlier.
    D(1,1) = f(1,N-i); %% f_{N-i-1,0} = 0 for all i.
    D(M-1, 1) = max(0,s_max - K) - C(M-1,1)*max(0,s_max - K);
else
    D(1,1) = f(1,N-i) - A(1,1)*K;
    D(M-1, 1) = 0; %% f_{N-1-i,M} and f_f_{N-i-2,M} = 0 for all i
end

%% Now that we have our vectors we may calculate our new coefficeints
Cprime = zeros(size(C));
Cprime(1,1) = C(1,1)/B(1,1);
for k = 2:M-1
Cprime(k,1) = C(k,1)./(B(k,1)-A(k,1)*Cprime(k-1,1));
end
Dprime = zeros(size(D));
Dprime(1,1) = D(1,1)/B(1,1);
for k = 2:M-1
    Dprime(k,1) = (D(k,1)-A(k,1)*Dprime(k-1,1))./(B(k,1)-A(k,1)*Cprime(k
        ↪ -1,1));
end
%% Calculate our solution through backward subsitution.

F = zeros(size(Dprime));
F(M-1,1) = Dprime(M-1,1);
for k = M-2:-1:1
   F(k,1) = Dprime(k,1) - Cprime(k,1)*F(k+1,1);
end

%% Calculate the payoff at these nodes.
```

```
if  strcmp(type, 'C') == 1;
    for j = 1: M-1
    PayOff(j,1) = max(0,j*dS + s_min -K );
    end
else
    for j = 1: M-1
    PayOff(j,1) = max(0, K-j*dS - s_min ) ;
    end
end

%% Now we may take the maximum to determine if early exercise is optimal.

F = max(PayOff, F);

f(1:M-1,N-i-1) = F(1:M-1,1);

end

%% Find the S_0 for the right solution.
for j = 0:M
        if s_0 - j*dS + s_min <= 0.0001 ;
     Value = f(j+1,1);
        jfound=1;
    end
end
if jfound ~= 1
    disp('Your stock price is not one of the paritions of the plane. Try
       ↪ again and make sure s_0 = s_min+j*dS for some j')
Value = 0;
end
```

E.2.2 Explicit Finite Difference Method MatLab Code

```
function Value = FiniteDifferenceMethodsAE(k,s_0,r,q,T,sigma,s_min, s_max
   ↪ , M ,N, type)


%% This is a function to use an explicit finite difference methods to
% approximate the value of a American option

%% Inputs:
% k - strike price
% s_0 - inital stock price
% r - risk fre intrest rate
```

```
% q- dividends during the lifetime of the option
% T - the time of the option to exirary
% sigma - the volatility of the stock
% s_min - the minimum value of the stock conisdered
% s_max - the maximum value of the stock considered
% M - the number of intervals of stock price considered. (M+1 points
    ↪ considered)
% N - the numer of intervals of time considered (N+1 points considerd)
% type - takes value 'C' for calls and 'P' for puts.

% s_min should be chosen s.t. the value for a call is zero here.
% s_max should be chosen s.t. the valye for a put is zero here.

%% Calculate dt and dS from T, s_min, s_max, N and M.
dt = T/N;
dS = (s_max - s_min)/M;

%% We impose the terminal conditions.

if  strcmp(type, 'C') == 1;
    %% Store Boundry contitions
    for j = 1:M+1
        D(j,1) = max(s_min + (j-1)*dS-k, 0);
    end
else
    %% Store Boundry conditions
    for j = 1:M+1
        D(j,1) = max(k - s_min - (j-1)*dS, 0);
    end
end

%% Calculate our coefficents a*_j, b*_j and c*_j.

for j = 1:M+1
   A(j,1) =  (1/(1+r*dt))*(-0.5*(r-q)*(j-1)*dt + 0.5*(sigma^2)*((j-1)^2)*
       ↪ dt);
   B(j,1) =  (1/(1+r*dt))*(1-(sigma^2)*((j-1)^2)*dt);
   C(j,1) =  (1/(1+r*dt))*(0.5*(r-q)*(j-1)*dt + 0.5*(sigma^2)*((j-1)^2)*
       ↪ dt);
end

%% We will perfrom vector operations to calculate at the N-1th step.
% We shift the vecotr to do this.
```

```
ShiftD = circshift(D,[-1 1]);
ShiftD2 = circshift(D,[1 1]);
%% These shifts leave non-zero values where there should be.
% We zero these now.
ShiftD(M+1,1) = 0;
ShiftD2(1,1) = 0;

%% Calculate the values one time inverval backwards
F(1:M+1,1) = A(1:M+1,1).*ShiftD2(1:M+1,1) + B(1:M+1,1).*D(1:M+1,1) + C(1:
    ↪ M+1,1).*ShiftD(1:M+1,1);

%% We have calculated the terms at the boundary of S (j=1 and j=M+1).
% We must set these to the boundary conditions.
if  strcmp(type, 'C') == 1;
        F(1,1) = 0;
        F(M+1,1) = max(s_max-k, 0);
else
        F(1,1) = k*exp(-r*dt);
        F(M+1,1) = 0;
end
%% Calculate the payoff at these nodes.
if  strcmp(type, 'C') == 1;
    for j = 1: M+1
    PayOff(j,1) = max(0,(j-1)*dS + s_min -k );
    end
else
    for j = 1: M+1
    PayOff(j,1) = max(0, k-(j-1)*dS - s_min );
    end
end

%% Now we may take the maximum to determine if early exercise is optimal.

F = max(PayOff, F);
%% These are our values at time step N-1. We store these in a matrix.
f = zeros(M+1,N);
f(1:M+1,N) = F(1:M+1,1);

%% Loop through back to time step 0.

for i = 0:N-2
```

```
D(1:M+1,1) = f(1:M+1,N-i);

%% We shift this again.

ShiftD = circshift(D,[-1 1]);
ShiftD2 = circshift(D,[1 1]);
%% These shifts leave non-zero values where there should be. We zero
   ↪ these now.
ShiftD(M+1,1) = 0;
ShiftD2(1,1) = 0;

%% Calculate the values one time inverval backwards

F(1:M+1,1) = A(1:M+1,1).*ShiftD2(1:M+1,1) + B(1:M+1,1).*D(1:M+1,1) + C(1:
   ↪ M+1,1).*ShiftD(1:M+1,1);

%% Here we have calculated the terms at the boundary of S (j=1 and j=M+1)
   ↪ .
% We must set these to the boundary conditions.
if  strcmp(type, 'C') == 1;
        F(1,1) = 0;
        F(M+1,1) = max(s_max-k, 0);
else
        F(1,1) = k;
        F(M+1,1) = 0;
end

%% Calculate the payoff at these nodes.
if  strcmp(type, 'C') == 1;
    for j = 1: M+1
    PayOff(j,1) = max(0,(j-1)*dS + s_min -k );
    end
else
    for j = 1: M-1
    PayOff(j,1) = max(0, k-(j-1)*dS - s_min );
    end
end

%% Now we may take the maximum to determine if early exercise is optimal.

F = max(PayOff, F);
%% These are our values at time step N-1. We store these in a matrix.
f(1:M+1,N-1-i) = F(1:M+1,1);
```

```
end

%% Find the S_0 for the right solution.
for j = 0:M
        if s_0 - j*dS + s_min <= 0.0001 ;
     Value = f(j+1,1);
        jfound=1;
    end
end
if jfound ~= 1
    disp('Your stock price is not one of the paritions of the plane. Try
       ↪ again and make sure s_0 = s_min+j*dS for some j')
Value = 0;
end
```