MTHM005: Mathematical Sciences Project

Pricing Asian Options in Matlab

Candidate Number: 145670

Supervised by Prof. Byott

January 23, 2023

Abstract

In this report we take a deep look into the fundamental theory surrounding pricing options. Starting from market definitions and assumptions we explore the statistical properties that underpin several asset pricing models and extend it to option valuation with constant reference to the relevant literature. Finally, we implement a well-established pricing model for Asian options with the use of Matlab and attempt to make it more efficient by reducing the time complexity of the algorithm.

All files available online:
https://github.com/leele2/Mathematics-in-Business-Project

Contents

List of Figures and Tables

Chapter 1: Introduction

This report studies a vital financial derivative in today's markets, namely options. The importance of the option market has been shown by empirical studies which suggest that option trading improves information efficiency in the broader stock market [PP06; Li21], and also that firms with listed options experience lower implied cost of equity capital [NNT13]; indicating that options trading reduces the cost of capital [Li21]. The popularity of the options market can easily be seen in Figure 1.1, which shows the exponential growth in trading volume since standardized, exchange-traded stock options were first listed in The Chicago Board Options Exchange in 1973 [Mar02]. In 2020 single stock option trading volume became higher than the underlying stock volume for the first time ever [Fin20].

It is this explosive popularity and significance which have motivated this report. We will begin by describing standard options and explore popular methods that are used to price them. We will then move onto Asian options and look at the literature surrounding how to price them before implementing several pricing methods with the use of MATLAB. Furthermore, we will make an adjustment to a well-established pricing model, in hopes of improving its efficiency. We will then take an analytical approach to determine if the change has impacted the models ability to price accurately and efficiently.

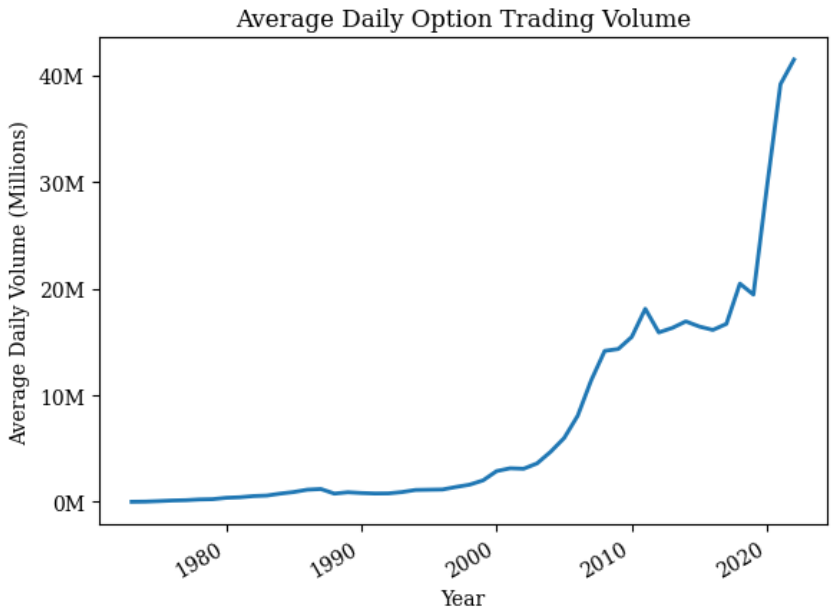

Figure 1.1: Time series plot of the average daily option trading volume per annum. Data from by the Options Clearing Corporation (OCC) [Cor22]. Source code: Listing B.1

1.1 A brief overview of call and put options

Options are a particular type of financial derivative, a contract that details the conditions under which payments are made between two counterparties. They are purchased for a set fee, and in return the buyer is granted the right, but not the obligation, to buy or sell an underlying asset — such as commodities, stocks or bonds — for a predetermined price (the strike price) on or before a determined date (the expiry date).

Call options allow the buyer to purchase an asset for the strike price at a future date. The buyer can make a return if the value of the asset is worth more than the strike price when exercised. Alternatively, put options allow the buyer to sell an asset for the strike price at a future date and the buyer can make a return if the value of the asset is less than the strike price when exercised.

The option market is widely considered a venue for informed trading [Li21; Hu14; CGM04], that is, investors trading with superior knowledge of the probability distribution of share prices, through either access to private information or skilful processing of public information [Gro75].

1.1.1 A short history of option trading

The history of financial options can be traced as far back as 6th century BCE when ancient Greek mathematician and philosopher Thales of Miletus predicted through his astrological knowledge there was going to be a great olive harvest. As he did not have much money, he used what he had as a deposit on the rights to the local olive presses; due to no competition he secured this at a relatively low price. When the harvest proved to be bountiful leading to high demand, Thales charged a high price for use of the presses and reaped a considerable profit. His deposit gave him the right but not the obligation to hire the presses, thus his losses were limited to his initial deposit [Ins12; Ari77].

Whilst this is quite a positive look on option trading, throughout history this has not always been the case. During the Dutch tulip bubble of the seventeenth century, tulips were seen as a status symbol which caused high demand and consequently drove their prices up, creating a bubble [Das11]. Tulip growers would buy puts to protect their profits in case the price of tulip bulbs went down and wholesalers would buy calls to protect against the risk of tulip bulbs going up. When the bubble eventually burst, there was no way to force investors to fulfil their obligations under the options contract, due to the unregulated nature of the option market. This ultimately led to options gaining a dubious reputation and bans were later placed on them within Britain between 1733–1860 [Poi08].

During the late nineteenth century, brokers started to arrange deals between buyers and sellers of options for particular stocks at prices that were arranged between the two parties. Trades were arranged similarly until the 1960s when the options market started to become regulated by the Chicago Board of Trade. In 1973, the Chicago Board of Options Exchange (CBOE) began trading and for the first time options contracts were properly standardized. At the same time, the Options Clearing Corporation was established for centralized clearing and enforcing the proper fulfilment of contracts, ensuring that they were honoured [Mar02].

1.1.2 Standard options

A standard option comes in two styles; European: which restricts the holder of the option to only exercise the option on the expiry date, and American: which allows the holder to exercise the option at anytime up till or on the expiry date. They will take the current value of the underlying asset as the spot price — that is the price that the asset can be purchased for on the open market. The payoff in this case then becomes the difference between the spot price and strike price.

Mathematically we write the payoff $\mathbb{V}(t)$ of a European put option with strike price E, expiry date T and underlying asset price at time t being $S(t)$:

$$\mathbb{V}(T) = \max(E - S(T), 0)$$

and similarly for a European call option

$$\mathbb{V}(T) = \max(S(T) - E, 0)$$

An American put or call option payoff is identical however it may be exercises at anytime up to the expiry date, giving: $\mathbb{V}(t) \quad \forall t \in [0, T]$.

1.1.3 Asian options

Whilst standard options involve using the spot price as the underlying value of the asset; this is not always the case with so-called exotic options. Exotic options differ in their payment structures, expiration dates, and/or strike prices. In the case of exotic fixed-strike price Asian options, the average price of the asset is used in place of the underlying asset value. This differs from fixed-price Asian options which instead use the average price of the asset to take place of the strike price. These are the two main variations of Asian style options but both of these can be varied further in how the averaging is calculated, for example: geometrically, arithmetically, average taken every day or average taken at the start of each month and so on. They can be varied further by having an expiry structure matching a European or American style option.

A fixed-strike price Asian option with strike price E, expiry time T and underlying asset value at time t being $S(t)$; has the following put and call payoff functions $\mathbb{V}(t)$:

$$\mathbb{V}(t) = \max(E - f(t, \delta t),\ 0) \qquad \mathbb{V}(t) = \max(f(t, \delta t) - E,\ 0)$$

For some averaging function $f(t, \delta t)$ where δt represents the time increment for which the average is calculated upon. For an arithmetic average taken every day when time, t units is days our averaging function is:

$$f(t, \delta t) = \left(\left\lfloor \frac{t}{\delta t} \right\rfloor \right)^{-1} \sum_{i=0}^{\lfloor t/\delta t \rfloor} S(i\delta t)$$

Note that $\forall x \in \mathbb{R}$ $\lfloor x \rfloor$ notation represents the greatest integer less than or equal to x. This allows the function to work for early exercise when $t \in [0, T]$ in the case of American type Asian options.

1.2 Market definition and assumptions

To begin pricing options we must first establish assumptions on the value of assets and the market in which they are traded. We will consider a market in which there are two classes of assets; risk-free assets and risky assets. We allow derivatives to exist on risky assets.

1.2.1 Asset definitions

We define our two classes of assets with the following assumptions:

Risk-free assets

Risk-free assets are assumed to have no uncertainty in regard to their future value and defined as returning interest continuously compounded at a fixed rate r. Hence, an investment of A made at time t will return $A^{r(T-t)}$ at time $T > t$ with no uncertainty. These can be positive e.g., making an investment, or negative e.g., taking out a loan. Risk-free assets are akin to fixed rate investments at a well-know bank or government bonds. However, even sovereign nations have been known to default on their bond repayments so in reality are not truly risk-free, but instead considered low-risk [Kit+15].

Risky assets

Risky assets are assets in which the future value can not be determined without at least some uncertainty. These are typically shares in a company or commodities such as gold and oil. These assets are traded on the open market and their value is heavily dependent on the balance of supply and demand of the asset; this in turn is dependent on so many variables that we assume their future value to be a random process. Depending on our model we can make different assumptions on the distribution of this process and thus how the value evolves over a time period. In practice has been arguments made for the future value of risky assets are truly random [Fam65], and other arguments that propose they are instead at least partly deterministic where patterns and trends exists [SP85].

1.2.2 Market Assumptions

We shall assume that an investor may hold any quantity (including negative and/or fractional) of an asset, known as the *divisibility* and *short-selling* assumption. We will also assume that our market is *frictionless* — that is that any quantity of assets can be bought and sold at the same price with no transaction cost and are entirely *liquid* meaning there is always a buyer for every seller and vice versa.

In practice these assumptions do not accurately represent real markets, for example it is common place for markets to have bid-ask spreads which represents the different in price at which you can buy and sell assets. Furthermore, transaction costs are impossible to fully eliminate — even in scenarios with no tax or commission fees, buying or selling assets (especially in large quantities) will in theory have an effect on the supply and demand of that asset; this introduces market impact costs [Mor+09]. There are other transaction fees to also consider such as: time spent deciding what assets to buy, computing power costs and their associated opportunity costs.

No-Arbitrage assumption

Perhaps our most important market assumption is the No-Arbitrage principle; sometimes referred to as the no free-lunch principle. This assumption states that there can be no risk-free way to get a better rate of return than the market risk-free interest rate, r. The logic behind this assumption is that if there exists a way to earn a higher rate of return than r with no initial outlay; than other traders would employ the same method and market forces would eliminate the arbitrage opportunity. This assumption assumes that markets are perfectly efficient meaning all traders have access to the same information and that market adjustments to supply and demand happen instantaneously, which we know not to be true in practice.

1.3 Asset pricing models

Having now established our market assumptions, we must now define our assumptions on the random process which describes how the asset value may evolve over a specified time period. We will first introduce the mathematical concepts on which these models are built upon. Then we will introduce a simple asset model and one which has more assumptions in hopes of better describing the future asset value in practice.

Stochastic processes

A stochastic process is defined as a collection of random variables defined on a common probability space $(\Omega, \mathcal{F}, P)$, where Ω is a sample space, $\mathcal{F}$ is a σ-algebra, and P is a probability measure; and the random variables, indexed by some set T, all take values in the same mathematical space S, which must be measurable with respect to some σ-algebra Σ [Lam77].

A discrete stochastic process is a sequence of random variables which represent observations taken at discrete times from a random system, e.g. observations $\{X_i : i \in \mathbb{N}_{[0,n]}\}$ at times $t = t_0, t_1, \ldots t_n$. We can let our market be the random system and our observations, X_i be the value of our asset observed on the i^{th} day. This is a typical example of structuring an asset market as a discrete stochastic process; we will later see how this is helpful for building our asset pricing models.

Brownian Motion (Wiener Process)

Brownian motion is a pattern of motion first described by the botanist Robert Brown in 1827 when observing the unpredictable motion of pollen of a plant through a microscope [Pea+10]. The motion typically consists of random fluctuations in a particle's position inside a subdomain, followed by relocation to another subdomain. This motion has applications in many fields of study such as pure and applied mathematics, economics and physics. In statistics, Brownian motion is described by the Wiener Process.

A Wiener process, $\mathcal{W}_t$ is a continuous-time stochastic process named after Norbert Wiener for his study into the mathematical properties of one-dimension Brownian motion [WM76]; it is characterized by the following properties [Dur19]:

1. $\mathcal{W}_0 = 0$
2. $\mathcal{W}$ has independent increments, i.e. $\forall t > 0$, the future increments $\mathcal{W}_{t+u} - \mathcal{W}_t, u \geq 0$, are independent of the past values $\mathcal{W}_s, s \leq t$.
3. $\mathcal{W}$ has Gaussian increments: $\mathcal{W}_{t+u} - \mathcal{W}_t$ is normally distributed with mean 0 and variance u, i.e. $\mathcal{W}_{t+u} - \mathcal{W}_t \sim \mathcal{N}(0, u)$.
4. $\mathcal{W}$ has continuous paths: $\mathcal{W}_t$ is continuous in t.

Asset Prices as Brownian Motion

The French mathematician Louis Bachelier is often attributed to pioneering using Brownian motion to model asset prices when he did so in his PhD thesis "The Theory of Speculation" (1900) [Bac07]. However, a lesser known French economist Jules Renault was first to do so in his 1863 publication "calcul des chances et philosophie de la bourse" [JP06].

Regnault hypothesized that asset prices change according to a game of "heads or tails":

> "On the stock market, the whole mechanism of the game comes down to two opposite possibilities: increasing and decreasing. Each one can always appear with an equal facility" [Reg63; JP06]

He also claimed "the deviation of prices is directly proportional to the square root of time" this is a result we would expect to see from a Weiner process (property 3). Fast-forward to 1900 and Bachelier further developed the theory of asset prices moving according to a random process; his ideas depended on markets being efficient. Bachelier theorized in his thesis that as soon as prices becomes predictable, it immediately becomes exploited. However, he believed all traders to have access to all available information which leads to these predictable patterns instantly disappearing. Bachelier believed prices took what we now call "random walks" above and below their true value, as competing traders attempt to beat the market. He would go on to conduct a study on French government bonds, concluding that their price changes were consistent with a random walk model; furthermore, Bachelier would formulate many of the mathematical properties of the stochastic process now known as Brownian motion [Bac07].

These contributions by Bachelier went largely unnoticed until the mid 20th century when several historic publications formalized and furthered the theory surrounding the Random-Walk and Efficient-Market Hypothesis [KH53; Coo64;

Fam65]. These hypotheses were heavily studied throughout the 20th century and pathed the way for many advances in mathematical finance. Over the decades there has been mixed evidence to support the hypothesis that markets can not be predicted. More recently studies have shown that market predictability has become more difficult due to advances in trading technology and investor learning [WG08; MP16]. However, for our market and asset assumptions, and our ultimate goal of implementing pricing models for financial options; random walk and Brownian motion models are sufficient.

1.3.1 Random-walk model

A random walk is a type of stochastic process that describes a path in which at every movement forward in time the movement in space is decided by a random process. A simple 2 dimension case for example; we construct a path in which we start at 0 and for every time step forward, we flip a fair coin to decide if we move up or down. We let S_i be our position of the path at time $t = t_i$ and y_i be the result of our i^{th} coin toss and Y_i be how the toss effects our path; where $Y_i = 1$ if $y_i =$ heads and $Y_i = -1$ if $y_i =$ tails. This random walk is called symmetric as it goes up or down by the same amount. If we were to plot our path it would look something like Figure 1.2:

The general case

It follows that we can construct a random walk for the general case where we may change the probabilities, the size of the step taken and also introduce a drift, μ and volatility, σ parameter. This is written as follows:

$$\begin{aligned} S_0 &= S_0 \\ S_{n+1} &= S_n + \mu + \sigma Y_i \end{aligned}$$

Where μ is the drift, σ scales the movement and Y_i are identically and independently distributed (i.d.d.):

$$Y_i = \begin{cases} \alpha & \text{with } P(Y_i = \alpha) = p; \\ \beta & \text{with } P(Y_i = \beta) = (1 - p) \end{cases}$$

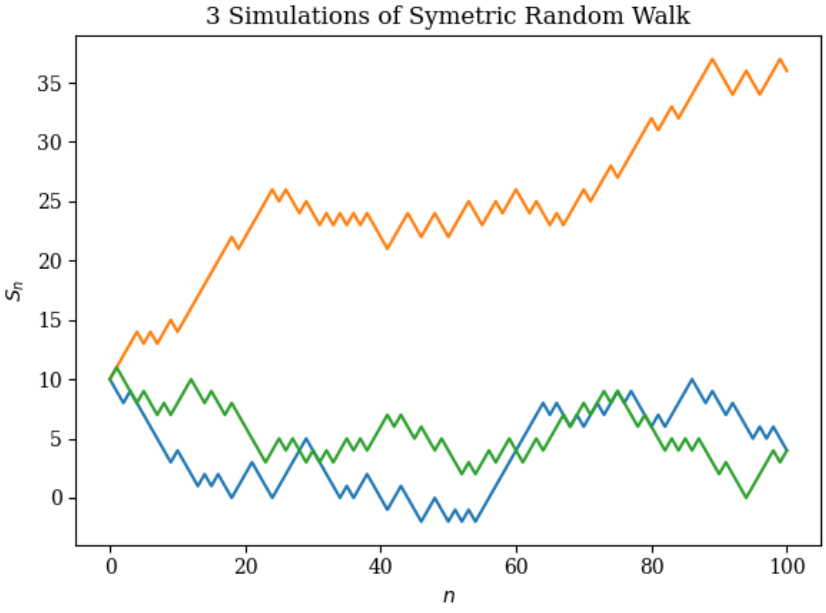

Figure 1.2: Three simulations of a symmetric random walk with probability $p = 0.5$, no drift and no volatility ($\mu = 0, \sigma = 1$). Source code: Listing B.2

We let S_i be the price of an asset on the i^{th} day, thus S_0 is the current price of the asset. The drift allows us to model in the expected increase/decrease in price of the asset; scale allows us to model the volatility (the magnitude of movements). There are different methods in which we could employ to calculate our drift and scale parameters, this will typically involve using historical data but can extend to take into account speculation about the underlying asset.

Drawbacks of a random walk to model asset prices

We can begin to see how this path may reflect the price an asset which future value is determined by a random event. However, this model comes with several drawbacks: This model only permits the price of the asset to go up and down by a fixed amount, further to this point, the amount is detached from the current value of the asset; we may instead expect that the value change would be proportional to the current value. Lastly, the random walk model also allows asset prices to go negative which typically is considered impossible. However, during April 2020, oil futures became sharply negative for the first time in history [Cho+22]. This has lead to people to reconsider the random walk model and see this instead as a positive in specific circumstances. An alternative approach instead applies the random walk model to rate of return instead of asset value, which of course can often be negative.

1.3.2 Geometric Brownian motion model

A better alternative is the (discrete) geometric Brownian model, shown in Equation 1.1. This model addresses the drawbacks of the simple random walk model. We can see from the left-hand side of the equation, that we now model the percentage increase from the previous time to the current time. This means our price change is not detached from the previous value and the value can never become negative.

$$\frac{S_{n+1} - S_n}{S_n} = \delta t\mu + \sqrt{\delta t}\sigma Y_n \tag{1.1}$$

Where μ is the drift, σ scales the movement and Y_i are identically and independently distributed (i.d.d.).

On the right-hand side: we introduce a new term, δt, this is taken to be a fixed time interval which is comparatively small to the time interval $t \in [0, T]$; where $t = T$ is the final time of the model. We let the price of the asset, $S_i = S(i)$ no longer be measured on the i^{th} day, but instead the i^{th} number of the time interval δt. It is common to calculate δt by considering your entire time range, $t \in [0, T]$ and splitting this into $N + 1$ δt increments, where N is some large number; hence $S_N = S(N\delta t) = S(T)$. Now, if we were to consider a drift parameter μ which represents the expected price increase in one unit of time, it would follow that the increase over our time increment δt would be $\delta t\mu$. The variance (percentage volatility) of the percentage increase at the final, N^{th} step would be N times the variance at each time interval step, so we would expect this variance to be proportional to δt. We let the random component of the percentage increase be proportional to $\sqrt{\delta t}$; it can be shown that letting σ be proportional to δt or $(\delta t)^{0.25}$ we end up with too little or too much randomness respectively (see [Hig04]). We will again let the values of μ and σ be determined in the same manner as previously where we use expert opinion and/or historical data.

Modified Bernoulli distributed Y_i

We can, as we did previously model the random process in our model, Y_i as a so-called symmetric random walk; that is:

$$Y_i = \begin{cases} 1 & \text{with } P(Y_i = 1) = 0.5; \\ -1 & \text{with } P(Y_i = -1) = 0.5 \end{cases}$$

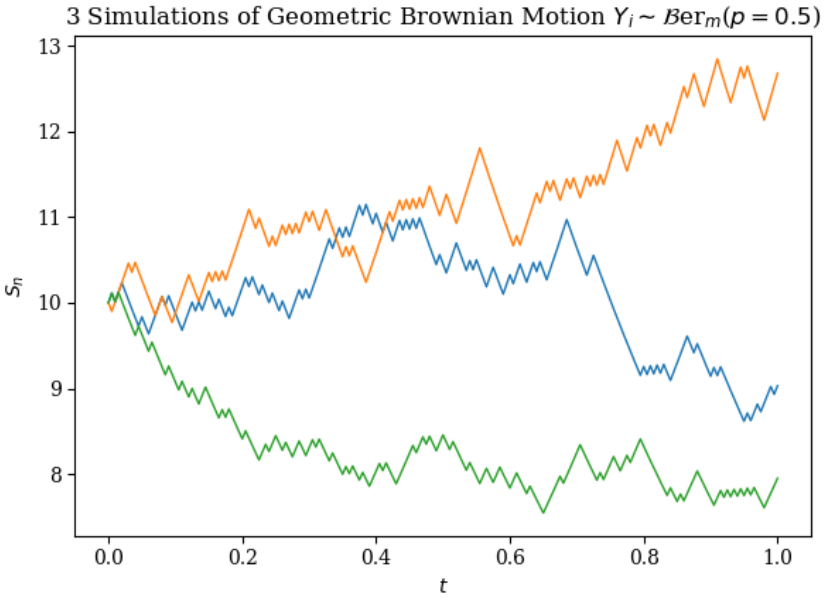

Figure 1.3: Three simulations of geometric Brownian motion random walk: with probability $p = 0.5$, 10% drift ($\mu = 0.10$), volatility 15% ($\sigma = 0.15$) and domain $t \in [0, 1]$ into 201 equal increments. Source code: Listing B.3

In Figure 1.3 we have simulated geometric Brownian motion over the time range $t \in [0, 1]$. Having $\mu = 0.10$ means that we expect the path to increase by 10% for every integer increment in t, thus we would expect $S_N = (1.1)S_0$ at $t = N\delta t = T$. Our percentage volatility, $\sigma = 0.15$ scales our movement at every increment before the drift is applied, this controls the variance or spread of our path. Whilst we have improved significantly from our previous model in Subsection 1.3.1, we still have a restriction in our price change. By having our random process, Y_i be Bernoulli distributed, we have limited our price change to either increase or decrease by a fixed percentage value.

Normally distributed Y_i

We can then instead distribute Y_i normally which allows us to add randomness to the amount at which the percentage value changes, instead of just being positive of negative. For instance, we can let $Y_i \sim \mathcal{N}(0, 1)$ then our random process still has equal probability of being positive or negative (since $\mu = 0$), but can now vary by random amounts; furthermore it has equal probability to be in some range $[a, b]$ as it is to be in $[-a, -b]$. We set our variance of Y_i to be one, $\sigma_Y = 1$, this gives us the standard normal distribution and allows us to alter percentage volatility, σ to account for higher or lower variance; as done previously. Figure 1.4 shows three simulations of geometric Brownian motion with standard normal distributed Y_i.

We see how this looks very similar to our previous model, with the exception that it much smoother. Our drift parameter μ and percentage volatility σ effect our model in the same way as previously, be it more randomness. You may suspect that in the long run these models will converge to have the same statistical properties. In the next section we will attempt to prove this analytically by letting $\delta t \to 0$ and $N \to \infty$ and finding a limiting equation of our Brownian motion model.

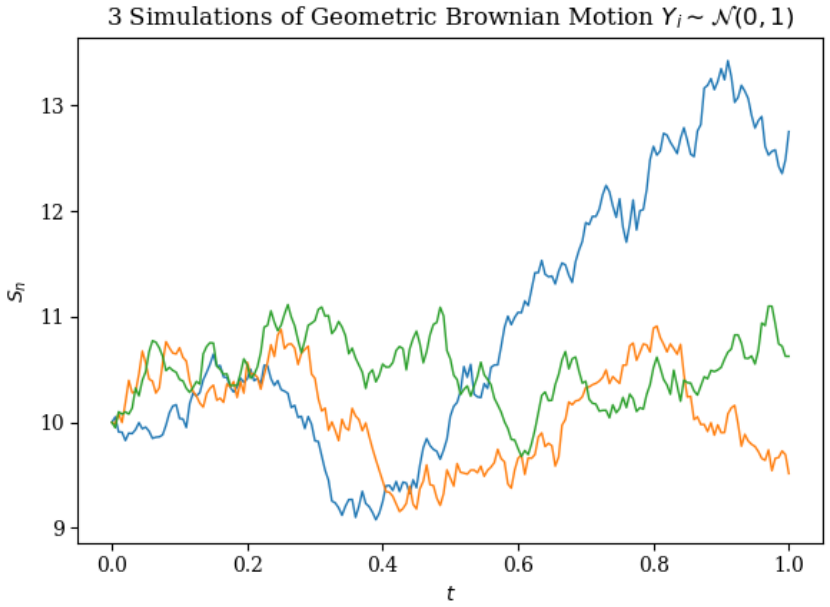

Figure 1.4: Three simulations of geometric Brownian motion with 10% drift ($\mu = 0.10$), volatility 15% ($\sigma = 0.15$) and domain $t \in [0, 1]$ into 201 equal increments. Source code: Listing B.4

Comparing normal and modified Bernoulli distributed Y_i

Let us rewrite our geometric Brownian motion model as follows:

$$S_{n+1} = S_n + S_n \delta t \mu + S_n \sqrt{\delta t} \sigma Y_n \tag{1.2}$$

Now, again consider a time interval $t \in [0, T]$ where $T = N\delta t$. We know $S(0) = S_0$ and Equation 1.2 gives us an expression for $S(\delta t) = S_1, \ \ldots \ , S(T) = S_N$. We now let $\delta t \to 0 \implies N \to \infty$ and find a limiting equation for S_n. Equation 1.2 tells us at each δt interval we multiply the asset price by $1 + \mu\delta t + \sqrt{\delta t}\sigma Y_n$. Thus, we can write $S(T)$ as:

$$S(T) = S_0 \prod_{n=0}^{N-1} \left(1 + \mu\delta t + \sqrt{\delta t}\sigma Y_n\right)$$

Now, diving by S_0 and taking the log:

$$\log\left(\frac{S(T)}{S(0)}\right) = \sum_{n=0}^{N-1} \log\left(1 + \mu\delta t + \sqrt{\delta t}\sigma Y_n\right) \tag{1.3}$$

We are interested in the limit $\delta t \to 0$, so we would like to exploit the Taylor expansion $\log(1 + x) \approx x - \frac{x^2}{2} + \mathcal{O}\left(x^3\right)$, for small x. It is worth noting the quantity Y_n in Equation 1.2 is a RV, not a number. It can however be shown that if $\mathrm{E}\left[\, Y_n^2 \,\right]$ is finite, then we can treat Y_i as a number for our approximation [Hig04]. Continuing such that the log expansion remains valid, we obtain:

$$\log\left(\frac{S(T)}{S(0)}\right) \approx \sum_{n=0}^{N-1} \left(\mu\delta t + \sigma\sqrt{\delta t} Y_n - \frac{1}{2}\sigma^2 \delta t Y_n^2\right) + \mathcal{O}\left(\delta t^{\frac{3}{2}}\right) \tag{1.4}$$

Where terms of $\delta t^{\frac{3}{2}}$ or higher have been omitted.

Now, if we consider the Expectation and Variance of $\left(\mu\delta t + \sigma\sqrt{\delta t} Y_n - \frac{1}{2}\sigma^2 \delta t Y_n^2\right)$:

$$\begin{aligned}
\mathrm{E}\left[\,\mu\delta t+\sigma\sqrt{\delta t}Y_n-\frac{1}{2}\sigma^2\delta t Y_n^2\,\right] &= \mu\delta t+\sigma\sqrt{\delta t}\mathrm{E}\left[\,Y_n\,\right]-\frac{1}{2}\sigma^2\delta t\mathrm{E}\left[\,Y_n^2\,\right] \\
&= \mu\delta t+\sigma\sqrt{\delta t}\mathrm{E}\left[\,Y_n\,\right]-\frac{1}{2}\sigma^2\delta t\left(\mathrm{Var}\left[\,Y_n\,\right]+\mathrm{E}\left[\,Y_n\,\right]^2\right) \\
\mathrm{Var}\left[\,\mu\delta t+\sigma\sqrt{\delta t}Y_n-\frac{1}{2}\sigma^2\delta t Y_n^2\,\right] &= \sigma^2\delta t\mathrm{Var}\left[\,Y_n\,\right]-\frac{1}{4}\sigma^4\delta t^2\mathrm{Var}\left[\,Y_n^2\,\right] \\
&= \sigma^2\delta t\mathrm{Var}\left[\,Y_n\,\right]+\mathcal{O}\left(\delta t^{\frac{3}{2}}\right)
\end{aligned}$$

We can see that both our equations above depend on the expectation and variance of Y_n. For both our modified Bernoulli and normally distributed Y_i we can show that $\mathrm{E}\left[\,Y_i\,\right]=0$ and $\mathrm{Var}\left[\,Y_i\,\right]=1$. Thus, for both of our Brownian motion models we can rewrite our above equations as:

$$\begin{aligned}
\mathrm{E}\left[\,\mu\delta t+\sigma\sqrt{\delta t}Y_n-\frac{1}{2}\sigma^2\delta t Y_n^2\,\right] &= \mu\delta t-\frac{1}{2}\sigma^2\delta t \\
\mathrm{Var}\left[\,\mu\delta t+\sigma\sqrt{\delta t}Y_n-\frac{1}{2}\sigma^2\delta t Y_n^2\,\right] &= \sigma^2\delta t+\mathcal{O}\left(\delta t^{\frac{3}{2}}\right)
\end{aligned}$$

Since our realizations of Y_i are i.i.d. and our RHS contains no RVs, we can use the following:

$$\mathrm{E}\left[\,\sum_{i=0}^{N}f(x)\,\right]=N\cdot\mathrm{E}\left[\,f(x)\,\right] \qquad\qquad \mathrm{Var}\left[\,\sum_{i=0}^{N}f(x)\,\right]=N\cdot\mathrm{E}\left[\,f(x)\,\right]$$

Now using the Central Limit Theorem we can approximate Equation 1.4 as a normally distributed variable with mean $N\left(\mu\delta t-\frac{1}{2}\sigma^2\delta t\right)=T\left(\mu-\frac{1}{2}\sigma^2\right)$ and variance $N\left(\sigma^2\delta t+\mathcal{O}\left(\delta t^{\frac{3}{2}}\right)\right)=T\sigma^2$, since $N\delta t=T$. So we have the following approximation for large N and relatively small δt over the range $t\in[0,T]$.

$$\log\left(\frac{S(T)}{S(0)}\right)\sim\mathcal{N}\left(\mu-\frac{1}{2}\sigma^2,\ T\sigma^2\right) \tag{1.5}$$

We may be surprised to see an extra term in the mean of our normal distribution in Equation 1.5. One may expect our mean to simply be the time period multiplied by expected rate of return (drift), but this extra term arrives from the random noise in our Brownian model. This occurs because the path is not smooth, although continuous; the path is not everywhere differentiable. This is why it was required that we went to the quadratic term in our Taylor expansion of $\log\left(1+x\right)$ to get the correct linear approximation of $\log\left(S(T)/S(0)\right)$.

Our limiting continuous time expression for our asset model, using either a modified Bernoulli or normally distributed Y_i (or any distribution centred zero, variance equal to 1 and finite second moment) becomes:

$$S(T)=S_0\exp^{\left(\mu-\frac{1}{2}\right)T+\sigma\sqrt{T}Z},\quad\text{Where } Z\sim\mathcal{N}\left(0,1\right). \tag{1.6}$$

It is worth noting that there was nothing in our derivation that required us to use $t=0$ and $t=T$, we could in fact use any two values $S_i,S_j\ :\ i<j\in[0,N]$. With $0\le i<j\le N$, we have:

$$\log\left(\frac{S(j\delta t)}{S(i\delta t)}\right)\sim\mathcal{N}\left(\left(\mu-\frac{1}{2}\sigma^2\right)\left(j-i\right)\delta t,\ \left(\sigma^2\left(j-i\right)\delta t\right)\right) \tag{1.7}$$

Since our Y_i are i.i.d. across non-overlapping intervals, the normal RVs that describe these changes will be independent. Thus, $\forall i,j,k\in[0,N]\ :\ i<j<k$, we have:

$$\log\left(\frac{S(k\delta t)}{S(j\delta t)}\right)\sim\mathcal{N}\left(\left(\mu-\frac{1}{2}\sigma^2\right)\left(k-j\right)\delta t,\ \left(\sigma^2\left(k-j\right)\delta t\right)\right)$$

Which is independent of $\log\left(\frac{S(j\delta t)}{S(i\delta t)}\right)$.

Thus, we can describe our asset price modelled by our geometric Brownian motion over any sequence of time points using the following equation:

$$S_{n+1}=S_n\exp^{\left(\mu-\frac{1}{2}\sigma^2\right)(\delta t)+\sigma\sqrt{\delta t}Z_i},\quad\text{With i.i.d. } Z_i\sim\mathcal{N}\left(0,1\right). \tag{1.8}$$

So as it turns out, it does not matter how we decide to distribute Y_i so long as the $\mathrm{E}[Y_i] = 0, \mathrm{Var}[Y_i] = 1$ and $\mathrm{E}\left[Y_i^2\right]$ is finite; we end up with a statistically equivalent model, for large N and a small δt relative to the time interval $t \in [0, T]$.

The above is an important derivation to know when deciding how to correctly choose parameters in the binomial method of pricing options, as done in Subsection 1.4.4.

1.4 Pricing standard options

Since the holder of the contract is not obliged to exercise the contract at the expiry time, they do not hold any liability in the absence of a price to purchase the option. The problem then becomes, what is the correct price to charge the holder of the option to balance this inequality of liability.

1.4.1 Black-Scholes Model

$$\frac{\partial \mathbb{V}}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \mathbb{V}}{\partial S^2} + rS\frac{\partial \mathbb{V}}{\partial S} - r\mathbb{V} = 0 \tag{1.9}$$

The Black Scholes Equation is a partial differential equation developed by two economists, Fisher Black and Myron Scholes. This equation shows the value of an option as a function of time and underlying asset value. Thus, solving this equation with an appropriately chosen boundary conditions you can in some cases find an analytical solution to price an option. Most famously, Black and Scholes showed that with a change of variables transformation, it was possible to transform the Black-Scholes equation for a European put or call option into the commonly solved heat equation. Upon solving and transforming the variables back, they had provided an analytical solution for a European put and call option.

1.4.2 Deriving the Black-Scholes equation

The following derivation is taken from an unpublished report [LG19], which follows works from Hull [Hul12] which is in turn is based on the classic argument given in the original Black-Scholes paper [BS73]. To derive the Black-Scholes Equation we must first make three assumptions about the price of an option.

1. The stock price follows a stochastic process, meaning that we can not use historical price to predict its future movement.
2. The option pays no dividends to the owner.
3. Risk-free rate, r and percentage volatility, σ are known and constant.

We again assume that the price of an asset follows a geometric Brownian motion. However, we instead opt for the continuous form, that is:

$$\frac{dS}{S} = \mu dt + \sigma dW \tag{1.10}$$

Where W is a stochastic variable (Brownian motion). Note that W, and consequently its infinitesimal increment dW, represents the only source of uncertainty.

The payoff of an option $\mathbb{V}(S, t)$ at the exercise date, $t = T$ is known. To find the value of the option at an earlier time, we need to know how $\mathbb{V}$ changes as a function of S and t.

As we have assumed $\mathbb{V}(S, t)$ is a stochastic process and also dependent on time we can find the differential of $\mathbb{V}$ using Itō's lemma.

By Itō's lemma for two variables we have:

$$d\mathbb{V} = \left(\mu S\frac{\partial \mathbb{V}}{\partial S} + \frac{\partial \mathbb{V}}{t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \mathbb{V}}{\partial S^2}\right)dt + \sigma S\frac{\partial \mathbb{V}}{\partial S}dW \tag{1.11}$$

Now, consider a specific portfolio, the delta-hedge portfolio, consisting of the negative value of the option and $\frac{\partial \mathbb{V}}{\partial S}$ shares at time t. The value of the portfolio is then:

$$\Pi = -\mathbb{V} + \frac{\partial \mathbb{V}}{\partial S}S$$

Over a given time period, the total profit or loss from changes in the values of the holdings (the option and shares) is

$$\Delta\Pi = -\Delta\mathbb{V} + \frac{\partial \mathbb{V}}{\partial S}\Delta S \tag{1.12}$$

Now discretizing equations (1.10) and (1.11):

$$\Delta S = \mu S \Delta t + \sigma S \Delta W, \qquad \Delta \mathbb{V} = \left(\mu S \frac{\partial \mathbb{V}}{\partial S} + \frac{\partial \mathbb{V}}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \mathbb{V}}{\partial S^2} \right) \Delta t + \sigma S \frac{\partial \mathbb{V}}{\partial S} \Delta W$$

and substituting these into Equation 1.12, we obtain

$$\Delta \Pi = \left(- \frac{\partial \mathbb{V}}{\partial t} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \mathbb{V}}{\partial S^2} \right) \Delta t \tag{1.13}$$

Note that ΔW term has vanished. Thus, uncertainty has been eliminated, and the portfolio is effectively risk-less. The rate of return on this portfolio must be equal to the rate of return on any other risk-less instrument; otherwise, there would be opportunities for arbitrage. Now given the risk-free rate of return is r, we must have over the same given time period:

$$r \Pi \Delta t = \Delta \Pi \tag{1.14}$$

Now equating our two formulas for $\Delta \Pi$, equations (1.13) and (1.14)

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

Simplifying, we arrive at the celebrated Black-Scholes equation:

$$\frac{\partial \mathbb{V}}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 \mathbb{V}}{\partial S^2} + rS \frac{\partial \mathbb{V}}{\partial S} - r\mathbb{V} = 0$$

With the assumptions of the Black-Scholes model, this second order partial differential equation holds for any type of option as long as its payoff function depends only on the underlying asset value at the expiry date and its price function $\mathbb{V}$ is twice differentiable with respect to S and once with respect to t.

1.4.3 Risk-neutrality assumption

Before the publication of the Black-Scholes model, it was thought that the value of an option should be equal to it's expected payoff, for a European styled option; that is:

$$W\left(T, S(T)\right) = \mathrm{E}\left[\, e^{-rT} \mathbb{V}\left(S(T)\right) \right]$$

Where T is the option's expiry date, r the risk-free interest rate, $\mathbb{V}(t)$ the payoff function of the option and $S(T)$ is the underlying asset value at time t. Remembering back to our asset model, we know $S(T)$ is a random variable from Equation 1.6. We may consider our option value at any $t \in [0, T]$ as being:

$$W\left(t, S(T)\right) = e^{-r(T-t)} \mathrm{E}\left[\, \mathbb{V}\left(S(T) \mid S(t)\right) \right] \tag{1.15}$$

Which may be written explicitly using our asset model and properties of RVs. We shall leave in this implicit form for brevity. This may seem like a reasonable approach to value an option. However, this equation has made no attempt to mitigate risk through the use of hedging; nor has it been derived with the no arbitrage principle we established in subsubsection 1.2.2 in mind. More importantly our equation depends on our drift parameter μ.

The no-arbitrage principle tells us that there can only be one fair value for an option, and that is the one quoted in the market. If the value was higher or lower an opportunity for arbitrage would occur giving investors risk-free profit. This restricts the value of the option to the Black-Scholes level, it follows that since our discounted expected payoff method depends on μ; that it cannot be used to get a fair value.

However, as it turns out if we assume what is called risk-neutrality assumption; that is we let our drift parameter be equal to our risk-free interest, i.e. $\mu = r$. Then we arrive at the remarkable conclusion that our expected discounted payoff, Equation 1.15 satisfies our Black-Scholes PDE, Equation 1.9.

We summarize the risk-neutrality assumption as follows: regardless of the parameters σ and μ we believe to be correct in our asset model Equation 1.8. If we suppose that our drift μ is instead equal to our risk-free interest rate r than by using the discounted expected payoff we can obtain the Black-Scholes option value.

In the risk-neutral world we have that the expected price of an asset is just the same as investing the value of the initial asset risk-free, i.e. $\mathrm{E}\left[\, S(t) \right] = S_0 \exp\left(rt\right)$. This kind of process, one in which the expected future value is equal to

the initial value is called a martingale process. Through martingale theory one can use the risk-neutrality assumption to price options consistent with the no-arbitrage principle [Kri00]. This proves to be a powerful tool when valuing options using numerical methods, we will see this in our next model Subsection 1.4.4 and Section 1.5.

1.4.4 Binomial method

The binomial option pricing model (BOPM) provides a generalized discrete numerical method for valuing options. The model uses a binomial tree where at each discrete time-step the underlying asset goes up or down by a fixed (percentage) value. It was first proposed by William Sharpe in 1978 [Sha78], but formalized by Cox, Ross and Rubinstein in 1979 [CRR79]. Since then, many developments and improvements have been made; here we will follow closely to D. Higham work [Hig04].

Binomial model over a single time-step

We will first consider a one-step binomial model. In this model we have an underlying asset with current (initial) price, S_0. We propose in one time-step from $t = 0$ to $t = 1 = T$ that the price will either change to $S_T = uS_0$ or $S_T = dS_0$ where $d < u$; with probabilities p and $(1 - p)$ respectively. We write this as follows:

$$S_T = \begin{cases} uS_0 = S_{(1,1)} & \text{with } P\left(S_T = S_{(1,1)}\right) = p \\ dS_0 = S_{(1,0)} & \text{with } P\left(S_T = S_{(1,0)}\right) = (1-p) \end{cases}$$

The notation $S_{(i,j)}$ simply represents the value of the underlying asset at the i^{th} time-step with j up steps, note that the number of down steps is simply $i - j$.

We may also construct a lattice that describe the potential paths of the underlying asset value, this becomes useful when we increase the number of steps. Similarly, we construct a lattice which describes the value of the option given the corresponding value of the underlying asset. These are both shown in Figure 1.5

(a) Possible asset values for $t \in [0, T]$

(b) Possible option values for $t \in [0, T]$

Figure 1.5: Two binomial trees which represent the evolution of the underlying asset value, and the value of the option from $t = 0$ to $t = T$.

We know how to calculate the options value at $t = T$, this simply requires inputting the corresponding S_T values into the options payoff function; i.e. for a standard European put option with strike price, E expiring at $t = T$ the payoff function is such:

$$\mathbb{V}(S_T) = \max(E - S_T,\ 0)$$

Substituting our potential S_T tells us the possible values of the option at $t = T$, let $dS_0 < E < uS_0$:

$$X(S_T) = \begin{cases} 0 & \text{if } S_T = S_{(1,1)} \\ E - dS_0 & \text{if } S_T = S_{(1,0)} \end{cases}$$

We must now determine the value of $\mathbb{V}_{(0,0)}$, a natural method may be to calculate the discounted expected payoff of the option; that is the amount you would need to invest risk-free to guarantee $\mathrm{E}[\,\mathbb{V}(S_T)\,]$ at $t = T$; this depends on the probability p. However, as mentioned previously, doing so leads to an answer which is not consistent with our No-Arbitrage assumption. To price the option consistent with the No-Arbitrage assumption we must calculate the risk-neutral probability, p_* this is the probability in which our discounted expected payoff value is equal to the initial option value; becoming a martingale process. It can be shown that the same probability p_*, satisfies expected discounted asset value being equal to the initial asset value [Kri00], i.e.:

$$\mathrm{E}_{p_*}\left[\, e^{-r\delta t} S_{\delta t}\,\right] = e^{-r\delta t}\left[p_* S_{(1,1)} + (1-p_*)\, S_{(1,0)}\right] = S_0$$

Here $\delta t = (T-0) = T$ which is the change in time in one step, it is also worth noting that this value holds across all steps. It follows that we can calculate the risk-neutral probability, p_* using the following:

$$p_* = \frac{S_0 e^{r\delta t} - uS_0}{uS_0 - dS_0} = \frac{e^{r\delta t} - u}{u - d} \tag{1.16}$$

We can then use p_* to find the value of the option at time $t = 0$.

$$\mathrm{E}_{p_*}\left[\, e^{-rT} \mathbb{V}_{\delta t}\,\right] = e^{-rT}\left[p_* \mathbb{V}_{(1,1)} + (1-p_*)\, \mathbb{V}_{(1,0)}\right] = \mathbb{V}_{(0,0)}$$

Extending to multiple steps

Now extending our method over multiple time-steps, let expiry time $T = 2$ and $\delta t = 1$. We construct the following lattice

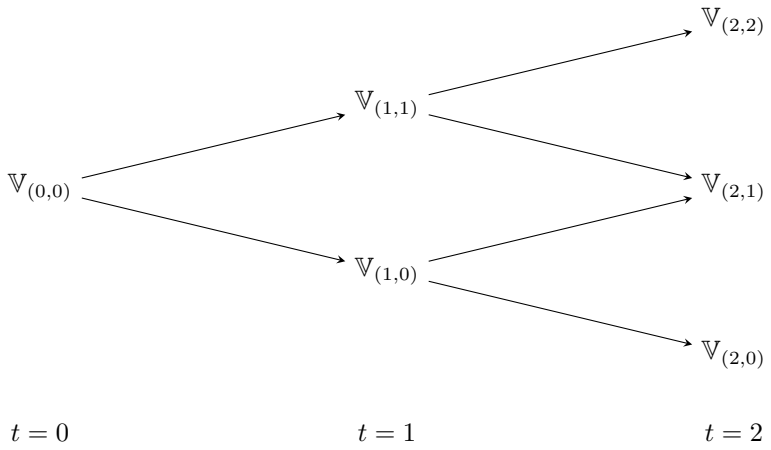

Figure 1.6: A binomial tree representing the possible option values from $t = 0$ to $t = 2$ with $\delta t = 1$

We can see at both nodes when time, $t = 1$, we essentially have two one-step binomial models. If we apply the one-step method to each, we can then use our calculated values to find our current option value.

It follows that we can do this for any number of steps, let expiry time $T = N$. Then we begin by calculating the payoff for each node in the final column, i.e. $\mathbb{V}_{(N,k)} = \mathbb{V}(u^k d^{N-k} S_0)$ at time $t = N$. Using these values we can then calculate the nodes at the previous time step, i.e. time $t = m = N - 1$:

$$\mathbb{V}_{(m,k)} = e^{-r\delta t}\left(p_* \mathbb{V}_{(m+1,k+1)} + (1-p_*)\, \mathbb{V}_{(m+1,k)}\right) \tag{1.17}$$

This can then be repeated for the next previous step, $t = m = N - 2$, and so forth. This is repeated until $m = 0$, at which point you will have calculated the initial option value.

Determining Parameters

Let us assume that you have determined appropriate values for drift, μ and percentage volatility, σ through use of historical data or expert speculation. We may ask how can these values be used to determine the value of our up step, u and or down step d.

Let us describe out underlying asset path using a Bernoulli distributed RV, β_i. Let $\beta_i = 1$ with when our asset steps up over a single time step from$(i-1)\delta t$ to $i\delta t$, and $\beta_i = 0$ when we step down. Hence, we can write out path as follows:

$$S(n\delta t) = S_0 u^{\sum_{i=1}^{n} \beta_i} d^{n - \sum_{i=1}^{n} \beta_i}$$

We rearrange this to the following:

$$\frac{S(n\delta t)}{S_0} = d^n \left(\frac{u}{d}\right)^{\sum_{i=1}^{n} \beta_i} \tag{1.18}$$

Then taking logs:

$$\log\left(\frac{S(n\delta t)}{S_0}\right) = n\log(d) + \sum_{i=1}^{n}\beta_i \log\left(\frac{u}{d}\right)$$

Now continuing as we did in Equation 1.4, whereby we calculated the expectation and variance of our log expression, using $\mathrm{E}[\beta_i] = p$, $\mathrm{Var}[\beta_i] = p(1-p)$ since β_i is a standard Bernoulli RV :

$$\begin{aligned}
\mathrm{E}\left[n\log(d) + \sum_{i=1}^{n}\beta_i\log\left(\frac{u}{d}\right)\right] &= n\log(d) + \log\left(\frac{u}{d}\right)\sum_{i=1}^{n}\mathrm{E}[\beta_i] \\
&= n\log(d) + np\log\left(\frac{u}{d}\right) \\
&= n\left[\log(d) + p(\log(u) - \log(d))\right] \\
&= n\left[p\log(u) + (1-p)\log(d)\right] \\
\mathrm{Var}\left[n\log(d) + \sum_{i=1}^{n}\beta_i\log\left(\frac{u}{d}\right)\right] &= \log\left(\frac{u}{d}\right)^2\sum_{i=1}^{n}\mathrm{Var}[\beta_i] \\
&= n\left[p(1-p)\log\left(\frac{u}{d}\right)^2\right]
\end{aligned}$$

Now, if we apply the Central Limit Theorem for large n and assume our underlying asset should match our Brownian motion equations. Then Equation 1.7 tells us the mean of $\log(S(n\delta t))/S_0$ must equal $\left(\mu - \frac{1}{2}\sigma^2\right)n\delta t$ and the variance be $\sigma^2 n\delta t$. Hence, we get the following equations:

$$\begin{aligned}
[p\log(u) + (1-p)\log(d)] &= \left(\mu - \frac{1}{2}\sigma^2\right)\delta t \\
\left[p(1-p)\log\left(\frac{u}{d}\right)^2\right] &= \sigma^2\delta t \\
\log(u) - \log(d) &= \sigma\sqrt{\frac{\delta t}{p(1-p)}}
\end{aligned} \tag{1.19}$$

Now, since we determine μ, σ and $\delta t = T/N$; it stands that we have three unknowns, u, d, p for two equations. It follows that we must then fix one of our unknowns, or determine a third equation. A popular choice is to fix $d = 1/u$ and consider that when we price our option we do not use parameter p, we do however use u and d in the calculation. The definition of our drift, μ tells us that it should be closely related to p. Thus, we can conclude that u and d should not dependent on μ, but instead solely on the percentage volatility σ. Whilst many solutions can exist we will use the following:

$$u = e^{\sigma\sqrt{\delta t}} \qquad\qquad d = e^{-\sigma\sqrt{\delta t}}$$

Which leads to the following expression for p:

$$p = \frac{1}{2} + \frac{1}{2\sigma}\left(\mu - \frac{1}{2}\sigma^2\right)\sqrt{\delta t}$$

Substituting these values into Equation 1.19 confirms their accuracy to $\mathcal{O}\left(\delta t^{\frac{3}{2}}\right)$ as required. It is worth noting this is not the only way to determine u, d, p from $\mu, \sigma, \delta t$. Some notable examples are: $u = \exp\left(\sigma\sqrt{\delta t} + (r - 1/2)\,\delta t\right), d = 1/u$ [Hig04] and $u = \exp(\delta t)\left(1 + \sqrt{\exp(\sigma^2\delta t) - 1}\right)$, $d = \exp(\delta t)\left(1 - \sqrt{\exp(\sigma^2\delta t) - 1}\right)$ [WHD95].

Overview of the BOPM

The BOMP is a very robust model allowing you to easily implement different payoff function into the algorithm, since you calculate values one node at a time. This makes it easy to implement break conditions, such as a so-called All or Nothing option; an option format where if a price of an underlying asset drops below an exercise price at any time $t \leq T$, it is worth nothing. It is for this reason the binomial method is a good choice to price American options since they can be exercised at any time $t \leq T$.

One of the biggest benefits is the relative simplicity of the model it is quite straight forward to programme an algorithm that implements the method without involving any complicated mathematics. However, a drawback to the method is that

for options in which the number of computations grow exponentially relative to the number of time-steps. i.e. For Asian options, we need to calculate the average off all different possible paths so as N becomes large, this becomes computationally unviable. Hence, it is required that we modify the method; we will explore this in Section 1.5.

1.5 Pricing Asian options

In this section we will explore several methods which can be used to price Asian options. Whilst an analytical solution exists for Asian options which use a geometric average [Zha97]; it not the case when we use the more popular arithmetic average. Asian options which use an arithmetic average must instead be solved using numerical approaches.

1.5.1 Pricing Asian options using BOPM

Let us again consider an asset with an initial value $S_0 = 10$, with a probability $p = 0.5$ of going up to uS_0 or down to dS_0 at every time-step $\delta t = 0.1$ with $u = 1.1,\ d = 1/u$. We can again show this in a binomial tree, however, we will also show the potential path dependent arithmetic averages that can be reached at each node. This can be seen in Figure 1.7.

Suppose $3\delta t = T$, then we are able to calculate the value of the option at $t = T$ for each possible path. We introduce the notation $S_{(i,j,k)}$ as being the asset (or option) value at the i time-step, with j up steps and k^{th} possible value, where $k_1 > k_2 > \ldots$. Thus, from figure Figure 1.7 $S_{(3,2,2)} = 10.50$.

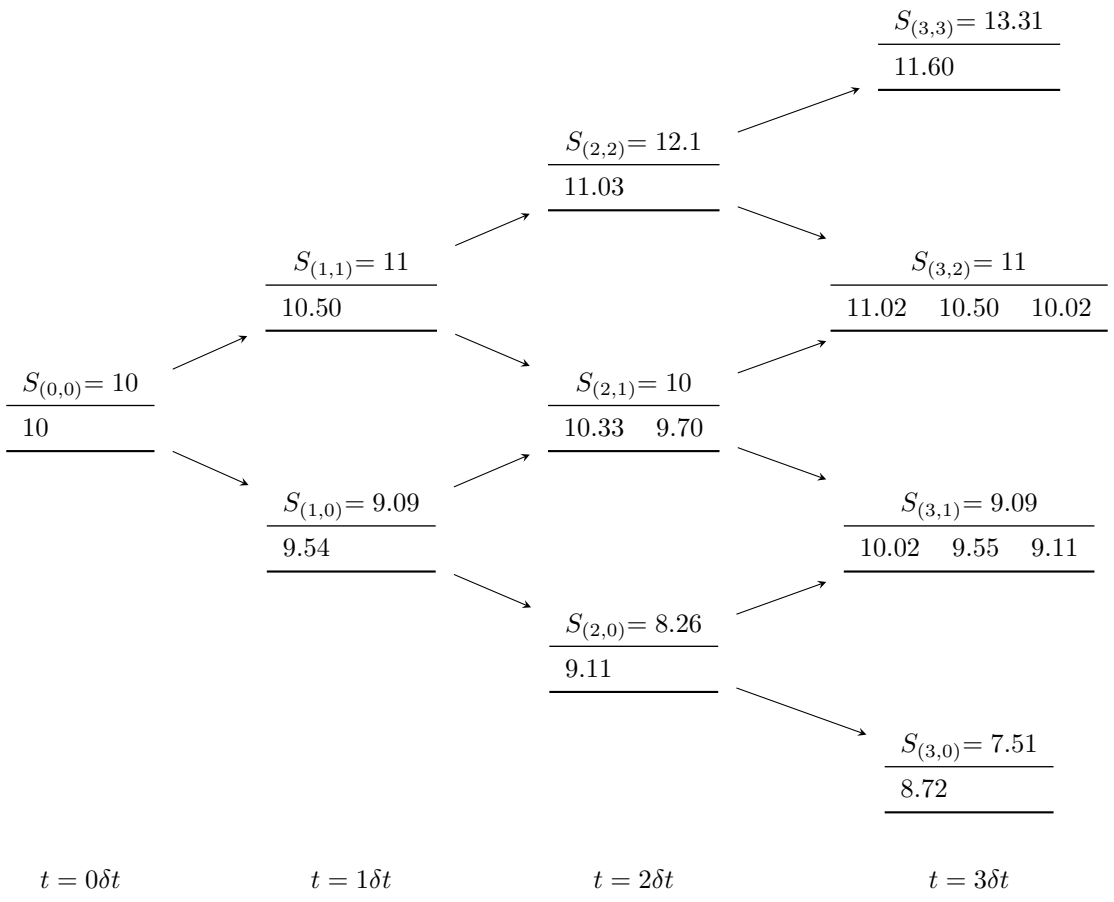

Figure 1.7: A binomial-tree representing the potential assets values with the corresponding path averages below. $S_0 = 10,\ u = 1.1,\ d = 1/u,\ \delta t = 0.1$

Let our option be a fixed-strike price Asian call option, in which the arithmetic average taken every $\delta t = 0.1$ is used as the spot price with strike price $E = 10$. Then using Figure 1.7 we can proceed to calculate our option value at $t = T$, we will calculate for both $\mathbb{V}_{(3,3,k)}$ and $\mathbb{V}_{(3,2,k)}$

$$\mathbb{V}_{(3,3,1)} = \max(13.31 - 10, 0) = 3.31$$

$$\mathbb{V}_{(3,2,1)} = 1.02$$
$$\mathbb{V}_{(3,2,2)} = 0.50$$
$$\mathbb{V}_{(3,2,3)} = 0.02$$

Now, as it stands we are able to use Equation 1.17 to work backwards and price $\mathbb{V}_{(2,2,1)}$. We just need to make a slight adjustment to the formula and select k_u such that the k_u^{th} option value is the value in which we arrive after an up step from $\mathbb{V}_{(i,j,k)}$; similarly k_d to arrive at the correct down step.

$$\mathbb{V}_{(i,j,k)} = e^{-r\delta t}\left(p_* \mathbb{V}_{(i+1,j+1,k_u)} + (1-p_*)\,\mathbb{V}_{(i+1,j,k_d)}\right) \tag{1.20}$$

This gives us the following equation for $\mathbb{V}_{(2,2,1)}$

$$\begin{aligned}\mathbb{V}_{(2,2,1)} &= e^{-r\delta t}\left(p_* \mathbb{V}_{(3,3,1)} + (1-p_*)\,\mathbb{V}_{(3,2,1)}\right)\\ &= e^{-r\delta t}\left(p_* 3.31 + (1-p_*)\,1.02\right)\end{aligned}$$

As we did with the standard BOPM we can repeat this process for all nodes and their respective path dependent averages, working backwards till we reach $\mathbb{V}_{(0,0,0)}$, giving us the present value of the option.

Difficulties with BOPM

As mentioned previously, the standard BOPM comes into problems when applied to Asian type options. Let us again consider a binomial tree to describe our asset value, then at each time-step (δt) our number of possible asset values increases by $\delta t + 1$. Thus, for N steps we have $\sum_{i=0}^{N} i + 1 = \frac{1}{2}(N+1)(N+2) \sim \mathcal{O}\left(N^2\right)$ different nodes. As seen in the binomial method, we have to calculate the value of the option at each corresponding node. For a standard option this is just one calculation per node, giving us $\approx N^2$ calculations. However, the value of an Asian option at each node depends on the path taken to reach it. Thus, to calculate the value at each node we must calculate for every potential path that can be taken; as for every deviation in the path taken our average will give a different value. We let node $(i,\ j)$ be the node reached at the i^{th} time-step with j up steps. A node with j up steps will consequently have $i-j$ down steps. Hence, the number of potential paths to reach node $(i,\ j)$ would simply be the number of permutations of j up steps and $i-j$ down steps; which can be calculated using $\binom{i}{j} = \frac{i!}{j!(i-j)!}$. If we were to consider all these paths then we would have one calculation per path, and thus the following number of calculations:

$$\sum_{i=1}^{N}\sum_{j=0}^{i}\binom{i}{j} = \sum_{i=1}^{N} 2^i = 2\left(2^N - 1\right) \sim \mathcal{O}\left(2^N\right)$$

So it is clear to see how this method becomes unviable as we increase N.

1.5.2 Hull-White model

Whilst not the first numerical solution to price Asian options, the method proposed by Hull and White [HW93] was the first to do so by using a binomial tree method. The authors showed that the method was faster than a Monte-Carlo Simulation and more accurate than the previously proposed log-normal approximation. Furthermore, a binomial tree method is similar to the standard BOPM, in that they are easily modified to allow for American and other variations to the option structure.

To overcome this problem, the model uses a list of so-called representative averages which is considerable smaller than the list which contains all the possible averages, the model would then interpolate between these values to approximate the option value.

Representative Averages

Hull and White decided admittedly arbitrarily, the representative averages would be of the form: $F_{(0,m)} = S(0)e^{hm}$, with h being some constant and m being a positive of negative integer. With our initial representative average being S_0 and at each time step we append an additional two averages.

Let us consider our previous binomial tree Figure 1.7, considering all of our maximum and minimum averages at time step 1, 10.50 and 9.54. We will set $h = 0.1$, it then follows that we can use $m = +1, -1$ to cover this range of averages. For our first time step then, our list of representative averages along with S_0 is:

$$S(0)e^{0.1} = 11.05 \qquad\qquad S(0)e^{-0.1} = 9.05$$

We can then calculate our maximum and minimum averages at the next time step, as follows: $\left(2 \cdot 10.50 + S_{(2,2)}\right)/3 = 11.03$ and $\left(2 \cdot 9.05 + S_{(2,0)}\right)/3 = 9.11$. Here, we use $m = +2\ -2$ along with our previous values to be our representative averages at our second time step; giving us:

$$S(0)e^{0.2} = 12.21 \qquad\qquad S(0)e^{-0.2} = 8.19$$

Now, considering our previous binomial tree Figure 1.7, but this time let $T = 2\delta t$. We will focus on node $S_{(1,1)}$ and the paths that come from it. We will again show the value of the asset at that node; but below we will this time opt for showing our representative averages and the corresponding values of a fixed-price Asian put option with $E = 10$, $\delta t = 0.1$ and $r = 0.3$. For $t = T$ we simply use $\max\left(F_{(T,i)} - E, 0\right)$ to calculate our option value, where $F_{(T,i)}$ are our representative averages at time T.

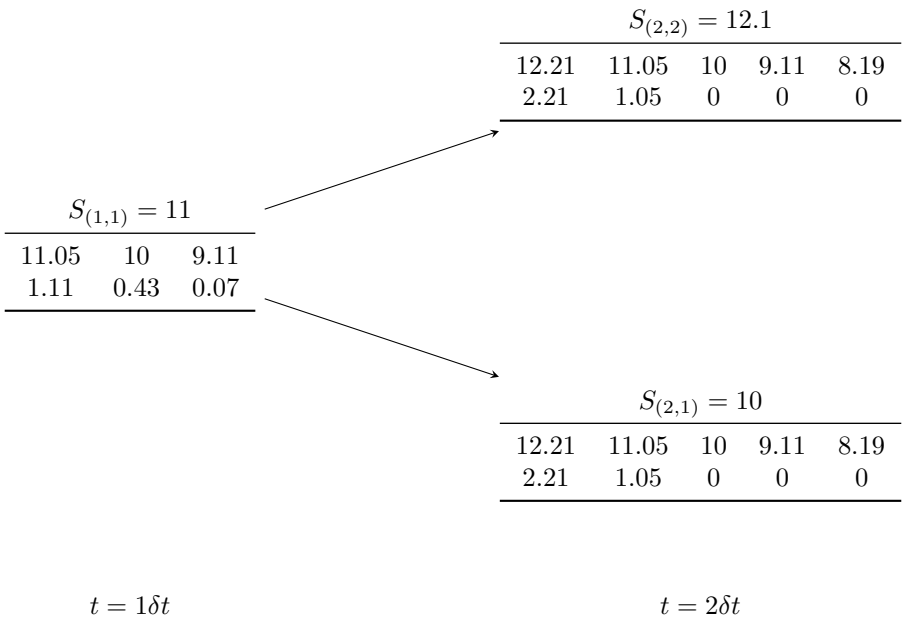

Figure 1.8: A binomial-tree representing the potential assets values with the corresponding possible averages below.

To calculate the option values show at $S_{(1,1)}$ we do for the following. We consider being at the representative average 11.05, if we were to jump up, our new average would become $(2 \cdot 11.05 + 12.1)/3 = 11.4$ we then interpolate between our two closest representative averages to get $\mathbb{V}_{(2,2,k_u)}$, similarly for $\mathbb{V}_{(2,1,k_d)}$:

$$\mathbb{V}_{(2,2,k_u)} = 1.05\left(\frac{12.21 - 11.4}{12.21 - 11.05}\right) + 2.21\left(\frac{11.4 - 11.05}{12.21 - 11.05}\right) = 1.4$$
$$\mathbb{V}_{(2,2,k_d)} = 1.05\left(\frac{12.21 - 10.7}{12.21 - 11.05}\right) + 2.21\left(\frac{10.7 - 11.05}{12.21 - 11.05}\right) = 0.7$$

Allowing us to calculate our $\mathbb{V}_{(1,1,1)}$ with Equation 1.20:

$$\begin{aligned}\mathbb{V}_{(1,1,1)} &= e^{-r\delta t}\left(p_* \mathbb{V}_{(2,2,k_u)} + (1 - p_*)\,\mathbb{V}_{(2,1,k_d)}\right) \\ &= 1.11\end{aligned}$$

This is then repeated for all representative averages for all nodes at that time step before moving backwards to repeat for the previous time step. This is then repeated till $\mathbb{V}_{(0,0,0)}$ is reached, in which the option has been valued for the current time.

Considerations

It stands that by using the Hull-White model we no longer need to consider all the possibly paths, we only consider the list of representative averages. We know that as the number of steps N increases we increase our number of representative averages by two. Meaning our number of computations as N increases is again linear, $\sim \mathcal{O}(N)$. This means that this method is computationally viable for Asian options with large N.

Hull and White showed in their paper that the method is very sensitive to the selection of h and m when calculating representative averages. Whilst you can simply select m deterministically by letting it be the smallest integer value that encompasses the range of averages; it still leaves h to be decided.

1.5.3 Costabile adjusted binomial method

Since the introduction of the first binomial tree method for solving Asian options by Hull and White; many similar schemes followed, each method used different methods to produce the list of representative averages [BP96; CJV97; Kla01]. Costabile et al. [CMR06] presented a binomial method of their own in 2006. The method also followed the same principles as the Hull and White model, in that it reduces the number of total averages to a representative list. The authors however present a method which rather than simulating the averages, instead realizes effective averages at each node.

What this means, is instead of producing an equation which estimates the representative average on parameters; which can lead to poor valuing and/or slow calculating when selected poorly. Instead, the representative averages are a subset of true averages which are selected through an algorithm we will describe below.

Selecting representative averages

We used similar notation as previously in that $S_{(i,j)}$ represents the asset value at time i, after j up steps and $i-j$ down steps, i.e.

$$S_{(i,j)} = S_0 u^j d^{i-j}$$

Then for each node we consider a set of representative averages, denoted by $A_{(i,j,k)}$. Let us consider the maximum and minimum averages that can realized at any node $(i,\ j)$. It is the case that for any node $(i,\ j)$ that the maximum average will be from the path which takes j up steps followed by $i-j$ down steps. For the minimum average it is the path which takes $j-i$ down steps followed by j up steps. We will call these two paths $\tau_{\max}(i,\ j)$ and $\tau_{\min}(i,\ j)$; these paths for node $(3,\ 2)$ are highlighted in Figure 1.9 as the red and blue paths respectively. We assign our maximum average $A_{(i,j,1)} = A_{\max}(i,\ j)$ and minimum average as last in our list $A_{(i,j,1+j(i-j))} = A_{\min}(i,\ j)$, we will see why $\max(k) = 1 + j(i-j)$ shortly.

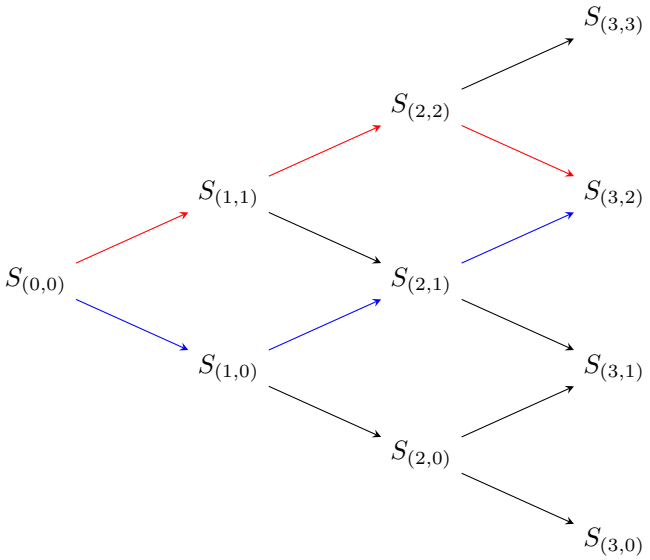

Figure 1.9: A binomial tree representing the possible asset values from $t = 0$ to $t = 3\delta t$; with $\tau_{\max}(3,\ 2)$ and $\tau_{\min}(3,\ 2)$ highlighted in red and blue respectively.

We calculate the other representative averages in the following way, let $S_{\max}(i,j,k)$ be the highest asset value reached on the trajectory that produces $A_{(i,j,k)}$ that is not along the path $\tau_{\min}(i,\,j)$, then:

$$A_{(i,j,k+1)} = A_{(i,j,k)} - \frac{1}{i+1}\left[S_{\max}(i,j,k) - S_{\max}(i,j,k)\,d^2\right]$$

This means that we find out $k^{\text{th}}+1$ average from our k^{th} average by simply substituting the highest value reached in the k^{th} path not belonging to $\tau_{\min}(i,\,j)$ with $S_{\max}(i,j,k)\,d^2$. Since trajectory $\tau_{\min}(i,\,j)$ produces the minimum average $A_{\min}(i,\,j)$. Starting from $\tau_{\max}(i,\,j)$ it is clear to see that we reach $\tau_{\min}(i,\,j)$ after $j\,(i-j)$ substitutions, hence our total number of representative averages at node $i,\,j$ is $1+j\,(i-j)$.

Instead of considering the path, we can instead use the following formulae for our maximum and minimum representative averages:

$$A_{(i,j,1)} = A_{\max}(i,\,j) = \frac{1}{i+1}\left(\sum_{h=0}^{j} S_0 u^h + \sum_{h=0}^{i-j-1} S_0 u^{h+2j-i}\right)$$
$$A_{(i,j,1+j(i-j))} = A_{\min}(i,\,j) = \frac{1}{i+1}\left(\sum_{h=0}^{i-j} S_0 d^h + \sum_{h=0}^{j-1} S_0 d^{i-2j+h}\right)$$

Once all representative averages have been constructed we can again use Equation 1.20 to work backwards through our binomial tree to reach the current value of the option at $t=0$, again interpolating when needed as done in the Hull-White model.

Analytical solution for geometric average Asian options

1.6 Monte-Carlo methods

The Monte-Carlo methods describes a method in which instead of trying to solve a statistical problem either analytically or through the use of numerical approximations; but rather, run multiple simulations and determine the outcome in a deterministic manner. We know that, by the law of large numbers for a series of independently identically distributed RVs $X_1, X_2, \ldots X_n$ that:

$$\mathrm{E}[\,X_i\,] = \frac{1}{n}\sum_{i=1}^{n} X_i \qquad\qquad \mathrm{Var}[\,X_i\,] = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \mathrm{E}[\,X_i\,])^2$$

Thus, if we were able to simulate realizations of X, for large n we can simply use the above formulae to approximate the expectation and variance.

Valuing options using Monte-Carlo methods

Recalling back to risk-neutrality assumption, we asserted that if we were to let $\mu = r$, then we are able to value options by simply taking the expectation of the payoff function $\mathrm{E}\left[\,e^{-r(T-t)}\mathbb{V}(S(t))\,\right]$. Remembering that our asset price $S(t)$ is modelled by Equation 1.6 and using $\mu = r$, we write the current value of a European option as:

$$e^{-rT}\mathrm{E}\left[\,\mathbb{V}\left(S_0 \exp^{\left(r-\frac{1}{2}\right)T+\sigma\sqrt{T}Z}\right)\right], \quad \text{Where } Z \sim \mathcal{N}(0,1)\,.$$

Thus, we can simulate many realizations of $\mathbb{V}(S(t))$ and take the mean of them all to find an approximation for our option value. It is worth noting that to implement a Monte-Carlo method to price Asian option, it would be necessary to compute realizations across multiple times according to averaging function of the option.

The Monte-Carlo method is often used at the "true" value when an analytical solution does not exist, of course providing that a large amount of realizations is used. Whilst providing highly accurate approximations, the method by nature is very computationally expensive. Despite that becoming less of an issue with computing power increasing globally, it still stands that more efficient methods are desirable. However, the simplicity of Monte-Carlo methods is allows for easy modification to apply to many kinds of options making the method very robust.

Chapter 2: Method

In this chapter we will discuss how we can computationally implement the Costabile pricing model with the use of MATLAB, we will also briefly outline how we implement a Monte-Carlo model. We will then compare the results of our implementation focusing on the accuracy and efficiency.

2.1 Costabile method

As mentioned in the previous chapter, the Costabile method relies on a subset of representative averages which are actually a subset of real averages realized at each node. Costabile et al. propose an algorithm to calculate the list of representative averages at each node. Whilst there is an explicit formula for the first and last averages, namely $A_{\max}(i,\, j)$ and $A_{\min}(i,\, j)$ respectively. The averages in between rely on a recursive method which first involves considering the path taken to reach node, finding the highest asset value reached that is not on the minimum path, $\tau_{\min}(i,\, j)$. After saving this value, the corresponding node to that value is multiplied by d^2, in which we find the next highest value not on the $\tau_{\min}(i,\, j)$. This is repeated until the path being considered becomes $\tau_{\min}(i,\, j)$, at which point you with have a set of $S_{\max}(i, j, k)$ values which can be used to produce the list of representative averages.

2.1.1 Finding $S_{\max}(i, j, k)$ values

Whilst the aforementioned method produces a set of representative averages without needing any additional parameters unlike the binomial methods which predate it; it becomes very computationally expensive for large binomial trees. Consider that for every representative average you need to first produce a corresponding path to reach $A_{\max}(i,\, j)$, search along said path to find the $S_{\max}(i, j, k)$ value, replace the value and then search along the path again, then repeat $j(i-j)$ times for each node.

Alternative approach

Whilst we can of course programme a path search and replace algorithm as shown in Listing A.4 lines: 48–75; however, here we propose an alternative. If we were to look at the lists of $S_{\max}(i, j, k)$ produced by the search and replace method, it is clear to see that a pattern emerges. We begin by stating that for any node $(i,\, j)$ such that $i = j$ or $j = 0$, then there will only be one $S_{\max}(i, j, 1) = S_0 u^j d^{i-j}$.

For all other nodes we have $k \in [1, j\,(i-j)]$. Intuitively, we know our first $S_{\max}(i, j, 1) = S_0^j$, it follows that our next value would then be $S_{\max}(i, j, 2) = S_0^{j-1}$. Upon investigating, if we set parameter $z = \min(j, i-j)$ then it turns out our $S_{\max}(i, j, k)$ values descend from $S_{\max}(i, j, 1) = S_0^j$ in repeated chunks which increase in length by one; until the repeated chunk length is equal to z. Then repeated chunks of length z are repeated ensuring enough space at the end of list to reduce the length back to one.

Take for example node $(7, 4)$, here $z = 3$ meaning we start at S_0^j, then the next two values are S_0^{j-1} and the three after that are S_0^{j-2}. Since our repeated chunk length is now equal to z we continue decreasing our value, but our chunk stays off length z. The list continues in this fashion ensuring enough room at the end of the list to decrease the chunk length to equal one. Figure 2.1 shows all $S_{\max}(7, 4, k)$ values:

Whilst this may seem quite complicated, it does make sense when you consider how the path moves when following the algorithm proposed by Costabile. It is also quite simple to implement into MATLAB, allowing you to calculate all required $S_{\max}(i, j, k)$ values by only considering S_0, u, d and $i,\, j$. We proceed as follows (code shown in Listing A.1 lines: 49–77):

We begin by calculating the size that our vector will be, $n = j\,(i-j)$. If $n = 1$ we simply set the one $S_{\max}(i, j, 1)$ value and proceed to the next node. We then calculate our parameter mentioned earlier $z = \min(j,\, i-j)$. We can now determine the number of unique values η, that occur in our $S_{\max}(i, j, k)$ vector with the following formula:

$$\eta = \frac{n}{z} + z - 1$$

Now, with η we can calculate the first and last $S_{\max}(i, j, k)$ values

$$S_{\max}(i, j, 1) = S_0 u^j \qquad\qquad S_{\max}(i, j, j(i-j)) = S_0 u^{j-\eta+1}$$

$S_{\max}(7,4,k)$	
k = 1	S_0u^7
k = 2	S_0u^6
k = 3	S_0u^6
k = 4	S_0u^5
k = 5	S_0u^5
k = 6	S_0u^5
k = 7	S_0u^4
k = 8	S_0u^4
k = 9	S_0u^4
k = 10	S_0u^3
k = 11	S_0u^3
k = 12	S_0u^2

Table 2.1: All $S_{\max}(7,4,k)$ values to be used in the calculation of our representative averages.

We then fill our vector from both the top and bottom, doing so in one for loop that fills in chunks that increase in length and decrease (increase) in value from the top (bottom); until current chunk length is equal to $z-1$. We then use another for loop which fills in the middle chunks of length z again decreasing value after every iteration. After which, we have successfully filled our $S_{\max}(i,j,k)$ vector for node $(i,\ j)$, this is then repeated for all nodes; allowing us to calculate the representative averages using the formulae shown in the Costabile paper.

Useful mentions on the Costabile method

It is certainly worth noting that the list of representative averages are sorted from highest to lowest. Keeping this in mind allows us to implement a basic but very fast binary search algorithm; when searching for k_u and k_d in our list of representative averages. This proves to be much faster than the find function included in MATLAB, our implementation of the binary search is shown in Listing A.2. It also stands that you can see some marginal gains from implementing your own interpolation function, in our implementation we opted for a simple linear interpolation which is shown to be sufficient, especially for small δt.

2.2 Monte-Carlo method

As mentioned previously, to extend the Monte-Carlo method to the price Asian option we must instead consider our asset module over $N\delta t$ time steps. This is so we can realize a simulated path the asset value took from $t=0$ to $t=N\delta t=T$. We then repeat this many times allowing us to utilize the law of large numbers and approximate the expectation of $\mathbb{V}(S(T))$.

We do this programmatically in MATLAB as follows: We first simulate $M\times N$ realizations of $Z\sim\mathcal{N}(0,\ 1)$. Then using a for loop which iterates forward in time by δt per iteration which calculates the next value off the path using Equation 1.8 for a vector of length M:

$$\vec{S}_{n+1}=\vec{S}_n\exp\left(\left(\mu-\frac{1}{2}\sigma^2\right)(\delta t)+\sigma\sqrt{\delta t}\vec{Z}_i\right),\quad \text{With i.i.d. } \vec{Z}_i\sim\mathcal{N}(0,1)$$

Upon the for loop finishing, we are left with a matrix of dimensions $(M\times N)$ in which every row contains a realized path for our asset value. All that is left is for us to take the expectation across every row, giving us M path averages A_i. Then using our strike price we can calculate the pay-off function for each realized path using $\mathbb{V}(E,A_i)$. Finally, we use the law of large numbers and take the expectation giving us an approximation for $\mathrm{E}[\,\mathbb{V}(E,A_i)\,]$, thus our option value is this value times the discounted factor e^{-rT}. Our implementation can be seen in Listing A.3.

We summarize as follows:

$$\mathbb{V}_0=e^{-rT}\frac{1}{M}\sum_{i=1}^{M}\mathbb{V}(E,A_i)$$

Chapter 3: Results

The results shown in Figure 3.1 is a comparison between our Costabile method using path search, and the alternative approach we previously outlined. Our parameters are as follows: $S_0 = 50, T = 1, r = 0.1, \sigma = 0.3$ with our option being an arithmetic Asian call option. Our table shows the alternative approach minus the path approach. We can see that for all values the difference between the two approaches is miniscule, thus validating our approach.

Row	E = 40	E = 45	E = 50	E = 55	E = 60
N = 10	0	0	0	0	0
N = 20	5e-07	5e-07	6e-07	6e-07	6e-07
N = 30	-0	-0	-0	-0	0
N = 40	6e-07	1e-07	-1e-07	-2e-07	-2e-07
N = 50	7e-07	1e-07	-1e-07	-1e-07	-1e-07

Figure 3.1: Here is a table which shows the difference between the alternative approach and the path search approach for varying strike price and number of steps

3.1 Accuracy

Next we will compare our model to the results published in the Costabile paper [CMR06], specifically table 3, and table 4. Our results are shown in Figure 3.2 and Figure 3.3 respectively. We can see that our results are very close to the results published by Costabile et al. especially in Figure 3.2, however there are some discrepancies in Figure 3.3, see $N = 70$.

Row	Alternative Costabile	Published Values	Monte-Carlo
N = 10	11.5244	11.5276	11.5187
N = 15	11.534	11.5348	11.5129
N = 20	11.5382	11.5384	11.5409
N = 30	11.542	11.5413	11.5484
N = 40	11.5439	11.5302	11.5545
N = 50	11.545	11.5449	11.5697
N = 60	11.5458	11.5458	11.5495
N = 70	11.5463	11.5463	11.5684
N = 80	11.5467	11.5467	11.5308
N = 90	11.547	11.547	11.5488

Figure 3.2: Here is a table which compares our alternative Costabile method against the published results in the original paper by Costabile et al. [CMR06] Parameters are such: $S_0 = 50, E = 40, T = 1, r = 0.1, \sigma = 0.3$ and the option is an arithmetic Asian call option

Row	Alternative Costabile	Published Values	Monte-Carlo
N = 10	28.3999	28.3491	28.2553
N = 15	28.3844	28.3687	28.22
N = 20	28.3804	28.2439	28.266
N = 30	28.3844	28.3478	28.3414
N = 40	28.3877	28.3866	28.4193
N = 50	28.3905	28.3899	28.3956
N = 60	28.3925	28.392	28.3984
N = 70	28.394	26.8899	28.3402
N = 80	28.3952	28.3934	28.4854
N = 90	28.3962	28.3875	28.4545

Figure 3.3: Here is a table which compares our alternative Costabile method against the published results in the original paper by Costabile et al. [CMR06] Parameters are such: $S_0 = 100, E = 100, T = 5, r = 0.1, \sigma = 0.5$ and the option is an arithmetic Asian call option

However, upon checking the values against our implemented path search Costabile method we arrive at the same discrepancies see Figure 3.4. This is quite unexpected and would require more research.

Row	Path Costabile	Published Values	Monte-Carlo
N = 10	28.3999	28.3491	28.3139
N = 15	28.3844	28.3687	28.3864
N = 20	28.3804	28.2439	28.2988
N = 30	28.3844	28.3478	28.3052
N = 40	28.3877	28.3866	28.4027
N = 50	28.3905	28.3899	28.3744
N = 60	28.3925	28.392	28.3155
N = 70	28.394	26.8899	28.4239
N = 80	28.3952	28.3934	28.401
N = 90	28.3962	28.3875	28.368

Figure 3.4: Here is a table which compares our path search Costabile method against the published results in the original paper by Costabile et al. [CMR06] Parameters are such: $S_0 = 100, E = 100, T = 5, r = 0.1, \sigma = 0.5$ and the option is an arithmetic Asian call option

3.2 Efficiency

The follow table Figure 3.5 shows the average speed taken for a variety of N values. We can see that whilst both methods give a similar time for small N as it increases a big improvement starts to materialize.

Row	Alternative Costabile	Path Costabile
N = 10	0.0209	0.0236
N = 15	0.0312	0.0399
N = 20	0.0463	0.0886
N = 30	0.1296	0.3667
N = 40	0.3532	1.0777
N = 50	0.8621	2.6479

Figure 3.5: Here is a table which compares our the speed of our alternative Costabile method against the traditional path search method, with execute time shown in seconds. Parameters are such: $S_0 = 50, E = 40, T = 1, r = 0.1, \sigma = 0.3$ and the option is an arithmetic Asian call option

Chapter 4: Conclusion

In conclusion, we can quite evidently see a big efficiency improvement to the Costabile method by taking an alternative approach to filling the $S_{\max}(i, j, k)$ vector. However, from our comparison of the results we found through our implementation to that off the original paper by Costabile et al. there does appear to be some unexpected results which at present time cannot be explained. Due to the nature of the Costabile method, it would be quite simple to modify the pay-off function to account for American style Asian options, an area I would like to look into in future research. Furthermore, it stands to reason that with adjustments to the representative average formula proposed by Costabile; it should be possible to adjust the method to be able to price geometric priced options. This would permit the ability to compare the results to that off the analytical solution.

References

Books and Academic Sources

[Ari77] Aristotle. *Aristotle's Politics I*. Aristotle's Politics Book 1. Longmans, Green, and Company, 1877. URL: https://books.google.co.uk/books?id=NvZCAQAAMAAJ.

[Bac07] Louis Bachelier. *Louis Bachelier's Theory of Speculation*. Ed. by Mark Davis and Alison Etheridge. Note: Translated reprint. Princeton: Princeton University Press, 2007. ISBN: 9781400829309. DOI: doi:10.1515/9781400829309.

[BP96] Jérôme Barraquand and Thierry Pudet. "Pricing of American path-dependent contingent claims". In: *Mathematical Finance* 6.1 (1996), pp. 17–51. DOI: 10.1111/j.1467-9965.1996.tb00111.x.

[BS73] Fischer Black and Myron Scholes. "The pricing of options and corporate liabilities". In: *Journal of political economy* 81.3 (1973), pp. 637–654. DOI: 10.1086/260062.

[CGM04] Sugato Chakravarty, Huseyin Gulen, and Stewart Mayhew. "Informed Trading in Stock and Option Markets". In: *The Journal of Finance* 59.3 (2004), pp. 1235–1257. DOI: 10.1111/j.1540-6261.2004.00661.x.

[CJV97] Prasad Chalasani, Somesh Jha, and Ashok Varikooty. *Accurate approximations for European-style Asian options*. Tech. rep. CARNEGIE-MELLON UNIV PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1997. DOI: 10.21314/JCF.1998.017.

[Cho+22] Jaehyuk Choi et al. "A Black–Scholes user's guide to the Bachelier model". In: *Journal of Futures Markets* 42.5 (2022), pp. 959–980. DOI: 10.1002/fut.22315.

[Coo64] P.H. Cootner. *The Random Character of Stock Market Prices*. Risk classics library. M.I.T. Press, 1964. ISBN: 9781899332847. URL: https://books.google.com.au/books?id=7XGvQgAACAAJ.

[CMR06] Massimo Costabile, Ivar Massabó, and Emilio Russo. "An adjusted binomial model for pricing Asian options". In: *Review of Quantitative Finance and Accounting* 27.3 (2006), pp. 285–296. DOI: 10.1007/s11156-006-9432-9.

[CRR79] John C. Cox, Stephen A. Ross, and Mark Rubinstein. "Option pricing: A simplified approach". In: *Journal of Financial Economics* 7.3 (1979), pp. 229–263. ISSN: 0304-405X. DOI: 10.1016/0304-405X(79)90015-1.

[Das11] M. Dash. *Tulipomania: The Story of the World's Most Coveted Flower and the Extraordinary Passions it Aroused*. Orion, 2011. ISBN: 9781780220574. URL: https://books.google.co.uk/books?id=AAX3nngAfjQC.

[Dur19] R. Durrett. *Probability: Theory and Examples*. Cambridge series on statistical and probabilistic mathematics. Cambridge University Press, 2019. ISBN: 9781108591034. URL: https://books.google.com.au/books?id=OQDoxQEACAAJ.

[Fam65] Eugene F. Fama. "Random Walks in Stock Market Prices". In: *Financial Analysts Journal* 21.5 (1965), pp. 55–59. DOI: 10.2469/faj.v21.n5.55.

[Gro75] Steven D Grossman. "An Application of communication theory to financial statement analysis". In: *Financial Review* 10.1 (1975), pp. 12–20. DOI: 10.1111/j.1540-6288.1975.tb00915.x.

[Hig04] D.J. Higham. *An Introduction to Financial Option Valuation: Mathematics, Stochastics and Computation*. An introduction to financial option valuation: mathematics, stochastics and computation v. 13. Cambridge University Press, 2004. ISBN: 9780521547574. URL: https://books.google.to/books?id=lEM%5C_ruhAnDwC.

[Hu14] Jianfeng Hu. "Does option trading convey stock price information?" In: *Journal of Financial Economics* 111.3 (2014), pp. 625–645. ISSN: 0304-405X. DOI: 10.1016/j.jfineco.2013.12.004.

[Hul12] J. Hull. *Options, Futures, and Other Derivatives*. Prentice Hall, 2012. ISBN: 9780132164948. URL: https://books.google.com.au/books?id=LzBTpwAACAAJ.

[HW93] John C Hull and Alan D White. "Efficient Procedures for Valuing European and American Path-Dependent Options". In: *The Journal of Derivatives* 1.1 (1993), pp. 21–31. ISSN: 1074-1240. DOI: 10.3905/jod.1993.407869.

[JP06] Franck Jovanovic and Geoffrey Poitras. "A 19th Century Random Walk: Jules Regnault and the Origins of Scientific Financial Economics". In: (Jan. 2006). eprint: https://www.researchgate.net/publication/280748880_A_19th_Century_Random_Walk_Jules_Regnault_and_the_Origins_of_Scientific_Financial_Economics.

[KH53] M. G. Kendall and A. Bradford Hill. "The Analysis of Economic Time-Series-Part I: Prices". In: *Journal of the Royal Statistical Society. Series A (General)* 116.1 (1953), pp. 11–34. ISSN: 00359238. URL: http://www.jstor.org/stable/2980947 (visited on 12/09/2022).

[Kit+15] Yakim Kitanov et al. "Are risk free government bonds risk free indeed". In: *Economy & Business* 9 (2015), pp. 523–530. eprint: https://www.scientific-publications.net/get/1000012/1440160097559642.pdf.

[Kla01] Timothy Klassen. "Simple, fast and flexible pricing of Asian options". In: *Final version in: Journal of Computational Finance* 4.3 (2001), pp. 89–124. DOI: 10.21314/JCF.2001.067.

[Kri00] M.P. Kritzman. *Puzzles of Finance: Six Practical Problems and Their Remarkable Solutions*. Wiley Investment. Wiley, 2000. ISBN: 9780471246572. URL: https://books.google.com.au/books?id=FvYlEAAAQBAJ.

[LG19] W. Gregory L. Luderman R. Deeley and J. Gill. "The Finite Difference Method in Corporate Finance — The Black-Scholes Equation". Unpublished report. Mar. 2019.

[Lam77] J. Lamperti. *Stochastic Processes: A Survey of the Mathematical Theory*. Applied mathematical sciences. Springer-Verlag, 1977. ISBN: 9783540902751. URL: https://books.google.com.au/books?id=Pd4cvgAACAAJ.

[Li21] Keming Li. "The effect of option trading". In: *Financial Innovation* 7.1 (2021), pp. 1–32. DOI: 10.1186/s40854-021-00279-5.

[Mar02] J.W. Markham. *A Financial History of the United States Volume III (1970-2001) From the Age of Derivatives to the New Millennium*. A Financial History of the United States. M.E. Sharpe, 2002. ISBN: 9780765607300. URL: https://books.google.co.uk/books?id=YRjmQLOscGoC.

[MP16] R David McLean and Jeffrey Pontiff. "Does academic research destroy stock return predictability?" In: *The Journal of Finance* 71.1 (2016), pp. 5–32.

[Mor+09] Esteban Moro et al. "Market impact and trading profile of hidden orders in stock markets". In: *Physical Review E* 80.6 (2009), p. 066102. DOI: 10.1103/PhysRevE.80.066102.

[NNT13] Vic Naiker, Farshid Navissi, and Cameron Truong. "Options trading and the cost of equity capital". In: *The Accounting Review* 88.1 (2013), pp. 261–295. DOI: 10.2308/accr-50275.

[PP06] Jun Pan and Allen M. Poteshman. "The Information in Option Volume for Future Stock Prices". In: *The Review of Financial Studies* 19.3 (Feb. 2006), pp. 871–908. ISSN: 0893-9454. DOI: 10.1093/rfs/hhj024.

[Pea+10] Philip Pearle et al. "What Brown saw and you can too". In: *American Journal of Physics* 78.12 (2010), pp. 1278–1289. DOI: doi/10.1119/1.3475685.

[Poi08] Geoffrey Poitras. "The Early History of Option Contracts". In: *Vinzenz Bronzin's Option Pricing Models: Exposition and Appraisal* (Jan. 2008). DOI: 10.1007/978-3-540-85711-2_24.

[Reg63] J. Regnault. *Calcul des chances et philosophie de la bourse*. Pilloy, 1863. URL: https://books.google.com.au/books?id=Lpr1gh7QJjIC.

[Sha78] W.F. Sharpe. *Investments*. Prentice-Hall International editions. Prentice-Hall, 1978. ISBN: 9780135046050. URL: https://books.google.com.au/books?id=4UgWAQAAMAAJ.

[SP85] Robert J Shiller and Pierre Perron. "Testing the random walk hypothesis: Power versus frequency of observation". In: (1985). eprint: https://elischolar.library.yale.edu/cgi/viewcontent.cgi?article=1971&context=cowles-discussion-paper-series.

[WG08] Ivo Welch and Amit Goyal. "A comprehensive look at the empirical performance of equity premium prediction". In: *The Review of Financial Studies* 21.4 (2008), pp. 1455–1508.

[WM76] N. Wiener and P. Masani. *Norbert Wiener - Collected Works: Mathematical Philosophy and Foundations - Potential Theory - Brownian Movement, Wiener Integrals, Ergodic and Chaos Theories, Turbulence and Statistical Mechanics*. Collected works. MIT Press, 1976. ISBN: 9780262230704. URL: https://books.google.com.au/books?id=UmGNzgEACAAJ.

[WHD95] P. Wilmott, S. Howison, and J. Dewynne. *The Mathematics of Financial Derivatives: A Student Introduction*. Cambridge University Press, 1995. ISBN: 9781139810975. URL: https://books.google.com.au/books?id=HAQgAwAAQBAJ.

[Zha97] P.G. Zhang. *Exotic Options: A Guide To Second Generation Options*. World Scientific Publishing Company, 1997. ISBN: 9789814549400. URL: https://books.google.com.au/books?id=VIzsCgAAQBAJ.

Online Articles

[Cor22] Options Clearing Corporation. *Annual Volume and Open Interest Statistics*. June 2022. URL: https://www.theocc.com/Market-Data/Market-Data-Reports/Volume-and-Open-Interest/Historical-Volume-Statistics (visited on 06/08/2022).

[Fin20] Yahoo! Finance. *Option Trading Volume Higher Than Underlying Stock Volume For First Time Ever*. June 2020. URL: https://finance.yahoo.com/news/option-trading-volume-higher-underlying-211006236.html (visited on 06/24/2022).

[Ins12] Business Insider. *The Story Of The First-Ever Options Trade In Recorded History*. Mar. 2012. URL: https://www.businessinsider.com/the-story-of-the-first-ever-options-trade-in-recorded-history-2012-3?r=US&IR=T (visited on 07/18/2022).

Appendix

A Matlab Files

All files can be found: https://github.com/leele2/Mathematics-in-Business-Project/tree/master/MATLAB%20Files

Listing A.1: ../../MATLAB Files/BinoAsian.m

```
function value = BinoAsian(S0,E,T,r,sigma,N,F)
addpath('Functions')
%% Function to evaluate European Call option by Binomial Method
%   Parameters:
%   S0 = initial share price
%   E = exercise price`
%   T = time to expiry
%   r = riskfree interest rate
%   sigma = volatility
%   N = Number of steps
%   F(E,S) = Option Payoff
%% Calculated parameters
dt   = T/N;                    %Timestep
u    = exp(sigma*sqrt(dt));    %Up price movement
d    = 1/u;                    %Down price movement
disf = exp(-r*dt);             %Discount factor over each timestep
p    = (1/disf - d)/(u-d);     %Risk-neutral probability
%% Initalizing Arrays and Functions
S    = zeros(N+1,N+1);         %Underlying Asset Price
S_k  = cell(N+1,N+1);          %S_max %% Cells are used so different sized vectors can be stored at each element in
    cell array
A_k  = cell(N+1,N+1);          %Representitive averages
C_k  = cell(N+1,N+1);          %Option price for given representitive avg
%% Functions to Calculate Maximum and Minimum Representative Averages
A_min = @(i,j) (1/(i+1)) * (sum(S0*d.^(0:i-j)) + sum(S0*d.^(0:(j-1)+i-(2*j))));
A_max = @(i,j) (1/(i+1)) * (sum(S0*u.^(0:j)) + sum(S0*u.^((0:(i-j-1))+(2*j)-i)));
%% Calculate Underlying asset price
for i = 0:N
    for j = 0:i
        S(i+1,j+1) = S0*u^(j)*d^(i-j);
    end
end
%% Calculating All S_max and Representitive Averages
for i = 0:N
    for j = 0:i %j indexes at j+1 due to matlab not allowing C_k{:,0)
        A_k{i+1,j+1} = zeros(j*(i-j)+1,1); %Create Vector to hold rep avgs
        S_k{i+1,j+1} = NaN(j*(i-j),1);     %Create Vector to hold S_max
        A_k{i+1,j+1}(1) = A_max(i,j);      %Assign A_max to first element in vector
        % Paths with only up (i = j) or down movements (j = 0) or i = 1 will only have one representative
            average
        if i == j || j == 0
            S_k{i+1,j+1}(1) = S0*(u^j)*d^(i-j);
            continue
        end
        %% Filling S_max Vector
        % Setting Paramaters
        n = j*(i-j);         %Size of vector
        rep = min([j;i-j]); %Maximum element repetition in vector
        % Calculating unique elements in vector
        unique = 2*(rep-1) + (n - ((rep-1)*rep))/rep;
        % First and last values
        S_k{i+1,j+1}(1)   = S0 * (u ^ j);
        S_k{i+1,j+1}(end) = S0 * (u ^ (j - unique + 1));
        %%The following if statement changes the final output%%
        %if n ~= 2
        % Ascending/Descending repeated values
        k     = 2; %iterator
        count = 2; %Number of elements filled
        while k < rep
            % Filling from top
            S_k{i+1,j+1}(sum(1:k-1)+1:sum(1:k-1)+k) = repmat(S0*(u^(j-k+1)), k, 1);
            % Filling from bottom
            S_k{i+1,j+1}(n-sum(1:k-1)+1-k:n-sum(1:k-1)) = repmat(S0*(u^(j-unique+k)), k, 1);
            count = count + 2*k;
            k = k + 1;
        end
        % "Middle" repeated values
        for l = 0 : (n-count)/rep - 1
            S_k{i+1,j+1}(count/2+1+l*rep:end) = [repmat(S0*u^(j-k+1), rep, 1);
                S_k{i+1,j+1}(count/2+1+l*rep+rep:end)];
            k = k + 1;
        end
        %end
        %% Calculating Representitive Averages Using S_max
        A_k{i+1,j+1}(j*(i-j) + 1) = A_min(i,j); %Assign A_min to last element in vector
```

```
        for k = 2:j*(i-j)
            A_k{i+1,j+1}(k) = A_k{i+1,j+1}(k-1) - (1/(i+1))* ...
                (S_k{i+1,j+1}(k-1) - S_k{i+1,j+1}(k-1)*(d^2));
        end
    end
end
%% Pricing Option Value at Final Time (N)
for j = 0:N
    C_k{N+1,j+1} = F(E,A_k{N+1,j+1});
end
%% Pricing Option
err = 1e-3;
for i = N-1:-1:0
    for j = 0:i
        C_k{i+1,j+1} = zeros(j*(i-j)+1,1);
        for k = 1:j*(i-j)+1
            %% Find K_u
            Ku  = ( (i+1)*A_k{i+1,j+1}(k) + u*S(i+1,j+1) )/(i+2);
            [loc, ubound, lbound] = findInSorted(Ku,A_k{i+2,j+2},err);
            % If found set accordingly
            if loc > 0
                Cu = C_k{i+2,j+2}(loc);
            else
                % If not found, interpolate between closest values
                Cu = C_k{i+2,j+2}(lbound)+(Ku-A_k{i+2,j+2}(lbound))*(...
                    (C_k{i+2,j+2}(ubound)-C_k{i+2,j+2}(lbound))/...
                    (A_k{i+2,j+2}(ubound)-A_k{i+2,j+2}(lbound)));
            end
            %% Find K_d
            Kd = ( (i+1)*A_k{i+1,j+1}(k) + d*S(i+1,j+1) )/(i+2);
            [loc, ubound, lbound] = findInSorted(Kd,A_k{i+2,j+1},err);
            %If found set accordingly
            if loc > 0
                Cd = C_k{i+2,j+1}(loc);
            else
                %If not, interpolate between closest values
                Cd = C_k{i+2,j+1}(lbound)+(Kd-A_k{i+2,j+1}(lbound))*(...
                    (C_k{i+2,j+1}(ubound)-C_k{i+2,j+1}(lbound))/...
                    (A_k{i+2,j+1}(ubound)-A_k{i+2,j+1}(lbound)));
            end
            %% Calculate option value at previous node (i,j decreasing)
            C_k{i+1,j+1}(k) = disf*(p*Cu + (1-p)*Cd);
        end
    end
end
value = C_k{1,1};
```

Listing A.2: ../../MATLAB Files/Functions/findInSorted.m

```
function [loc, A, B] = findinSorted(x,range,err)
%%%%%%%%%%%%%%%% inputs %%%%%%%%%%%%%%%%
% x = value to find                     %
% range = (sorted) vector to search     %
% err = tolerence on search             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Outputs
A   = 1;            %left boundary
B   = numel(range); %right boundary
loc = 0;            %initially set to 0 implying not found
%% Binary Search Algorithm
while B-A > 1
    mid = floor((A+B)/2);
    if x > range(mid)
        B = mid;
    else
        A = mid;
    end
end
%% Returning returned value if found (within tolerance)
if abs(range(A) - x) < err
    loc = A;
    return
elseif abs(range(B) - x) < err
    loc = B;
    return
end
```

Listing A.3: ../../MATLAB Files/MC.m

```
function value = MC(S0,E,T,r,sigma,N,F,M)
%% Evaluate Asian type option using Monte-Carlo
%   Parameters:
%   S0 = initial share price
%   E = exercise price`
%   T = time to expiry
%   r = riskfree interest rate
```

```
%   sigma = volatility
%   N = number of steps
%   F(E,S) = payoff function
%   M = number of realizations;
%% Calculated parameters
dt   = T/N;        %% Timestep
disf = exp(-r*T); %% Discount factor over each timestep
%% Initalizing Arrays and Functions
S = zeros(M, N+1);
Z = randn(M, N+1);
S(:,1) = S0;
%% Functions to Simulate Realisation
for i = 1:N
    S(:,i+1) = S(:,i).*exp((r-(.5*sigma^2))*dt + (sigma*sqrt(dt).*Z(:,i)));
end
%% Pricing Option
value = disf*mean(F(E, S));
```

Listing A.4: ../../MATLAB Files/BinoAsianPath.m

```
function value = BinoAsianPath(S0,E,T,r,sigma,N,F)
addpath('Functions')
%% Function to evaluate European Call option by Binomial Method
%   Parameters:
%   S0 = initial share price
%   E = exercise price`
%   T = time to expiry
%   r = riskfree interest rate
%   sigma = volatility
%   N = Number of steps
%   F(E,S) = Option Payoff
%% Calculated parameters
dt   = T/N;                    %% Timestep
u    = exp(sigma*sqrt(dt));    %% Up price movement
d    = 1/u;                    %% Down price movement
disf = exp(-r*dt);             %% Discount factor over each timestep
p    = (1/disf - d)/(u-d);     %% Risk-neutral probability
%% Initalizing Arrays and Functions
S     = zeros(N+1,N+1);   %% Underlying Asset Price
S_k   = cell(N+1,N+1);    %% S_max %% Cells are used so different sized vectors can be stored at each element
    in cell array
A_k   = cell(N+1,N+1);    %% Representitive averages
C_k   = cell(N+1,N+1);    %% Option price for given representitive avg
%% Functions to Calculate Maximum and Minimum Representative Averages
A_min = @(i,j) (1/(i+1)) * (sum(S0*d.^(0:i-j)) + sum(S0*d.^(0:(j-1)+i-(2*j))));
A_max = @(i,j) (1/(i+1)) * (sum(S0*u.^(0:j)) + sum(S0*u.^((0:(i-j-1))+(2*j)-i)));
%% Calculate Underlying asset price
for i = 0:N
    for j = 0:i
        S(i+1,j+1) = S0*u^(j)*d^(i-j);
    end
end
%% Calculating All Representitive Averages
for i = 0:N
    for j = 0:i %j indexes at j+1 due to matlab not allowing C_k{:,0)
        A_k{i+1,j+1} = zeros(j*(i-j)+1,1); % Create Vector to hold rep avgs
        S_k{i+1,j+1} = zeros(j*(i-j),1); % Create Vector to hold S_max
        A_k{i+1,j+1}(1) = A_max(i,j); %Assign A_max to first element in vector
        %Paths with only up (i = j) or down movements (j = 0) or i = 1 will only have one representative
            average
        if i == j || j == 0
            S_k{i+1,j+1}(1) = S0*(u^j)*d^(i-j);
            continue
        end
        S_k{i+1,j+1}(1) = S0*(u^j); %S_max for A_max
        %Creates initial path with j up steps followed by i-j downsteps
        %begins at node 1, +x: x steps above s0, -x: x steps below s0
        Tau = [1:j, j-1:-1:j-(i-j)+1];
        %Creates a "minimum" path with i-j downsteps followed by j upsteps
        Tau_min = [-1:-1:-(i-j),-(i-j):-(i-j)+(j-1)];
        %Removes duplicate from min path generation method
        Tau_min(numel(-1:-1:-(i-j))+1) = [];
        A_k{i+1,j+1}(j*(i-j) + 1) = A_min(i,j); %Assign A_min to last element in vector
        for k = 2:j*(i-j)
            A_k{i+1,j+1}(k) = A_k{i+1,j+1}(k-1) - (1/(i+1))* ...
                (S_k{i+1,j+1}(k-1) - S_k{i+1,j+1}(k-1)*(d^2));
            %filter where current path not equal to minimum path
            Tau_filtered = Tau ~= Tau_min;
            %Find first index which is not minimum
            [~, Tau_match] = find(Tau_filtered,1);
            %Index highest point on path that isn't on minimum path
            [~, Index] = max(Tau(Tau_filtered));
            %minus two from highest point not on minimum path
            Tau(Tau_match-1+Index) = Tau(Tau_match-1+Index) - 2;
            %Find new highest point not on min path by repeating above steps
            %1: Filter from not being on minimum path
            Tau_filtered = Tau ~= Tau_min;
```

```
            %2: Find first index not on minimum path
            [~, Tau_match] = find(Tau_filtered,1);
            %3: Find maximum not on minimum path
            [~, Index] = max(Tau(Tau_filtered));
            %Use new highest point to calculate the next S_max value
            S_k{i+1,j+1}(k) = S0 * u^(Tau(Tau_match-1+Index)); %Next S_max is the new Max
        end
    end
end
%% Pricing Option Value at Final Time (N)
for j = 0:N
    C_k{N+1,j+1} = F(E,A_k{N+1,j+1});
end
%% Pricing Option
err = 1e-6;
for i = N-1:-1:0
    for j = 0:i
        C_k{i+1,j+1} = zeros(j*(i-j)+1,1);
        for k = 1:j*(i-j)+1
            %Find K_u
            Ku = ( (i+1)*A_k{i+1,j+1}(k) + u*S(i+1,j+1) )/(i+2);
            [loc, ubound, lbound] = findInSorted(Ku,A_k{i+2,j+2},err);
            if loc > 0
                % Set value if found
                Cu = C_k{i+2,j+2}(loc);
            else
                % Otherwise, interpolate
                Cu = C_k{i+2,j+2}(lbound)+(Ku-A_k{i+2,j+2}(lbound))*(...
                    (C_k{i+2,j+2}(ubound)-C_k{i+2,j+2}(lbound))/...
                    (A_k{i+2,j+2}(ubound)-A_k{i+2,j+2}(lbound)));
            end
            %Find K_d
            Kd = ( (i+1)*A_k{i+1,j+1}(k) + d*S(i+1,j+1) )/(i+2);
            [loc, ubound, lbound] = findInSorted(Kd,A_k{i+2,j+1},err);
            if loc > 0
                % Set value if found
                Cd = C_k{i+2,j+1}(loc);
            else
                % Otherwise, interpolate
                Cd = C_k{i+2,j+1}(lbound)+(Kd-A_k{i+2,j+1}(lbound))*(...
                    (C_k{i+2,j+1}(ubound)-C_k{i+2,j+1}(lbound))/...
                    (A_k{i+2,j+1}(ubound)-A_k{i+2,j+1}(lbound)));
            end
            % Price option using discounted expected value
            C_k{i+1,j+1}(k) = disf*(p*Cu + (1-p)*Cd);
        end
    end
end
value = C_k{1,1};
```

B Python Files

All files can be found: https://github.com/leele2/Mathematics-in-Business-Project/tree/master/Python%20Files

Listing B.1: ../../Python Files/OptionVolume.py

```
# -*- coding: utf-8 -*-
"""
Created on Wed Jun  8 18:05:55 2022
Scraping and plotting data from theocc.com
@author: leele2
"""
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from datetime import datetime
from bs4 import BeautifulSoup
from os import listdir
from pathlib import Path

def string_to_int(string):
    if not string:
        return 0
    if string[0] == "$":
        string = string[1:]
    return int(string.replace(',', ''))

# Ensuring html file is on path
"""
html file taken from:
https://www.theocc.com/Market-Data/Market-Data-Reports/Volume-and-Open-Interest/Historical-Volume-Statistics
"""
html_file = [f for f in listdir('.') if f.endswith('.html')]
```

```
if len(html_file) != 1:
    raise ValueError('should be exactly one html file in the current directory')
# Importing html file
with open(html_file[0], 'r', encoding='utf-8') as file:
    soup = BeautifulSoup(file, 'lxml')
# Finding Table
table = soup.find_all("table")
# Selecting Data
table_data = table[0].find_all("tr")
# Finding Table Names
table_names = []
for names in table_data[0]:
    table_names.append(str(names.find("span").string))
del table_names[0]
del table_names[-1]
# Finding Headers
headers = []
for header in table_data[1]:
    headers.append(str(header.find("span").string))
# Populating Data
Data = {}
for rows in table_data[2:]:
    temp = {}
    for i, data in enumerate(rows):
        if i > 6 and i < 10 or i == 0:
            temp[headers[i]] = data.get_text()
    Date = datetime.strptime(temp["Date"], "%Y")
    del temp["Date"]
    Data[Date] = {k: string_to_int(v) for k, v in temp.items()}
# Converting data into pandas dataframes
Data = pd.DataFrame.from_dict(Data, orient="index")
# Plotting time-series
plt.rc('font', family='serif')
fig, ax1 = plt.subplots()
plt.gcf().set_size_inches(6, 4.5)
ax1.plot(Data['Options']/1e6, linewidth=2, zorder=1, label="Options")
# ax1.plot(Data[Data['Futures'] != 0]['Futures']/1e6, linewidth=2, zorder=1, label="Futures")
# ax1.legend()
ax1.set_title('Average Daily OptionS Trading Volume')
ax1.set_xlabel('Year')
ax1.xaxis.set_minor_formatter(mdates.DateFormatter("%Y"))
ax1.set_ylabel('Average Daily Volume (Millions)')
ax1.yaxis.set_major_formatter(mpl.ticker.StrMethodFormatter('{x:,.0f}M'))
fig.autofmt_xdate()
plt.tight_layout()
plt.show()
# Saving plot
cwd = str(Path(__file__).parent.parent.absolute())
fig.savefig(cwd + "\Latex_Files\Main\Chapters\C1\plots\OptionVolume.png", bbox_inches='tight')
```

Listing B.2: ../../Python Files/SymmetricRW_Sim.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 19 17:11:24 2022

@author: leele2
"""
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.stats import binom

def Simulate(prev, increment):
    return (prev + mu + sigma * increment)

# Parameters
mu    = +0.0 #Drift
sigma = +1.0 #Scale
S_0   = +10. #Initial point
p     = +0.5 #Probability of +1; P(Y = -1) = (1-p)
alpha = +1.0 #Step up
beta  = -1.0 #Step down

# Creating Random Walk Data
N   = 3   #Number of simulations to run
T   = 100 #How many forward steps to add to each run

sim = {}
for i in range(1, N + 1):
    sim[i] = []
    sim[i].append(S_0)
    for j, k in enumerate(binom.rvs(1, p, size = T)):
        if k == 1:
            sim[i].append(Simulate(sim[i][j], alpha))
        else:
            sim[i].append(Simulate(sim[i][j], beta))

```

```
# Plotting Simulations
plt.rc('font', family='serif')
fig, ax1 = plt.subplots()
plt.gcf().set_size_inches(6, 4.5)
for i in range(1, N + 1):
    ax1.plot(sim[i], linewidth=1.5)
ax1.set_title(str(N) + ' Simulations of Symetric Random Walk')
ax1.set_xlabel('$n$')
ax1.set_ylabel('$S_n$')
plt.tight_layout()
plt.show()
# Saving plot
cwd = str(Path(__file__).parent.parent.absolute())
fig.savefig(cwd + "\Latex_Files\Main\Chapters\C1\plots\RW_Simulations.png", bbox_inches='tight')
```

Listing B.3: ../../Python Files/GeometricRW_Sim.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec 14 16:46:24 2022

@author: leele2
"""
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.stats import binom

def Simulate(prev, increment):
    return (prev*(mu*delta_t + (delta_t**0.5)*sigma*increment) + prev)

# Parameters
mu    = 0.10 #Drift (Percentage increase per unit of time)
sigma = 0.15 #Scale (Percentage increase/decrease per step)
S_0   = 10.0 #Initial point
p     = 0.50 #Probability of +1; P(Y = -1) = (1-p)
alpha = 1.00 #Step up
beta  = -1.0 #Step down

# Creating Random Walk Data
N       = 3    #Number of simulations to run
T       = 1    #Final end time for each simulation
T_N     = 200  #How many increments to split time range into
delta_t = (T)/T_N

sim = {}
for i in range(1, N + 1):
    sim[i] = []
    sim[i].append(S_0)
    for j, k in enumerate(binom.rvs(1, p, size = T_N)):
        if k == 1:
            sim[i].append(Simulate(sim[i][j], alpha))
        else:
            sim[i].append(Simulate(sim[i][j], beta))

# Plotting Simulations
plt.rc('font', family='serif')
fig, ax1 = plt.subplots()
plt.gcf().set_size_inches(6, 4.5)
for i in range(1, N + 1):
    ax1.plot([x * delta_t for x in range(0, T_N + 1)], sim[i], linewidth=1)
ax1.set_title(str(N) + ' Simulations of Geometric Brownian Motion $Y_i \sim \mathcal{B}$er$_m$($p=0.5$)')
ax1.set_xlabel('$t$')
ax1.set_ylabel('$S_n$')
plt.tight_layout()
plt.show()
# Saving plot
cwd = str(Path(__file__).parent.parent.absolute())
fig.savefig(cwd + "\Latex_Files\Main\Chapters\C1\plots\BM_Simulations.png", bbox_inches='tight')
```

Listing B.4: ../../Python Files/GeometricNorm_Sim.py

```
# -*- coding: utf-8 -*-
"""
Created on Sat Dec 14 16:46:24 2022

@author: leele2
"""
import matplotlib.pyplot as plt
from pathlib import Path
from scipy.stats import norm

def Simulate(prev, increment):
    return (prev*(mu*delta_t + (delta_t**0.5)*sigma*increment) + prev)

# Parameters
mu    = 0.10 #Drift (Percentage increase per unit of time)
```

```
sigma = 0.15 #Scale (Percentage increase/decrease per step)
S_0   = 10.0 #Initial point

# Creating Random Walk Data
N       = 3     #Number of simulations to run
T       = 1     #Final end time for each simulation
T_N     = 200   #How many increments to split time range into
delta_t = (T)/T_N

sim = {}
for i in range(1, N + 1):
    sim[i] = []
    sim[i].append(S_0)
    for j, k in enumerate(norm.rvs(0, 1, size = T_N)):
        sim[i].append(Simulate(sim[i][j], k))

# Plotting Simulations
plt.rc('font', family='serif')
fig, ax1 = plt.subplots()
plt.gcf().set_size_inches(6, 4.5)
for i in range(1, N + 1):
    ax1.plot([x * delta_t for x in range(0, T_N + 1)], sim[i], linewidth=1)
ax1.set_title(str(N) + ' Simulations of Geometric Brownian Motion $Y_i \sim \mathcal{N}(0,1)$')
ax1.set_xlabel('$t$')
ax1.set_ylabel('$S_n$')
plt.tight_layout()
plt.show()
# Saving plot
cwd = str(Path(__file__).parent.parent.absolute())
fig.savefig(cwd + "\Latex_Files\Main\Chapters\C1\plots\BM_Norm_Simulations.png", bbox_inches='tight')
```