UNIVERSITY OF IOANNINA

DEPARTMENT OF ECONOMICS

STOCHASTIC PROCESSES FOR ASSET PRICES

Author: Christos Delivasilis

Supervisor: Theodore Simos

January 2023

To my parents

Acknowledgements

I would like to express my gratitude to my supervisor, Theodore Simos for his guidance and support throughout the preparation of my thesis. Additionally, I want to express my appreciation to my family and friends for their support during my postgraduate studies.

Contents

List of Figures

List of Tables

Abstract

Stochastic processes have been widely employed in the literature to value a variety of stock market assets, forecast their prices, and even determine and forecast their risk. In this thesis, I first discuss and simulate the most fundamental stochastic processes, Wiener and Poisson, and the characteristics that contribute to the explanation of a variety of stock market phenomena using a high-level programming language. In the empirical analysis Black-Scholes-Merton and Heston SV models are analyzed using Total Energies' stock price and options. The empirical evidence suggests that there are arbitrage opportunities as TTE's option price in the market is not equal to the Black-Sholes-Merton fair price. The sensitivity analysis suggests that TTE's option prices are not that sensitive to interest rate and dividend yield increases. Alongside, Heston SV specification is undoubtedly better to explain the fluctuations of TTE's stock price over the constant volatility framework. Finally, Heston SV predicting capabilities typically yield modest errors and accurately approximate the real price of the market.

Keywords: Stochastic processes, Wiener process, Poisson process, Geometric Brownian Motion, Option pricing, Forecasting Stochastic Volatility

1. Introduction

In the literature, stochastic processes have been extensively employed to estimate risk and uncertainty as well as to price different securities. Stochastic processes have been a part of economic theory since 1900, thus they are not new to the literature. The Wiener process, often known as Brownian Motion, is the most well-known stochastic process and is currently essential for producing stochastic volatility models.

Investors and portfolio managers need to be aware of the risk they are taking in their investments, let alone be able to predict it with relative accuracy. In the past literature there is a great deal of reference to AR, MA, ARMA, ARIMA, ARCH, GARCH etc. models which, however, cannot predict volatility with such high accuracy. The introduction of stochastic processes in financial models and mathematical finance models, has given a huge advantage to the mentioned for better management of their investments.

On the stock exchange, the prices of securities fluctuate erratically in either direction. Using the Wiener process, Black and Scholes (1973) were able to offer a closed-form solution for the precise pricing of securities. Their model was so precise that it is still employed as a benchmark in several studies today. Of course, their model had some shortcomings, most of which had been "fixed" by the literature. We now refer to stochastic variance utilizing stochastic processes. Stochastic variance is a concept used in every model that scholars have developed recently that involves the pricing of assets.

Consequently, we recognize the significance of stochastic process understanding because it currently serves as the basis for any new venture in the development of existing theory.

In this thesis, an extensive and in-depth analysis of the basic stochastic processes and their various applications is presented. The aim of the thesis is to present the elementary stochastic processes now used in the literature, and to highlight the ease of computing them through high-level programming languages (e.g., Python). At the same time, some models involving stochastic processes are used to study their capability in pricing and valuing securities. The Heston SV model seems to offer a valid and accurate pricing method, as well as a prediction.

Specifically, chapter 2 presents the literature review that discusses stochastic processes, their application in securities pricing and their evolution within various models that are now considered fundamental. Indicatively, the Black and Scholes (1973) model which was originally used for pricing European put, call options, the key Stochastic Volatility model of Heston (1993) where it is now used as a basis for other models, the Bates (1996) model combining Heston SV with Merton Jump Diffusion (Merton, 1976), etc.

Chapter 3 explains in mathematical terms the most important stochastic processes used to build the above models. These processes are the Wiener and Poisson processes. Indicatively, it is discussed how a Wiener process is built from scratch, properties, Quadratic Variation, etc. as well as a Poisson process. We conclude with several other forms that these processes take which are a step closer to reality. (e.g., Geometric Brownian Motion, Compound Poisson). In the fourth section, the process of simulating the above processes with a high-level language is carried out. In the fifth section, the models of Black and Scholes (1973) and Heston (1993) are analyzed and simulated for Total Energies (TTE) Stock price and Options. A sensitivity analysis for Black-Scholes is given. Calibration for the Heston (1993) variables with stock market data is also performed. Furthermore, a comprehensive forecast procedure is conducted and evaluated. Finally, a conclusion section is provided.

2. Literature Review

At the beginning of the 20th century, Bachelier (1900) in his thesis "Theory of Speculation" showed that financial markets are dominated by the laws of probabilities and statistics. In particular, he observed that the behavior of securities on the stock market is similar to the movement of particles in a fluid. This observation by Bachelier (1900) was the clue to link Brownian motion of particles to economic theory. We reflect that the motion of particles, which is random, in a fluid has quite similar properties to the stock market. Therefore, we could build models based on the randomness of variables to simulate the stock market. This is how Brownian motion began to be applied in finance. Fifty years later Wiener studied the properties of this Brownian motion in one dimension, and from then on, the Brownian motion got the name Wiener process in honor of Nobert Wiener. To this day the Wiener process is the building block in finance.

Later, the need to make the pricing of securities in the stock market more accurate, the Brownian motion or Wiener process was further developed into the geometric Brownian motion. It is otherwise called Brownian motion with drift. In this stochastic process, the logarithm of a random variable follows the Brownian motion with drift. Geometric Brownian motion is this stochastic process used in the model of Black and Scholes (1973).

Black and Scholes (1973) develop a mathematical model for the dynamics of the financial markets containing derivative instruments. This parabolic differential equation of Black and Scholes (1973) gives a closed-form solution to the pricing of European-style options. Although the Black and Scholes formula is still used today, it has been criticized for its biases (Rubinstein, 1985).

The Black and Scholes model assumes that the price of the asset can move upwards or downwards by the same probability. Yalincak (2012) states that this assumption does not reflect the reality, as stock prices are influenced by many factors that cannot

be assigned the same probability in the way they will affect the movement of stock prices. Black and Scholes (1973) assume that asset returns are normally distributed. Hull (2018) empirically disproves that assumption showing that returns are leptokurtic that is, they tend to have outliers. Another imperfect assumption is that volatility is constant, so as the interest rate. Cox, Ingersoll and Ross (1985) derive a model with stochastic interest rate but still the volatility term is not stochastic in nature. The CIR model was an extension of the Vasicek's (1977) model. Finally, Heston (1993) provides a closed-form solution for options with stochastic volatility and stochastic interest rates. The correlation between volatility and spot-asset returns can be arbitrary in his model. When the spot asset is connected with volatility, it offers a closed-form solution for the price of a European call option and updates the model to include stochastic interest rates. As a result, both bond options and currency options can be used with the model, in contrast to the Black and Scholes' (1973) model that tends to underperform with the latter.

In addition to Brownian motion and geometric Brownian motion, jump processes are needed to accurately describe the stock market. We have so far been able to reproduce the market using Brownian motion, but we are aware that stocks and derivatives can experience unpredictable jumps. The Poisson process adds the jumping element to this endeavor while Brownian motion builds the fundamental tools for discovering more accurate models. Such a model is Merton's (1976) which incorporates random jumps. This paper's major goal was to expand the Black-Scholes model to include more sensible presumptions that cope with the reality that empirical studies of market returns do not follow a constant variance log-normal distribution.

Even though the Black and Scholes (1973) model has been a benchmark model in finance and derivatives pricing, stochastic volatility has been developing from 1960. Early comments for stochastic volatility can be found in the work of Mandelbrot (1963) and Fama (1965). The SV technique uses the model's structure to specify the predicted distribution of returns rather than doing it explicitly. This predictive distribution can be explicitly estimated for a limited subset of SV models, but it must always be numerically computed for empirically accurate depictions. There are certain benefits to moving away from one-step predictions directly. Particularly in continuous time, modeling asset price volatility as possessing its own stochastic process is more practical and perhaps more natural without having to be concerned about the implied one-step-ahead distribution of returns recorded over an arbitrary frequency such as a daily or monthly data. Another important aspect of SV is its ability to account for an asymmetric return-volatility relationship, which is frequently referred to as a leverage effect (Black, 1976), despite the fact that it is generally acknowledged that the asymmetry has little to do with any underlying financial leverage.

The first stochastic volatility model was made by Hull and White (1987). They analyze European call options on a stock price subject to stochastic volatility. Using Taylor

series expansion they derive an accurate formula for call options, though it was a semi-closed solution without any correlation between returns and volatility. A few years later, Stein and Stein (1991) come up with another stochastic volatility model using a mean reverting process. They used the Olhestein-Ulhbeck (OU) process to model variance, but like the Hull and White (1987) there is no correlation between returns and volatility. After the Hull-White model, Heston (1993) comes up with a stochastic volatility model that incorporated a square root process of variance. As mentioned above, the Heston SV model was a closed form solution model that does require numerical integration, the variance is always positive, and it incorporates the leverage effect of Black (1976).

Furthermore, Dupire (1994) proposed a local volatility model. In this local volatility model, market prices were used to determine a local volatility surface in terms of time and strike. This deterministic surface could be used to value other options. At the same time Derman and Kani (1998)[1] were working on a similar model. Later this decade, Bates (1996) combines Heston's (1993) stochastic volatility model along with Merton's (1976) log-normal jump diffusion into the Stochastic Volatility with Jump (SVJ) model. Bates tests his model into Deutsche Mark options and finds that jump fears can explain the volatility smile.

The latest development in stochastic volatility literature is the work of Christoffersen, Heston and Jacobs (2009), also known as Double-Heston model. Through the use of multiple stochastic volatility variables, their paper employs an easy method to include a stochastic correlation. They show that the level and slope of the smirk can be controlled significantly more easily in two-factor models. Two-factor models also offer greater modeling flexibility for the volatility term structure, which is an additional benefit. They conduct a thorough investigation into the dimensions along which the estimated two-factor model differs from the one-factor model in order to shed additional light on the variations in pricing performance. They discover that the two-factor model significantly outperforms the one-factor model in both the term structure and moneyness dimensions. Additionally, they show that the typical one-factor model's modeling of conditional skewness and kurtosis is very constrained and that the estimated conditional higher moments have a strong correlation with the estimated conditional variance. The two-factor model, on the other hand, allows for more modeling freedom when describing conditional skewness and kurtosis at specific degrees of conditional variance, which is in line with the result that the slope of the smirk evolves substantially independently of the magnitude of volatility.

Nowadays, there is an index for measuring volatility that is used by financial analysts, investors, traders, etc. Utilizing the S&P 500 index, Cboe's Volatility Index (VIX) gauges

[1] Their work was being done at the same time as Dupire's (1994) but their paper was published in 1998.

the 30-day market's anticipated volatility. Investors use the VIX to evaluate the level of risk in the market. Because of this, it moves in the opposite direction of stock prices when they rise, suggesting low stress, and the other direction when they fall, indicating high stress. Options on the S&P 500 that are posted on Cboe make up VIX (SPX Options).

The most recent option literature uses both the Double-Heston model and the VIX index to price different types of derivatives, such as variance Swaps. Yoon and Kim (2021) provide a closed form solution for pricing variance swaps under the Double Heston model. In order to reduce the number of model parameters and to explicitly derive a closed form analytic solution formula for variance swaps, they rescale the double Heston model. They demonstrate that the rescaled double Heston model can match the VIX market data as well as the original double Heston model in a stable environment while taking substantially less time to compute than the double Heston model did. However, we agree that even the double Heston model falls short in a chaotic circumstance following the commencement of the COVID-19 pandemic.

3. Wiener, Poisson, and Lévy Processes

Nobert Wiener was the first to define and produce Brownian motion using a strict mathematical methodology. In his honor, this stochastic process is known as Wiener's process. Though, the first application of Brownian motion was in Bachelier (1900) controversial Thesis. The proof that such a process occurs, satisfies all the requirements, and is not a physical model termed Brownian motion as was previously believed and which one can approach with other models was one of Wiener's greatest discoveries.

3.1 Wiener Process

3.1.1 Random Walk and Binomial Distribution

The derivation of the Wiener Process or Brownian Motion arises from a random walk process (Vasiliou, 2001). Let's say we have the following binomial experiment:

$$X_1 = \begin{cases} 1, The\ price\ of\ the\ asset\ is\ increasing \\ 0, The\ price\ of\ the\ asset\ is\ decreasing \end{cases}$$

The fluctuation of the Asset Price comes with a probability (P) in case of success (The price goes up) and (1-P) in case of failure (The price goes down).

$$\mathbb{P}\{X_1 = 1\} = P$$
$$\mathbb{P}\{X_1 = 0\} = 1 - P$$

Assuming that S_t is the Price of the Asset at time t then we have the following contingencies: At time $t = 0$ there is a probability P that the price will increase and a probability $1 - P$ that the price will decrease. So S_1 either equals αS_0 with probability

P, or βS_0 with probability $1-P$, where α and β are positive multipliers on either event, $\alpha > 1, \beta \in (0,1)$.

$$S_0 \xrightarrow{P} S_1 = \alpha S_0$$
$$S_0 \xrightarrow{1-P} S_1 = \beta S_0$$

Although we have independence of the contingencies $X_1 = 1$, $X_2 = 0$, the probabilities P and $1-P$ remain constant and time-independent. So, in the next time period we will again have:

$$\text{(Success, Success, or Success, Failure)} \begin{cases} S_1 \xrightarrow{P^2} S_2(11) \\ S_1 \xrightarrow{P(1-P)} S_2(10) \end{cases}$$

$$\text{(Failure, Success, or Failure, Failure)} \begin{cases} S_1 \xrightarrow{P(1-P)} S_2(01) \\ S_1 \xrightarrow{(1-P)^2} S_2(00) \end{cases}$$

$S_{t=2}(11)$: If a successful event occurs after a successful event in the first period. $S_{t=2}(10)$: If a failed event occurs after a successful event in the first period. $S_{t=2}(01)$: If a successful event occurs after a failed event in the first period. $S_{t=2}(00)$: If a failed event occurs after a failed event in the first period.

This procedure continues for t periods. The stochastic process $\{X_t\}_{t=0}$ is also called random walk and it's a stochastic process in discrete time and space. It is clear that such process must be studied in continuous time and space. As a result, we take into account the time interval $[0, T]$ from the continuum and a very subtle partition of this interval that has a step:

$$0 = t_0 < t_1 < t_2 < \ldots < t_k < \ldots < t_n = T,$$
$$t_k = t_0 + k\Delta t, k = 0,1, \ldots, n$$

3.1.2 Symmetric Random Walk

Now we assume a "coin toss" example of the above process in order to construct a symmetric random walk. This process has two different outcomes $\omega_j = \{Heads, Tails\}$ and the probability of getting "Heads" or "Tails" in each toss is $p = \frac{1}{2}$:

$$X_j = \begin{cases} 1, if\ \omega_j = Heads \\ -1, if\ \omega_j = Tails \end{cases}$$

If we assume that the initial value of this process equals $Q_0 = 0$, the summation of the (k) outcomes would be $Q_k = \sum_{j=1}^{k} X_j$. This is a symmetric random walk process.

A symmetric random walk process comes with the property of independent increments between time periods. This increment can be shown as:

$$Q_{t_1} = (Q_{t_1} - Q_{t_0}), (Q_{t_2} - Q_{t_1}), \ldots, (Q_{t_n} - Q_{t_{n-1}})$$

Its increment has an expected value of $\mathbb{E}\left(Q_{t_{j+1}} - Q_{t_j}\right) = 0$ and variance $Var\left(Q_{t_{j+1}} - Q_{t_j}\right) = t_{j+1} - t_j$. As a result, the variance equals the difference in time or, the distance in time.

A symmetric random walk process also comes with the properties of a martingale. To observe that a symmetric random walk is a martingale, we choose nonnegative integers $k < l$ within $[0, T]$. So, in the time period $0 < k < l < T$ the conditional expectation of Q_l based on information up to time k is $\mathbb{E}[Q_l|\mathcal{F}_t]$, where $\mathcal{F}_t$ is a filtration, the σ-algebra information corresponding to first k coin tosses. This conditional expectation equals to

$$\mathbb{E}[Q_l|\mathcal{F}_t] = \mathbb{E}[(Q_l - Q_t) + Q_t|\mathcal{F}_t] = \mathbb{E}[(Q_l - Q_t)|\mathcal{F}_t] + \mathbb{E}[Q_t|\mathcal{F}_t]$$
$$= \mathbb{E}[(Q_l - Q_t)|\mathcal{F}_t] + M_t = M_t$$

as the $\mathbb{E}[M_l] = 0$.

3.1.3 Scaled Symmetric Random Walk and Log-Normal Distribution

Shreve (2008) states that in order to approximate a Brownian motion we scale down the step size of a symmetric random walk while accelerating time. The scaled symmetric random walk is defined as

$$W^{(n)}(t) = \frac{1}{\sqrt{n}} M_{nt}$$

We obtain the Brownian motion in the limit as $n \to \infty$. Like the random walk, the scaled random walk has independent increments. Also, he states that as we increase the number of the processes (n) we tend to get more precise approximations of the normal distribution. This approximation is valid according to the central limit theorem.

Additionally, the limit of a properly scaled binomial asset-pricing model leads to a stock price with a log-normal distribution, according to the Central Limit Theorem. These findings demonstrate that the geometric Brownian motion model, which serves as the foundation for the Black-Scholes option-pricing formula, is a discrete-time equivalent of the binomial model. Shreve (2008) constructs a formula by a binomial example of a stock price with two factors, the up factor, which is denoted by $u_n = 1 + \frac{\sigma}{\sqrt{n}}$, and the down factor denoted by $d_n = 1 - \frac{\sigma}{\sqrt{n}}$. The process of the stock price with $S(0)$ being the initial price, converges to the distribution of $S(t) = S(0)e^{\sigma W(t) - \frac{1}{2}\sigma^2 t}$ as $n \to \infty$. $W(t)$ is the Wiener process with mean zero and variance t.

The distribution of $S(t)$ is referred to as log-normal. In a broader sense, a log-normal distribution is defined as any random variable with the form ce^x, where c is a constant and x is normally distributed. With a mean of $-\frac{1}{2}\sigma^2 t$ and a variance of $\sigma^2 t$, $X = \sigma W(t) - \frac{1}{2}\sigma^2 t$ in this instance is normal. Shreve (2008) concludes that as $n \to \infty$, the distribution of $\log S(t)$ approaches the distribution of $\log S(0) - \frac{1}{2}\sigma^2 t + \sigma W(t)$.

3.1.4 Brownian Motion

As previously said, we obtain Brownian Motion as the limit of scaled random walks $W^{(n)}(t)$ approach infinity ($n \to \infty$). The Brownian Motion, as scaled random walk, has similar properties. That is, independent increments but normally distributed, zero mean and $t_{j+1} - t_j$ variance. The definition of Brownian Motion is given below.

Definition of Brownian Motion (Shreve, 2008; Karatzas and Shreve, 1998):

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. For each $\omega \in \Omega$, suppose there is a continuous function $W(t)$ of $t \geq 0$ that satisfies $W(0) = 0$ and that depends on ω. Then $W(t), t \geq 0$, is a Brownian motion if for all $0 = t_0 < t_1 < t_2 < \ldots < t_k < \ldots < t_n$

$$W(t_1) = W(t_1) - W(t_0), W(t_2) - W(t_1), \ldots, W(t_m) - W(t_{m-1})$$

are independent and each of these increments is normally distributed with

$$\mathbb{E}[W(t_{i+1}) - W(t_i)] = 0,$$

$$Var[W(t_{i+1}) - W(t_i)] = t_{i+1} - t_i.$$

A scaled random walk, such as $W^{(100)}(t)$, and a Brownian motion, $W(t)$, differ in that the latter has no linear components while the former has a natural time step and is linear between these time steps. The Brownian motion is exactly normal, in contrast to the scaled random walk $W^{(100)}(t)$, which is only approximately normal for each t. It follows from the Central Limit Theorem that this is the case. The increments $W(t) - W(s)$ are normally distributed for all $0 \leq s \leq t$, and $W(t) = W(t) - W(0)$ is normally distributed for each t.

In the definition given above, there are two ways to conceptualize ω. To put it another way, one should consider to be the Brownian motion's path. The value of this path at time t is therefore represented by $W(t)$, and it apparently depends on whatever path emerged from the random experiment. Alternately, one can consider as something more fundamental than the path itself, comparable to the result of a series of faster-moving coin tosses. The course of the Brownian motion can be depicted after the series of coin tosses has been completed and the outcome has been determined. A different path will be drawn if the tossing is repeated and a different is achieved.

In either scenario, the sample space Ω is the set of all outcomes that could result from a random experiment, $\mathcal{F}$ is the σ-algebra of subsets of Ω whose probabilities are

defined, and $\mathbb{P}$ is a probability measure. The probability of A is a number $\mathbb{P}(A)$ between zero and one for each $A \in \mathcal{F}$. The Brownian motion distributional statements apply to $\mathbb{P}$.

Furthermore, we have to investigate the covariances matrix of the Brownian motion. As previously said, the increment of the Brownian motion $W(t_1) - W(t_0), W(t_2) - W(t_1), \dots, W(t_m) - W(t_{m-1})$ are independent and normally distributed, so the random variables $W(t_1), \dots, W(t_m)$ are jointly normally distributed as well. From the properties of Brownian motion, the expected value of $W(t_i)$ is zero and the covariance between two times $s < t$ is $\mathbb{E}[W(s)W(t)] = \mathbb{E}\big[W(s)\big(W(t) - W(s)\big) + W^2(s)\big] = Var[W(s)] = s$. The variance-covariance matrix would be:

$$\begin{bmatrix} \mathbb{E}[W^2(t_1)] & \mathbb{E}[W(t_1)W(t_2)] & \cdots & \mathbb{E}[W(t_1)W(t_m)] \\ \mathbb{E}[W(t_2)W(t_1)] & \mathbb{E}[W^2(t_2)] & \cdots & \mathbb{E}[W(t_2)W(t_m)] \\ \vdots & \vdots & & \vdots \\ \mathbb{E}[W(t_m)W(t_1)] & \mathbb{E}[W(t_m)W(t_2)] & \cdots & \mathbb{E}[W^2(t_m)] \end{bmatrix} = \begin{bmatrix} t_1 & t_1 & \cdots & t_1 \\ t_1 & t_2 & \cdots & t_2 \\ \vdots & \vdots & \ddots & \vdots \\ t_1 & t_2 & \cdots & t_m \end{bmatrix}$$

By specifying the joint density[2] or the joint moment-generating function of the random variables, $W(t_1), W(t_2), \dots, W(t_m)$ one can determine the distribution of Brownian increments.

Another key element of the Brown motion is that it accumulates quadratic variation at rate one per unit time. To exhibit this fact, we have to define quadratic variation:

$$[W, W](T) = \lim_{\|\Pi\| \to 0} \sum_{j=0}^{n-1} [W(t_{j+1}) - W(t_j)]^2$$

The above equation demonstrates quadratic variation with $f(t)$ being a function of the time step, $0 \le t \le T$, $\Pi = \{t_0, t_1, \dots, t_n\}$ is the set of time steps which they belong in $[0, T]$ and $\|\Pi\|$ is the Euclidean norm of Π. Shreve (2008) proves that quadratic variation of Brownian motion is almost surely T for all $T \ge 0$. Also, the mean from a squared increment of Brownian motion equal $t_{j+1} - t_j$ and the variance is $2(t_{j+1} - t_j)^2$. By definition $t_{j+1} - t_j$ is a very small number, so the square of this number would be even smaller. Hence, the squared increment of Brownian motion from j to $j + 1$ is with high probability near its mean. Consequently, we argue that $[W(t_{j+1}) - W(t_j)]^2 \approx t_{j+1} - t_j$. Since both sides of this approximation are close to zero when $t_{j+1} - t_j$ is small, it is trivially true. It would also hold true if we changed or significantly altered $t_{j+1} - t_j$. In other words, the above approximation is of no content. It makes more sense if we subtract both sides with the right-hand-side. This would give $\frac{[W(t_{j+1}) - W(t_j)]^2}{t_{j+1} - t_j} \approx 1$.

[2] Normal distribution density function: $f(x) = \frac{1}{\sqrt{2\pi t}} e^{-\frac{x^2}{2t}}$

If we compute the quadratic variation of Brownian motion over the time interval $[0, T_1]$, we get $[W, W](T_1)$. Similarly, if we compute quadratic variation over $[0, T_2]$, $T_1 < T_2$, we get $[W, W](T_2) = T_2$. Hence, if we separate the interval $[T_1, T_2]$ into subintervals, square the increments of the subintervals, then sum them and take the limit as the maximal step size approaches zero, we will get the limit $[W, W](T_2) - [W, W](T_1) = T_2 - T_1$. Then, Brownian motion accumulates $T_2 - T_1$ units of quadratic variation over the interval $[T_1, T_2]$. So Brownian motion accumulates quadratic variation at rate one per unit time.

3.1.5 Geometric Brownian Motion

The stochastic differential equation that yields the Geometric Brownian Motion is

$$dS_t = \mu dt S_t + \sigma dW_t S_t, \qquad t \in [0, T]$$

The first part of the right-hand side contains the *drift* term, and the second part contains the *diffusion* term. In order to use this SDE to price various assets we have to solve it. A solution can be derived from combining the log-transformation of the stock price with Itô's lemma.

Itô's lemma where $y(t, X_t)$ represents a time-dependent function of a stochastic process

$$dy(t, X_t) = \frac{\partial y}{\partial t} dt + \frac{\partial y}{\partial X_t} dX_t + \frac{1}{2} \frac{\partial^2 \varphi}{\partial X_t^2} dX_t^2$$

Now consider the following log-transformation of the stock price: $\varphi(S_t) = \ln(S_t)$. We can apply Itô's lemma to the log-transformation function

$$d\varphi(S_t) = \frac{\partial y}{\partial t} dt + \frac{\partial \varphi}{\partial S_t} dS_t + \frac{1}{2} \frac{\partial^2 \varphi}{\partial S_t^2} dS_t^2$$

The differentiation of g with respect to t yields zero. Similarly, the first derivative of φ with respect to the stock price gives $\frac{1}{S_t}$ and the second derivative $-\frac{1}{S_t^2}$, mathematically:

$$\frac{\partial \varphi}{\partial t} = 0, \frac{\partial \varphi}{\partial S_t} = \frac{1}{S_t}, \frac{\partial^2 \varphi}{\partial S_t^2} = -\frac{1}{S_t^2}$$

$$dS_t^2 = \sigma^2 S_t^2 (dW_t)^2 = \sigma^2 S_t^2 dt$$

Substituting those results into $d\varphi(S_t)$:

$$d\varphi(S_t) = 0 + \frac{1}{S_t}(\mu dt S_t + \sigma dW_t S_t) + \frac{1}{2}(\frac{-1}{S_t^2})\sigma^2 S_t^2 dt \Leftrightarrow$$

$$d\varphi(S_t) = \left(\mu - \frac{1}{2}\sigma^2\right) dt + \sigma dW_t$$

We integrate both sides from t_0 to T:

$$\int_{t_0}^{T} d\varphi(S_t) = \int_{t_0}^{T} \left(\mu - \frac{1}{2}\sigma^2\right) dt + \int_{t_0}^{T} \sigma dW_t \Leftrightarrow$$

$$\varphi(S_t) - \varphi(S_0) = \left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0})$$

We substitute $\varphi(S_t) = \ln(S_t)$ to the above equation:

$$\ln(S_t) - \ln(S_0) = \left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0}) \Leftrightarrow$$

$$\ln\frac{S_t}{S_0} = \left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0}) \Leftrightarrow$$

$$e^{\ln\frac{S_t}{S_0}} = e^{\left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0})} \Leftrightarrow$$

$$\frac{S_t}{S_0} = e^{\left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0})} \Leftrightarrow$$

$$S_t = S_{t_0} e^{\left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0})},$$

Substituting $t_0 = 0$ and $W_{t_0} = 0$ in the equation above yields the equation of geometric Brownian motion used empirically, which is used in the next chapter of this thesis.

$$S_t = S(0) e^{\left(\mu - \frac{1}{2}\sigma^2\right)T + \sigma W_T}$$

The volatility of geometric Brownian motion is σ which is called realized volatility and it can be computed by calculating the sum of squares of log returns.

$$\sum_{j=0}^{m-1} \left(\ln\frac{S(t_{j+1})}{S(t_j)}\right)^2 = \left[\left(\mu - \frac{1}{2}\sigma^2\right)(T - t_0) + \sigma(W_T - W_{t_0})\right]^2$$

Shreve (2008) proves that when the maximum step size is small, the realized volatility of geometric Brownian motion approximates $\sigma^2(T_2 - T_1)$ in any interval $[T_1, T_2]$. Therefore,

$$\sum_{j=0}^{m-1} \left(\ln\frac{S(t_{j+1})}{S(t_j)}\right)^2 \frac{1}{T_2 - T_1} \approx \sigma^2$$

As a result, when the asset price exhibits a geometric Brownian motion, we calculate the realized volatility by taking the square root of the left hand side of the equation above. We can produce a more accurate approximation for realized volatility if we reduce the step size.

3.2 Poisson Process

To effectively model the stock market, we also require jump processes in addition to Brownian motion and geometric Brownian motion. Brownian motion has allowed us to replicate the market up to this point, but we are aware that stocks and derivatives are capable of having uncontrollable jumps. While Brownian motion and the Poisson process build the fundamental instruments for finding more accurate models, the Poisson process introduces the jumping component in this endeavor.

We cannot have random jumps in the time interval with the Poisson process, just as with the Brownian motion, where the drift term was the difficulty and that is why the geometric Brownian motion was invented. We therefore employ the Compound Poisson process, which enables us to have arbitrary jumps in either direction, up or down. To set the stage for the investigation in the next chapters, we will first examine the standard Poisson process.

3.2.1 Poisson Process

We assume the random variable τ with density

$$f(t) = \begin{cases} \lambda e^{-\lambda t}, & t \geq 0 \\ 0, & t < 0 \end{cases}$$

with λ being a positive constant. The expected value of τ is $\mathbb{E}\tau = \frac{1}{\lambda}$, as τ is an exponential random variable. In addition, $\mathbb{P}\{\tau \leq t\} = F(t) = \int_0^t \lambda e^{-\lambda v} dv = 1 - e^{-\lambda t}$. So, the probability of $\mathbb{P}\{\tau > t\} = 1 - \mathbb{P}\{\tau \leq t\} = e^{-\lambda t}$.

A major property for exponential distribution is memorylessness. This can be shown by calculating the following probability: $\mathbb{P}\{\tau > t + s \,|\tau > s\} = \frac{\mathbb{P}\{\tau > t+s \text{ and } \tau > s\}}{\mathbb{P}\{\tau > s\}} = \frac{e^{-\lambda(t+s)}}{e^{-\lambda s}} = e^{-\lambda t}$.

An intuitive example of this property could be the waiting time of an investor. Suppose he has been waiting for a stock to reach his desired price and he knows that the distribution of the time of this event is exponential with mean $\mathbb{E}\tau = \frac{1}{\lambda}$. Consider the case where he has already waited s time units and is interested in the probability that he will need to wait t more time units. In other words, the likelihood that he would have to wait an additional t time units after waiting s time units is the same as the likelihood that he would have to wait t time units had he started at time zero. The distribution of the remaining time is unchanged by the fact that he has already waited s time units.

The construction of a Poisson process begins with a sequence of independent exponential random variables with mean $\frac{1}{\lambda}$, that is $\tau_1, \tau_2, \ldots, \tau_n$. In this process, the first jump occurs at τ_1, the second at τ_2, etc.

$$S_n = \sum_{k=1}^{n} \tau_k$$

The τ_k exponential random variables are called interarrival times and S_n are the arrival times, where the index n indicates the n^{th} jump. The number of jumps that take place at or before time t is counted by the Poisson process $N(t)$. The jumps are happening on average at a rate of λ per unit time since the predicted duration between jumps is $\frac{1}{\lambda}$. The Poisson process $N(t)$ has intensity λ.

Additionally, we must determine the Poisson process increments' distribution. To accomplish that, we must ascertain the distribution of S_k jumps first. Shreve (2008), Embrechts, Frey and Furrer (2001) support that the random variable S_k has the gamma density. The Poisson process $N(t)$ with λ intensity has distribution

$$\mathbb{P}\{N(t) = k\} = \frac{\lambda t^k}{k!} e^{-\lambda t}, k = 0,1,2, \dots$$

Furthermore, the increments of the Poisson process are stationary. Let s be the present time and $t + s > s$ be the time after s. We are interested in $N(t + s) - N(s)$. As we assert the property of memorylessness of exponential variables, it turns out that the information about what happened up to and including time s is irrelevant. To compute the distribution of the jump from time s to time $t + s$, we are interested in time of the next jump after s. The time between s and the subsequent jump does not depend on the time between s and the previous jump, which we know at time s. Indeed, regardless of whatever that has occurred up to time s, the interval between s and the first jump after s has an exponential distribution with mean $\frac{1}{\lambda}$, as well as the succeeding jumps. Therefore, $N(t + s) - N(s)$ is independent of the filtration of s. The distribution of increments in a process is considered to be stationary when it has the property that it depends solely on the time interval between the two time points. Thus, the increments of Poisson process are stationary and independent with probability

$$\mathbb{P}\{N(t_{j+1}) - N(t_j) = k\} = \frac{\lambda^k (t_{j+1} - t_j)^k}{k!} e^{-\lambda(t_{j+1} - t_j)}, k = 0,1,2, \dots$$

The mean and of the increments of Poisson process equal $\lambda(t - s)$. More specifically, we know the distribution of the increments and recalling the exponential power series the expected value of the increment is $\mathbb{E}[N(t) - N(s)] = \sum_{k=0}^{\infty} k \frac{\lambda^k (t-s)^k}{k!} e^{-\lambda(t-s)}$ which is equal to $\lambda(t - s)$.

The variance is a product of the second moment. $\mathbb{E}[(N(t) - N(s))^2] = \sum_{k=0}^{\infty} k^2 \frac{\lambda^k (t-s)^k}{k!} e^{-\lambda(t-s)} = \lambda^2(t - s)^2 + \lambda(t - s)$. So, taking the formula for variance we derive $Var[N(t) - N(s)] = \lambda(t - s)$.

3.2.2 Compound Poisson Process

A Compound Poisson process is a continuous time stochastic process which builds on the standard Poisson process. Let $N(t)$ be a Poisson process with intensity λ, and $Y_1, Y_2, \ldots$ a sequence of identical and independently distributed random variables with mean $\mathbb{E}Y_i = \beta$. We assume those random variables are independent of the Poisson process $N(t)$.

In the Poisson process the size of the jump was fixes. The Compound Poisson process has random size jumps and those are occurring the same time as the $N(t)$. The first jump is Y_1, the second Y_2 and goes on till $N(t)$ forming a path. Hence, the Compound Poisson process is given by

$$Q(t) = \sum_{i=1}^{N(t)} Y_i\,, \quad t \geq 0$$

Also, the increments of the Compound Poisson process are independent and have the same distribution with $Q(t-s)$, as a consequence of $N(t-s)$ having the same distribution with $N(t) - N(s)$. The increment of Compound Poisson is specified as follows $Q(t) - Q(s) = \sum_{i=1+N(t)}^{N(t)} Y_i$.

The mean of this process is $\mathbb{E}Q(t) = \beta\lambda t$ as there are λt jumps in the interval $[0, t]$ and the average jump size is β.

Lastly, there are two equally valid perspectives on a Compound Poisson process with a finite number of potential leap sizes, according to Shreve (2008). It may be compared to a single Poisson process with random-sized leaps in place of the one-sized jumps. Alternately, it may be thought of as a collection of separate Poisson processes, each of which has leaps of a defined size in place of the size one jumps.

As a consequence, if $\{y_1, \ldots, y_M\}$ is a set of non-zero numbers, and the probability for each element of the set is $\{p(y_1), \ldots, p(y_M)\}, m = 1, \ldots, M$ with sum 1, we define the Compound Poisson process as

$$Q(t) = \sum_{i=1}^{N(t)} Y_i$$

with Y_i being a sequence of i.i.d. random variables with probability $\mathbb{P}\{Y_i = y_m\} = p(y_m)$. If $N_m(t)$ denotes the number of jumps in Q, then

$$N(t) = \sum_{m=1}^{M} N_m(t)\ , Q(t) = \sum_{m=1}^{M} y_m N_m(t)$$

This way, the Poisson processes N_m are independent and have $\lambda p(y_m)$ intensity.

3.3 Lévy Processes

The above studied processes (Wiener, Poisson) belong to a larger group of stochastic processes called Lévy processes (Satō, 2013; Bertoin, 1996; Eberlein, 2009). Those processes have the following properties: First of all, the initial value in the time interval is, almost surely, zero i.e., $X_0 = 0$. Then, the increments of those processes are mutually independent, (see Brownian motion where the time steps were $0 = t_0 < t_1 < t_2 < \ldots < t_k < \ldots < t_n$ and the increments $W(t_1) = W(t_1) - W(t_0), W(t_2) - W(t_1), \ldots, W(t_m) - W(t_{m-1})$). Next, Lévy processes have stationary increments which means that for any $s < t$ the distribution of $X_t - X_s$ is equal to X_{t-s}. Lastly, there is continuity in probability, i.e., for any number $\varepsilon > 0$ and time $t \geq 0$, $\lim_{t \to s} P(|X_t - X_s| > \varepsilon) = 0$. In the case where all conditions apply (namely, X is a Lévy process) then $\lim_{t \downarrow 0} P(|X(t)| > \varepsilon) = 0$, (Applebaum, 2005).

3.3.1 Normal and rare events

There are two types of events in the stock market, the so called “normal” and “rare” events. According to Hirsa and Neftci (2014), the size of the events and the likelihood that they will occur are the primary distinctions between "normal" and "rare" events. Since this interval tends to be zero, it becomes irrelevant when the observation interval shrinks along with the size of "normal" events. There is always a non-zero chance that some unnoticeable news may come in, even over a brief period of time. In contrast to "regular" events, "rare" events (such as shocks) can cause a considerable shift in the random variable's value in a short amount of time. For example, a market crash is considered to be a “rare” event. Put differently, the chance of a "rare" event moves to zero as the interval of observations tends toward zero, but its size may not change.

In the case that markets are driven by “normal” events, Brownian motion can be used to price a derivative. But this Lévy process by itself is not able to describe the stock market as a whole. For this reason, further models must be used, such as Merton’s jump diffusion model (Merton, 1976). This model uses Brownian motion along with Poisson process and allows for both “normal” events and “rare” event to occur.

3.3.2 Infinite divisibility and characteristic exponents

A Lévy process is a Markov process by default due to the characteristics of stationary and independent increments. Lévy processes can also be demonstrated to be robust Markov processes. We need to introduce the concept of an infinitely divisible distribution in order to better grasp Lévy processes. A random variable Θ does have an infinitely divisible distribution if a series of identical random variables exist that: $\Theta = \Theta_{1,n} + \cdots + \Theta_{n,n}$.

As an alternative, characteristic exponents can be used to express this relation (Schoutens, 2005). If Θ has a characteristic exponent $\psi(u) = -\log \mathbb{E}(e^{iu \cdot \Theta})$ then Θ is infinitely divisible if and only if for all $n \geq 1$ there exist a characteristic component of

a probability distribution ψ_n such that $\psi_u = n\psi_n$ for all $u \in \mathbb{R}^d$ (Kyprianou, 2014). That random variable Θ has an infinitely divisible distribution if there exists a triplet (α, Σ, Π), $\alpha \in \mathbb{R}^d$ and its characteristic function satisfies the Lévy - Khintchine formula which is:

$$\psi(u) = ia \cdot u + \frac{1}{2} u \cdot \Sigma u + \int_{\mathbb{R}^d} \left(1 - e^{iu \cdot x} + iu \cdot x \mathbf{1}_{(\|x\|<1)}\right) \Pi(dx),$$

where Π is the Lévy measure and Σ is a $d \times d$ matrix whose eigenvalues are nonnegative. The operator $(\cdot)$ is used to characterize the inner product between two matrices.

Brownian motion and Compound Poisson process have been analyzed in section 2.2. Those processes are the foundations for any other Lévy process. To be more precise, Brownian motion is the Lévy process with characteristic exponent $\psi(u) = \frac{1}{2} u \cdot \Sigma u$ and as a result has increments with Gaussian distribution across time periods of length t, together with a covariance matrix Σt. On the other hand, the Compound Poisson process has characteristic exponent $\psi(u) = \int_{\mathbb{R}^d} \left(1 - e^{iu \cdot x}\right) \lambda F(dx)$, where $\lambda > 0$ is the Poisson's process intensity and F a probability distribution.

A measurably infinite sum of Lévy processes can be demonstrated to converge appropriately to a Lévy process under specific circumstances. Any finite number of independent Lévy processes added together also constitutes a Lévy process. The Lévy-Itô decomposition, covered in the following section, is based on this concept. Either a Brownian motion with drift or a Compound Poisson process with drift make up the Lévy processes that are added.

3.3.3 Lévy – Itô decomposition

The path of a certain Lévy process is represented in the Lévy-Khintchine formula. Every sample path of a Lévy process can be represented as the sum of two independent processes according to the Lévy-Itô decomposition. One is a continuous Lévy process, which is the Brownian motion (Kyprianou, 2014), and the other is a compensated sum of independent jumps, which is a Compound Poisson process with drift (Matsuda, 2006).

To be more precise, a Lévy process can be decomposed in four parts as seen in Satō (2013). The first part is a deterministic linear drift, the second is a continuous Lévy process, i.e., the Brownian motion, a Compound Poisson process and a square integrable martingale, known as pure jump process. So, the characteristic exponent of a Lévy process consists of four individual characteristic exponents. The decomposition of Lévy processes devised by Lévy (1934;1954) and proved by Itô (1942).

4. Simulation of Stochastic Processes[3]

4.1 Wiener Process

A random walk process serves as the foundation for the Wiener Process or Brownian Motion. Because of this, we will begin expanding upon the random walk, to eventually obtain the Wiener process and the geometric Brownian motion. In order to perform the simulations of the processes mentioned in the previous chapter, Python 3.0 and some packages such as matplotlib, scipy, pandas etc.

The symmetric random walk is derived from the binomial problem in the second chapter of this thesis. This random walk can go either up or down by one. I start by initializing the parameters that is, the number of simulations and the time period. I start by creating a list that has two values in it, -1 and 1 and that's because along the path we can go up or down (i.e., +1 going up or -1 going down). Using numpy, I randomly select for t=10 years a value from the list [-1,1]. Then I initialize a vector with zeros with size 10 (i.e., the number of simulations). Then I join the two arrays along the same axis using numpy.concatenate and sum the values with numpy.cumsum. Figure 1 depicts the simulation of a symmetric random walk.

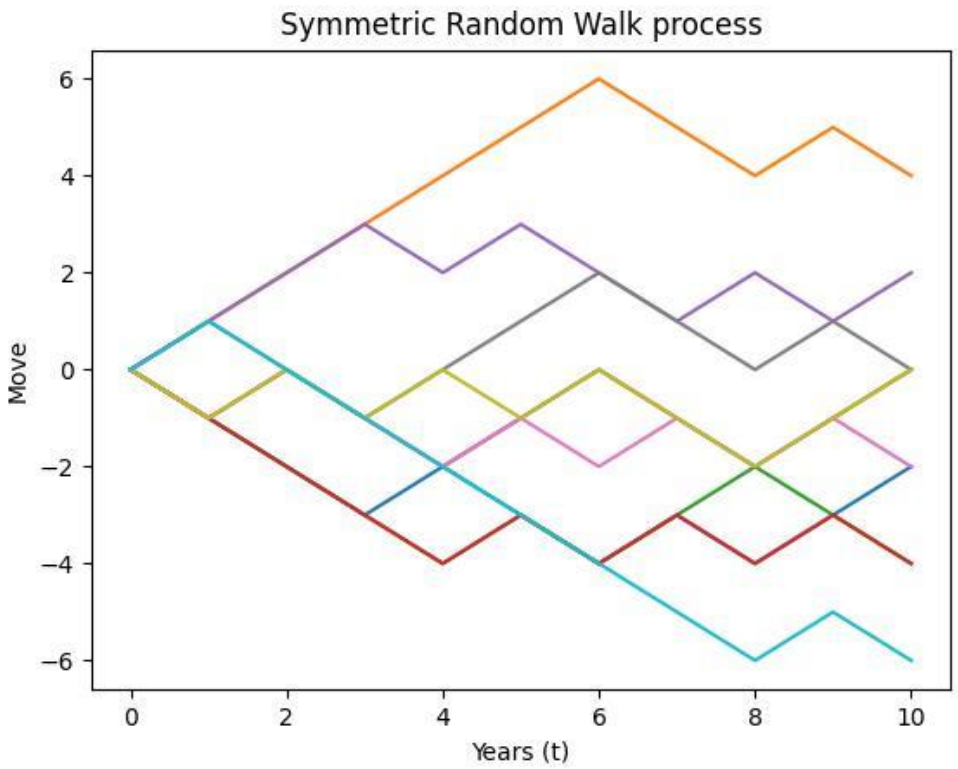

Figure 1: Simulation of Symmetric Random Walk Process using Python

[3] The code that has been used to derive the Figures is given in the Appendix.

The quadratic variation estimation follows. The quadratic variation is computed path by path and it's the square sum of the independent increments. That sum of the squared increments is the actual distance between the time steps. To prove that I create a function called quadratic variation and a function called variance. I am also using python's lambda function be able to sum across all the time steps and to better manage the code. The result of the code is that quadratic variation equals to the time step. On the other hand, variance equals to the current time step (i.e., at the first time step the variance is one, at the second time step the variance is two, …). To get a better estimate of the variance I ran the code for ten million simulations. As I increase the number of simulations, I get a better estimate (as we approach infinity, we get convergence).

The next step in order to get Brownian motion and Geometric Brownian motion is Scaled symmetric random walk. The equation I am trying to simulate is $W^{(n)}(t) = \frac{1}{\sqrt{n}} M_{nt}$. Similarly to Symmetric random walk I set the parameters that is, the number of simulations, the time and (n) which is the steps for every year. So, for ten years we will have (nt) steps. The code is similar to that of scaled random walk, but the time vector now is more "concentrated" in time. Scaled symmetric random walk still holds the same properties. Variance at a particular point along all those paths are equal to the time period and the quadratic variation along those paths is exactly the time, even though there are (nt) time steps. Figure 2 shows the simulated process and can observe that the movement of a particular path deviates from the rest, which is completely random.

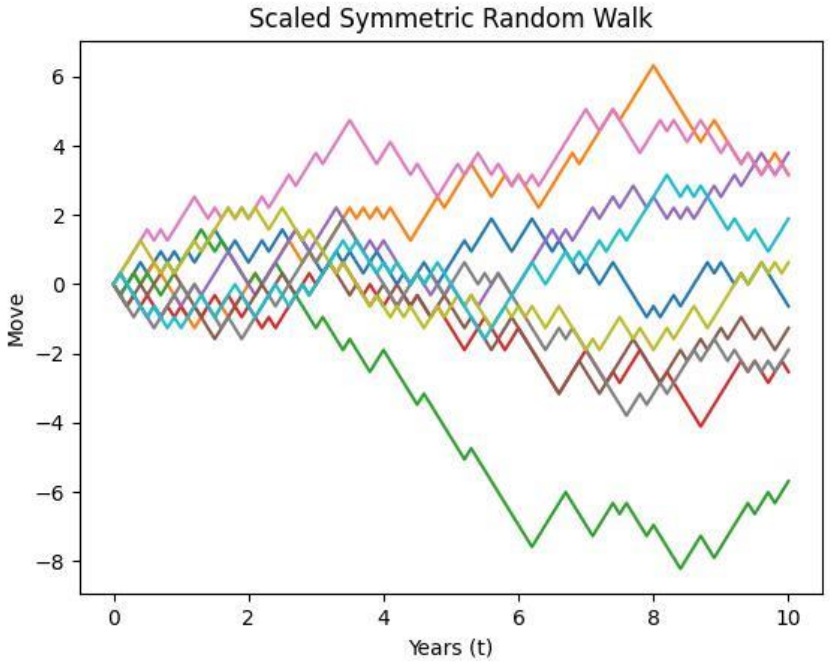

Figure 2: Simulation of Scaled Symmetric Random Walk using Python

In chapters 2.1.3 and 2.1.4 it has been stated that as we increase the number of the processes n, we are able to get Brownian motion, and this is valid according to central limit theorem. Also, as n increases the binomial distribution converges to the normal distribution with variance t.

In terms of binomial distribution, we can look at the possible permutations that these values take on given the amount of time steps that we are using. I am using the combinations formula in python to work out the binomial probabilities for each outcome. The combination formula determines the coefficient for each outcome. The permutations are the coefficient stated above times a half to the power of time steps (i.e., if we have ten years and ten steps over a year for all those years, the time steps are ten times ten equals a hundred). Next, I am using a lambda function to get each of these probabilistic outcomes. After the computation of $W^{(n)}(t)$ I am using a histogram to plot both the probabilistic outcomes and the normal distribution in Figure 3.

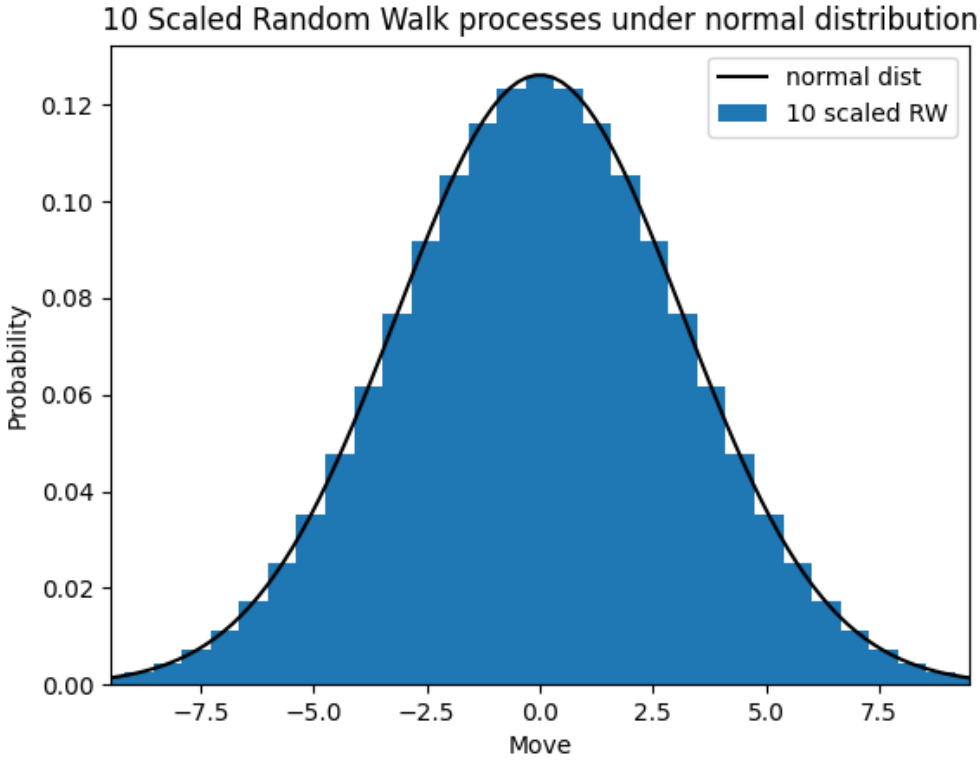

Figure 3: 10 Scaled Random Walks under the Normal Distribution

What we can conclude of this Figure is that as we increase n, we are getting closer to the normal distribution with a variance of t. So, if I increase n to 100 I should get a proper approximation of normal distribution.

We can fairly infer from Figure 4 that as n is increased, the normal distribution becomes more closely approximated. Of course, the issue that arose in this specific area was that the program's memory issues occurred as I increased n above 100 to detect convergence. Because it would take a considerable number of resources and capital, a model like this could not be used to accurately depict, for instance, the price of a stock. We therefore require a more practical and affordable approach. Thus, we can now reach the topic of studying the Wiener process.

After obtaining the Figures 3 and 4, having shown that the dual distribution converges to the normal distribution as we increase n, it follows that we need to show the derivation of the Brownian motion. Following (Shreve, 2008; Karatzas and Shreve, 1998) definition, the properties of the Brownian motion are $\mathbb{E}[W(t_{i+1}) - W(t_i)] = 0$ and $Var[W(t_{i+1}) - W(t_i)] = t_{i+1} - t_i$.

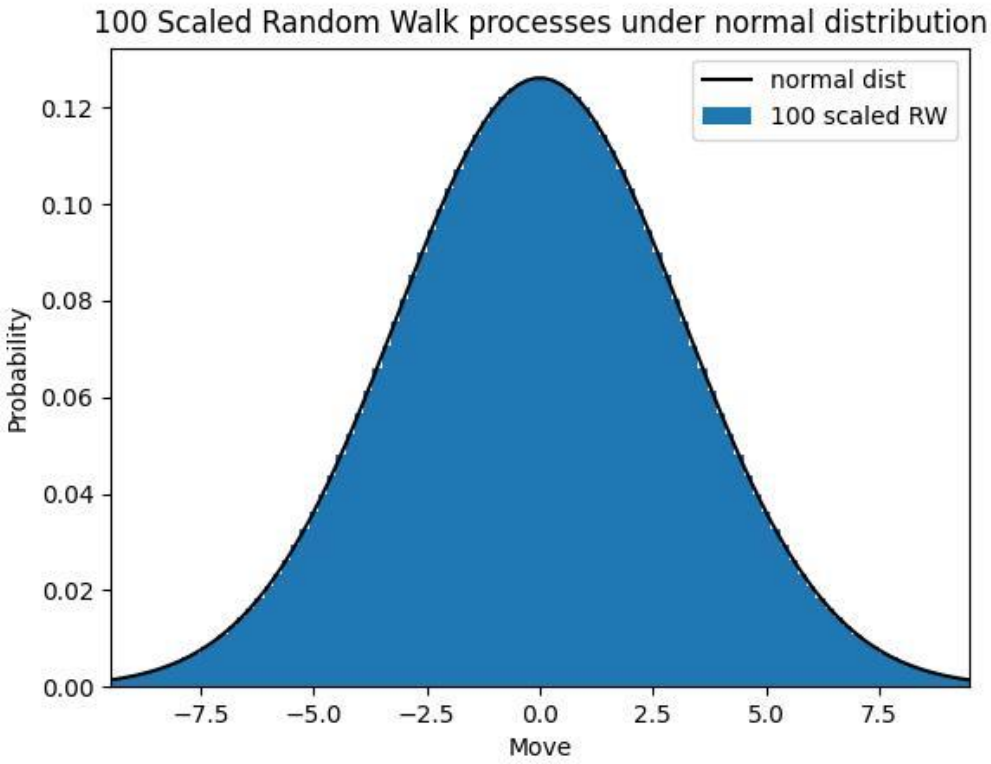

Figure 4: 100 Scaled Random Walks under the Normal Distribution

I am going to sample from the normal distribution with mean zero and variance the time difference at a particular time period. First of all, I set the parameters which are the number of simulations, the time, the steps that we want to see and the time step. I am using numpy.random.normal transposed to be able to sample this process. Then I am using an array containing zeros which is the initial values of the process, and setting the Brownian motion paths as I join the steps with the initial values and summating along axis x, using numpy.concatenate and numpy.cumsum. Then again, I

set the time array using linspace and shaping the time array in order to have the same length as the process array. The result of this process is shown in Figure 5.

We can see the Brownian motion paths that are non-differentiable for each time period, which is an extremely important case and stochastic in nature. The quadratic variation is exactly the time, same as random walk and symmetric random walk. This is shows as the time-steps value is set to ten million. Variance along these paths for one hundred simulations is the step itself. The problem is that Brownian motion can take negative values, so we need something closer to the real stock market in order to use it in models like the famous Black-Scholes model.

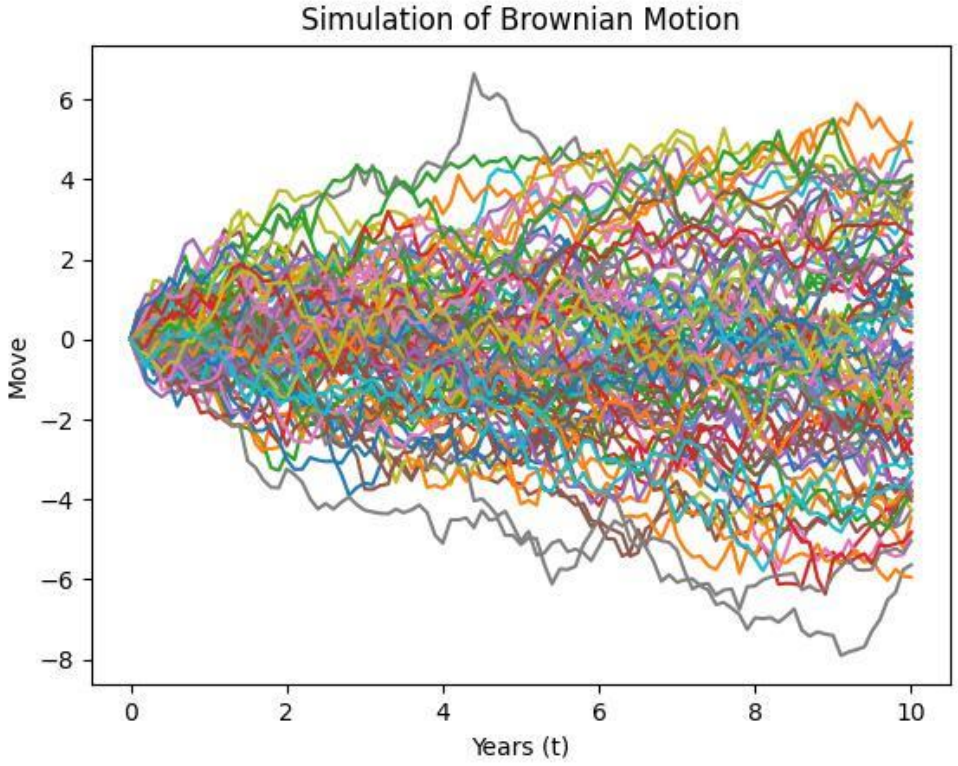

Figure 5: Simulation of the Wiener Process using Python

Geometric Brownian motion is the tool when developing various models. In the Geometric Brownian motion, we're trying to estimate the following stochastic differential equation: $dS_t = \mu dt S_t + \sigma dW_t S_t$. In Geometric Brownian motion we have an initial price which is greater than zero, μ is the drift term, σ is the constant term for volatility as a percentage point and $\sigma dW_t S_t$ is the stochastic integral component. The log of the stock price follows the normal distribution, and it has a variance that is defined by $\sigma^2 t$. The explicit expression of geometric Brownian motion is $S_t = S(0)e^{\left(\mu - \frac{1}{2}\sigma^2\right)T + \sigma W_T}$ and this is what we are trying to simulate.

I am employing the following methodology. I start by importing the python libraries that I am going to use. That is, numpy and matplotlib. The next step is to define the

parameters. So, I define the drift coefficient, the number of steps, time in years, the number of simulations, the initial stock price which has to be greater than zero and the volatility in terms of standard deviation.

The next step of the process is to simulate the Geometric Brownian Motion paths. First of we need to calculate each time step because we want to return a vector for all these paths for all these time steps. After that, I am using numpy to write the explicit form of the Geometric Brownian motion equation, and I use np.random.normal get a random number from the normal distribution with mean zero and variance the square root of the time step dt. The size of the vector is going to be the number of simulations by the number of the time steps, thus $M \times n$. This random component of the equation is going to be transposed in order to get the simulation for each time step. After that, I am going to include an array of ones. The dimension of this array will be the number of simulations and I am combining this vector with the equation S_t. In order to derive the simulation path, I need to get the cumulative product of S_t along axis zero and multiply it by the initial stock price.

The last step to get paths of Geometric Brownian motion is to define the interval in years and for this purpose I am using numpy's linspace. The reason I am doing that is that I have the time defined at one year, but I have n evenly spaced time steps. The length is now $n + 1$ because we added the array of ones. To plot the Geometric Brownian Motion we need the time vector to be the same as the S_t equation, and for this reason I am going to use numpy.full to get the same shape of S_t. The Figure 6 of Geometric Brownian Motion is given bellow.

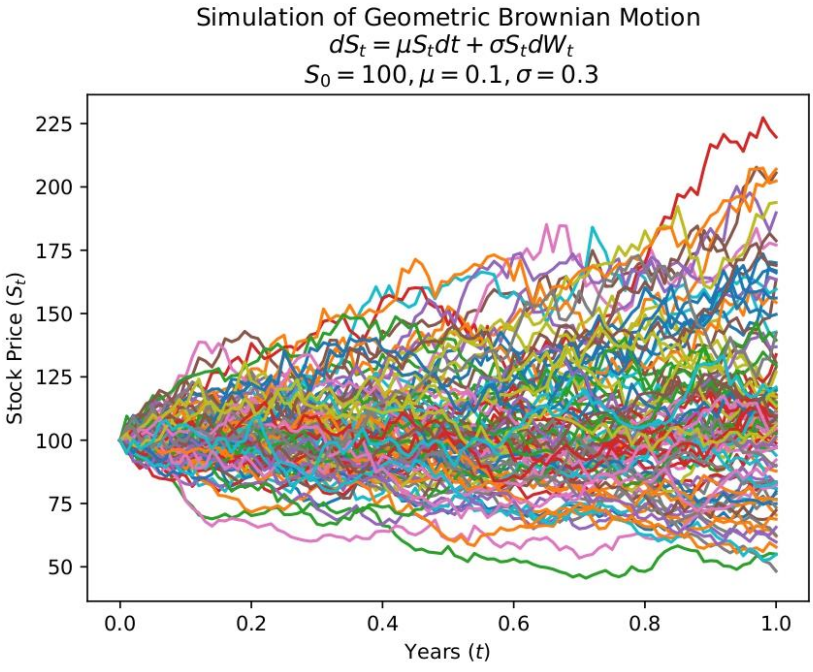

Figure 6: Simulation of Geometric Brownian Motion with Initial Stock price of 100, drift term 0.1 and Volatility 0.3

4.2 Poisson Process

To simulate the Poisson process, I use the inverse Cumulative Distribution Function[4]. This technique is called inverse transform sampling and we can use it to generate random numbers from any probability distribution using its inverse cumulative distribution. As a result, we are provided with the corresponding inter-arrival times for the distinct probabilities.

So, to simulate this process we need the inverse Cumulative Distribution Function feeding it with values from the $\boldsymbol{U(0,1)}$. The inverse Cumulative Distribution Function is given by the following formula, where $\boldsymbol{\lambda}$ is the Poisson intensity:

$$F_X^{-1}(t) = -\frac{ln\,1-t}{\lambda}$$

First, I defined the parameters utilized in both the Poisson and the Compound Poisson processes after importing the necessary libraries into Python. Instead of using a for/while loop, I defined the function that generates the process to speed up and simplify the code. I utilized the inverse CDF, which provides the time intervals, in this function. The total events are the cumulative sum. I then used Pandas to store the quantity of events in a data frame and plotted the function for a given Poisson intensity factor λ and the number of events to simulate. Figure 7 shows the Poisson process.

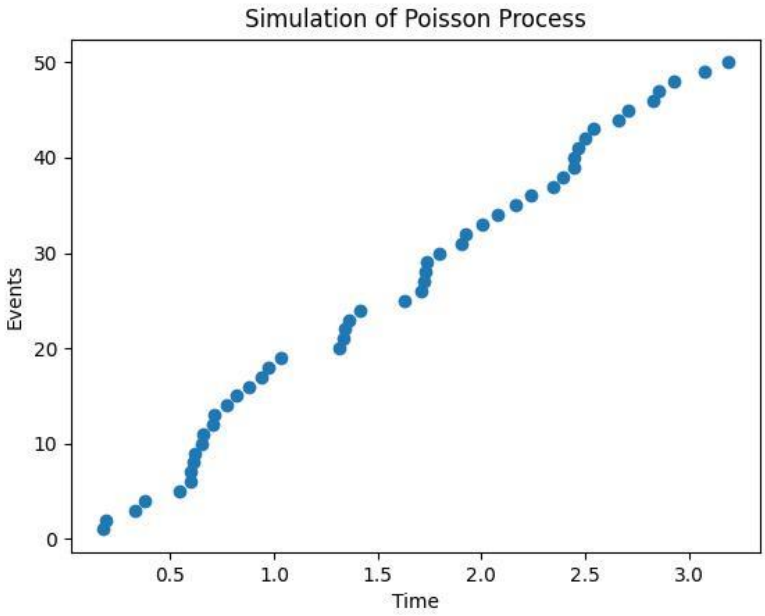

Figure 7: Simulation of Poisson Process with $\lambda = 15$ and 50 events

[4] CDF of X returns the probability that the interval of time between consecutive arrivals will be less than or equal to some value t.

To confirm that this is a Poisson process, I generate a hundred thousand loops and compare the generated Poisson process to the generated variates from the Poisson distribution. I plot the histogram for both the simulated process and the generated values from the Poisson distribution.

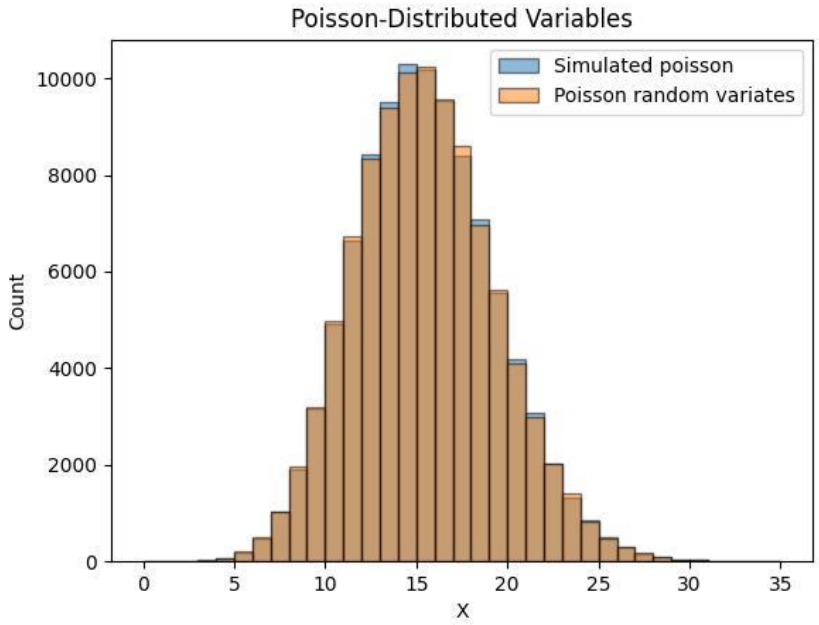

Figure 8: Poisson distributed variables and simulated Poisson process

Figure 8 demonstrates the histogram mentioned above. We may observe that the simulated process appears to fit the Poisson distributed variables fairly well.

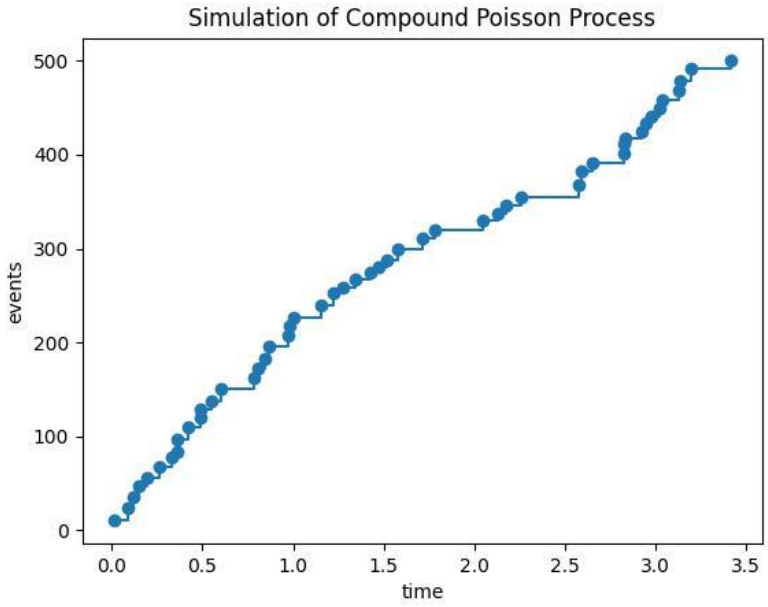

Figure 9: Simulation of Compound Poisson Process

The Compound Poisson process is an extension of the Poisson process as acknowledged in the previous Chapter. So as to simulate the Compound Poisson process we need to know a secondary distribution and generate variables distributed according to it. In this case I generated a Compound Poisson process using a binomial distribution with 20 trials and a half probability of success, $\boldsymbol{B}(\mathbf{20}, \mathbf{0.5})$.

First, I defined the binomial generator function which returns values according to $\boldsymbol{B}(\mathbf{20}, \mathbf{0.5})$. Then I defined the Compound Poisson process function which is similar to the Poisson process above. Plotting the process gives Figure 9.

5. Empirical Analysis

5.1 Stylized facts of the returns

By choosing a common factor among the characteristics noted in research of many markets and instruments, stylized facts can be obtained. The generality gained by doing this is obvious, but the generalizations one can make regarding asset returns tend to be less precise. In fact, stylized facts may not be precise enough to distinguish between various parametric models because they are frequently expressed in terms of qualitative aspects of asset returns according to Cont (2001).

The first stylized fact is the heavy tails of the returns. Compared to the normal distribution, the unconditional distribution of returns has fatter tails. Also, the returns are subscribed of asymmetry. Because of the negatively skewed distribution of the returns, it is more likely to come up with extreme negative values. Another fact is that we have absence of serial autocorrelations for non-high frequency data. Moreover, we have volatility clustering - over several days, various volatility indices show a positive autocorrelation, which quantifies the tendency of high-volatility occurrences to cluster over time.

5.2 Implied Volatility

An indicator that measures the market's perception of the possibility of price movements for a specific security is known as implied volatility. Investors can use implied volatility to predict future movements, supply, and demand, and it is widely used to price option contracts. Implied Volatility is entirely based on probability, meaning that it reflects an estimate of future prices.

Implied Volatility does not help predict the direction of prices. A high IV simply means that a stock is more likely to have a large price swing. This could mean very low or very high or both, low volatility on the other hand just means the price is not likely to make violent unpredictable moves in either direction. Although many investors will use Implied Volatility as a tool when making investment decisions, it is important to note that there is no guarantee an option’s price will follow the predicted pattern modeled

by Implied Volatility. At the same time, it helps in understanding market opinion which in turn shapes option pricing, something that Implied Volatility represents fairly well.

When the market is doing well and prices are moving up, Implied Volatility is usually lower, however when prices decline, and the market is bearish Implied Volatility increases. This is largely due to a standard belief by investors that bearish markets are riskier than bullish markets.

There are many variables affecting this measure. The two major ones are demand and time value. When demand for a particular stock or security is high, its Implied Volatility tends to increase with it. higher prices because of demand cause risk to rise as well as premiums. The inverse is true when demand is low, Implied Volatility falls and the option's price becomes cheaper. The time value of options, or the amount of time left until the options expire is another major influence. An option expiring relatively quickly will have low Implied Volatility, while the one expiring well into the future will have high Implied Volatility because there is more time baked into the option's price leaving it exposed longer to time, giving it a higher chance to experience changes and react to market events.

The Black-Scholes formula can be used to estimate implied volatility. With the Black-Scholes model, we can accurately price a call or put option based on market sentiment given a stock price, the strike price, the risk-free rate, the time to expiration, and a constant measure of volatility in standard deviation. We can therefore work backwards and determine what the market believes the volatility to be by using the call price formatted by the transaction in the market and all the variables in the Black-Scholes model.

5.3 Black and Scholes Model

The price of any derivative reliant on a non-dividend paying stock must satisfy the Black and Scholes (1973) differential equation. The model's justifications call for creating a riskless portfolio with positions in both stocks and derivatives. The returns from the portfolio must be the risk-free interest rate in the absence of arbitrage opportunities. The Black-Scholes differential equation results from this.

Since both the stock price and the derivative price are influenced by the same fundamental source of uncertainty—stock price fluctuations—a riskless portfolio can be created. The price of the derivative has a perfect short-term correlation with the price of the underlying stock. The entire value of the portfolio at the end of the short period of time is known with certainty when a suitable portfolio of the stock and the derivative is constructed. This is because the gain or loss from the stock position invariably offsets the gain or loss from the derivative position.

According to Hull (2018), the assumptions of the Black-Scholes model are the following. First of all, the stock price follows the Geometric Brownian Motion (see

Chapter 3) with both mean and volatility being constant. Second, it is acceptable to sell securities short with a full utilization of the proceeds, and there are no taxes or transaction fees, the division of all securities is perfect. Also, there are no dividends during the life of the derivative and no riskless arbitrage opportunities. Lastly, security trading is continuous, and the risk-free interest rate is constant and the same for all maturities.

5.3.1 Black – Scholes differential equation

As stated above, the stock price follows the Geometric Brownian motion. The differential equation of the stock price is

$$dS = \mu S dt + \sigma S dz$$

Assuming that f is the call option price of the stock and that f is a function of the stock price and time t, then by applying the Itô's lemma (see chapter 3) we should get the following formula:

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

The discretization of the above formulas for a small-time interval δt:

$$\delta S = \mu S \delta t + \sigma S \delta z,$$

$$\delta f = \left(\frac{\partial f}{\partial S}\mu S + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2\right)\delta t + \frac{\partial f}{\partial S}\sigma S \delta z$$

We define V as the value of the portfolio:

$$V = -f + \frac{\partial f}{\partial S}S$$

The investor is short one derivative and long $\frac{\partial f}{\partial S}$ shares. If we differentiate both sides of the equation in discrete time we should get:

$$\delta V = -\delta f + \frac{\partial f}{\partial S}\delta S$$

Then substituting δf and δS in the above equation we get:

$$\delta V = -\left(\frac{\partial f}{\partial S}\mu S + \frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2\right)\delta t - \frac{\partial f}{\partial S}\sigma S \delta z + \frac{\partial f}{\partial S}(\mu S \delta t + \sigma S \delta z) \Leftrightarrow$$

$$\delta V = -\frac{\partial f}{\partial S}\mu S \delta t - \frac{\partial f}{\partial t}\delta t - \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 \delta t - \frac{\partial f}{\partial S}\sigma S \delta z + \frac{\partial f}{\partial S}\mu S \delta t + \frac{\partial f}{\partial S}\sigma S \delta z \Leftrightarrow$$

$$\delta V = -\frac{\partial f}{\partial t}\delta t - \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 \delta t \Leftrightarrow$$

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

Note that the above equation does not include the stochastic variance term, so we could say that the investor's portfolio is risk-free in discrete time.

According to the assumptions outlined previously, the portfolio must instantly generate the same rate of return as other short-term risk-free securities. Arbitrageurs could not make a risk-free profit by lending money to the portfolio if it earned more than this return, but they could do so by shorting the portfolio and investing in risk-free securities if it earned less. Thus:

$$\delta V = rV\delta t$$

Substituting V and δV in the above equation, gives:

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

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

$$-\frac{\partial f}{\partial t} - \frac{1}{2}\frac{\partial^2 f}{\partial S^2}\sigma^2 S^2 = -rf + r\frac{\partial f}{\partial S}S \Leftrightarrow$$

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

The above equation is the Black-Scholes-Merton partial differential equation, where r is the risk-free rate. The boundary conditions for a European call option imply that the payoff is $\max(S_T - K, 0)$. if $S_T > K$ then the payoff is bigger than zero. If the opposite is true, then the payoff is zero. Similarly, the put option payoff is $\max(K - S_T, 0)$. K is the exercise price of the option.

The call and put option pricing formulas of the Black-Scholes model are:

$$c = S_t N(d_1) - Ke^{-r(T-t)}N(d_2)$$

$$p = Ke^{-rT}N(d_2) - S_t N(-d_1)$$

Where $N(x)$ is the cumulative probability function for a standardized normal distribution. d_1, d_2 are

$$d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{(T-t)}}$$

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

The likelihood that the stock price at expiration will be higher than the strike price (for out-of-the-money options) or continue above it is represented by $N(d_2)$ (for In-the-Money options). In other words, for a call option holder, $N(d_2)$ is the probability of exercising the option, whereas for a put option holder, who anticipates that the stock will be below K at expiration, $N(d_2)$ or $1 - d_2$ is the likelihood of exercising the option.

5.3.2 Total Energies (TTE) Option Pricing

In this section I will demonstrate a call/put option pricing for TTE, which is a company operating in integrated gas, Renewables and power, exploration and production, refining and chemicals and marketing and services.

The implementation of the Black and Scholes formula gives the call and put option price of TTE. The data[5] put into the code are given below along with the results.

	3-month maturity	1-month maturity
Stock Price	$60.26	$60.60
Strike Price	$62.50	$60
Time to expiry (Current date: 16/11/2022)	93 Days (Expiration Date: 17/02/2023)	30 Days (Expiration Date: 16/12/2023)
Interest rate	4.31%	3.77%
Implied Volatility	33.11%	31.28%
Dividend Yield	4.79%	4.79%
Last call option price	$2.46	$2.40
Black-Scholes Call price	**$2.99**	**$2.43**
Black-Scholes Put price	**$5.28**	**$1.88**

Table 1: Black and Scholes Call/Put prices for Total Energies (TTE)

In table 1 we can see the stock price of the asset, the strike price, the time to expiry, the interest rate, volatility, dividend payment, the last call option price, and the results from the Black-Scholes formula. The first column refers to one-month maturity of the option.

The stock price used to derive the Black-Scholes prices was $60.26 and the strike price was $62.50. in this case the call option is "Out-of-the-Money" and the put option is "In-the-Money". In the code I needed the annualized time, so I divided 93 days with 365 days. The interest rate is the three-month treasury bill, which was 4.31% at 16/11/2022. Lastly, the dividend payment was 4.79% and the last call price was $2.46. The call option price according to the Black and Scholes model has to be $2.99 but the

[5] The code used to produce the prices of the options is given in the Appendix. All the data used for this section was provided by Yahoo Finance.

observed price was $2.46. This could mean one of two things: first, the Black and Scholes model did not accurately approximate the realistic price. Due to the maturity being far later than the current date, this could be a miscalculation. The call option's lower value than expected is the second matter. This could be a buy signal to take advantage of the lower price.

In the case that the maturity of the option is one month, the stock price is $60.60, and the strike price of the option is $60. In this case the call option is “In-the-Money” and the put option is “Out-of-the-Money”. The one-month treasury bill is 3.77% and the implied volatility of the option is 31.28%. The dividend payment remains the same as the previous case, and the last call option price was $2.40. In this case we observe that the Black and Scholes formula approximated better the last seen call option price. This may have happened because the time to expiration is far less than the previous case, so we can get a more realistic approximation.

5.3.3 Sensitivity Analysis for Total Energies (TTE) Derivatives

In this part of the analysis, we are also interested in seeing how the derivative values vary when the input variables for the Black-Scholes equation change. To accomplish that, I run a sensitivity analysis changing the stock price, strike price, time, interest rate, volatility, and dividend yield in a ceteris paribus context. The initial data used to simulate the figures bellow are those in Table 1 with 1-month maturity. The code used to generate the Figures is given in the Appendix.

Figure 10: Stock Price Change and Black-Scholes Call Value

To generate the slope of the curve in Figure 10, I started with the price being $55 i.e., $5.6 below the true price to check the call's value "Near-the-Money". Then I added an $1 increase until the price hits $80, ceteris paribus. We can see in Figure 10 that as the value of the asset rises, so does the value of the call option. This makes intuitive sense because when we have a long position on a call option, we are making an estimate that the value of the underlying asset will rise, and we want that to happen, if we are just buying the call option by itself. Also, we can see that in the range [$55, $60] there is a different slope than the one in ($60, $80). This happens because the call option is "Near-the-Money". Options contracts that are at or near the money often cost more than those that are out-of-the-money, where the price of the underlying instrument is much higher or lower than the strike price. If near-the-money options are slightly out of the money, they have intrinsic value; however, if they are slightly in the money, they have both intrinsic and extrinsic value.

Figure 11: Stock Price Change and Black-Scholes Put Value

The negative slope of put option value in Figure 11 makes intuitive sense as well. A put allows us to sell the underlying asset, as the price of TTE rises, the value of our put falls. But as the TTE stock price falls in value, then the value of our put option will increase because we can sell it at a given price and avoid having these losses. That's also why put options are used as insurance in case the value of the stock does start to fall.

Figure 12: Strike Price Change and Black-Scholes Put, Call Value

In Figure 12 the strike price for the call option is $60. Again, I started the price at $55 and put in a $1 increment. As we start to increase the strike price for a call option, the value of the call option is falling. Given strike price is the point at which we can buy Total Energies' stock. If TTE shares were trading at $55 but the call option that we hold is $60, we would not exercise it because we can just buy it in the spot market, and the call option would expire worthless. However, if the TTE's share is trading at $58 and we have a call option with a strike price $55, we would just purchase shares at $55 and sell at $55, making $3/share profit.

For the put option, as the strike price increases the value of the option increases as well (Figure 12). E.g., TTE's stock is trading at $65 but I bought a put option that allows me to sell TTE's shares at $75. If that is the case, I would sell the shares I am holding making a $10 profit per share.

Volatility is a very intriguing input since we cannot actually see it in the market. Figure 13 shows that as volatility rises, both the value of a call option and the value of a put option rise as well. In both situations, the value of the derivative rises as volatility does. There is a rationale for that. Our option could end considerably above the current strike price or well below the current strike price and could generate a big return for a trader, making it more desirable if there is increased volatility. The volatility term I used to generate the Black-Scholes values, is the implied volatility, which was 31.28%. the starting value of the volatility is 1% and the increment I apply is also 1% till 50%.

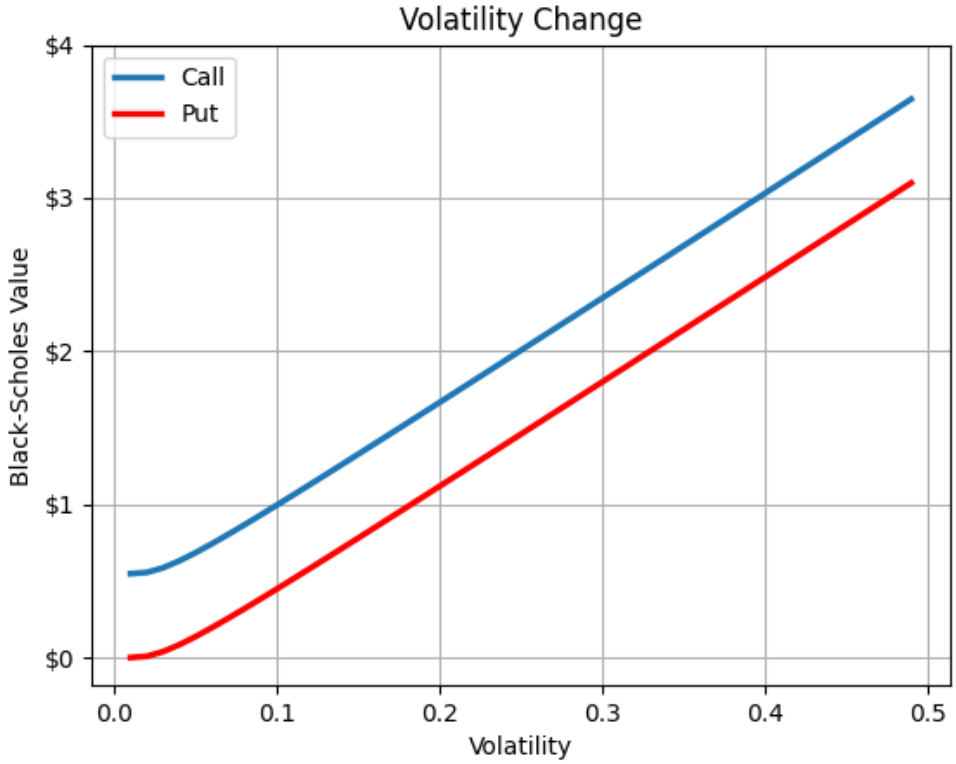

Figure 13: Volatility Change and Black-Scholes Put, Call Value

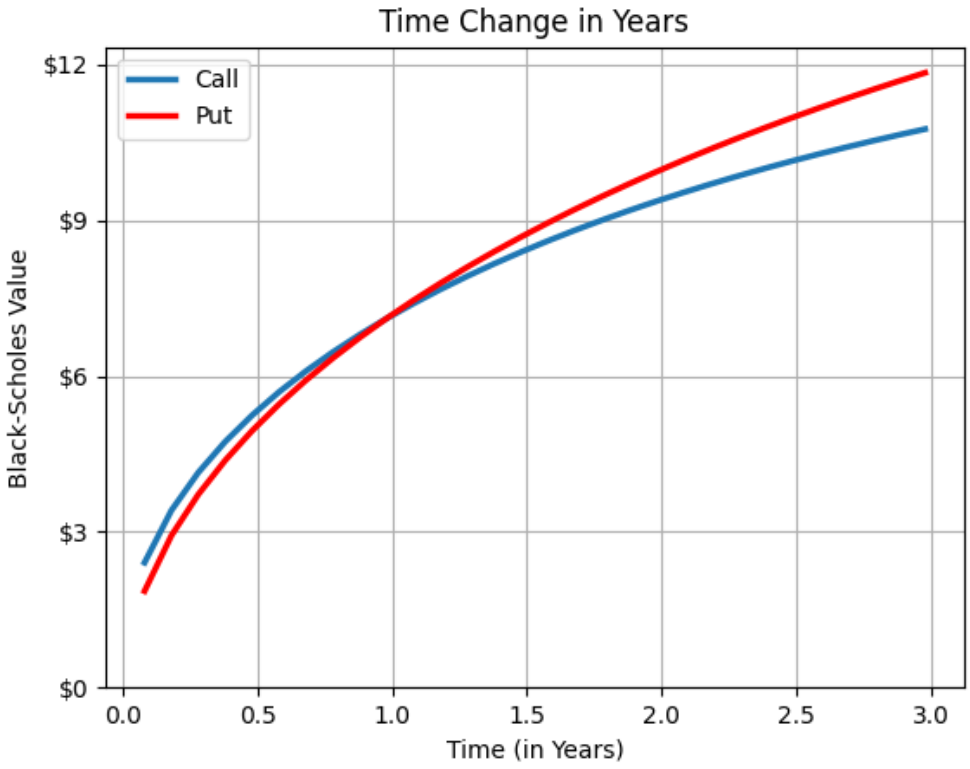

Figure 14: Time Change in Years and Black-Scholes Put, Call Value

The next input to change in a ceteris paribus context[6] is time to expiry. In order to generate Figure 14, I expressed the time to expire in annualized terms. That is, the derivatives' expiration was 30 calendar days, so $T = \frac{30}{365} = 0.08219$ years. The starting time is the first month, where the call's price is higher than the put's, and the increment is 0.1 or 1.2 months or 36 days. The call option is "In-The-Money" and the put option "Out-Of-The-Money".

These curves both have positive slopes. This makes sense as, with longer time, the price can move more favorably up or down in the direction we wanted. However, it is very unlikely that the price of TTE will move widely in a very limited time frame, which is why time in general is more valuable.

The next parameter to change is interest rate. For interest rate I used the 1-month treasury bill because the option's time to expiry was 1 month. The interest rate in 16/11/2022 was 3.77% but in Figure 15 the starting value is 0.1% with 1% increment.

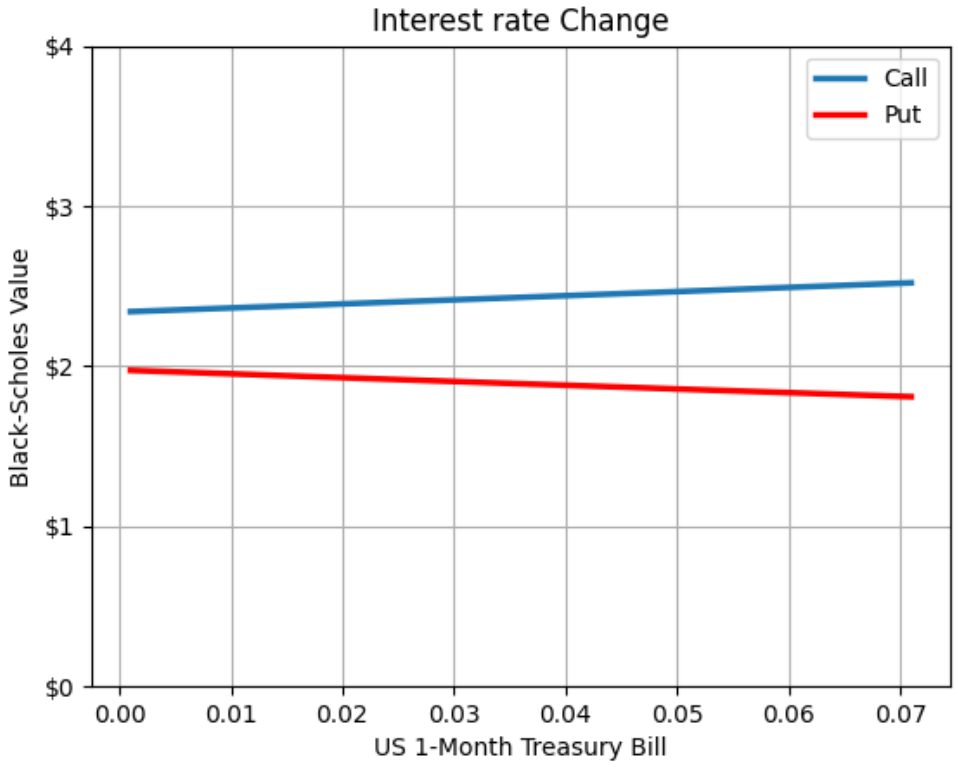

Figure 15: Interest rate Change and Black-Scholes Put, Call Value

As the interest rate increase we can see that the value of the call option increases because in this case, holding a call option is akin to being short bonds and as interest rates increase bond values decrease. On the other side we have put options

[6] This indicates that the research also assumes that there are no dividend payments. The Figure would be different if dividend payments were made. (Hull, 2018)

decreasing as the interest rates increase since going long a put is similar to going long a bond. Though the slope of the curves is very elastic – almost perfectly elastic – so we cannot certainly say that interest rates are affecting much TTE's derivatives.

The last parameter to change is dividend yield. The dividend yield of the stock is 4.79%, though in Figure 16 it starts from 0.0% and a 1% increment is applied until it reaches 7.5%. The value of a call option declines as dividend values rise. This is due to the fact that following a dividend payment, the stock's value will drop or will likely drop. The same is true of a put value. The value of the put will rise because we anticipate a decline in the share price once it goes ex-dividend.

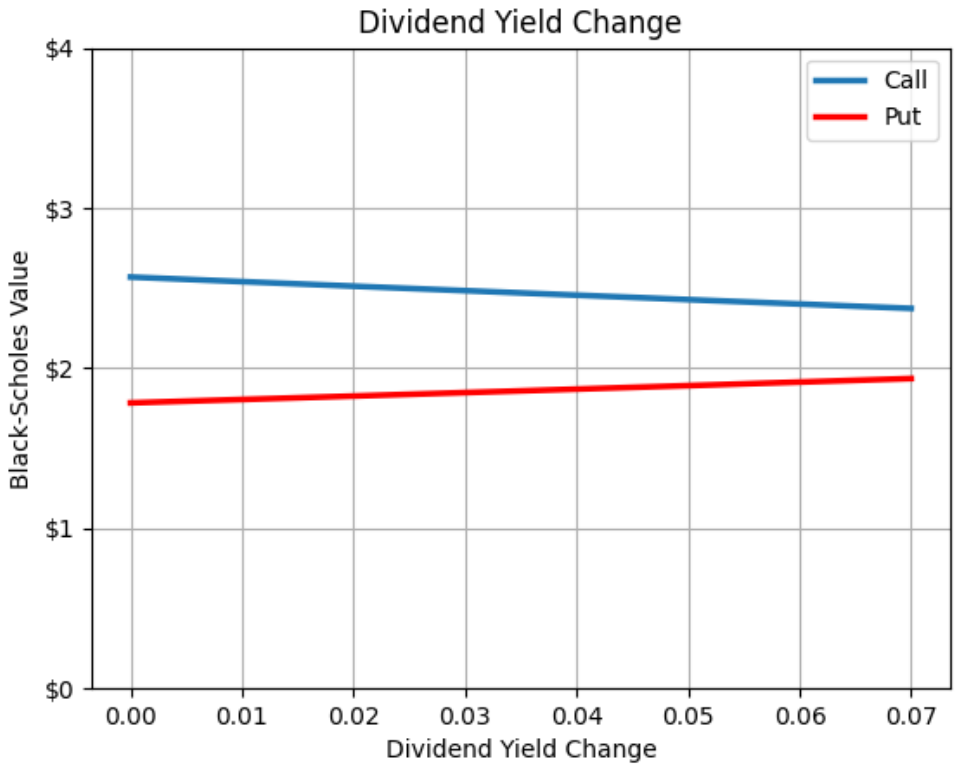

Figure 16: Dividend Yield Change and Black-Scholes Put, Call Value

5.4 Heston SV

The most significant stochastic model, which served as the basis for a number of versions in contemporary literature, is that of Heston (1993). When the underlying is correlated with volatility, Heston suggests a stochastic volatility model that is not dependent on the Black and Scholes (1973) model. He also further adjusts his model to take into consideration stochastic interest rates. Heston specifically makes the assumption that returns are produced by the following relationship in order to explain the dynamics of variance using a square root technique.

For the price of a European call option when the underlying assets are correlated with a volatility stochastic process, Heston has suggested a stochastic volatility model with a closed-form solution.

The first differential equation of the model describes the stock price. We can think of this equation as a random walk with heteroskedastic variance term. Suppose that the price of a stock in the spot market is S_t, where t is the specific time period. μ is the rate of return and v_t is the instantaneous variance at a specific time period. The process followed by the stock price is as follows:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^1$$

The second differential equation of the model describes the variance process which is also embedded in the first differential equation. The variance equation utilizes the logic of mean reversion – Ornstein-Uhlenbeck process. θ is the long-run average price variance and κ is the rate at which variance reverts to the long-run average price variance. ξ is the volatility of volatility v_t. The variance process is as follows:

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t} dW_t^2$$

W_t^1, W_t^2 are the Wiener processes whose properties are discussed in Chapter 3. W_t^1 is the Wiener process of the first differential equation and W_t^2 is the Wiener process of the second differential equation. The model requires that the two distinct Weiner processes that comprise the randomness be correlated, with instantaneous constant correlation. Hence,

$$\mathbb{E}[dW_t^1 dW_t^2] = \rho dt$$

The dynamics of the model under a risk-neutral measure is as follows:

$$dS_t = rS_t dt + \sqrt{v_t} S_t dW_t^1$$
$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t} dW_t^2$$

Where the only change is in the stock price process. r is the risk-free rate which replaces μ.

Heston (1993) derives a closed form solution for European option pricing, using the characteristic functions. Those characteristic functions offer positive variances as opposed to Ornstein – Uhlenbeck process where we can get negative variances.

5.4.1 Simulation of the Heston SV

As discussed above, Heston SV offers a closed form solution using the characteristic functions for European Option pricing, so there is no need to discretize the model in order to get the option prices in a certain time frame. Though, there are certain cases where we would like to use Monte-Carlo simulation. In these cases, we should discretize our model to obtain outputs and one way to discretize the model is the

Euler's discretization. The other one is Milstein's discretization. For this purpose, I will use the Euler's discretization (not Log-Euler) as described in Broadie and Kaya (2006), Van Haastrecht and Pelsser (2010), and Rouah (2013). So, the discretized model is:

$$dS_{i+1} = S_i e^{\left(r - \frac{v_i}{2}\right)\Delta t + \sqrt{v_i}\Delta t W_{i+1}^1}$$

$$v_{i+1} = v_i + \kappa(\theta - v_t)\Delta t + \xi\sqrt{v_i}\Delta t W_{i+1}^2$$

The downside of Euler's discretization is that we can get negative variances, so we will need to keep only the positive in the algorithm.

Initial Asset Price (S_0)	100
Time in Years (T)	1
Number of Steps (N)	252
Number of Simulations (M)	1000
Risk-free rate (r)	5%
Mean reversion of variance (κ)	3
Long-term mean of variance (θ)	4%
Initial variance (v_0)	6.25%
Correlation between returns and asset prices (ρ)	0.7
Volatility of volatility (ξ)	60%

Table 2: Heston SV Simulation Parameters

The methodology is straightforward. Using linear algebra, we are able to get the results much faster. First, we need to define a function in python that takes the inputs of the table 2 and outputs the stock price and the variance.

The first sub step is to initialize three parameters, dt, μ, ρ. dt is the time in years over the number of the steps. The number of the steps is 252 because those are the stock market trading days, though the simulation will not change dramatically if we just substitute it with 365 days. Because we need to simulate a random multivariate process between two Wiener processes, we need to set:

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$

Where μ is a vector with the expectations of the two processes which are zero, and Σ which is the Variance-Covariance matrix describing the relationship between these two processes. The main diagonal is the variance of those two processes and the correlation is 0.7 from table 2.

The second sub step is to make two arrays that store the asset prices and variances and sample correlated wiener processes with size $(N + 1 \times M)$.

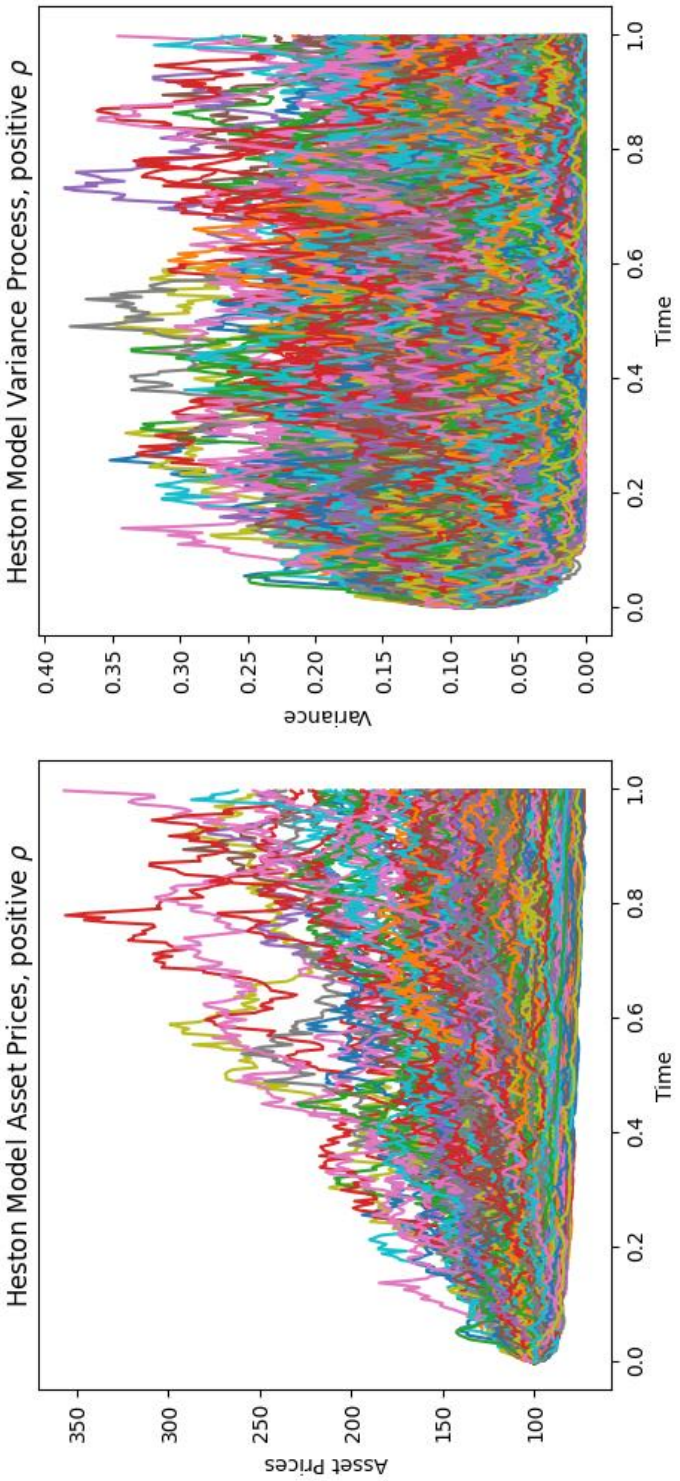

Figure 17: Heston SV Simulation with $\rho = 0,98$

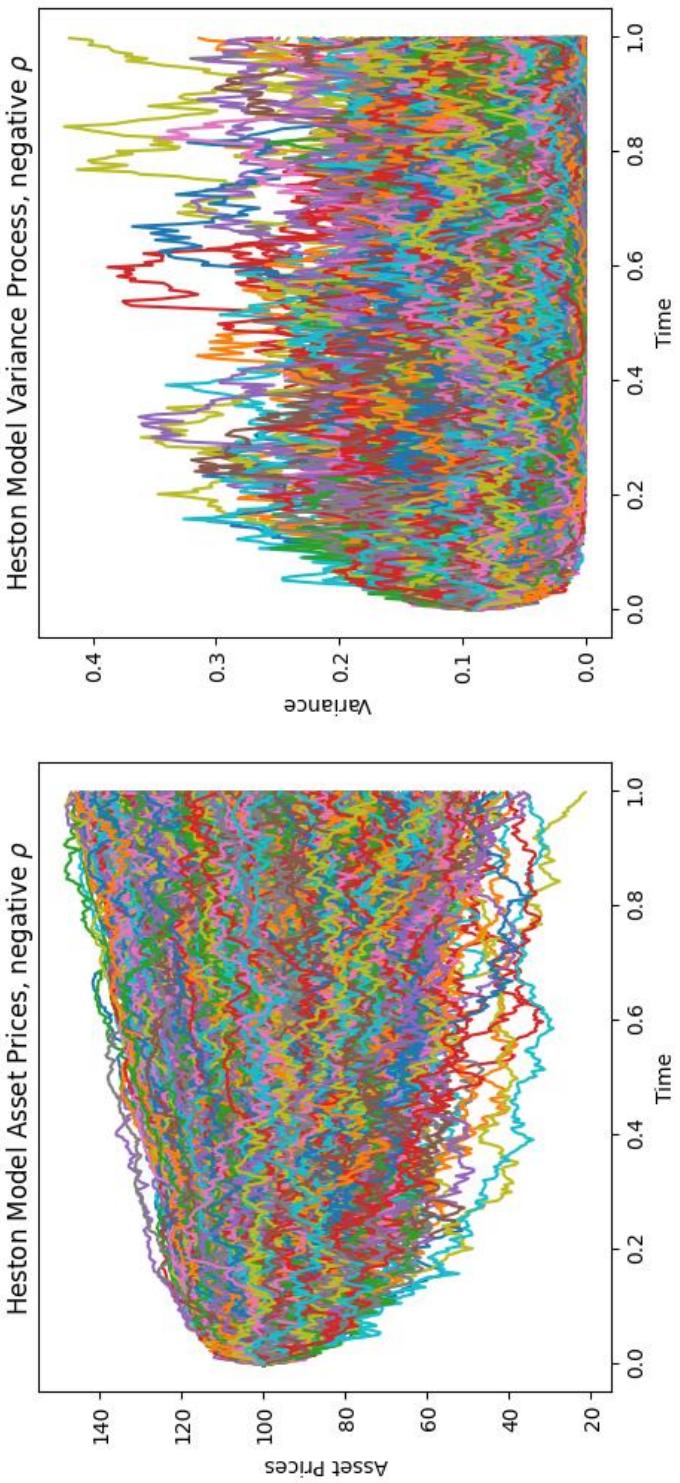

Figure 18: Heston SV Simulation with $\rho = -0,98$

To sample correlated Wiener processes, we are using the multivariate normal distribution with mean μ and covariance Σ with size $(N \times M)$.

Lastly, we apply the recursive function with a simple loop from 1 to $N + 1$. The Asset price and the variance functions are lists and the formula applied is the Euler discretization. For the variance recursion we need to use a maximum because, as previously mentioned, the Euler discretization may give negative variances, which is something we need to avoid.

For visualization purposes I have got two cases for the correlation. A very positive one, $\rho = 0{,}98$ and a very negative one, $\rho = -0{,}98$. Figures 17, 18 show the simulated Heston SV for each scenario. Figure 19 shows the asset price density

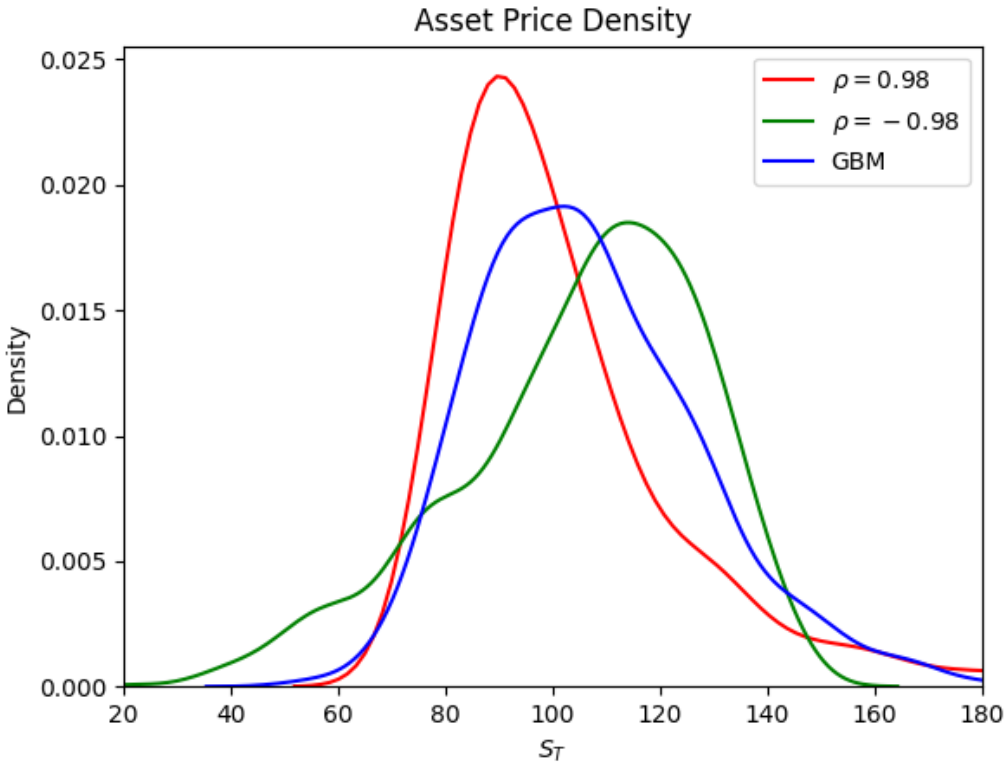

Figure 19: Asset Price Density under different correlations in Heston SV

The red curve shows the almost perfect correlation between returns and volatility and as we can see it has a very long tail. The leverage effect captured by Black (1976) was the opposite effect. This is where high volatility led to negative asset price returns. We can see that here in the negatively skewed distribution of $\rho = -0{,}98$. So the Heston SV model can capture the leverage effect of Black (1976) and that is a really desired property of the model.

5.4.2 Heston SV Calibration – Total Energies (TTE) Stock

In order to calibrate the Heston SV model, we need data from the financial markets[7]. Once the model is calibrated can be used to describe the market's characteristics, but also be used for predictions.

	TTE returns
Observations	251
Mean	0.118336
Max	7.783977
Min	-7.929653
Standard Deviation	2.193419
Skewness	-0,3801763
Kurtosis	3,986263
Jarque – Bera p-value	0,0041
Shapiro – Wilk p-value	0,00914

Table 3: TTE Returns – Descriptive Statistics

Table 3 represents the descriptive statistics of TTE's stock price. The observations are 251 for the period December 1, 2021 – November 30, 2022 (one year of data).

The mean return is approximately 11.83%. The Jarque – Bera test for normality of the returns indicates that are not normally distributed. Though, the sample is not sufficient for Jarque – Bera test to be precise. To avoid any statistical mistake, I performed the Shapiro – Wilk test, which also indicates that returns are not normally distributed. The value of kurtosis indicate that we have a very heavy tail as it is greater than 3. Skewness indicates a distribution skewed to the left as it is less than 0.

Figure 20 shows the Stock price time series and returns. The trend of the stock cannot be identified immediately, as there are periods with positive linear trend (December 2021 – March 2022) and periods with negative linear trend (June 2022 – July 2022). This would easily be tested with Chow's test for structural breaks, but it would be out of the scope of this chapter. Looking at the returns graph in Figure 20, we cannot seem to find any volatility clustering effects, except March 2022.

The calibration process[8] of Heston SV model is conducted in two phases. The first phase of the process is to calculate the returns, residuals, realized variance, expected variance and the log likelihood function. The returns are calculated by log-differencing the stock price. Then the mean of the returns is calculated, and that is the drift term μ of our model. The residuals of the model are calculated taking the difference between the return and the mean (drift term). Realized variance is calculated by

[7] The data are derived from Yahoo Finance. The descriptive statistics (Table 3) is produced using Stata 14. Figure 20 is produced by Stata 14 as well.

[8] The calibration process is conducted in Microsoft excel using Solver and non-linear gradient descent algorithm.

squaring the residuals. The expected variance is calculated. The first value of expected variance is just the long run variance. The next day and so on, the variance will change given kappa and theta (κ, θ). So, the expected variance at time t is the realized variance plus kappa multiplied by the difference between theta and the realized variance of the previous day. So:

$$Expected\ Variance_t = Realized\ Variance_{t-1} + \kappa(\theta - Realized\ Variance_{t-1})$$

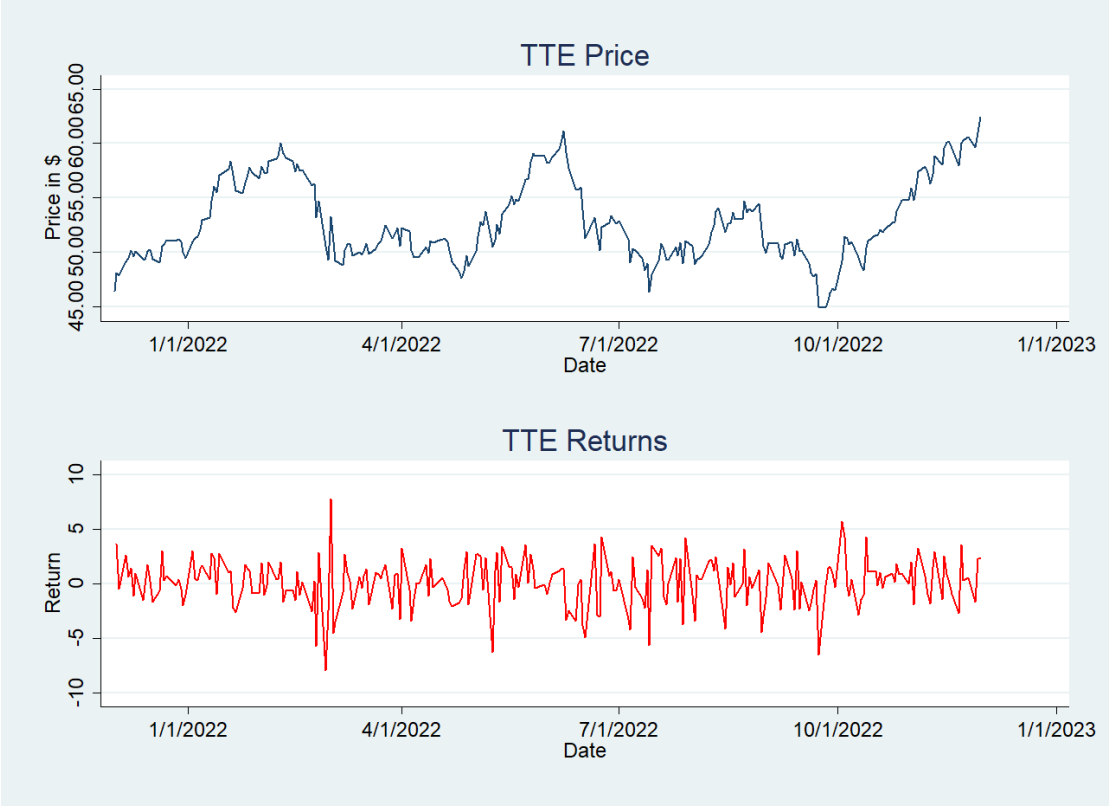

Figure 20: TTE Price and Returns

Finally, before the optimization phase Log-Likelihood must be calculated. Log-Likelihood is estimated using the joint probability density:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} e^{-\frac{\left(\frac{x-\mu_x}{\sigma_x}\right)^2 - 2\rho\left(\frac{x-\mu_x}{\sigma_x}\right)\left(\frac{y-\mu_y}{\sigma_y}\right) + \left(\frac{y-\mu_y}{\sigma_y}\right)^2}{2(1-\rho^2)}}$$

After estimating the log-likelihood values, I sum them up. At the same time, I use the IFERROR function in excel to speed up convergence and relax constraints on variables, so I do not need to put them manually in Solver. So, if there is an error, the function will give me a very large negative value that would teach the Solver to not venture to domains where we have errors, or not well-behaved functions.

So, the starting phase parameters that will later be calibrated are shown in table 4:

Long-run Variance $(\boldsymbol{\theta})$	4,811089
Variance mean-reversion $(\boldsymbol{\kappa})$	1
Volatility of volatility $(\boldsymbol{\xi})$	8,362627
Correlation $(\boldsymbol{\rho})$	0
Starting Variance $(\boldsymbol{v_0})$	4,811089
Drift $(\boldsymbol{\mu})$	0,118334
Starting Log-likelihood	-1439,5593519793

Table 4: Heston SV Pre-Calibration Values

The long-run variance equals to the sample variance of the residuals. The pre-calibration value of kappa is assumed to be 1. The starting specification assumes constant variance, which means that we have perfect mean reversion. No matter what happens, the next day variance reverts to the long run value. If $\kappa < 1$ that would imply that the mean reversion process is not perfect – variance reverts to the long run value after many days or even weeks. The volatility of volatility is calculated by taking the sample standard deviation of realized variance. Also, the correlation between returns and variance is assumed to be zero – variance shocks are independent. The starting variance of the specification is equal to the long run variance. The drift term is just the mean of the returns. The starting log-likelihood value of the specification is -1439.56. As the specification gets optimized, we manage to increase the log-likelihood.

As the specification of the model assumes kappa to be 1, the expected variance will be the same for each time step. Changing it, gives us a stochastic specification of the model, as variance changes randomly.

The second phase of the calibration is the optimization of log-likelihood with respect to the parameters in Table 4.

Long-run Variance $(\boldsymbol{\theta})$	4,711798009
Variance mean-reversion $(\boldsymbol{\kappa})$	0,85645659
Volatility of volatility $(\boldsymbol{\xi})$	8,25773478
Correlation $(\boldsymbol{\rho})$	-0,263081559
Starting Variance $(\boldsymbol{v_0})$	13,72501634
Drift $(\boldsymbol{\mu})$	0,18
Optimized Log-likelihood	-1426,9898203758

Table 5: Heston SV Calibrated Parameters

Because I used the IFERROR function there is no need to manually put the constraints in Solver. The optimization problem we are facing can be written as follows:

$$\max Log\ Likelihood\ s.t.$$

$$\begin{cases} \theta, v_0, \xi, \mu > 0 \\ 0 < \kappa < 1 \\ -1 \leq \rho \leq 1 \end{cases}$$

Solver using Non-Linear Gradient Descent gives the optimized values in table 5. The results are quite interesting. The correlation between returns and variance is negative, meaning that innovations to mean and variance is inversely related. This is somewhat expected as the underlying is a stock, which holds higher risk than e.g., bonds. That could also represent flight-to-quality or shocks to overall certainty that represents investors discouraged from holding a risky asset which is a stock and preferring a riskless asset such as a bond. The drift is almost 63% larger than the uncalibrated mean and the mean reversion is very high, meaning that almost instantly the variance returns to its long-run level. The long-run variance is quite the same as the uncalibrated one, but the starting variance is much bigger from the corresponding uncalibrated, almost 185% higher. Lastly, the log-likelihood has been increased by almost 12.58 units.

However, we have to test if this increase in Log-likelihood is statistically significant. To answer this question, I apply a Likelihood ratio test:

$$LL = 2(LL_0 - LL_1) \sim \chi_6^2$$

Where LL_0 is the uncalibrated Log-likelihood and LL_1 is the calibrated Log-likelihood. The distribution following this statistical test is chi-squared with six degrees of freedom.

Starting Log-likelihood	-1439,5593519793
Optimized Log-likelihood	-1426,9898203758
Likelihood ratio	25,13906321
p-value	0,000

Table 6: Heston SV Calibration Evaluation

The p-value of the test shows strong statistical significance at 1% significance level. This means that the Heston model adds explanatory power over the constant variance specification. So, the stochastic volatility can explain more accurately the variation of Total Energies' Stock price at the given time period than the constant volatility models.

Though accurate, Heston SV cannot be used to explain prices for all assets in the financial markets. The same analysis has been used for NASDAQ Composite for the same time period, but Heston SV was rejected over the constant volatility specification. Though the calibration process has given no errors, the Likelihood ratio test indicated that the constant volatility model has better fit over the data with a p-value of 0.99.

5.4.3 Heston SV – Forecasting Total Energies (TTE) Stock Price

In order to forecast the stock price of Total Energies, we need the calibrated parameters of the previous section (see table 5). The prediction is conducted for the next day, 1/12/2022. The formula used for this purpose is the Euler discretization of the previous section and the whole process is performed in python.

In order to get a good approximation, I simulate the process 1000 times. Therefore, the predicted stock price is a (1000×1) matrix containing different prices for different Wiener process values.

Stock price$_t$ (30/11/2022)	$62.42
Actual closing price (01/12/2022)	$61.70
1-year Treasury Bill	4.66%
Realized variance$_t$ (30/11/2022)	4.563114037
Mean predicted closing price (01/12/2022)	$61.98
Mean Absolute Percentage Error (MAPE)	0.7748%
Root Mean Square Error (RMSE)	0.6015
Jarque – Bera (predicted price)	0.05611
Jarque – Bera p-value (predicted price)	0.97233

Table 7: Data and Results from Heston SV Forecast

The process from which I got the predicted price is similar to that on the simulations section. The mean matrix is filled with the calibrated drift terms, as the data showed that the calibrated drift term performs better than the specification with zero mean. The covariance array is the same as the simulations' section, but the covariance values are the calibrated rho.

Then I fill two particular arrays (Stock price and variance) with the starting values. The size of those matrices is (252×1000) as we have a thousand simulations of the closing price the next day and 252 trading days. The stock price array is filled with the closing price of 30/11/2022 that is, $62.42. The variance array is filled with the calibrated starting variance of table 5.

Next, I sample correlated Wiener processes from the multivariate normal distribution with mean the matrix of the calibrated drift terms and covariance, the calibrated covariance matrix. The size of the Wiener array is (252×1000) to be able to calculate the Euler discretization.

Then I calculate the predicted prices and variances arrays given the realized variance of the previous day, the 1-year interest rate and the closing price of the previous day. So, the prediction for (01/12/2022) is an array with a thousand values, all with respect to (01/12/2022). The predicted values are distributed as shown in Figure 21.

As we can see, the predicted price is slightly bigger than the real price. The difference between the predicted price and the mean predicted price is about 28 cents. MAPE is 0.77% which is a desired value. It demonstrates that the difference between the real price and the predicted price is below the 1% on average. The root mean square error is also significantly low.

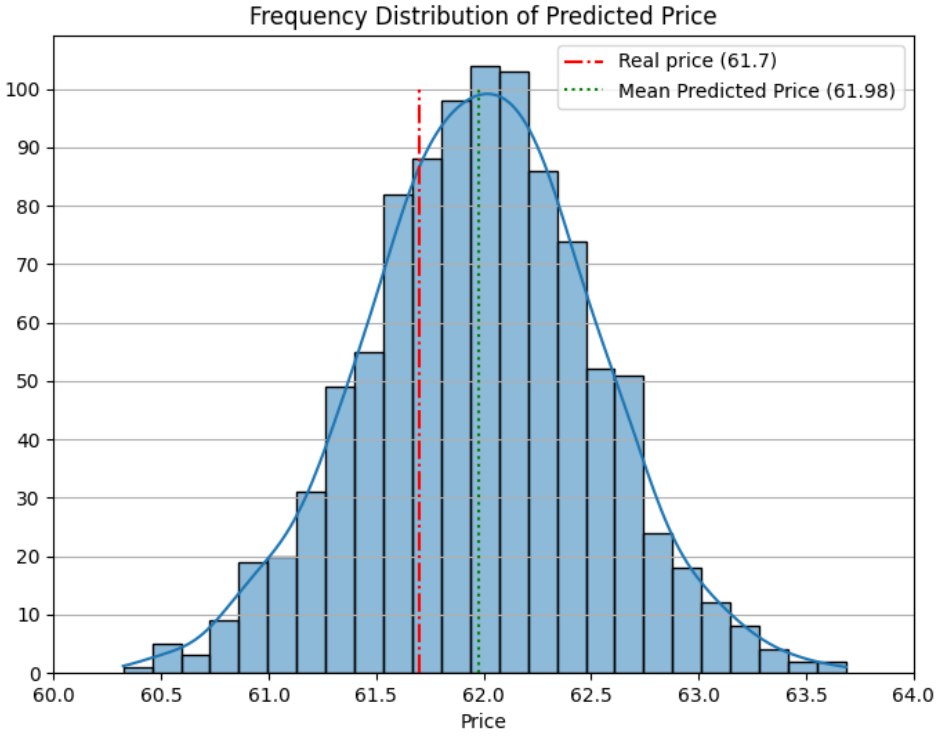

Figure 21: Frequency Distribution of Heston SV predicted Price

What we can see from Figure 21 is that the kernel density of the predicted price seems to be close to normal distribution. For this reason, I conducted a normality test via Jarque – Bera test. The p-value of the test indicates that we can safely not reject the null hypothesis. The predicted price is normally distributed. The result of this Figure is that the predicted price is normally distributed. The predicted price is $\pm 1\sigma$ around the mean as well as the real price, as seen in the figure.

6. Conclusion

In the literature, stochastic processes have been used extensively to price various stock market assets, to predict their prices or even to calculate and predict their risk. In this thesis, I first present the most basic stochastic processes and the properties they exhibit that help explain various stock market phenomena (e.g., high volatility); the Wiener process is used to describe the variance of an underlying, while the Poisson process is used to model jumps in volatility.

The above processes are then simulated using Python. In this thesis an attempt is made to implement the above processes, their statistics, distributions etc. The difficulty of simulating these processes lies in the difficulty of coding them in a programming language, having been the biggest obstacle of this Thesis.

Then, the empirical analysis is presented. In this section, two models were analyzed. The most basic model in the literature, the Black-Scholes-Merton and then the Heston

SV, which implements the feature of random variance, that is missing in the former. In Black-Scholes-Merton, Total Energies' derivatives were used to achieve fair pricing. Total Energies' call option with a three-month expiration and an exercise price of $62.50 is overpriced according to Black-Scholes-Merton. The fair price of the option is $2.99 while the market is trading at $2.46. The same is true for the call option with a one-month expiration as the fair price is $2.43 while the market price is $2.40. This finding indicates that arbitrage opportunities exist in the market, and we expect option traders to buy en masse as the market price is less than the fair price. The second finding of the Black-Scholes-Merton analysis is that the effect of the interest rate and dividend yield does not affect the price of both the call option and the put option as much. The relationship between Total Energies' options to an increase in interest rates or dividend yields is almost completely elastic over the period studied.

In Heston SV, I used the stock price of Total Energies to calibrate the parameters of the model. The necessary tests performed showed that the specification with stochastic volatility is clearly better than fixed volatility. The calibrated parameters are then used to predict the stock price. The forecasting methodology used in this thesis appears to provide valid and accurate results. The prediction error is infinitesimal. However, the Heston SV may not always be the most reliable model as comparison with a corresponding fixed-variance specification showed that the latter is clearly better.

Finally, it would be quite interesting for future literature to examine the strength of stochastic variability models trained by intelligent methods such as neural networks. Large Silicon Valley companies such as Google and Microsoft have developed APIs that could be used to train stochastic volatility models (Tensorflow, Keras, Microsoft Azure etc) and optimize their predictive capability.

References

Applebaum, D. (2005). *Lévy processes and stochastic calculus*. Cambridge Etc.: Cambridge University Press.

Bachelier, L. (1900). Théorie de la spéculation. Annales scientifiques de l'École normale supérieure, 17, pp.21–86.

Bates, D.S. (1996). Jumps and Stochastic Volatility: Exchange Rate Processes Implicit in Deutsche Mark Options. *Review of Financial Studies*, 9(1), pp.69–107.

Bertoin, J. (1996). *Lévy Processes*. Cambridge University Press.

Black, F. (1976). Studies of stock price volatility changes. *Proceedings of the Business and Economic Statistics Section, American Statistical Association*, 177–181.

Black, F. and Scholes, M. (1973). The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3), pp.637–654.

Broadie, M. and Kaya, Ö. (2006). Exact Simulation of Stochastic Volatility and Other Affine Jump Diffusion Processes. *Operations Research*, 54(2), pp.217–231.

Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2), pp.223–236.

Cox, J.C., Ingersoll, J.E. and Ross, S.A. (1985). A Theory of the Term Structure of Interest Rates. *Econometrica*, 53(2), p.385.

Christoffersen, P., Heston, S. and Jacobs, K. (2009). The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well. *Management Science*, 55(12), pp.1914-1932.

Derman, E. and Kani, I. (1998) Stochastic implied trees: Arbitrage pricing with stochastic term and strike structure of volatility, *International Journal of Theoretical and Applied Finance*, 1 (1), pp.61–110.

Dupire, B. (1994) Pricing with a smile. *Risk Magazine,* 7 (1), pp.18–20.

Eberlein, E. (2009). Jump–Type Lévy Processes. *Handbook of Financial Time Series*, pp.439–455.

Embrechts, P., Frey, R. and Furrer, H. (2001). Stochastic processes in insurance and finance. *Handbook of Statistics*, pp.365–412.

Fama, E.F. (1965). The Behavior of Stock-Market Prices. *The Journal of Business*, 38(1), pp.34–105.

Hirsa, A. and Neftci, S.N. (2014). *An introduction to the mathematics of financial derivatives*. Amsterdam: Academic Press.

Hull, J. and A. White (1987). The pricing of options on assets with stochastic volatilities. *Journal of Finance* 42, 281–300.

Hull, J.C. (2018). *Options, Futures, and Other Derivatives*. 9th ed. Harlow Etc.: Pearson Educational Limited.

Itô, K. (1942). On stochastic processes (I) Infinitely divisible laws of probability. In *Japanese journal of mathematics: transactions and abstracts*, 18, pp.261-301. The Mathematical Society of Japan.

Karatzas, I. and Shreve, S.E. (1998). *Brownian motion and stochastic calculus*. New York: Springer, Cop.

Kyprianou, A.E. (2014). *Fluctuations of Lévy Processes with Applications Introductory Lectures*. Berlin, Heidelberg Springer.

Lévy, P. (1934). Sur les intégrales dont les éléments sont des variables aléatoires indépendantes. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 3(3-4), pp.337-366.

Lévy, P. (1954). *Théorie de l'addition des variables aléatoires*. Gauthier-Villars.

Mandelbrot, B. (1963). The Variation of Certain Speculative Prices. *The Journal of Business*, 36(4), p.394.

Matsuda, K. (2006). *Lévy option pricing models: Theory and application*. City University of New York.

Merton, R.C. (1976). Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1-2), pp.125–144.

Rouah, F. (2013). *Euler and Milstein Discretization.* [online] *frouah.com*, pp.1–8. Available at: https://frouah.com/finance%20notes/Euler%20and%20Milstein%20Discretization.pdf [Accessed Nov. 10. 2022].

Rubinstein, M. (1985). Nonparametric Tests of Alternative Option Pricing Models Using All Reported Trades and Quotes on the 30 Most Active CBOE Option Classes from August 23, 1976 through August 31, 1978. *The Journal of Finance*, 40(2), pp.455–480.

Satō, K. (2013). *Lévy processes and infinitely divisible distributions*. Cambridge: Cambridge University Press, Cop.

Schoutens, W. (2005). *Lévy processes in finance: pricing financial derivatives*. Wiley.

Shreve, S.E. (2008). *Stochastic calculus for finance II. Continuous-time models. New York*; London: Springer.

Stein, E.M. and Stein, J.C. (1991). Stock Price Distributions with Stochastic Volatility: An Analytic Approach. *The Review of Financial Studies*, 4(4), pp.727–752.

Vasiliou, P.C. (2001). Stochastika Chrimatooikonomika. 1st ed. Ziti, pp.1–480.

Vasicek, O. (1977). An equilibrium characterization of the term structure. *Journal of Financial Economics*, 5(2), pp.177–188.

Van Hastrecht, A. and Pelsser, A. (2010). Efficient, Almost Exact Simulation of the Heston Stochastic Volatility Model. *International Journal of Theoretical and Applied Finance*, 13(01), pp.1–43.

Yalincak, H. (2012). Criticism of the Black-Scholes Model: But Why is it Still Used?: (The Answer is Simpler than the Formula). SSRN Electronic Journal.

Yoon, Y. and Kim, J.-H. (2021). A Closed Form Solution for Pricing Variance Swaps Under the Rescaled Double Heston Model. *Computational Economics*.

Appendix

The code written below simulates the Figures 1-5. First of all, the ***symmetric random walk*** using Numpy. I set a list of two numbers [-1,1] so the path goes up by one or down by one, then I simulate the steps of the paths for the number of simulations and make use of Numpy's Concatenate and Cumsum to be able to plot the paths into one single graph. I also show the quadratic variation and variance for this process. Second, I simulate the scaled ***symmetric random walk*** with the same methodology and plot the graph. Third, I show the ***limit of the binomial distribution*** making use of the nCr and a for loop in python. I ran the code 3 times, the first time I was able to get Figure 3, the second time Figure 4, and the last time I ran the code for 1000 permutations, but I was unable to get a result due to memory errors. Lastly, I simulate the ***Brownian Motion*** by implementing the formula. The code exports Figure 5.

```
# Imports
import math
import itertools
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# SYMMETRIC RANDOM WALK

# Parameters
M = 10 # number of simulations
t = 10 # Time

random_walk = [-1, 1]
steps = np.random.choice(random_walk, size=(M,t)).T
origin = np.zeros((1,M))
rwPaths = np.concatenate([origin, steps]).cumsum(axis=0)

plt.plot(rwPaths)
plt.xlabel("Years (t)")
plt.ylabel("Move")
plt.show()

# Quadratic Variation and Variance Functions
# Create Quadratic variation and Variance functions
quadratic_variation = lambda x: round(np.square(x[:-1]-
x[1:]).sum(),3)
variance = lambda x: round(np.var(x, axis=0), 3)
[quadratic_variation(path) for path in rwPaths.T[:4]]

[variance(path) for path in rwPaths[1:11]]

# SCALED SYMMETRIC RANDOM WALK
# Parameters
M = 10 # number of simulations
t = 10 # Time
n = 10
random_walk = [-1, 1]
```

```
steps = (1/np.sqrt(n)) * np.random.choice(random_walk,
size=(M,t*n)).T
origin = np.zeros((1,M))
srwPaths = np.concatenate([origin, steps]).cumsum(axis=0)
time = np.linspace(0,t,t*n+1)
ttime = np.full(shape=(M, t*n+1), fill_value=time)
ttime = ttime.T

plt.plot(ttime,srwPaths)
plt.xlabel("Years (t)")
plt.ylabel("Move")
plt.title("Scaled Symmetric Random Walk")
plt.show()

# Create Quadratic variation and Variance functions
quadratic_variation = lambda x: round(np.square(x[:-1]-
x[1:]).sum(),3)
variance = lambda x: round(np.var(x,axis=0),3)
[quadratic_variation(path) for path in rwPaths.T[:4]]

[variance(path) for path in rwPaths[1:11]]


# LIMIT OF BINOMIAL DISTRIBUTION


# As I change (n), I tend to get more precise approximation of
Normal Distribution
n = 10
t = 10

# Combinations
def nCr(n,k):
    f = math.factorial
    return f(n) / (f(k) * f(n-k))

permutations = [nCr(n*t,k)*(0.5)**(n*t) for k in
range(int(n*t)+1)]

W_nt = lambda n,t: 1/np.sqrt(n) * np.arange(-n*t,n*t+1,2)

outcomes = W_nt(n,t)
plt.bar(outcomes,[perm/(outcomes[1]-outcomes[0]) for perm in
permutations],outcomes[1]-outcomes[0],
        label='{0} scaled RW'.format(n))

x = np.linspace(-3*np.sqrt(t), 3*np.sqrt(t), 100)
plt.plot(x, stats.norm.pdf(x, 0, np.sqrt(t)), 'k-',label='normal
dist')

plt.xlim(-3*np.sqrt(t),3*np.sqrt(t))
plt.title("10 Scaled Random Walk processes under normal
distribution")
plt.ylabel("Probability")
plt.xlabel("Move")
plt.legend()
```

```
plt.show()


# BROWNIAN MOTION

# Parameters
M = 100 # number of simulations
t = 10 # Time
n = 100 # steps we want to see
dt = t/n # time step

steps = np.random.normal(0, np.sqrt(dt), size=(M, n)).T
origin = np.zeros((1,M))
bmPaths = np.concatenate([origin, steps]).cumsum(axis=0)

time = np.linspace(0,t,n+1)
ttime = np.full(shape=(M, n+1), fill_value=time)
ttime = ttime.T
plt.plot(ttime,bmPaths)
plt.xlabel("Years (t)")
plt.ylabel("Move")
plt.show()

[quadratic_variation(path) for path in bmPaths.T[:4]]

[variance(path) for path in bmPaths[1:11]]
```

The code bellow simulates the ***Geometric Brownian Motion*** and exports Figure 6. First of all, I set the values to the parameters and simulate the Geometric Brownian Motion paths. The formula that I use to simulate the stock price is given in Chapter 3.1.5. Then I define the time interval correctly and giving the arrays the same size in order to get Figure 6.

```
import numpy as np
import matplotlib.pyplot as plt

# PARAMETERS

# drift coefficient
mu = 0.1
# number of steps
n = 100
# time in years
T = 1
# number of simulations
M = 100
# initial stock price
S0 = 100
# volatility in terms of standard deviation
sigma = 0.3


# SIMULATING GBM PATHS

```

```
# calc each time step
dt = T/n

# simulation using numpy arrays
St = np.exp(
    (mu - sigma ** 2 / 2) * dt
    + sigma * np.random.normal(0, np.sqrt(dt), size=(M, n)).T #
Transposed in order to get the simulation for each time step
)

# include array of ones
St = np.vstack([np.ones(M), St])

St = S0 * St.cumprod(axis=0)

# CONSIDER TIME INTERVALS IN YEARS

# Define time interval correctly
time = np.linspace(0, T, n+1)

# Require numpy array that is the same shape as St
ttime = np.full(shape=(M, n+1), fill_value=time).T

plt.plot(ttime, St)
plt.xlabel("Years $(t)$")
plt.ylabel("Stock Price $(S_t)$")
plt.title(
    "Simulation of Geometric Brownian Motion\n $dS_t = \mu S_t dt
+ \sigma S_t dW_t$\n $S_0 = {0}, \mu = {1}, \sigma =
{2}$".format(S0, mu, sigma)
)
plt.show()
```

The code below simulates Poisson and Compound Poisson process. The first part of the code is the ***Poisson process***. To simulate that process, I define a function in python with two arguments, the Poisson intensity which is mu[9] and the number of events to simulate. To get the simulated Poisson values I used the inverse cumulative distribution function (Chapter 4.2). The code exports Figure 7. The Second part of the code is the ***Histogram of the Poisson process*** to make sure the simulated process is correct. I make a hundred thousand iterations and store them to a variable. Then I used Scipy's stats package to give me the same amount of Poisson random distributed variables. I plot both of the variables into one single plot to check if they match. The result is Figure 8. The last part of the code is the simulation of the ***Compound Poisson process***. To simulate the Compound Poisson, we need a secondary distribution and generate variables according to it. I used the binomial distribution with 20 trials and probability 0.5. Plotting the process gives Figure 9.

[9] I did not use "Lambda" to make sure that the reader does not get confused by Python's lambda function.

```
# POISSON PROCESS

# Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import poisson

# Parameters
mu = 15
event_sims = 50

# Poisson process
def gen_poisson_proc(mu, num_events):
    time_intervals = -np.log(np.random.random(num_events)) / mu
    total_events = time_intervals.cumsum()
    events = pd.DataFrame(np.ones(num_events),
index=total_events)
    events[0] = events[0].cumsum()

    return events

# Poisson process plot
plt.plot(gen_poisson_proc(mu, event_sims), marker='o',
linestyle='none')
plt.title("Simulation of Poisson Process")
plt.xlabel("Time")
plt.ylabel("Events")
plt.show()

# Histogram of the Poisson process
results = []
for x in range(100000):
    process = gen_poisson_proc(mu, event_sims)
    results.append(process[:1][0].iloc[-1])

plt.hist(results, bins=np.linspace(0, 35, 36), alpha=0.5,
label='Simulated poisson', ec='black')
# Generating Poisson random variates and plotting both the
process and the random variates in the same histogram
r = poisson.rvs(mu, size=100000)
plt.hist(r, bins=np.linspace(0, 35, 36), alpha=0.5,
label='Poisson random variates', ec='black')
plt.title("Poisson-Distributed Variables")
plt.ylabel("Count")
plt.xlabel("X")
plt.legend()
plt.show()

# COMPOUND POISSON
def binomial_gen(num_events):
    return np.random.binomial(20, 0.5, num_events)

def gen_Compound_poisson_proc(mu, num_events, generator):
    time_intervals = -np.log(np.random.random(num_events)) / mu
    total_events = time_intervals.cumsum()
```

```
    events = pd.DataFrame(generator(num_events),
index=total_events)
    events[0] = events[0].cumsum()

    return events

plt.plot(gen_Compound_poisson_proc(mu, event_sims, binomial_gen),
marker='o', drawstyle='steps-post')
plt.title("Simulation of Compound Poisson Process")
plt.xlabel("time")
plt.ylabel("events")
plt.show()
```

The code below uses the Stock price, the Strike Price, the interest rate, the days to expiry, the volatility (implied volatility), and the dividend payment in order to ***simulate the call and put option prices in Table 1***. To simulate the prices with Black and Scholes formula, I define a function in Python which takes the above as inputs and returns the call and put price. The first part of the function calculates d_1 and d_2 as shown in the 5th chapter. Then, it calculates the call and put prices using the cumulative distribution function. In the end of the code, I calculate the prices for given data.

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# Definitions

# Current Stock Price
S = 60.60
# Strike Price
K = 60
# Interest rate
int_rate = 0.0377
# Days till expiration - annualized (16-11-2022 : 17-2-2023)
T = 30 / 365
# Volatility
sigma = 0.3128
# Dividend
dividend = 0.0479


# Black and Scholes Function
def black_scholes(S, K, int_rate, T, sigma, dividend):
    # Calculation of d1, d2
    d1 = (np.log(S / K) + (int_rate - dividend + sigma ** 2 / 2)
* T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    # Call, Put prices calculation
    call = stats.norm.cdf(d1) * S * np.exp(1) ** (-dividend * T)
- stats.norm.cdf(d2) * K * np.exp(1) ** (-int_rate * T)
    put = stats.norm.cdf(-d2) * K * np.exp(1) ** (-int_rate * T)
- stats.norm.cdf(-d1) * S * np.exp(1) ** (
```

```
                -dividend * T)

    return print('The call price is:', call, '. The put value
is:', put)


black_scholes(S, K, int_rate, T, sigma, dividend)

# Three-month
black_scholes(60.26, 62.50, 0.0431, 93/365, 0.3311, 0.0479)
# The call price is: 2.998730525746865 . The put value is:
5.287115560748681

# One-month
black_scholes(60.60, 60, 0.0377, 30/365, 0.3128, 0.0479)
# The call price is: 2.4338242126586813 . The put value is:
1.8863064922104549
```

The following code outputs ***Figures 10-16.*** First of all, I make sure I define every input of the ***Black-Scholes function*** so I do not have to write down values every time I have to call a plot. Those values are the one-month data described at Table 1. Then I use the define utility of Python to create a function called "black_scholes", which uses d_1 and d_2 to calculate the call and put prices. The function returns a list with two values. The first value [0] gives the call price and the second value [1] gives the put price.

Then I start changing the parameters. First of all, TTE's share price. The starting value is 55 and the ending value is 80. Within the plt.plot command I use a for loop to start changing the Stock price, but all other parameters remain the same (ceteris paribus). I am doing the same for the put price and get **Figures 10, 11**.

Then I start changing the strike price. The starting value is 55 and the ending value is 80. Similar to the previous case, I use a for loop within the plt.plot command which changes only the value of the Strike price, ceteris paribus. I apply the same methodology for the put option and run the two pieces of code together to get **Figures 12, 13.**

I am using the same methodology for all parameters. The resulting **Figures** are **14-16.**

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt

# Definitions

# Current Stock Price
S = 60.60
# Strike Price
K = 60
# Interest rate
int_rate = 0.0377
```

```
# Days till expiration - annualized (16-11-2022 : 17-2-2023)
T = 30 / 365
# Volatility
sigma = 0.3128
# Dividend
dividend = 0.0479


# Black and Scholes Function
def black_scholes(S, K, int_rate, T, sigma, dividend):
    # Calculation of d1, d2
    d1 = (np.log(S / K) + (int_rate - dividend + sigma ** 2 / 2)
* T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)

    # Call, Put prices calculation
    call = stats.norm.cdf(d1) * S * np.exp(1) ** (-dividend * T)
- stats.norm.cdf(d2) * K * np.exp(1) ** (-int_rate * T)
    put = stats.norm.cdf(-d2) * K * np.exp(1) ** (-int_rate * T)
- stats.norm.cdf(-d1) * S * np.exp(1) ** (
                -dividend * T)

    return [call, put]
# In order to visualize the prices of the call and the put
option, I had to return a list on the function

black_scholes(S, K, int_rate, T, sigma, dividend)

# ------------------Parameter changes------------------

# Underlying Asset Price - CALL
plt.plot(range(55, 80), [black_scholes(x, K, int_rate, T, sigma,
dividend)[0] for x in range(55, 80)], lw=2.5, label='Call')
plt.yticks(range(5, 30, 5), ['$'+str(i) for i in range(5, 30,
5)])
plt.xticks(range(55, 90, 5), ['$'+str(i) for i in range(55, 90,
5)])
plt.xlabel('TTE Stock Price')
plt.ylabel('Black-Scholes Call Value')
plt.title("Stock Price Change")
plt.grid()

# Underlying Asset Price - PUT
plt.plot(range(55, 75), [black_scholes(x, K, int_rate, T, sigma,
dividend)[1] for x in range(55, 75)], lw=2.5, color='red',
label='Put')
plt.yticks(range(0, 10), ['$'+str(i) for i in range(0, 10)])
plt.xticks(range(55, 75, 5), ['$'+str(i) for i in range(55, 75,
5)])
plt.xlabel('TTE Stock Price')
plt.ylabel('Black-Scholes Put Value')
plt.title("Stock Price Change")
plt.grid()

# Strike Price - CALL

```

```
plt.plot(range(55, 80), [black_scholes(S, x, int_rate, T, sigma,
dividend)[0] for x in range(55, 80)], lw=2.5, label='Call')
plt.yticks(range(0, 8), ['$'+str(i) for i in range(0, 8)])
plt.xticks(range(50, 90, 5), ['$'+str(i) for i in range(50, 90,
5)])
plt.xlabel('Strike Price')
plt.ylabel('Black-Scholes Value')
plt.title("Strike Price Change")
plt.legend()

# Strike Price - PUT
plt.plot(range(55, 80), [black_scholes(S, x, int_rate, T, sigma,
dividend)[1] for x in range(55, 80)], lw=2.5, color='red',
label='Put')
plt.yticks(range(0, 25, 5), ['$'+str(i) for i in range(0, 25,
5)])
plt.xticks(range(50, 90, 5), ['$'+str(i) for i in range(50, 90,
5)])
plt.xlabel('Strike Price')
plt.ylabel('Black-Scholes Value')
plt.title("Strike Price Change")
plt.legend()
plt.grid()

# Volatility - CALL
plt.plot(np.arange(0.01, 0.5, 0.01), [black_scholes(S, K,
int_rate, T, x, dividend)[0] for x in np.arange(0.01, 0.5,
0.01)], lw=2.5, label='Call')
plt.yticks(range(0, 5), ['$'+str(i) for i in range(0, 5)])
plt.xlabel('Volatility')
plt.ylabel('Black-Scholes Value')
plt.title("Volatility Change")
plt.legend()

# Volatility - PUT
plt.plot(np.arange(0.01, 0.5, 0.01), [black_scholes(S, K,
int_rate, T, x, dividend)[1] for x in np.arange(0.01, 0.5,
0.01)], lw=2.5, color='red', label='Put')
plt.yticks(range(0, 5), ['$'+str(i) for i in range(0, 5)])
plt.xlabel('Volatility')
plt.ylabel('Black-Scholes Value')
plt.title("Volatility Change")
plt.legend()
plt.grid()

# Time - CALL
plt.plot(np.arange(0.08, 3, 0.1),[black_scholes(S, K, int_rate,
x, sigma, dividend)[0] for x in np.arange(0.08, 3, 0.1)],
lw=2.5, label='Call')
plt.yticks(range(0, 15, 3), ['$'+str(i) for i in range(0, 15,
3)])
plt.xlabel('Time (in Years)')
plt.ylabel('Black-Scholes Value')
plt.title("Time Change in Years")
plt.legend()

```

```
# Time - PUT
plt.plot(np.arange(0.08, 3, 0.1), [black_scholes(S, K, int_rate,
x, sigma, dividend)[1] for x in np.arange(0.08, 3, 0.1)],
lw=2.5, color='red', label='Put')
plt.yticks(range(0, 15, 3), ['$'+str(i) for i in range(0, 15,
3)])
plt.xlabel('Time (in Years)')
plt.ylabel('Black-Scholes Value')
plt.title("Time Change in Years")
plt.legend()
plt.grid()

# Interest Rates - CALL
plt.plot(np.arange(0.001, 0.075, 0.01), [black_scholes(S, K, x,
T, sigma, dividend)[0] for x in np.arange(0.001, 0.075, 0.01)],
lw=2.5, label='Call')
plt.yticks(range(0, 5), ['$'+str(i) for i in range(0, 5)])
plt.xlabel('US 1-Month Treasury Bill')
plt.ylabel('Black-Scholes Value')
plt.title("Interest rate Change")
plt.legend()

# Interest Rates - PUT
plt.plot(np.arange(0.001, 0.075, 0.01), [black_scholes(S, K, x,
T, sigma, dividend)[1] for x in np.arange(0.001, 0.075, 0.01)],
lw=2.5, color='red', label='Put')
plt.yticks(range(0, 5), ['$'+str(i) for i in range(0, 5)])
plt.xlabel('US 1-Month Treasury Bill')
plt.ylabel('Black-Scholes Value')
plt.title("Interest rate Change")
plt.legend()
plt.grid()

# Dividend Yield - CALL
plt.plot(np.arange(0.0, 0.075, 0.01), [black_scholes(S, K,
int_rate, T, sigma, x)[0] for x in np.arange(0.0, 0.075, 0.01)],
lw=2.5, label='Call')
plt.yticks(range(0, 5), ['$'+str(i) for i in range(0, 5)])
plt.xlabel('Dividend Yield Change')
plt.ylabel('BSM Euro Call Value')
plt.title("Dividend Yield Change")
plt.legend()

# Dividend Yield - PUT
plt.plot(np.arange(0.0, 0.075, 0.01), [black_scholes(S, K,
int_rate, T, sigma, x)[1] for x in np.arange(0.0, 0.075, 0.01)],
lw=2.5, color='red', label='Put')
plt.yticks(range(0, 5), ['$'+str(i) for i in range(0, 5)])
plt.xlabel('Dividend Yield Change')
plt.ylabel('Black-Scholes Value')
plt.title("Dividend Yield Change")
plt.legend()
plt.grid()
```

The following python code generates **Figures 17-19**. First of all, I set the parameters for the Heston SV model simulation. Then I use a python function in order to produce the simulation using Euler's discretization. The time difference is the time in years over the steps. I am using a year's trading days for N. Then I construct a numpy array containing the expected values of the Wiener processes, which are zero. The covariance matrix is a $2x2$ matrix with variance 1 and covariance ρ. Then I make an empty numpy array for the asset price and the variance with $(N+1 \times M)$ shape. We need this shape because we are taking into account the initial value of the asset price and variance. Then I sample the Wiener processes from multivariate normal distribution and the inputs of this function are the expected value and the covariance matrices I constructed before. The shape we need is $(N \times M)$. Then I simply apply the recursion from 1 to $N+1$, because again, we are taking into account the initial value of the process. The recursion stores in the empty lists of asset price and variance the values according to Euler's discretization. Lastly, the function I made outputs the asset price and variance. The visualization technique for this simulation starts from row 45 and ends in row 67. The output is **Figures 17-18**.

In order to show the asset price densities, I have taken two distinct cases. The first one is an almost perfect positive correlation between returns and variance, and the second one is an almost perfect negative correlation between these two. I receive the simulation for asset price and variance according to those two different cases and visualize the result using seaborn and matplotlib. The code for visualization begins from row 68 and ends in row 82, generating **Figure 19**.

```
 1 import numpy as np
 2 import seaborn as sns
 3 import matplotlib.pyplot as plt
 4
 5 # Parameters
 6 S0 = 100.0
 7 T = 1.0
 8 N = 252
 9 M = 1000
10 r = 0.05
11 kappa = 3
12 theta = 0.20**2
13 v0 = 0.30**2
14 ksi = 0.5
15 rho = 0.8
16
17 def heston_SV(S0, v0, rho, kappa, theta, ksi, T, N, M):
18     dt = T/N
19     mu = np.array([0, 0])
20     cov = np.array([[1, rho],
21                     [rho, 1]])
22
23     S = np.full(shape=(N+1, M), fill_value=S0)
24     v = np.full(shape=(N+1, M), fill_value=v0)
25
26     Wiener = np.random.multivariate_normal(mu, cov, (N, M))
27
```

```
    for i in range(1, N+1):
        S[i] = S[i-1] * np.exp((r - 0.5 * v[i-1]) * dt +
np.sqrt(v[i-1] * dt) * Wiener[i-1, :, 0])
        v[i] = np.maximum(v[i-1] + kappa * (theta - v[i-1]) * dt
+ ksi * np.sqrt(v[i-1] * dt) * Wiener[i-1, :, 1],
                          0)
    return S, v

positive_rho = 0.98
S_positive, v_positive = heston_SV(S0, v0, positive_rho, kappa,
theta, ksi, T, N, M)

negative_rho = -0.98
S_negative, v_negative = heston_SV(S0, v0, negative_rho, kappa,
theta, ksi, T, N, M)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))
time = np.linspace(0, T, N+1)
ax1.plot(time, S_positive)
ax1.set_title(r'Heston Model Asset Prices, positive $\rho$')
ax1.set_xlabel('Time')
ax1.set_ylabel('Asset Prices')
ax2.plot(time, v_positive)
ax2.set_title(r'Heston Model Variance Process, positive $\rho$')
ax2.set_xlabel('Time')
ax2.set_ylabel('Variance')
plt.show()

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12,5))
time = np.linspace(0, T, N+1)
ax1.plot(time, S_negative)
ax1.set_title(r'Heston Model Asset Prices, negative $\rho$')
ax1.set_xlabel('Time')
ax1.set_ylabel('Asset Prices')
ax2.plot(time, v_negative)
ax2.set_title(r'Heston Model Variance Process, negative $\rho$')
ax2.set_xlabel('Time')
ax2.set_ylabel('Variance')
plt.show()

# GBM simulation at T
gbm = S0 * np.exp((r - theta ** 2 / 2) * T + np.sqrt(theta) *
np.sqrt(T) * np.random.normal(0, 1, M))
fig, ax = plt.subplots()
ax = sns.kdeplot(S_positive[-1], label=r"$\rho= 0.98$", ax=ax,
color='red')
ax = sns.kdeplot(S_negative[-1], label=r"$\rho= -0.98$", ax=ax,
color='green')
ax = sns.kdeplot(gbm, label="GBM", ax=ax, color='blue')
plt.title(r'Asset Price Density')
plt.xlim([20, 180])
plt.xlabel('$S_T$')
plt.ylabel('Density')
plt.legend()
plt.show()
```

The following python code generates the **figures and tables of chapter 5.4.3**. First of all, I import the dependencies and set the calibrated parameter values. Then I have to calculate the time step which is one year over 252 trading days. The mu matrix contains the calibrated drift terms and cov is the calibrated variance – covariance matrix. Then I fill S1 and v matrices with the starting values. Then using numpy I sample correlated Wiener processes utilizing mu and cov matrices. S1 is the predicted value through Euler discretization and v is the variance prediction. Next I use seaborn and matplotlib to figure out the frequency distribution of the predicted price. lastly I define two functions, the mean absolute percentage error and the root mean square error, and output the result. Also, I make use of scipy dependency to perform the Jarque – Bera normality test for the predicted price.

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

# Parameters - Calibrated
S0 = 62.42
T = 1.0
N = 252
M = 1000
r = 0.0466
kappa = 0.85645659
theta = 4.711798009
v0 = 13.72501634
ksi = 8.25773478
rho = -0.263081559
drift = 0.18
realized_var_prev = 4.563114037

# Heston SV Forecast
dt = T/N
mu = np.array([drift, drift])
cov = np.array([[1, rho],
                [rho, 1]])

S1 = np.full(shape=(N, M), fill_value=S0)
v = np.full(shape=(N, M), fill_value=v0)

Wiener = np.random.multivariate_normal(mu, cov, (N, M))

S1 = S0 * np.exp((r - 0.5 * realized_var_prev) * dt +
np.sqrt(realized_var_prev) * dt * Wiener[0, :, 0])
v = np.maximum(realized_var_prev + kappa * (theta -
realized_var_prev) * dt + ksi * np.sqrt(realized_var_prev * dt) *
Wiener[0, :, 1], 0)

# Plot
fig, ax = plt.subplots()
ax = sns.histplot(S1, kde=True)
ax.vlines(x=61.7, ymin=0, ymax=100, colors='red',
linestyles='dashdot', label="Real price (61.7)")
```

```
ax.vlines(x=61.98, ymin=0, ymax=100, colors='green',
linestyles='dotted', label="Mean Predicted Price (61.98)")
plt.title("Frequency Distribution of Predicted Price")
plt.xlabel("Price")
plt.ylabel("Percentage")
plt.yticks(range(0, 105, 10))
plt.xlim([60, 64])
plt.legend()
plt.grid(axis='y')

# Mean Absolute Percentage Error
S_actual = np.full(shape=M, fill_value=61.7)

def Mean_Absolute_Percentage_Error(S1, S_actual):
    mape = np.mean(np.abs((S_actual - S1) / S1)) * 100
    return mape

# Root Mean Square Error
def RMSE(S1, S_actual):
    MSE = np.mean(np.square(S1 - S_actual))
    RMSE = np.sqrt(MSE)
    return RMSE

S1
S_actual
v
Mean_Absolute_Percentage_Error(S1, S_actual)
RMSE(S1, S_actual)

jb = stats.jarque_bera(S1)
jb
```