Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

Julie Johanne Uv

Pricing European Options with Lévy Market Models and Deep Learning

A comparison of parametric and non-parametric models in finance

Master's thesis in Applied Physics and Mathematics
Supervisor: Espen Robstad Jakobsen
June 2019

NTNU
Norwegian University of Science and Technology

Julie Johanne Uv

Pricing European Options with Lévy Market Models and Deep Learning

A comparison of parametric and non-parametric models in finance

Master's thesis in Applied Physics and Mathematics
Supervisor: Espen Robstad Jakobsen
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Mathematical Sciences

NTNU
Norwegian University of
Science and Technology

Preface

This master thesis concludes five years of my M.Sc Degree in Applied Physics and Mathematics, specialized in Industrial Mathematics, at the Norwegian University of Science and Technology (NTNU). The work was carried out in the spring of 2019 at the Institute of Mathematical Sciences under the supervision of professor Espen Robstad Jakobsen.

In particular, I wish to express my gratitude towards my supervisor, professor Espen Robstad Jakobsen, for our weekly meetings and conversations and for letting me write a thesis combining the topic of finance and artificial intelligence. He has been of great help and support.

I would also like to thank family, friends and fellow students for support, (many) coffee breaks and five remarkable years.

Trondheim, June 2019
Julie Johanne Uv

Abstract

In this thesis, five Lévy models and a multilayer perceptron has been implemented to compare the pricing of European call options against the Geometric Brownian motion stock price dynamics of the Black-Scholes formula. Statistical analysis has been done on the underlying assets, where it was found that the Lévy models clearly are a better fit than the Geometric Brownian motion. However, the pricing performance did not reflect this. On the contrary, for the option prices, the Geometric Brownian motion outperforms several of the models. It is also found that the multilayer perceptron generalizes well, despite of few observations, and outperforms all the models for the longest maturity option which is held completely out of the training data. In the end, it is concluded that more data is needed to say anything definite.

Sammendrag

I denne oppgaven har fem Lévy-modeller og et kunstig nevralt nettverk blitt implementert for å sammenligne prising av europeiske kjøpsopsjoner mot den geometrisk brownske bevegelsesdynamikken i Black-Scholes-formelen. Statistisk analyse er utført på de underliggende aksjene, hvor det ble funnet at Lévy-modellene er en mer beskrivende modell enn den geometrisk brownske bevegelsen. Prissettingspressisjonen reflekterte imidlertid ikke dette. Derimot presterer den gemoetrisk brownske bevegelsesmodellen bedre enn flere av de andre modellene. Det blir også funnet at det nevrale nettverket presterer og generaliserer bra, til tross for få observasjoner, og overgår alle modellene for den lengste forfallsdatoen som er holdt utenfor treningsdataen. Til slutt konkluderes det med at mer data er nødvendig for å kunne si noe mer konkret.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Modelling financial markets is a great been an area of interest and started with Bachelier [6] modelling stock prices as a Brownian motion with drift. In 1973, Black & Scholes [13] and Merton [43] made an important contribution to the field of financial derivatives. In particular, they derived the Black-Scholes formula - a closed form solution of the Black-Scholes partial differential equation for European options, which they in 1997 received the Nobel Prize in Economics for. The Black-Scholes model has since dominated the pricing of financial derivatives, mainly because of its simplicity. In particular, it models the underlying asset as a Geometric Brownian motion which, in light of empirical data, has proven to be a poor fit.

In the late 1980s and early 1990s, several Lévy models were proposed that take the empirical observations in to account with several stylized features to the market. Barndorff-Nielsen [8] applied the Normal Inverse Gaussian process on financial data in 1995. The same year, the Hyperbolic process on financial data was published as well by Eberlein & Keller [20]. Madan *et al.* [36] proposed the general Variance Gamma process. It was first considered for the symmetric case by [39] and [40] along with [38]. Carr *et al.* [14] extended the three parameter Variance Gamma, to the four parameter CGMY, providing more flexibility.

The Normal Inverse Gaussian, Variance Gamma and the Hyperbolic process are special cases of the Generalized Hyperbolic process, also proposed by Barndorff-Nielsen [7]. This model was originally considered as a model for the log particle size of sand in 1977. The whole family of Generalized Hyperbolic distributions were studied for financial modelling by Eberlein and Prause [21] and Prause [47] in 1998 and 1999 respectively.

At around the same time, studies comparing the performance of the non-parametric artificial neural networks to Black-Scholes were published, e.g. by Hutschinson *et al.* [31], finding them to be superior to Black-Scholes, indicating that artificial neural networks can learn option prices with high precision for historical prices.

The data sets that have been used in earlier literature is usually from the 1990s. This motivates for training deep artificial neural networks on more recent data. A lot of the earlier artificial neural networks are also small and shallow, but with evolving computing power, deeper and more computationally expensive networks can be built. The evolving computer power has caused a renaissance of artificial neural networks in recent years.

This thesis aims to compare not only the exponential Lévy models and artificial neural networks to the Geometric Brownian motion framework of the Black-Scholes model, but also comparing the proposed models to each other to examine if the artificial neural networks can compete with the improved option pricing models. This is an extension to the work done in my project thesis where an artificial neural network was implemented to compare pricing accuracy against Black-Scholes for European call options based on daily closing stock prices from S&P 500 [53].

In chapter 2, some fundamental theory on finance and probability will be covered. Chapter 3 will start by introducing the Geometric Brownian motion as a stock price model, then give a brief motivation for Lévy models in general before presenting five Lévy models. The mathematical preliminaries will be given. Chapter 4 will set them in a market context, doing a statistical analysis on the fitted market models to empirical data.

In chapter 5, the theory that exists on artificial neural networks today will be given[1], before the model set up and network architecture particularly for option pricing will be presented. Finally, in chapter 6, we will see numerical results and a discussion concerning these, on the data set considered. We will discuss wether the non-parametric artificial neural network can compete with the more intuitive Lévy models.

For the exponential Lévy models, a brute-force Monte Carlo method has been used. This is a computationally expensive and time-consuming method. It should be noted that Monte Carlo operates better in higher dimensions where it does better time wise with regards to complexity, but as computational speed has not been the main focus in this thesis, there was not spent time researching refined Monte Carlo methods or other pricing methods such as exploiting the closed form of the characteristic functions with Fast Fourier transform, but more on this can be found in e.g. [16] and [27].

[1] Note that this is still a major area of research.

Chapter 2

Theory

In this chapter, an introduction of the necessary financial basics of options and their underlying, as well as pricing rules will be given in section 2.1. In section 2.2, probability theory needed in order to present the Lévy models of chapter 3 will be presented.

2.1 Financial Theory

Financial Market

A financial market is is a common term for markets that trade different financial securities and derivatives. The market liquidity tells us about the degree to which the purchase and sale of assets influence their price. When there exists a high number of buyers and sellers on a market, the price a buyer offers, called the *bid* price, and the price a seller accepts, called the *ask* price, is close and the market is highly liquid. When the spread between bid and ask price increases, the market is becoming more illiquid.

Stocks

Stocks are issued by a company and its value reflect both the value of the company's real assets as well as the company's earning power. Through stocks, an investor may obtain partial ownership of a company. Stocks of publicly quoted companies are quoted and traded on a stock exchange.

Indices

Stock indices measures the performance of a section of the stock market. It may be thought of as a portfolio consisting of a collection of stocks representing some segment of the market, usually constructed by some weighted average. There are several types of indices. A broad-based index for instance, represents the performance of the whole market, while a narrow-based index contains only a few stocks

usually representing a certain sector of industry. The weighting of the stocks can also be done in several ways. A price-weighted index for instance, weighs the stocks based on the price per share, while a capitalization-weighted index weighs the stocks based on total market value, i.e. number of outstanding shares times price per share.

2.1.1 Pricing Rules

In this section, we will follow the theory of Cont & Tankov [18] closely.

Let a market scenario space, defined by $(\Omega, \mathcal{F}, \{\mathcal{F}_t, t \in [0,T]\})$, describe all possible evolutions for time $t \in [0,T]$ in the market, with information flow at time t, $\mathcal{F}_t$ where $\mathcal{F}_t$ is such that

- $\mathcal{F}_t \subseteq \mathcal{F} \quad \forall t$
- $\mathcal{F}_s \subseteq \mathcal{F}_t$ for $s \leq t$.

An underlying asset on this market, may be described as an $\mathbb{F}$-adapted process[1]

$$S : [0,T] \times \Omega \mapsto \mathbb{R}^{d+1}$$
$$(t,\omega) \mapsto \left(S_t^0(\omega), S_t^1(\omega), \dots S_t^d(\omega)\right)$$

such that S_t^i is the value of asset i at time t.

S_t^0 is called a numeraire, usually taken to be a cash account with interest rate r, that is $S_t^0 = S_0 e^{rt}$. For any portfolio with value θ_t at time t, the discounted price is then $\hat{\theta}_t = \theta_t / S_t^0$ and the discount factor is $B(t,T) = S_t^0 / S_T^0$, which is equal to $e^{-r(T-t)}$ if $S_t^0 = S_0 e^{rt}$.

Contingent Claims

A contingent claim is a financial contract whose value at expiration T is determined by the price process of its underlying assets up to time T.

Let $\{\mathcal{F}_t, t \in [0,T]\}$ be the information flow, for the history of an asset, up to time t. A contingent claim with expiry T can be represented by a terminal payoff function $H(\omega)$ for each scenario $\omega \in \Omega$. H may depend on the entire price process $S_t(\omega), t \in [0,T]$, or only on S_T^i.

The pricing rule assigns each H with a value $\Pi_t(H)$ for each point in time. There are some requirements for the pricing rule, $\Pi_t(H)$.

- $\mathbb{F}$-*adapted.* See definition 2.1.3. Means that any information given at time t should be used to compute $\Pi_t(H)$.

[1] See definition 2.1.3.

- *Positiveness.* A claim with a positive payoff has a positive value.

$$\forall \omega \in \Omega, H(\omega) \geq 0 \implies \forall t \in [0,T], \Pi_t(H) \geq 0 \tag{2.1}$$

- *Linearity.* The value of a portfolio is given by the value of its components[2].

$$\Pi_t\Big(\sum_{j=1}^{J} \alpha_j H_j\Big) = \sum_{j=1}^{J} \alpha_j \Pi_t(H_j), \quad \alpha_j \in \mathbb{R}, \quad j = 1, \ldots, J \tag{2.2}$$

Next, we want to conclude with the risk-neutral pricing formula given in definition 2.1.1. To do so, we must define the probability measure $\mathbb{Q}$ which will be defined by first considering the event $A \in \mathcal{F}$, such that the terminal payoff be represented by the indicator function $\mathbb{1}_A$[3]. Then the discount factor $\Pi_t(\mathbb{1}_\Omega)$ is equal to $e^{-r(T-t)}$. That is the present value of one unit of currency paid out at time T. Now, let $\mathbb{Q}\colon \mathcal{F} \mapsto \mathbb{R}$ be such that

$$\mathbb{Q}(A) = \frac{\Pi_0(\mathbb{1}_A)}{\Pi_0(\mathbb{1}_\Omega)} = e^{rT}\Pi_0(\mathbb{1}_A). \tag{2.3}$$

From (2.1) and (2.2), the following holds for $\mathbb{Q}$,

- $0 \leq \mathbb{Q}(A) \leq 1$, because $0 \leq \mathbb{1}_A \leq 1$, and $\mathbb{Q}(\Omega) = 1$.
- If A and B are disjoint events ($A \cap B = \emptyset$), then $\mathbb{1}_{A\cup B} = \mathbb{1}_A + \mathbb{1}_B$, and by (2.2)

$$\mathbb{Q}(A \cup B) = \mathbb{Q}(A) + \mathbb{Q}(B).$$

By extending to an infinite sum, we find that $\mathbb{Q}$ becomes a probability measure over $(\Omega, \mathcal{F}, \{\mathcal{F}_t, t \in [0,T]\})$.

Now we will show that if $H = \sum_i c_i \mathbb{1}_{A_i}$, by linearity, the risk-neutral pricing formula holds. First, for $H = \mathbb{1}_A$,

$$\Pi_0(H) \overset{4}{=} e^{-rt}\mathbb{Q}(A) \tag{2.4}$$

$$= e^{-rt}E^{\mathbb{Q}}[H]. \tag{2.5}$$

Now, if $H = \sum_i c_i \mathbb{1}_{A_i}$,

$$\Pi_0(H) \overset{5}{=} \sum_i c_i \Pi_0(\mathbb{1}_{A_i}) \tag{2.6}$$

$$= e^{-rt}\sum_i c_i \mathbb{Q}(A_i) \tag{2.7}$$

$$= e^{-rt}E^{\mathbb{Q}}[H]. \tag{2.8}$$

[2]This may however not hold for large portfolios given a discount market price.
[3]equal to 1 if $x \in A$ and 0 otherwise.
[4]by (2.3)
[5]by (2.2)

As every H can be approximated by $\sum_i c_i \mathbb{1}_{A_i}$, by the monotone approximation theorem, the risk-neutral pricing formula holds.

Definition 2.1.1 ***Risk-neutral pricing formula.*** *For a probability measure $\mathbb{Q}$ on $(\Omega, \mathcal{F}, \{\mathcal{F}, t \in [0,T]\})$ and any random payoff function $H \in \mathcal{H}$ for a set $\mathcal{H}$ containing all contingent claims of interest such that a dominated convergence theorem holds, the value of a random payoff is given by its discounted expectation under $\mathbb{Q}$,*

$$\Pi_0(H) = e^{-rT} E^{\mathbb{Q}}[H]. \tag{2.9}$$

Arbitrage

A fundamental requirement for a pricing rule is that it does not allow for arbitrage, meaning an opportunity to make a profit without risk. If that would be possible, arbitrageurs could make profit of the market in an unlimited quantity, without exposure to risk, making it impossible for the market to be in equilibrium.

To be able to define arbitrage, we must first specify what is meant by a self-financing strategy.

Definition 2.1.2 ***Self-financing strategy.*** *Let a portfolio, θ_t, contain n stocks and let h_t^i denote the number of shares of stock i at time t. If S_t^i is the value of stock i at time t, then*

$$\theta_t = \sum_{i=1}^{n} h_t^i S_t^i.$$

The portfolio is self-financing if

$$\sum_{i=1}^{n} \mathrm{d}h_t^i S_t^i = 0$$

which means that

$$\mathrm{d}\theta_t = \sum_{i=1}^{n} h_t^i \mathrm{d}S_t^i.$$[6]

An arbitrage opportunity is the value process of a self-financing strategy, θ, that may generate a terminal profit without any intermediate loss,

$$\mathbb{P}(\forall t \in [0,T], V_t(\theta) \geq 0) = 1$$
$$\mathbb{P}(V_T(\theta) > V_0(\theta)) \neq 0.$$

$\mathbb{P}$ is often called the real world probability measure and tells us something about the probability of the scenarios $(\Omega, \mathcal{F}, \{\mathcal{F}_t, t \in [0,T]\})$ such as investors belief in the future. Any arbitrage-free pricing rule is given by an equivalent martingale measure. Hence, to conclude arbitrage-free pricing rules, we must define an equivalent martingale measure which will be derived by the following set of definitions.

[6] for continuous time, in a frictionless market

Definition 2.1.3 ***$\mathbb{F}$-adapted.*** *A stochastic process $X = \{X_t, t \geq 0\}$ is $\mathbb{F}$-adapted if the random variable X_t is $\mathcal{F}_t$-measurable*[7] *for each $t \in I$.*

$\mathbb{F}$-adapted means that the stochastic process is forward-looking - it can only *see* history up to time $\leq t$.

Definition 2.1.4 ***Martingale.*** *A stochastic process $X = \{X_t, t \geq 0\}$ is a martingale relative to $(P, \mathbb{F})$ if*

- *X is $\mathbb{F}$-adapted.*
- *$E[|X_t|] < \infty \quad \forall t \geq 0$.*
- *$E[X_t|\mathcal{F}_s] = X_s$*[8] *almost surely with respect to P for $0 \leq s \leq t$.*

A martingale models fair game because the best guess about the future value of X based on the current information, is the value of X at the current moment.

Definition 2.1.5 ***Equivalent probability measures.*** *$\mathbb{P}$ and $\mathbb{Q}$ are equivalent probability measures if they define the same sets of impossible events, i.e. they have the same null sets,*

$$\mathbb{P} \sim \mathbb{Q} \quad : \forall A \in \mathcal{F} \qquad \mathbb{Q}(A) = 0 \iff \mathbb{P}(A) = 0. \tag{2.10}$$

Definition 2.1.6 ***Equivalent martingale measures for the pricing problem.*** *An equivalent martingale measure, $\mathbb{Q}$, fulfills for an asset S^i, traded at price S_t^i at time t, generating a terminal payoff S_T^i,*

$$E^{\mathbb{Q}}[e^{-rT} S_T^i | \mathcal{F}_t] = e^{-rt} S_t^i.$$

Now, we can conclude with two propositions on arbitrage-free pricing.

Proposition 2.1.1 ***Arbitrage-free pricing.*** *In a market described by the probability measure $\mathbb{P}$, any arbitrage-free pricing rule is given by*

$$\Pi_t(H) = e^{-r(T-t)} E^{\mathbb{Q}}[H|\mathcal{F}_t] \tag{2.11}$$

for an equivalent martingale measure $\mathbb{Q}$.

Proposition 2.1.2 ***Fundamental theorem of asset pricing.*** *A market defined by scenario $(\Omega, \mathcal{F}, \{\mathcal{F}_t, t \in [0,T]\})$, probability measure $\mathbb{P}$ and asset prices $\{S_t, t \in [0,T]\}$ is arbitrage-free if and only if there exists a probability measure $\mathbb{Q} \sim \mathbb{P}$ such that the discounted prices $\{\hat{S}_t, t \in [0,T]\}$ are martingales with respect to $\mathbb{Q}$.*

Proposition 2.1.1 shows that if an equivalent martingale measure exists, then the market is arbitrage free while proposition 2.1.2 shows the converse.

[7] A random variable is $\mathcal{F}_t$-measurable if its value is *revealed* at time t. See [18] for details.
[8] $E[X_t|\mathcal{F}_t] = E[X_t|X_r, 0 \leq r \leq s] = X_s \ \forall 0 \leq s \leq t$.

2.1.2 Options

An option is a contingent claim where the holder of the option buys the right to sell or purchase an underlying asset at some specified price K, called strike price or just strike, at (or by) a specified time T, called maturity or expiry. The value of the underlying asset at time t will be denoted by S_t.

- A *call* option gives the holder the right to *buy* the underlying asset.
- A *put* option gives the holder the right to *sell* the underlying asset.

European Options

A European option[9] is one of the most common types of options where the option can only be exercised *at* maturity, T. The terminal payoff is therefore given by

$$H^{Call} = \max\{S_T - K, 0\} \tag{2.12}$$

$$H^{Put} = \max\{K - S_T, 0\}. \tag{2.13}$$

Hence, the value of the European option at time $t = 0$ is the discounted risk-neutral expectation

$$\Pi_0^{Call} = e^{-rT} E^{\mathbb{Q}}[H^{Call}] \tag{2.14}$$

$$\Pi_0^{Put} = e^{-rT} E^{\mathbb{Q}}[H^{Put}]. \tag{2.15}$$

Put-call parity

The put-call parity describes a relationship between a European call and put option and is a result of the no-arbitrage assumption. Consider a portfolio, θ_t, consisting of a put option, P_t and a short position in a call, C_t, on the same underlying asset and with the same strike price and expiry, K, T, and one unit of the underlying asset S_t,

$$\theta_t = S_t + P_t - C_t. \tag{2.16}$$

The terminal payoff is then given by

$$\theta_T = \begin{cases} S_T + 0 - (S_T - K) = K & \text{if} \quad S_T \geq K \\ S_T + (S_T - K) - 0 = K & \text{if} \quad S_T \leq K \end{cases}$$

meaning the portfolio will always have a terminal payoff K. K can be obtained risklessly at time $t < T$ by depositing $Ke^{-r(T-t)}$ at the bank. By the no-arbitrage assumption, the value of the portfolio at time $t < T$ must therefore be

$$\theta_t = S_t + P_t - C_t = Ke^{-r(T-t)}.$$

Hence, we may always compute the value of a put option given the price of a call option on the same underlying asset, with the same strike price and expiry, and vice versa.

[9] also called plain vanilla option.

2.1.3 Dividends

The risky asset that has been considered until now has been assumed to pay out no dividend. Assume now that a continuously compounded dividend at rate q per annum is paid out to the shareholders. The stock price then follows the process

$$S_t = e^{-qt}\bar{S}_t$$

where $\bar{S}_t$ is the price process without any dividends. For the previously obtained results, this only means we discount by a rate $r - q$ instead of r. For the put-call parity, that is

$$\theta_t = e^{-q(T-t)}S_t + P_t - C_t = Ke^{-r(T-t)}.$$

Definition 2.1.7 ***Equivalent martingale measures with dividends.*** *An equivalent martingale measure, $\mathbb{Q}$, fulfills for an asset S^i with constant dividend rate q, traded at price S_t^i at time t, generating a terminal payoff S_T^i,*

$$E^{\mathbb{Q}}[e^{-(r-q)T}S_T^i|\mathcal{F}_t] = e^{-(r-q)t}S_t^i$$

Proposition 2.1.3 ***Arbitrage-free pricing with dividends.*** *In a market described by the probability measure $\mathbb{P}$, any arbitrage-free pricing rule is given by*

$$\Pi_t(H) = e^{-(r-q)(T-t)}E^{\mathbb{Q}}[H|\mathcal{F}_t] \tag{2.17}$$

for an equivalent martingale measure $\mathbb{Q}$.

2.2 Probability Theory

When modelling stock prices in a financial market, it is common to assume that it follows the *Efficient Market Hypothesis* to some degree. That is, the price at time t, fully reflects the information given at time t. This suggests the stock price is a Markov process.

Definition 2.2.1 ***Markov process.*** *A stochastic process, $X = \{X_t, t \geq 0\}$, is said to have the Markov property if, for an information flow $\{\mathcal{F}_t, t \in [0,T]\}$,*

$$P(X_t = x|\mathcal{F}_s) = P(X_t = x|x_s) \quad \text{for} \quad s < t. \tag{2.18}$$

Next, we will define some properties of random variables.

Definition 2.2.2 ***Characteristic function.*** *The characteristic function of a random variable, X, is given by*

$$\phi_X(u) = E[\exp(iuX)] = \int_{-\infty}^{\infty} \exp(iux)\mathrm{d}F(x) \tag{2.19}$$

where $F(x) = P(X \leq x)$ is the distribution function of X.

Assuming X has a kth order moment[10], it can be derived from ϕ_X by the following equation

$$E[X^k] = \mathrm{i}^{-k} \frac{\mathrm{d}}{\mathrm{d}u^k} \phi_X(u)\Big|_{u=0}.$$

Definition 2.2.3 ***Infinite Divisibility.*** *Suppose a distribution F on $\mathbb{R}^d$, has a characteristic function $\phi(u)$. Then we say that the distribution is infinitely divisible if, for every integer $n \geq 2$, $\phi(u)$ is also the nth power of a characteristic function, $\phi_n(u)$. I.e.*

$$\phi(u) = (\phi_n(u))^n. \tag{2.20}$$

Definition 2.2.3 is the same as saying that a random variable $Z \sim F$ is infinitely divisible if

$$Z \overset{F}{=} Y_1 + \cdots + Y_n$$

for independent and identically distributed variables $Y_i, i = 1, \ldots n$.

Definition 2.2.4 ***Skewness.*** *For a random variable, X, with mean μ_X and variance σ_X^2, its skewness is defined as its third order standardized moment,*

$$\gamma_1 = \frac{E[(X - \mu_X)^3]}{(\sigma_X^2)^{3/2}}. \tag{2.21}$$

The skewness measures the degree of asymmetry in a distribution. A symmetric distribution will have skewness equal to zero, while distributions with longer left tail than right tail is said to have negative skewness and vice versa for positive skewness. In finance, skewness is a result of risk adverse investors in encounter with the risk of price jumps.

Definition 2.2.5 ***Kurtosis.*** *For a random variable, X, with mean μ_X and variance σ_X^2, its kurtosis is defined as its fourth order standardized moment,*

$$\gamma_2 = \frac{E[(X - \mu_X)^4]}{(\sigma_X^2)^2}. \tag{2.22}$$

A distribution with kurtosis equal to 3 is said to be mesokurtic. If a distribution has kurtosis larger than 3 it is said to be leptokurtic and will have a higher peak and heavier tails than a mesokurtic distribution, while a distribution with kurtosis less than 3 will have a flatter top and less heavy tails is said to be platykurtic. Excess kurtosis is a result of the price jumps and is reflected in the risk premium on deep in- and out-of-the-money options[11].

[10] $E[|X|^k] < \infty$

[11] If an option expires with value of the underlying far from the strike price.

2.3 Lévy Processes

Definition 2.3.1 ***Lévy process.*** *A cadlag*[12] *stochastic process* $X = \{X_t, t \geq 0\}$ *with values in* $\mathbb{R}^d$ *on* $(\Omega, \mathcal{F}, \mathbb{P})$ *such that* $X_0 = 0$*, is a Lévy process if*

- *it has independent increments, i.e. for an increasing time sequence* $t_0, t_1, \dots t_n$*, the increments* $X_{t_0}, X_{t_2} - X_{t_1}, \dots X_{t_n} - X_{t_{n-1}}$ *are independent.*
- *it has stationary increments, i.e.* $X_{t+h} - X_t$ *has the same distribution as* X_h*.*
- *it is stochastically continuous, i.e.* $\forall \epsilon > 0$, $\lim_{h \to 0} \mathbb{P}(|X_{t+h} - X_t| \geq \epsilon) = 0$.

For Lévy processes, the following theorem is true for infinite divisibility.

Theorem 2.3.1 ***Infinite divisibility for Lévy Processes.*** *If* $X = \{X_t, t \geq 0\}$ *is a Lévy process, then for every* t*,* X_t *has an infinitely divisible distribution* F*, or conversely, if* F *is an infinitely divisible distribution, then there exists a Lévy process* $X = \{X_t, t \geq 0\}$ *such that* $X_1 \sim F$*.*

Definition 2.3.1 tells us there is a bijection between Lévy processes and infinitely divisible distributions.[13] Furthermore, a Lévy process is uniquely determined by its Lévy triplet $[\gamma, \Sigma, \nu(\mathrm{d}x)]$ which comes from the Lévy-Khintchine formula.

Theorem 2.3.2 ***Lévy-Khintchine formula.*** *For a Lévy process* $X = \{X_t, t \geq 0\}$ *taking values in* $\mathbb{R}^d$*,* $\gamma \in \mathbb{R}^d$*, a positive definite matrix* $\Sigma \in \mathbb{R}^{d \times d}$ *and* $\nu(\mathrm{d}x)$ *a nonnegative measure on* $\mathbb{R} \setminus \{0\}$ *with* $\int_{\mathbb{R}\setminus\{0\}} (1 \wedge |x|^2)\nu(\mathrm{d}x) < \infty$*, the characteristic exponent* $\psi(u)$ *satisfies the following*

$$\phi_{X_t}(u) = E[e^{iuX_t}] = e^{-t\psi(u)} \tag{2.23}$$

$$\psi(u) = -iu^\top \gamma + \frac{1}{2} u^\top \Sigma u - \int_{-\infty}^{\infty} (e^{iu^\top x} - 1 - iu^\top x \mathbb{1}_{|x| \leq 1})\nu(\mathrm{d}x) \quad u \in \mathbb{R}^d. \tag{2.24}$$

γ represents the deterministic drift components, Σ the Brownian components and $\nu(\mathrm{d}x)$ is the Lévy measure which represents the jump components. If $\nu(\mathrm{d}x)$ is on the form $\nu(\mathrm{d}x) = u(x)\mathrm{d}x$, $u(x)$ is called the Lévy density and follows the same mathematical restrictions as a probability density function apart from having to be integrable and it must have zero mass at the origin. The jumps in a Lévy process with Lévy measure $\nu(\mathrm{d}x)$, of sizes in a set A, occur according to a Poisson process with intensity $\int_A \nu(\mathrm{d}x)$.

Next, two properties of Lévy processes will be presented, namely variation and activity.

Proposition 2.3.1 ***Variation.*** *Let* $X = \{X_t, t \geq 0\}$ *be a Lévy process with Lévy triplet* $[\gamma, \Sigma, \nu(\mathrm{d}x)]$*. Then if*

- $\Sigma = 0$ *and* $\int_{|x| \leq 1} |x|\nu(\mathrm{d}x) < \infty$*, almost all paths of* X_t *is of finite variation.*

[12] right-continuous with left-limits

[13] See [51] for details.

- $\Sigma \neq 0$ *or* $\int_{|x|\leq 1} |x|\nu(\mathrm{d}x) < \infty$*, almost all paths of* X_t *is of infinite variation.*

Proposition 2.3.2 ***Activity.*** *Let* $X = \{X_t, t \geq 0\}$ *be a Lévy process with Lévy triplet* $[\gamma, \Sigma, \nu(\mathrm{d}x)]$*. Then if*

- $\nu(\mathbb{R}) < \infty$*,* X_t *is of finite activity.*

- $\nu(\mathbb{R}) = \infty$*,* X_t *is of infinite activity.*

Activity tells us if the number of jumps on any finite interval is finite or not. For proof, please see Theorem 21.3 and 21.9 in Sato [51].

Concluding this chapter, we will define a useful class of Lévy processes.

Definition 2.3.2 ***Subordinator.*** *A subordinator is a nonnegative nondecreasing Lévy process. Its Lévy triplet is* $[\gamma, 0, \nu(\mathrm{d}x)]$ *such that* $\gamma \geq 0$ *and* $\nu(\mathrm{d}x)|_{\mathbb{R}_{x\leq 0}} = 0$.

Chapter 3

The Lévy Market Model

In this chapter, we will have a look at the Black-Scholes framework and motivate for, and present, five Lévy models.

We will consider a financial market on $(\Omega, \mathcal{F}, \mathbb{P})$, with information flow $\{\mathcal{F}_t, t \in [0,T]\}$, consisting of two assets, namely the risk-free asset $\{B_t, t \in [0,T]\}$ satisfying

$$\begin{cases} dB_t = rB_t \mathrm{d}t, & t \in [0,T] \\ B_0 = 1, \end{cases}$$

where r is the risk free interest rate, and the risky asset, $(S_t)_{t\in[0,T]}$, being a stock or index, satisfying

$$S_t = S_0 e^{X_t}. \tag{3.1}$$

When the log returns $X_t = \log(S_t) - \log(S_0)$ follow the increments of length t of a Lévy process, equation (3.1) is called the exponential Lévy model.

3.1 The Brownian Motion

The first Lévy process we will consider for the log returns is the Brownian motion which implies that (3.1) is of the form

$$S_t = S_0 \exp(X_t^{\text{Normal}}) \tag{3.2}$$

(3.2) solves the stochastic differential equation

$$\mathrm{d}S_t = S_t(\mu \mathrm{d}t + \sigma \mathrm{d}W_t) \tag{3.3}$$

where $\{W_t, t \geq 0\}$ is a standard Brownian motion[1]. Applying Itô's formula[2], (3.2) yields that $X_{s+t}^{\text{Normal}} - X_s^{\text{Normal}} \sim \text{Normal}\Big((\mu - \frac{1}{2}\sigma^2)t, \sigma^2 t\Big)$.

[1] A Lévy process with Normally distributed increments with mean 0 and variance equal to the length of the time increment.

[2] See Appendix C.

Definition 3.1.1 ***Normal distribution.*** *The Normal distribution,* $\text{Normal}(\mu, \sigma^2)$*, can be expressed through its density function and characteristic function respectively,*

$$f_{\text{Normal}}(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.4}$$

$$\phi_{\text{Normal}}(u; \mu, \sigma^2) = e^{iu\mu} e^{-\frac{1}{2}\sigma^2 u^2}. \tag{3.5}$$

As we can see from definition 3.1.1, the Normal distribution does not have a jump component, hence its Lévy triplet is given by $[\mu, \sigma^2, 0]$. Moments of the Normal distribution are listed in table 3.1.

Table 3.1: Moments of the Normal distribution.

	$\text{Normal}(\mu, \sigma^2)$
mean	μ
variance	σ^2
skewness	0
kurtosis	3

From table 3.1, we can see that the Normal distribution is a symmetric and mesokurtic distribution. In addition, the Brownian motion is of infinite variation[3].

When the log returns follow a Brownian motion, the stock price itself follows a Geometric Brownian motion (GBM), and is lognormally distributed.

The stock price under the risk-neutral price measure $\mathbb{Q}$, is unique since the market model is complete[4], and can be derived by Girsanov theorem [26]. The risk-neutral log returns, $\tilde{X}_t$, are then given by

$$\tilde{X}_t^{\text{Normal}} \sim \text{Normal}\Big((r - q - \frac{1}{2}\sigma^2)t, \sigma^2 t\Big).$$

We can see that there has been introduced a risk-neutral drift, $\tilde{\mu} = r - q$ where r is the risk-free interest rate and q is the dividend rate.

3.1.1 Shortcomings of the Geometric Brownian Market Model

The lognormally distributed underlying asset is the framework of the Black-Scholes formula, which is built upon several assumptions[5], which has several weaknesses.

[3] does not make sense to give meaning to activity for a Brownian motion.
[4] Every contingent claim can be perfectly replicated by a dynamic trading strategy.
[5] See e.g. [52].

In the Geometric Brownian motion pricing dynamics, we have a complete market, meaning that options are redundant as all risk can be hedged away by dynamically trading the underlying asset. In reality, we do not have a complete market, and options are not redundant.

The properties of table 3.1 shows that the Normal distribution is a symmetric and mesokurtic distribution. However, log returns are often observed with significant negative skewness. Large movements in asset price are also observed more frequently than in the Normal distribution. This can be seen as a higher peak and heavier tails in the empiric distribution of the log returns than in the Normal distribution and indicate a leptokurtic distribution. These two features motivate for distributions with more flexibility in the third and fourth order standardized moments.

Another important observation is the volatility. In the Black-Scholes model, the volatility is assumed to be constant during the life-span of the option, while empirically, we observe that the volatility changes stochastically. We may also observe *volatility clusters* by considering the absolute value of the log returns which shows that periods with high variance more often are followed by new periods with high variance and periods with low variance more often are followed by new periods with low variance.

In addition, the Brownian motion is a continuous process which means it does not allow for any discontinuous jumps, whereas asset prices are often observed with jumps. This motivates for finding processes which allow for jumps to occur. As mentioned in section 2.2, the skewness is a consequence of the risk associated with the jumps. The Geometric Brownian motion, being a continuous model, hence, can not account for this.

For stock prices we may also observe aggregated normality. Stock prices may be collected several times a day, daily, weekly or monthly and so on. The larger the intervals between price observations are, the closer to a Brownian motion the price paths appear. This is observed by e.g. Eberlein & Keller [20] for estimated parameters of the Hyperbolic distribution for increased time lags. However, the shorter the intervals, the more evident the discontinuities become. Hence, the time horizon is important to consider.

The processes which are presented next, have both the wanted flexibility in skewness and kurtosis as well as the ability to model jumps. In addition, they are incomplete, meaning options are not redunant. They can also be expanded to allow for stochastic volatility.[6]

[6] See for example [52].

3.2 The Normal Inverse Gaussian Process

The Normal Inverse Gaussian distribution (NIG) was introduced by Barndorff-Nielsen [8] in 1995.

Definition 3.2.1 ***Normal Inverse Gaussian distribution.*** *The Normal Inverse Gaussian distribution,* $\mathrm{NIG}(\alpha, \beta, \delta)$*, can be expressed through its density function and characteristic function respectively,*

$$f_{\mathrm{NIG}}(x; \alpha, \beta, \delta) = \frac{\alpha\delta}{\pi} \exp(\delta\sqrt{\alpha^2 - \beta^2} + \beta x) \frac{K_1(\alpha\sqrt{\delta^2 + x^2})}{\sqrt{\delta^2 + x^2}}$$
$$\phi_{\mathrm{NIG}}(u; \alpha, \beta, \delta) = \exp(-\delta(\sqrt{\alpha^2 - (\beta + iu)^2} - \sqrt{\alpha^2 - \beta^2}))$$

for $\alpha > 0$*,* $|\beta| < \alpha$ *and* $\delta > 0$*, where* K_1 *is the modified Bessel function of the third kind with index* 1.[7]

From definition 3.2.1, we can define the Normal Inverse Gaussian process, $X^{\mathrm{NIG}} = \{X_t^{\mathrm{NIG}}, t \geq 0\}$. As the characteristic function of the process satisfies

$$\begin{aligned} E[\exp(iuX_t^{\mathrm{NIG}})] =& (\phi_{\mathrm{NIG}}(u; \alpha, \beta, \delta))^t \\ & \phi_{\mathrm{NIG}}(u; \alpha, \beta, t\delta), \end{aligned}$$

the increments of the process follow $X_{s+t}^{\mathrm{NIG}} - X_s^{\mathrm{NIG}} \sim \mathrm{NIG}(\alpha, \beta, t\delta)$. Moments of the Normal Inverse Gaussian are listed in table 3.2.

Table 3.2: Moments of the Normal Inverse Gaussian process.

	$\mathrm{NIG}(\alpha, \beta, \delta)$
mean	$\delta\beta/\sqrt{\alpha^2 - \beta^2}$
variance	$\alpha^2\delta(\alpha^2 - \beta^2)^{-3/2}$
skewness	$3\beta\alpha^{-1}\delta^{-1/2}(\alpha^2 - \beta^2)^{-1/4}$
kurtosis	$3\Big(1 + \frac{\alpha^2 + 4\beta^2}{\delta\alpha^2\sqrt{\alpha^2 - \beta^2}}\Big)$

The Lévy triplet is given by $[\gamma, 0, \nu_{\mathrm{NIG}}]$ such that

$$\gamma = \frac{2\delta\alpha}{\pi} \int_0^1 \sinh(\beta x) K_1(\alpha x) \mathrm{d}x$$

and

$$\nu_{\mathrm{NIG}}(\mathrm{d}x) = \frac{\delta\alpha}{\pi} \frac{\exp(\beta x) K_1(\alpha|x|)}{|x|} \mathrm{d}x.$$

[7] See Appendix B.

The process is of infinite variation and infinite activity and may be written as an Inverse Gamma subordinated Brownian motion. Let $X_t^{\mathrm{IG}} \sim \mathrm{IG}(t, \delta\sqrt{\alpha^2-\beta^2})$ be an Inverse Gamma process[8]. For a standard Brownian motion $W = \{W_t, t \geq 0\}$,

$$X_t^{\mathrm{NIG}} = \beta\delta^2 X_t^{\mathrm{IG}} + \delta W_{X_t^{\mathrm{IG}}}.$$

3.3 The Variance Gamma Process

The Variance Gamma process (VG) was first introduced by Madan & Seneta [39], and was considered along with [40] and [38] for the symmetric case, before the general case allowing skewness was presented by Madan *et al.* [36].

Definition 3.3.1 ***Variance Gamma distribution.*** *The Variance Gamma distribution,* $\mathrm{VG}(\sigma, \nu, \theta)$*, can be expressed through its density function and characteristic function respectively,*

$$f_{\mathrm{VG}}(x;\sigma,\nu,\theta) = \frac{\sqrt{2}\exp(\theta x/\sigma^2)}{\sigma\sqrt{\pi}\nu^{1/\nu}\Gamma(\frac{1}{\nu})}\left(\frac{|x|}{\sqrt{\frac{2\sigma^2}{\nu}+\theta^2}}\right)^{\frac{1}{\nu}-\frac{1}{2}} K_{\frac{1}{\nu}-\frac{1}{2}}\left(\frac{|x|\sqrt{\frac{2\sigma^2}{\nu}+\theta^2}}{\sigma^2}\right) \quad (3.6)$$

$$\phi_{\mathrm{VG}}(u;\sigma,\nu,\theta) = (1 - iu\theta\nu + \frac{1}{2}\sigma^2\nu^2)^{-1/\nu} \quad (3.7)$$

for $\sigma > 0$*,* $\nu > 0$*,* $\theta \in \mathbb{R}$*.*

From definition 3.3.1, we can define the Variance Gamma process, $X^{\mathrm{VG}} = \{X_t^{\mathrm{VG}}, t \geq 0\}$. As the characteristic function of the process satisfies

$$\begin{aligned} E[\exp(iuX_t^{\mathrm{VG}})] =& (\phi_{\mathrm{VG}}(u;\sigma,\nu,\theta))^t \\ & \phi_{\mathrm{VG}}(u;\sigma\sqrt{t},\nu/t,t\theta), \end{aligned}$$

the increments of the process follow $X_{s+t}^{\mathrm{VG}} - X_s^{\mathrm{VG}} \sim \mathrm{VG}(\sigma\sqrt{t}, \nu/t, t\theta)$. Moments of the Variance Gamma are listed in table 3.3.

Table 3.3: Moments of the Variance Gamma process.

	$\mathrm{VG}(\sigma,\nu,\theta)$
mean	θ
variance	$\sigma^2 + \nu\theta^2$
skewness	$\theta\nu(3\sigma^2 + 2\nu\theta^2)/(\sigma^2 + \nu\theta^2)^{3/2}$
kurtosis	$3(1 + 2\nu - \nu\sigma^4(\sigma^2 + \nu\theta^2)^{-2})$

The Variance Gamma process is of finite variation and infinite activity and may be

[8] See Appendix A.

written as a Gamma subordinated Brownian motion. Let $X_t^{\mathrm{G}} \sim \mathrm{Gamma}(t/\nu, 1/\nu)$ be a Gamma process[9]. For a standard Brownian motion $W = \{W_t, t \geq 0\}$,

$$X_t^{\mathrm{VG}} = \theta X_t^{\mathrm{G}} + \sigma W_{X_t^{\mathrm{G}}}.$$

The Variance Gamma process is also often encountered with two other parametrizations. If we write

$$\begin{aligned} C &= 1/\nu > 0 \\ G &= (\sqrt{\frac{1}{4}\theta^2\nu^2 + \frac{1}{2}\sigma^2\nu} - \frac{1}{2}\theta\nu)^{-1} \\ M &= (\sqrt{\frac{1}{4}\theta^2\nu^2 + \frac{1}{2}\sigma^2\nu} + \frac{1}{2}\theta\nu)^{-1}, \end{aligned}$$

we can express the Lévy triplet as $[\gamma, 0, \nu_{\mathrm{VG}}]$, such that

$$\gamma = \frac{-C(G(e^{-M} - 1) - M(e^{-G} - 1))}{MG}$$

and

$$\nu_{\mathrm{VG}}(\mathrm{d}x) = \begin{cases} C\exp(Gx)|x|^{-1}\mathrm{d}x, & x < 0 \\ C\exp(-Mx)x^{-1}\mathrm{d}x, & x > 0. \end{cases}$$

The characteristic function is then

$$\phi_{\mathrm{VG}}(u; C, G, M) = \Big(\frac{GM}{GM + (M - G)iu + u^2}\Big)^C.$$

From this parametrization, we can see that the process may also be expressed as the difference between two independent Gamma processes,

$$X_t^{\mathrm{VG}} = X_t^{G_1} - X_t^{G_2}$$

where $X_t^{G_1} \sim \mathrm{Gamma}(Ct, M)$ and $X_t^{G_2} \sim \mathrm{Gamma}(Ct, G)$.

3.4 The Generalized Hyperbolic Process

The Generalized Hyperbolic process (GH) was introduced by Barndorff-Nielsen [7] for the purpose of modelling distributions of the mass size of aeolian sand deposits in 1977. The subclasses of the distribution were first proposed for financial data, before the generalized distribution itself was studied as a model for financial log-returns by Eberlein & Prause [21] and Prause [47].

Definition 3.4.1 ***Generalized Hyperbolic distribution.*** *The Generalized Hyperbolic distribution,* $\mathrm{GH}(\alpha, \beta, \delta, \lambda)$*, can be expressed through its density function*

[9] See Appendix A.

and characteristic function respectively,

$$f_{\mathrm{GH}}(x;\alpha,\beta,\delta,\lambda) = a(\alpha,\beta,\delta\lambda)(\delta^2+x^2)^{(\lambda-1/2)}K_{\lambda-1/2}(\alpha\sqrt{\delta^2+x^2})\exp(\beta x) \quad (3.8)$$

$$\phi_{\mathrm{GH}}(u;\alpha,\beta,\delta,\lambda) = \Big(\frac{\alpha^2-\beta^2}{\alpha^2-(\beta+iu)^2}\Big)^{\lambda/2}\frac{K_\lambda(\delta\sqrt{\alpha^2-(\beta+iu)^2})}{K_\lambda(\delta\sqrt{\alpha^2-\beta^2})} \quad (3.9)$$

where

$$a(\alpha,\beta,\delta,\lambda) = \frac{(\alpha^2-\beta^2)^{\lambda/2}}{\sqrt{2\pi}\alpha^{\lambda-1/2}\delta^\lambda K_\lambda(\delta\sqrt{\alpha^2-\beta^2})},$$

K_λ is the modified Bessel function of the third kind with index λ[10] *and*

$$\begin{aligned}\delta &\geq 0 \quad |\beta|<\alpha \quad \textit{if } \lambda>0\\ \delta &> 0 \quad |\beta|<\alpha \quad \textit{if } \lambda=0\\ \delta &> 0 \quad |\beta|\leq\alpha \quad \textit{if } \lambda<0.\end{aligned}$$

From definition 3.4.1, we can define the Generalized Hyperbolic process, $X^{\mathrm{GH}} = \{X_t^{\mathrm{GH}}, t\geq 0\}$. As the distribution is infinitely divisible, we can see that the increments follow

$$E[\exp(iuX_t^{\mathrm{GH}})] = (\phi_{\mathrm{GH}}(u;\alpha,\beta,\delta,\lambda))^t.$$

However, the Generalized Hyperbolic process is not closed under convolution, so it is not as straightforward changing the time-scale of the process. Moments of the Generalized Hyperbolic are listed in table 3.4.

Table 3.4: Moments of the Generalized Hyperbolic process. $\xi = \delta\sqrt{\alpha^2-\beta^2}$

	$\mathrm{GH}(\alpha,\beta,\delta,\lambda)$
mean	$\delta\beta(\alpha^2-\beta^2)^{-1/2}K_{\lambda+1}(\xi)K_\lambda^{-1}(\xi)$
variance	$\delta^2\Big(\frac{K_{\lambda+1}(\xi)}{\xi K_\lambda(\xi)}+\frac{\beta^2}{\alpha^2-\beta^2}\Big(\frac{K_{\lambda+2}(\xi)}{K_\lambda(\xi)}-\frac{K_{\lambda+1}^2(\xi)}{K_\lambda^2(\xi)}\Big)\Big)$

The Generalized Hyperbolic process does not have a Brownian component. Its Lévy measure is given by equation (3.10). The process is of infinite variation and infinite activity.

$$\nu_{\mathrm{GH}}(\mathrm{d}x) = \begin{cases}\dfrac{\exp(\beta x)}{|x|}\Big(\int_0^\infty \dfrac{\exp(-|x|\sqrt{2y+\alpha^2})}{\pi^2 y(J_\lambda^2(\delta\sqrt{2y})+N_\lambda^2(\delta\sqrt{2y}))}\mathrm{d}y + \lambda\exp(-\alpha|x|)\Big), & \lambda\geq 0\\ \dfrac{\exp(\beta x)}{|x|}\int_0^\infty \dfrac{\exp(-|x|\sqrt{2y+\alpha^2})}{\pi^2 y(J_{-\lambda}^2(\delta\sqrt{2y})+N_{-\lambda}^2(\delta\sqrt{2y}))}\mathrm{d}y, & \lambda<0\end{cases} \quad (3.10)$$

[10] See Appendix B

$J_\lambda(z)$ and $N_\lambda(z)$ are Bessel functions of first and second kind respectively[11].

The Generalized Hyperbolic process may also be written as a Generalized Inverse Gaussian subordinated Brownian motion.

The processes described in section 3.2 and 3.3 are special cases of the Generalized Hyperbolic distribution when $\lambda = -1/2$ and $\delta \to 0$ respectively. The change of parametrization for the Variance Gamma process is given by $\lambda = 1/\nu$, $\alpha = (\sqrt{(2\sigma^2/\nu) + \theta^2})/\sigma^2$, $\beta = \theta/\sigma^2$. The last special case of the Generalized Hyperbolic process used in financial modelling we will consider is when $\lambda = 1$ which is called the Hyperbolic process .

3.5 The Hyperbolic Process

The Hyperbolic distribution (H) was first applied to financial data by Eberlein & Keller [20]. The model is also studied in Bingham & Kiesel [12].

Definition 3.5.1 ***Hyperbolic distribution.*** *The Hyperbolic distribution,* $\mathrm{H}(\alpha, \beta, \delta)$*, can be expressed through its density function and characteristic function respectively,*

$$f_{\mathrm{H}}(x; \alpha, \beta, \delta) = \frac{\sqrt{\alpha^2 - \beta^2}}{2\delta\alpha K_1(\delta\sqrt{\alpha^2 - \beta^2})} \exp(-\alpha\sqrt{\delta^2 + x^2} + \beta x) \tag{3.11}$$

$$\phi_{\mathrm{H}}(u; \alpha, \beta, \delta) = \Big(\frac{\alpha^2 - \beta^2}{\alpha^2 - (\beta + iu)^2}\Big)^{1/2} \frac{K_1(\delta\sqrt{\alpha^2 - (\beta + iu)^2})}{K_1(\delta\sqrt{\alpha^2 - \beta^2})}. \tag{3.12}$$

for $\alpha > 0$, $0 \leq |\beta| < \alpha$ *and* $\delta > 0$.

From definition 3.5.1, we can define the Hyperbolic process, $X^{\mathrm{H}} = \{X_t^{\mathrm{H}}, t \geq 0\}$. As the distribution is infinitely divisible, its increments follow

$$E[\exp(iuX_t^{\mathrm{H}})] = (\phi_{\mathrm{H}}(u; \alpha, \beta, \delta))^t.$$

Moments of the Hyperbolic are listed in table 3.5.

The Hyperbolic process does not have a Brownian component. Its Lévy measure is given by equation (3.13). The process is of infinite variation and infinite activity.

$$\nu_{\mathrm{H}}(\mathrm{d}x) = \frac{\exp(\beta x)}{|x|}\Big(\int_0^\infty \frac{\exp(-|x|\sqrt{2y + \alpha^2})}{\pi^2 y(J_1^2(\delta\sqrt{2y}) + N_1^2(\delta\sqrt{2y}))}\mathrm{d}y + \exp(-\alpha|x|)\Big) \tag{3.13}$$

$J_\lambda(z)$ and $N_\lambda(z)$ are Bessel functions of first and second kind respectively[12].

[11] See Appendix B.
[12] See Appendix B.

Table 3.5: Moments of the Hyperbolic process.
$\xi = \delta\sqrt{\alpha^2 - \beta^2}$

	$\mathrm{H}(\alpha, \beta, \delta)$
mean	$\delta\beta(\alpha^2 - \beta^2)^{-1/2} K_2(\xi) K_1^{-1}(\xi)$
variance	$\delta^2\Big(\frac{K_2(\xi)}{\xi K_1(\xi)} + \frac{\beta^2}{\alpha^2 - \beta^2}\Big(\frac{K_3(\xi)}{K_1(\xi)} - \frac{K_2^2(\xi)}{K_1^2(\xi)}\Big)\Big)$

3.6 The CGMY Process

The last Lévy model we will consider is the CGMY process which was introduced by Carr *et al.* [14] as a more flexible alternative to the VG process by adding an extra parameter, Y, to the C, G, M-parametrization. It was later generalized by Carr *et al.* [15] to a six parameter distribution by splitting C and Y into a positive and negative part, but only the four parameter model will be considered in this thesis.

Definition 3.6.1 ***CGMY distribution.*** *The CGMY distribution,* $\mathrm{CGMY}(C, G, M, Y)$*, can be expressed through its characteristic function,*

$$\phi_{\mathrm{CGMY}}(u; C, G, M, Y) = \exp(C\Gamma(-Y)((M - iu)^Y - M^Y + (G + iu)^Y - G^Y)) \quad (3.14)$$

for $C, G, M > 0$ *and* $Y < 2$*.*

The CGMY density function does not exist on closed form, but its characteristic function does. From definition 3.6.1 we can define the CGMY process, $X^{\mathrm{CGMY}} = \{X_t^{\mathrm{CGMY}}, t \geq 0\}$. As the characteristic function of the process satisfies

$$\begin{aligned} E[\exp(iuX_t^{\mathrm{CGMY}})] =& (\phi_{\mathrm{CGMY}}(u; C, G, M, Y))^t \\ & \phi_{\mathrm{CGMY}}(u; Ct, G, M, Y), \end{aligned}$$

the increments of the process follow $X_{s+t}^{\mathrm{CGMY}} - X_s^{\mathrm{CGMY}} \sim \mathrm{CGMY}(Ct, G, M, Y)$. Moments of the CGMY are listed in table 3.6.

Table 3.6: Moments of the CGMY process.

	$\mathrm{CGMY}(C, G, M, Y)$
mean	$C(M^{Y-1} - G^{Y-1})\Gamma(1 - Y)$
variance	$C(M^{Y-2} - G^{Y-2})\Gamma(2 - Y)$
skewness	$C(M^{Y-3} - G^{Y-3})\Gamma(3 - Y)/(C(M^{Y-2} - G^{Y-2})\Gamma(2 - Y))^{3/2}$
kurtosis	$C(M^{Y-4} - G^{Y-4})\Gamma(4 - Y)/(C(M^{Y-2} - G^{Y-2})\Gamma(2 - Y))^2 + 3$

The Lévy triplet is given by $[\gamma, 0, \nu_{\text{CGMY}}]$ such that

$$\gamma = C\Big(\int_0^1 (\exp(-Mx) - \exp(-Gx))x^{-Y}\mathrm{d}x\Big)$$

and

$$\nu_{\text{CGMY}}(\mathrm{d}x) = C|x|^{-1-Y}(\exp(Gx)\mathbb{1}_{(x<0)} + \exp(-Mx)\mathbb{1}_{(x>0)})\mathrm{d}x).$$

The variation and activity of the CGMY process is dependent on Y. The properties are listen in table 3.7. The CGMY process may also be represented as a

Table 3.7: Variation and activity of the CGMY process.

	Variation	Activity
$Y < 0$	Finite	Finite
$Y \in (0, 1)$	Finite	Infinite
$Y \in (1, 2)$	Infinite	Infinite (quadratic)

subordinated Brownian motion. For details, please see Madan & Yor [37].

3.7 Adding Drift

We will introduce an additional drift term for all the processes. This is merely a translation by a value $\mu \in \mathbb{R}$ and does not influence the properties of infinite divisibility or activity and variation. Denote the original Lévy process, its density function, characteristic function and Lévy triplet respectively by $\bar{X}$, $\bar{f}$, $\bar{\phi}$, $[\bar{\gamma}, \bar{\sigma}^2, \bar{\nu}]$. The obtained process with additional drift, μ, in terms of the original process becomes, for the characteristic function, an additional factor

$$\phi(u) = \bar{\phi}(u)\exp(iu\mu),$$

for the process, an additional deterministic term,

$$X_t = \bar{X}_t + \mu t,$$

for the Lévy triplet, an additional term for the drift component,

$$\gamma = \bar{\gamma} + \mu \qquad \sigma^2 = \bar{\sigma}^2 \qquad \nu(\mathrm{d}x) = \bar{\nu}(\mathrm{d}x)$$

and for the density function, a translation,

$$f(x) = \bar{f}(x - \mu).$$

3.8 Equivalent Martingale Measure

For all the Lévy market models presented, apart from the Geometric Brownian motion, the market models are incomplete. Hence, there are several equivalent martingale measures that satisfies the risk-neutral pricing. In this thesis, the mean-correcting martingale measure have been found, but another popular choice in literature is the Esscher transform[13].

The mean-correcting equivalent martingale measure changes the drift term μ, presented in section 3.7, such that the discounted exponential Lévy model becomes a martingale. This is done by introducing a new drift term, $\tilde{\mu}$, satisfying

$$\tilde{\mu} = \mu + r - q - \log(\phi(-i)). \tag{3.15}$$

The new mean-corrected equivalent martingale measure are listed in table 3.8.

Table 3.8: Mean-corrected martingale measure.

Model	Mean-corrected drift, $\tilde{\mu}$
GBM	$r - q$
VG	$r - q - (1 + \theta\nu - \frac{1}{2}\sigma^2\nu)^{-1/\nu}$
NIG	$r - q + \delta(\sqrt{\alpha^2 - (\beta+1)^2} - \sqrt{\alpha^2 - \beta^2})$
GH	$r - q - \log\Big(\Big(\frac{\alpha^2 - \beta^2}{\alpha^2 - (\beta+1)^2}\Big)^{\lambda/2} \frac{K_\lambda(\delta\sqrt{\alpha^2 - (\beta+1)^2})}{K_\lambda(\delta\sqrt{\alpha^2 - \beta^2})}\Big)$
H	$r - q - \log\Big(\Big(\frac{\alpha^2 - \beta^2}{\alpha^2 - (\beta+1)^2}\Big)^{1/2} \frac{K_1(\delta\sqrt{\alpha^2 - (\beta+1)^2})}{K_1(\delta\sqrt{\alpha^2 - \beta^2})}\Big)$
CGMY	$r - q - C\Gamma(-Y)((M-1)^Y - M^Y + (G+1)^Y - G^Y)$

[13] See e.g. [52].

Chapter 4

Analysis of Market Fit

In this chapter, we will consider how we fit the Lévy models to the market, before testing them on empirical data.

4.1 Parameter Estimation

The Class of Generalized Hyperbolic Processes

The classes of Generalized Hyperbolic processes was handled with the *ghyp*-package in `R` [35]. In *ghyp*, these distributions can take three parametrizations. For the Normal Inverse Gaussian, Generalized Hyperbolic and Hyperbolic process, the *alpha.delta*-parametrization was used. It represents the multivariate Generalized Hyperbolic process as follows

$$f_{\mathbf{X}}(\mathbf{x};\alpha,\boldsymbol{\beta},\delta,\boldsymbol{\Delta},\lambda,\mu) = \frac{(\alpha^2-\boldsymbol{\beta}^\top\boldsymbol{\Delta}\boldsymbol{\beta})^{\lambda/2}}{(2\pi)^{d/2}\sqrt{|\boldsymbol{\Delta}|}\alpha^{\lambda-\frac{d}{2}}\delta^\lambda K_\lambda(\delta\sqrt{\alpha^2-\boldsymbol{\beta}^\top\boldsymbol{\Delta}\boldsymbol{\beta}})} \times \frac{K_{\lambda-\frac{d}{2}}(\alpha\sqrt{\delta^2+(\mathbf{x}-\mu)^\top\boldsymbol{\Delta}^{-1}(\mathbf{x}-\mu)})e^{\boldsymbol{\beta}^\top(\mathbf{x}-\mu)}}{(\sqrt{\delta^2+(\mathbf{x}-\mu)^\top\boldsymbol{\Delta}^{-1}(\mathbf{x}-\mu)})^{\frac{d}{2}-\lambda}},$$

such that

$\alpha>0 \quad \boldsymbol{\beta}\in\{\mathbf{x}\in\mathbb{R}^d\colon \alpha^2-\mathbf{x}^\top\boldsymbol{\Delta}\mathbf{x}>0\} \quad \delta>0 \quad \boldsymbol{\Delta}\in\{A\in\mathbb{R}^{d\times d}\colon |A|=1\} \quad \lambda\in\mathbb{R}$
$\mu\in\mathbb{R}^d$.

For the Generalized Hyperbolic process presented in chapter 3 and its special cases, denoting parameters of *ghyp* by ˜, this representation implies that

$$\tilde{\alpha}=\alpha \quad \tilde{\boldsymbol{\beta}}=\beta \quad \tilde{\delta}=\delta \quad \tilde{\boldsymbol{\Delta}}=1 \quad \tilde{\lambda}=\lambda \quad \tilde{\mu}=\mu \quad d=1.$$

For the Variance Gamma distribution, the *chi.psi*-parametrization was used which

is given by the following parameter change,
$(\alpha, \boldsymbol{\beta}, \delta, \boldsymbol{\Delta}, \lambda, \mu) \to (\chi, \psi, \gamma, \Sigma, \lambda, \mu)$:

$$\Sigma = \boldsymbol{\Delta}, \quad \gamma = \boldsymbol{\Delta}\boldsymbol{\beta}, \quad \chi = \delta^2, \quad \psi = \alpha^2 - \boldsymbol{\beta}^\top \boldsymbol{\Delta}\boldsymbol{\beta} \tag{4.1}$$

The parameters are found through maximum likelihood using an EM-scheme. For details, please see [35].

The CGMY Process

As the CGMY process does not have an analytical expression for the density function, parameter estimation through maximum likelihood is more complicated. Several approaches to parameter estimation have been explored. One can exploit that the characteristic function is of a known analytic form by finding the empirical characteristic function of the data, $\hat{\phi}(u)$ through Fast Fourier transformation. Then do a curve fitting with $\phi_{\text{CGMY}}(u; C, G, M, Y, \mu)$ as given in equation (3.14) with additional drift.[1]

Another approach is the method of moments where one considers the difference of the moments of the distribution and the empirical moments. As we, with the additional drift, have five unknowns (C, G, M, Y, μ), one can find the fifth order standardized moment such that one obtains a system of five equations and five unknowns. .

Definition 4.1.1 ***Hyperskewness.*** *For a random variable X with mean μ_X and variance σ_X^2, its hyperskewness is defined as its fifth order standardized moment,*

$$\gamma_3 = \frac{E[(X - \mu_X)^5]}{(\sigma_X^2)^{5/2}}.$$

The system obtained is given in table 4.1 and was solved using the *nleqslv*-package in `R` [29], by the Broyden method.

Table 4.1: System of equations for method of moments, CGMY.

$$\begin{aligned}
\mu_X = \;& C(M^{Y-1} - G^{Y-1})\Gamma(1-Y) + \mu \\
\sigma_X^2 = \;& C(M^{Y-2} - G^{Y-2})\Gamma(2-Y) \\
\gamma_1 = \;& C(M^{Y-3} - G^{Y-3})\Gamma(3-Y)/(C(M^{Y-2} - G^{Y-2})\Gamma(2-Y))^{3/2} \\
\gamma_2 = \;& C(M^{Y-4} - G^{Y-4})\Gamma(4-Y)/(C(M^{Y-2} - G^{Y-2})\Gamma(2-Y))^{2} + 3 \\
\gamma_3 = \;& C(M^{Y-5} - G^{Y-5})\Gamma(5-Y)/(C(M^{Y-2} - G^{Y-2})\Gamma(2-Y))^{5/2} \\
& +10C(M^{Y-3} - G^{Y-3})\Gamma(3-Y)/(C(M^{Y-2} - G^{Y-2})\Gamma(2-Y))^{3/2}
\end{aligned}$$

[1] This was implemented without success due to difficulties with fitting complex curves.

4.2 Test of Fit

In this section, the models presented in chapter 3 will be fit to the log returns on the adjusted closing prices of four data assets - two of which represents indices of different sizes and liquidities, and two representing single assets with different liquidities. An assumption in the option pricing models is that the market considered has required liquidity. Hence, it is of interest to see how the models adjust to the different liquidities.

Though the Lévy models considered in this thesis, are univariate, they have been considered on indices previously, e.g. by Madan *et al.* [36] and is in fact only considered for the S&P 500 index in Schoutens [52]. We might see if this affects the fit.

Data was downloaded using the *quantmod*-package in `R` [50] from Yahoo Finance. The details are given in table 4.2. For the parameter estimation, Madan *et al.* [36] used 691 data points on S&P 500 from January 1992 to September 1994, while Schoutens [52] considers both S&P 500 over a 30 year period as well as a 2 year period. Including historical data covering more than 10 years does not seem relevant. Both a 3 and a 5 year historical data set was considered, with 750 and 1250 data points respectively. These showed however little difference, so a 5 year historical data set was chosen.

Table 4.2: Details of data set of underlying assets.

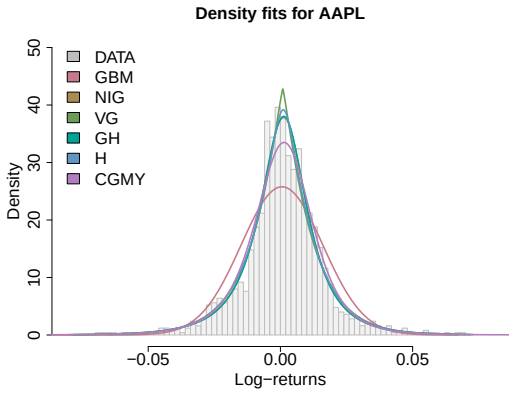
Asset/Index	Ticker	Data	Quoted in
Apple Inc.	AAPL	2014-06-12 to 2019-05-31	USD
OMX Stockholm	^OMX	2014-06-16 to 2019-05-31	SEK
Norsk Hydro ASA	NHY.OL	2014-06-06 to 2019-05-31	NOK
Oslo Stock Exchange Index	OSEBX.OL	2013-02-15 to 2018-02-07	NOK

4.2.1 Apple Inc.

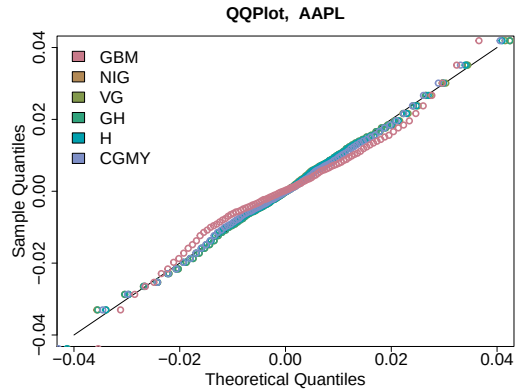
Apple Inc. is a multinational technology company, considered as one of the Big Four tech companies. It is the largest information technology company by revenue and its stocks are traded on Nasdaq, the second largest stock exchange in the world by market capitalization. I.e. it is a highly liquid asset. Fitted densities and quantile-quantile plots are given in figure 4.1a and 4.1b, while estimated parameters and moments are given in table 4.3 and 4.4 respectively.

4.2.2 OMX Stockholm 30 Index

The OMX Stockholm 30 Index is an index of the 30 most traded stock on the Stockholm Exchange. OMX is a capitalization-weighted index, i.e. the stocks are weighted according to the company's overall market value. This weighting changes



(a) Fitted densities for Apple Inc. log returns.



(b) Quantile-Quantile plot for Apple Inc. log returns.

Figure 4.1: Density and Quantile-Quantile plot for Apple Inc. log returns.

Table 4.3: Table of fitted parameters for Apple Inc. log returns.

GBM	$\mu = 0.00058$	$\sigma = 0.01547$			
NIG	$\mu = 0.00146$	$\alpha = 49.78274$	$\beta = -3.66282$	$\delta = 0.01192$	$\lambda = -0.5$
VG	$\mu = 0.00040$	$\Sigma = 0.01513$	$\psi = 2.23396$	$\gamma = -0.00032$	$\lambda = 1.11698$
GH	$\mu = 0.00146$	$\alpha = 54.13586$	$\beta = -3.61052$	$\delta = 0.01121$	$\lambda = -0.3674$
H	$\mu = 0.00115$	$\alpha = 97.06485$	$\beta = -2.51911$	$\delta = 0.0031$	$\lambda = 1$
CGMY	$\mu = 0.00194$	$C = 0.00303$	$G = 36.61029$	$M = 48.04055$	$Y = 1.11252$

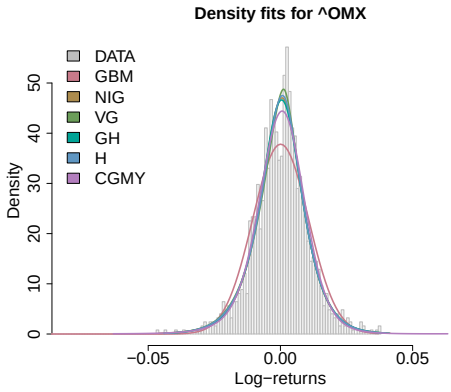
Table 4.4: Table of estimated moments for Apple Inc. log returns.

	Empirical	GBM	NIG	VG	GH	H	CGMY
μ_X	0.00057	0.00057	0.00057	0.00057	0.00055	0.00057	0.00057
σ_X^2	0.00024	0.00024	0.00024	0.00023	0.00024	0.00023	0.00024
γ_1	−0.35056	0	−0.28583	−0.05789	−0.2749	−0.10044	−0.35056
γ_2	7.25733	3	8.16529	5.68466	7.9016	5.68548	7.25733

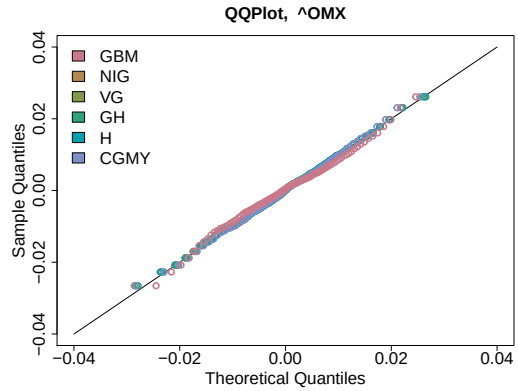
daily. All the underlying assets have perfect liquidity. Fitted densities and quantile-quantile plots are given in figure 4.2a and 4.2b, while estimated parameters and moments are given in table 4.5 and 4.6 respectively.

4.2.3 Norsk Hydro ASA

Norsk Hydro ASA is a Norwegian aluminium and renewable energy company traded on the New York Stock Exchange (NYSE) and Oslo Stock Exchange (OSE). Though



(a) Fitted densities for OMX Stockholm 30 Index log returns.



(b) Quantile-Quantile plot for OMX Stockholm 30 Index log returns.

Figure 4.2: Density and Quantile-Quantile plot for OMX Stockholm 30 Index log returns.

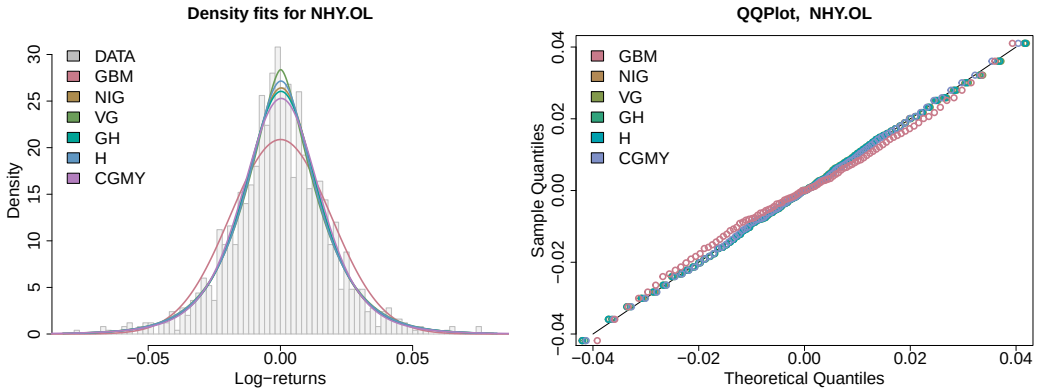
Table 4.5: Table of fitted parameters for OMX Stockholm 30 Index log returns.

GBM	$\mu = 0.00008$	$\sigma = 0.01055$			
NIG	$\mu = 0.00106$	$\alpha = 113.2662$	$\beta = -9.02901$	$\delta = 0.01227$	$\lambda = -0.5$
VG	$\mu = 0.00152$	$\Sigma = 0.01035$	$\psi = 3.80486$	$\gamma = -0.00144$	$\lambda = 1.90243$
GH	$\mu = 0.00080$	$\alpha = 122.9841$	$\beta = -6.66119$	$\delta = 0.01237$	$\lambda = -0.37137$
H	$\mu = 0.00121$	$\alpha = 162.2381$	$\beta = -10.49482$	$\delta = 0.00701$	$\lambda = 1$
CGMY	$\mu = 0.00163$	$C = 0.00017$	$G = 29.88516$	$M = 58.55552$	$Y = 1.52606$

Table 4.6: Table of estimated moments for OMX Stockholm 30 Index log returns.

	Empirical	GBM	NIG	VG	GH	H	CGMY
μ_X	0.00008	0.00008	0.00008	0.00008	0.00008	0.00008	0.00008
σ_X^2	0.00011	0.00011	0.00011	0.00011	0.00011	0.00011	0.00011
γ_1	−0.54744	0	−0.20321	−0.21977	−0.13441	−0.18856	−0.54744
γ_2	7.83977	3	5.22127	4.56587	4.97066	4.77225	7.83977

it is a large company in Norway and it is traded on NYSE, it is still smaller and more illiquid than large international companies such as Apple Inc.. Fitted densities and quantile-quantile plots are given in figure 4.3a and 4.3b, while estimated parameters and moments are given in table 4.7 and 4.8 respectively.



(a) Fitted densities of Norsk Hydro ASA log returns.(b) Quantile-Quantile plot for Norsk Hydro ASA log returns.

Figure 4.3: Density and Quantile-Quantile plot for Norsk Hydro ASA log returns.

Table 4.7: Table of fitted parameters for Norsk Hydro ASA log returns.

GBM	$\mu = 0.00007$	$\sigma = 0.01911$			
NIG	$\mu = 0.00044$	$\alpha = 57.81315$	$\beta = -1.03205$	$\delta = 0.02092$	$\lambda = -0.5$
VG	$\mu = 0.00013$	$\Sigma = 0.01894$	$\psi = 3.20022$	$\gamma = -0.00006$	$\lambda = 1.60011$
GH	$\mu = 0.00027$	$\alpha = 48.75914$	$\beta = -0.2975$	$\delta = 0.0247$	$\lambda = -1.0441$
H	$\mu = 0.00029$	$\alpha = 84.41623$	$\beta = -0.62699$	$\delta = 0.00994$	$\lambda = 1$
CGMY	$\mu = 0.00057$	$C = 0.00138$	$G = 28.70874$	$M = 31.45093$	$Y = 1.32063$

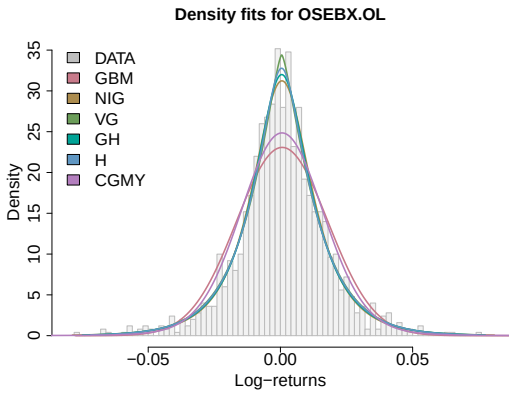
Table 4.8: Table of estimated moments for Norsk Hydro ASA log returns.

	Empirical	GBM	NIG	VG	GH	H	CGMY
μ_X	0.000007	0.000007	0.000006	0.000007	0.00017	0.000006	0.000007
σ_X^2	0.00037	0.00037	0.00036	0.00036	0.00036	0.00036	0.00037
γ_1	−0.09068	0	−0.04870	−0.00558	−0.01457	−0.02392	−0.09068
γ_2	6.48457	3	5.48442	4.87506	5.5747	5.01366	6.48457

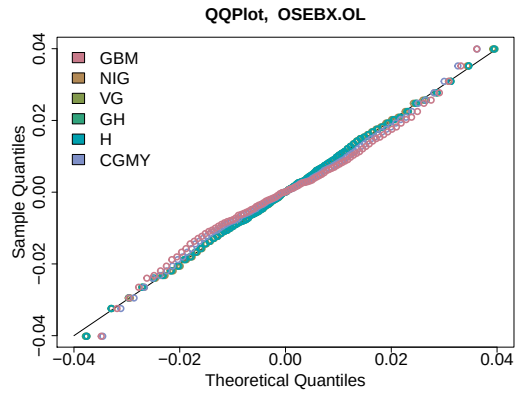
4.2.4 Oslo Stock Exchange Index

Oslo Stock Exchange Index is Oslo Stock Exchange's index which contains 70 companies and is supposed to be a representative selection of all traded stocks on OSE. OSE is the only independent stock exchange for the Nordic countries and is Norway's only regulated market for securities trading. The index is audited twice a year. Fitted densities and quantile-quantile plots are given in figure 4.4a and 4.4b,

while estimated parameters and moments are given in table 4.9 and 4.10 respectively.



(a) Fitted densities for Oslo Stock Exchange Index log returns.



(b) Quantile-Quantile plot for Oslo Stock Exchange Index log returns.

Figure 4.4: Density and Quantile-Quantile plot for Oslo Stock Exchange Index log returns.

Table 4.9: Table of fitted parameters for Oslo Stock Exchange Index log returns.

GBM	$\mu = 0.00044$	$\sigma = 0.00969$			
NIG	$\mu = 0.00128$	$\alpha = 101.2096$	$\beta = -8.94183$	$\delta = 0.0094$	$\lambda = -0.5$
VG	$\mu = 0.00114$	$\Sigma = 0.00956$	$\psi = 2.89454$	$\gamma = -0.00069$	$\lambda = 1.44727$
GH	$\mu = 0.00127$	$\alpha = 92.07335$	$\beta = -8.77391$	$\delta = 0.01013$	$\lambda = -0.71974$
H	$\mu = 0.00126$	$\alpha = 161.5059$	$\beta = -8.8891$	$\delta = 0.00389$	$\lambda = 1$
CGMY	$\mu = 0.00111$	$C = 0.00384$	$G = 78.87235$	$M = 93.10434$	$Y = 1.00857$

Table 4.10: Table of estimated moments for Oslo Stock Exchange Index log returns.

	Empirical	GBM	NIG	VG	GH	H	CGMY
μ_X	0.00045	0.00045	0.00045	0.00045	0.00045	0.00045	0.00045
σ_X^2	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009	0.00009
γ_1	−0.19865	0	−0.2742	−0.15487	−0.28551	−0.1952	−0.19865
γ_2	5.94092	3	6.26477	5.04038	6.44086	5.28717	5.94092

For all the assets, the Geometric Brownian motion clearly has the worst fit based on the density plots and quantile-quantile plots. In addition, because of its limited third and fourth order standardized moments, it is not able to pick up the negative skewness and leptokurtic behaviour in the data.

CGMY is the best model based on the estimated moments, but its parameters are found through the method of moments so it should perform well for these. Aside from the CGMY model, it may look as if the Normal Inverse Gaussian distribution and the Generalized Hyperbolic distribution comes closest in estimating the moments. Apart from looking at the OSEBX.OL data set, in which the Hyperbolic and the Variance Gamma process' moments come quite close to the empirical moments. For the AAPL, OMX and NHY.OL data set, the Variance Gamma distribution looks to be underestimating the skewness and leptokurtic behaviour.

Note on parameter estimation with CGMY

The Lévy models were fitted to several assets not presented here, which showed that the CGMY model had problems estimating parameters to some of the data sets. For instance for S&P 500, it estimated Y to be -8 which corresponds to a compound Poisson with finite activity and finite variation and makes a very poor fit. Perhaps it is the combination of empirical moments that makes the system of equations in table 4.1 hard to optimize. One possible solution could be to solve the constrained optimization problem by constraining the parameters to their given range, in particular $Y \in (0, 2)$.

Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is a test constructed to check the equality of continuous one-dimensional distributions to compare a sample with a reference distribution. It does so by quantifying the largest distance between the cumulative distribution function of the reference distribution and the empirical cumulative distribution function of the observations. Denote $F(x)$ the cumulative distribution function of the model we want to test and $F_n(x)$ as the empirical cumulative distribution function such that for n independent observations X_i,

$$F_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{X_i \leq x}$$

where $\mathbb{1}$ is the indicator function. The null hypothesis is then

$$H_0 \colon F_n(x) = F(x).$$

Theorem 4.2.1 ***Kolmogorov's Theorem.*** *Denote the Kolmogorov-statistic by* D_n *such that*

$$D_n = \sup_x |F_n(x) - F(x)|.$$

Then, for every $z \geq 0$,

$$\lim_{n\to\infty} P(\sqrt{n}D_n \leq z) = K(z)$$

where $K(z)$ *is the cumulative Kolmogorov distribution,*

$$K(z) = \frac{\sqrt{2\pi}}{x} \sum_{k=1}^{\infty} e^{-(2k-1)^2\pi^2/(8x^2)}.$$

We thus reject the null hypothesis if $\sqrt{n}D_n > K_\alpha$, where

$$P(K \leq K_\alpha) = 1 - \alpha.$$

Table 4.11 shows the qauntiles of the Kolmogorov distribution and table 4.12

Table 4.11: Quantiles of the Kolmogorov distribution.

α	0.5	0.3	0.2	0.1	0.05	0.025	0.01	0.001
K_α	0.8276	0.9731	1.0728	1.2238	1.3581	1.4802	1.6276	1.9495

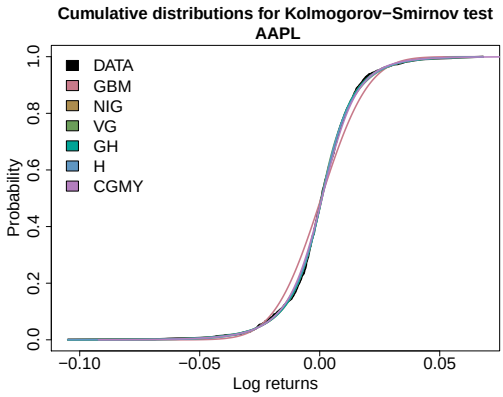
Table 4.12: Observed values of K_α.

	AAPL	^OMX	NHY.OL	OSEBX.OL
GBM	3.0493	1.59	1.9408	2.045
NIG	0.6633	0.7008	0.6106	0.5132
VG	0.9070	0.6380	0.4493	0.6638
GH	0.6498	0.8033	0.6800	0.5056
H	0.9005	0.6483	0.5036	0.6060
CGMY	1.3088	0.6903	0.7752	0.7578

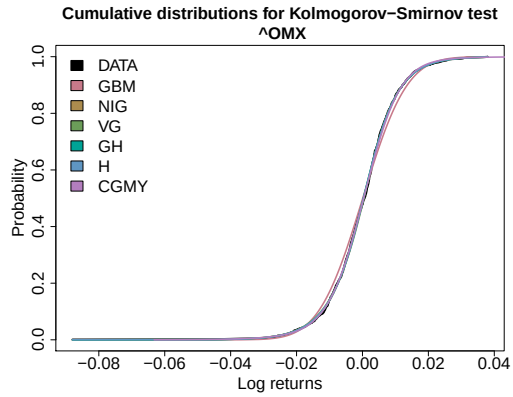
shows calculated test statistics. First, note that we would reject the Geometric Brownian motion at a 0.025 significance level for all data sets and at a 0.01 level for all assets apart from OMX. Though the CGMY model came closest to estimating the moments for all data sets, it is rejected at a 0.1 significance level for the AAPL data set. The Generalized Hyperbolic and the Variance Gamma looks to have p-values somewhere between 0.5 and 0.3, but this would not have been of significance in a typical hypothesis test. All the values not mentioned, have p-values

> 0.5, meaning we would not reject the null hypothesis.

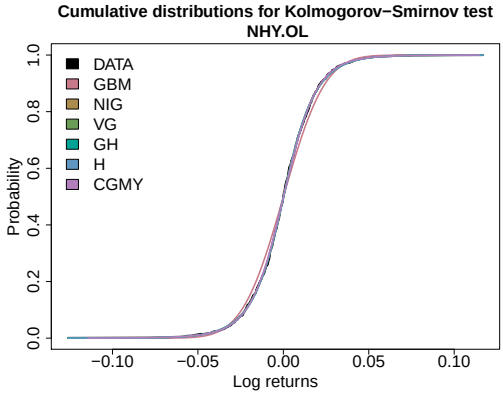
We note that the liquidities and index or single underlying asset does not seem to have any implications on the fit. In fact, the Variance Gamma, Normal Inverse Gaussian, Hyperbolic and Generalized Hyperbolic have better values for the Kolmogorov-Smirnov test for the least liquid assets.



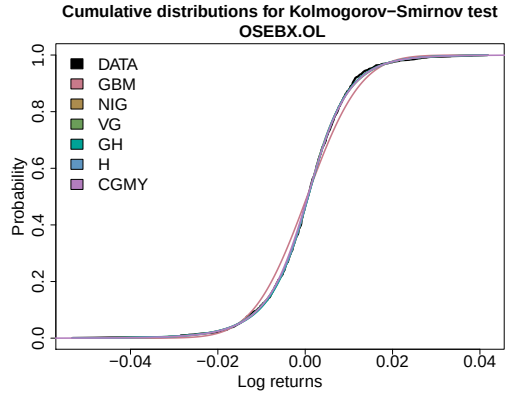
(a) Cumulative distributions for densities fitted to Apple log-returns.



(b) Cumulative distributions for densities fitted to OMX Stockholm log returns.



(c) Cumulative distributions for densities fitted to Norsk Hydro log returns.



(d) Cumulative distributions for densities fitted to Oslo Stock Exchange Index log returns.

Figure 4.5: Cumulative distributions for all fitted densities.

In figure 4.5a-4.5d the cumulative distributions of the models are compared to the empirical cumulative distribution. The Geometric Brownian motion seems to fit worst for the AAPL and OSEBX.OL log returns. This is reflected in the Kolmogorov-Smirnov values in table 4.12. The other Lévy models seem to fit the

empirical cumulative density function quite well which is also confirmed by table 4.12.

Value at Risk

The Value at Risk (VaR) is another way of measuring the goodness of the fit by considering the behaviour of the tails of the distributions. It was introduced as a way of quantifying risk in complex portfolios. The VaR is defined such that for a given probability level, α, it is the maximum loss possible for a portfolio over a future holding period.

Definition 4.2.1 ***VaR*$_\alpha$.** *The VaR of a random variable, X, with a cumulative distribution function, $F_X(x)$, at a probability level α,*

$$VaR_\alpha(X) = -\inf\{x \in \mathbb{R} \colon F_X(x) > \alpha\}.$$

In figure 4.6a-4.6d, the VaR has been visualized for probability levels $\alpha \in [0.05, 0]$, i.e. we are looking at extreme cases which there are very few observations for.

For all the data sets, the Geometric Brownian motion is definitely the worst fit of the tail behaviour. For the smaller Norwegian asset and index, NHY.OL and OSEBX.OL, there is quite similar tail behaviour of all the Generalized Hyperbolic distributions, but for the larger, more liquid index and asset, OMX and AAPL, the distributions seem to be more spread.

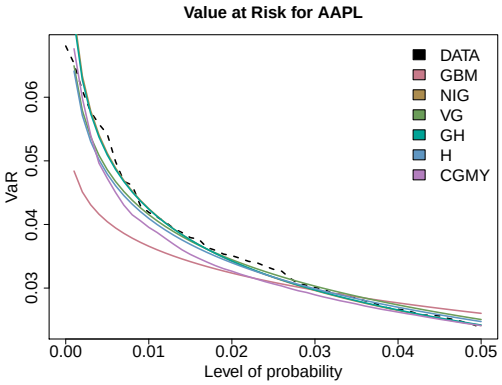
4.3 Monte Carlo Simulation

In this section, we will go through how the pricing of options are done through Monte Carlo simulation after the models have been fitted to empirical data.

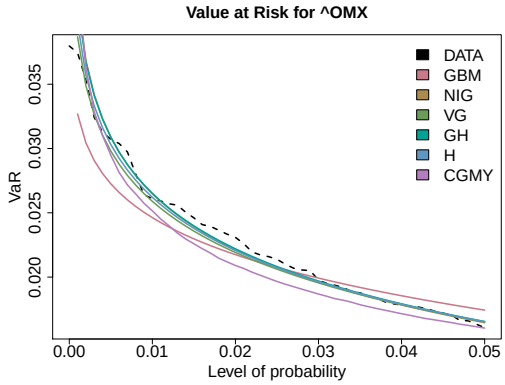
Generating Random Variables

Our market model is given by equation (3.1). First, the time series of the log returns are simulated by drawing random variables from the Lévy processes. This can be done in several ways. One can exploit the fact that all of the Lévy processes described, can be expressed as subordinated Brownian motions of simpler processes. Asmussen and Rosinski [5] showed that small jumps in a Lévy process can be approximated by a Brownian motion. Then the process can be divided into a compound Poisson process and a Brownian motion for small jumps. In this thesis, the random variables from the class of Generalized Hyperbolic processes were simulated using the *ghyp*-package in `R` [35]. For the the CGMY process, the analytic expression of the characteristic function was vectorized, before it was Fast Fourier transformed[2] to a vectorized probability distribution which could be used to generate the random variables using the *sample*-function in `R`.

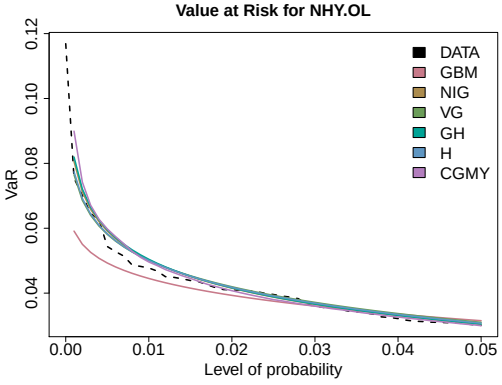
[2] See [1].



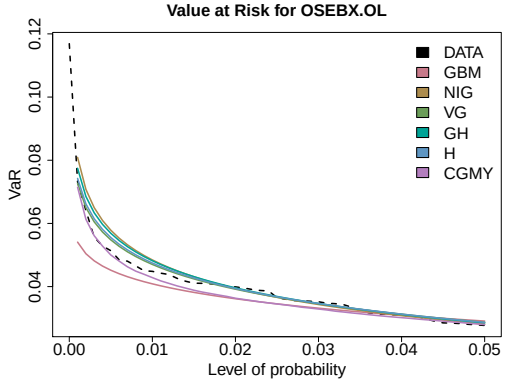
(a) VaR visualized for Apple Inc. log-returns.



(b) VaR visualized for OMX Stockholm 30 Index log returns.



(c) VaR visualized for Norsk Hydro ASA log returns.



(d) VaR visualized for Oslo Stock Exchange Index log returns.

Figure 4.6: VaR visualized for all fitted densities.

Generating Price Paths and Pricing Options

Assume that N random variables from the Lévy processes has been generated,

$$X^j_{\Delta t_i}, \quad \Delta t_i = t_{i+1} - t_i, \quad t_i = t_0, \dots t_N, \quad j = 1 \dots N_{sim}.$$

Stock price paths are then simulated as

$$S^j_{t_i} = S^j_0 (\sum_{k=1}^{i} X^j_{\Delta t_k}), \quad t_i = t_0, \dots t_N, \quad j = 1 \dots N_{sim}.$$

for N_{sim} simulations and an equally spaced time grid such that $t_0 = 0$ and $t_N = T$.

Because of the put-call parity, we may always find the price of a put option given the price of a call option and vice verse. Hence, we limit ourselves to only finding the call option price. The estimate of the call option price at time $t = 0$, $\hat{\Pi}_0^{Call}$, is then found by calculating the discounted expectation of the terminal payoff,

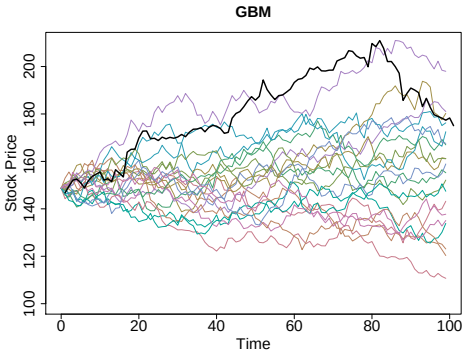
$$\hat{\Pi}_0^{Call} = e^{-rT} \frac{1}{N_{sim}} \sum_{j=1}^{N_{sim}} \max\{S_T^j - K, 0\}.$$

Two remarks about the described approach is that this is an exact solution to the stochastic differential equation,

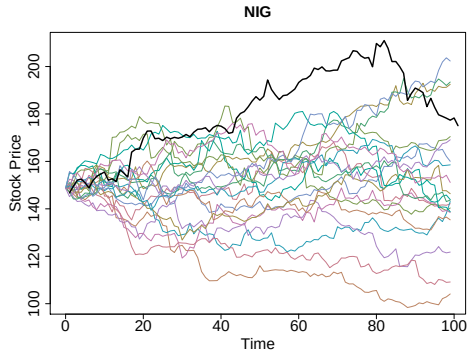
$$\mathrm{d}S_t = S_t \mathrm{d}X_t. \tag{4.2}$$

Hence, the spacing in the time grid should not make a significant difference. Also, this is a very straight forward simulation approach, there are several ways in which to improve the simulation. Please see e.g. Glasserman [27].

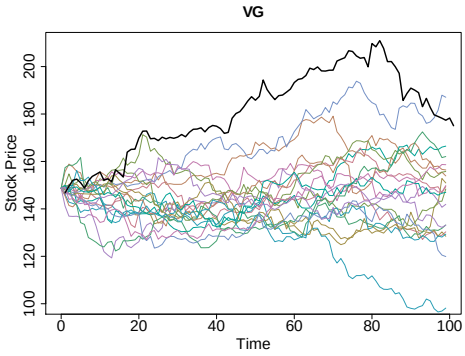
In figure 4.7a-6.2f, 20 stock price paths have been simulated for 100 days with a time space grid of 1 day for AAPL adjusted closing prices. The black price path is the true AAPL adjusted closing price path and the coloured paths are simulations.



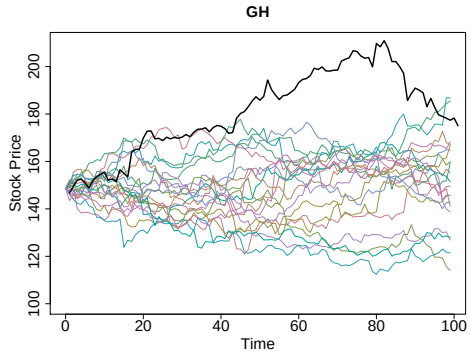
(a) 20 Monte Carlo simulated Geometric Brownian Motion price paths.



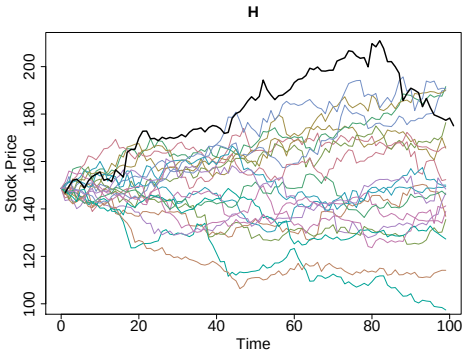
(b) 20 Monte Carlo simulated Normal Inverse Gaussian price paths.



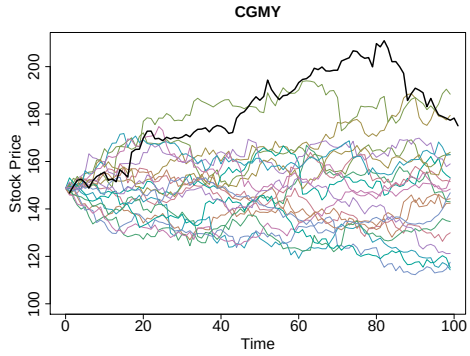
(c) 20 Monte Carlo simulated Variance Gamma price paths.



(d) 20 Monte Carlo simulated Generalized Hyperbolic price paths.



(e) 20 Monte Carlo simulated Hyperbolic price paths.



(f) 20 Monte Carlo simulated CGMY price paths.

Figure 4.7: Monte Carlo simulated price paths for Apple Inc. adjusted closing prices.

Chapter 5

Artificial Neural Networks

Artificial Neural Networks are loosely inspired by the biological neural network in the brain and how it processes information, thereby the name. McCulloch and Pitts [42] were the first who tried to represent the brain of a mammal through an artificial neural network represented by basic brain cells they called neurons.

Rosenblatt [48] a bit later introduced the perceptron built up of linear threshold units (LTUs). An LTU sums over weighted inputs and applies the step function which outputs 1 if the sum is larger than some threshold and 0 if it is below. An LTU is illustrated in figure 5.1. The perceptron is then constricted by an input layer, a layer of LTUs connected to all the inputs and produces an output vector containing only binary values. However, the perceptron was restricted. Later, the neocognitron was introduced by Fukushima [23], a hierarchical multilayer neural network - what motivated for further work on multilayer perceptrons.

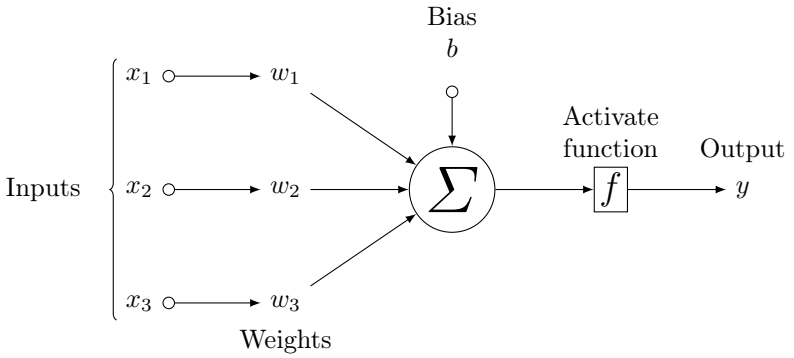


Figure 5.1: An LTU with input of dimension three.

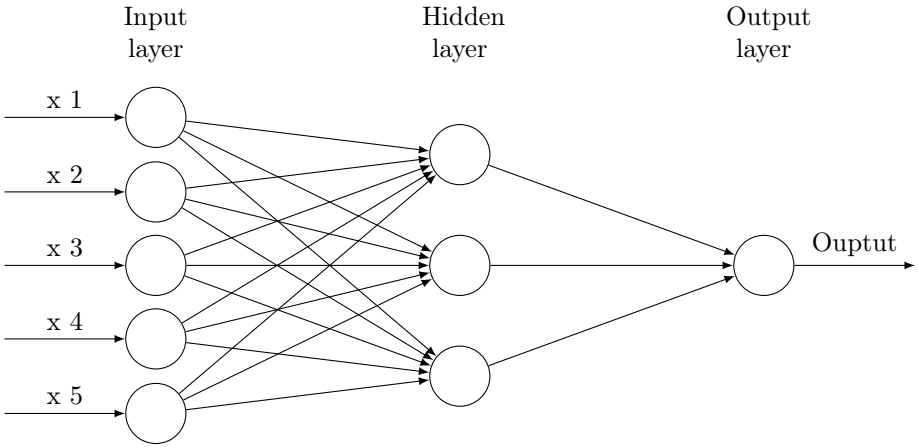


Figure 5.2: Illustration of a MLP with input dimension of five, a one-dimensional output and one hidden layer with three neurons.

5.1 Multilayer Perceptron

Multilayer perceptrons (MLPs) are networks consisting of multiple layers of neurons. The first layer is an input layer, the last layer is an output layer and the layers between are called hidden layers. The network is deep if it consists of two or more hidden layers.

MLPs are a class of feedforward neural networks. Feedforward neural networks are the most essential deep learning models. It is called feedforward because it is a directed acyclic graph built up of chain structures and the input $\boldsymbol{x}$ floats *forward* through the network to produce an output $\boldsymbol{y}$. The goal of the network is to approximate some function $\boldsymbol{y} = f^*(\boldsymbol{x})$ by defining a mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ and learning the parameters $\boldsymbol{\theta}$ that best fit f^*.

An MLP is constructed similar to a perceptron, consisting of an input layer with an input vector and a bias term usually initialized to zero. The input layer is connected to the first hidden layer such that every unit in the hidden layer is connected to every input unit. The difference from the perceptron is in the nonlinear *activation function*. Where the perceptron would use a binary step function, the MLP units can use a wide range of activation functions producing a real-valued output which it passes on to units in the next layer. The process continues like this until it reaches the output layer.

5.1.1 Activation Function

The reason for using activation functions is the nonlinear element, namely the possibility of solving nonlinear functions. If all the activation functions are linear, one simply has multiple linear regression. An activation function also gives a restriction on the outputs of the neurons. This can be useful if one wants to squeeze the output into a range. In addition it increases training stability. Different layers can have different activation functions. The choice of activation function is closely related to the choice of cost function[1].

In the 1980s, the sigmoid activation function was very popular as it performed well on small neural networks. One avoided the use of rectified linear units until as late as the early 2000s because of its undefined derivative. However, in practice, the nondifferentiability is not a problem as the optimizer does not actually find a local minimum, but just a very low value in addition to the fact that there is some numerical error which means that when the analytic gradient is zero, the estimated gradient will not be exactly zero. Jarrett *et al.* [32] observed that the use of rectified nonlinearity was the most important factor of improvement among several other factors that they examined. For this reason, ReLU and its variations such as Leaky ReLU, Parametric ReLU and ELU has had increased popularity as of recently.

In this thesis, ELU and the softplus function, given by the equations in table 5.1, has been used as activation functions. Both ELU and softplus are very similar to ReLU. Softplus only outputs positive values, but has a softer transition around $z = 0$. ELU outputs small values also for negative input values.

Table 5.1: Activation functions.

ELU	$f(z) = \begin{cases} \alpha(e^z - 1) & \text{if } z < 0 \\ x & \text{if } z \geq 0 \end{cases}$
Softplus	$f(z) = \ln(1 + e^z)$

5.1.2 Cost Function

The cost function is a performance measure for the neural network and what is actually optimized with respect to the parameters $\boldsymbol{\theta}$. Important properties of the cost function is to have large and predictable enough gradients. However, if activation functions saturates for some values, the cost function tends to saturate as well. Hence, the combination of activation functions and cost functions should be chosen with care.

Another problem that occurs in deep neural networks is that the cost functions

[1] See section 5.1.2.

one usually evaluates become non-convex, i.e. they are harder to optimize as we might not find a global minimum, but merely just a very low value. We therefore use iterative and stochastic gradient-based optimizers. They do not have any global convergence guarantee and may be sensitive to initial values. The weights are usually initialized to small random values and the bias to small positive values or even zero.

A common choice of cost function for regression tasks that is used in this thesis, is the root-mean-square error which is given by (5.1).

$$J(\theta) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(\boldsymbol{x}_i;\boldsymbol{\theta}) - y_i)^2} \tag{5.1}$$

5.1.3 Backpropagation

In order to minimize the cost function, the gradient of the model has to be calculated. This is done using the backpropagation algorithm. The backpropagation algorithm was popularized and first used on multilayer neural networks by Rumelhart *et al.* [49]. The backpropagation algorithm consists of two parts, the forward pass and the backward pass.

Forward Pass

In the forward pass, the training samples are propagated forward through the network, calculating the output of each neuron in a layer and passing forward to the next layers to produce the output.

Let $\boldsymbol{x} \in \mathbb{R}^d$, $y \in \mathbb{R}$, $b^{(l)} \in \mathbb{R}$ for $l = 1, \ldots, n$ and w_l be the activation function of layer l. In addition, let $W^{(1)} \in \mathbb{R}^{m_1 \times d}$, $W^{(l)} \in \mathbb{R}^{m_l \times m_{l-1}}$ for $l = 2, \ldots n-1$ and $W^{(n)} \in \mathbb{R}^{m_{n-1}}$ be the matrices such that element W^l_{ij} is the weight from node j in layer $(l-1)$ to node i in layer l. Then the forward pass is given by

$$\begin{aligned}
\boldsymbol{h}^{(1)} &= w_1(b^{(1)} + W^{(1)}\boldsymbol{x}) \\
\boldsymbol{h}^{(2)} &= w_2(b^{(2)} + W^{(2)}\boldsymbol{h}^{(1)}) \\
&\vdots \\
\boldsymbol{h}^{(n-1)} &= w_{n-1}(b^{(n-1)} + W^{(n-1)}\boldsymbol{h}^{(n-2)}) \\
y &= w_n(b^{(n)} + W^{n\top}\boldsymbol{h}^{(n-1)}).
\end{aligned}$$

The algorithm is presented in Algorithm 1.

Backward Pass

The goal of the backpropagation algorithm is to calculate the gradient of the cost function with respect to any weight or bias in the network,

$$\frac{\partial J}{\partial W_{ij}^{(l)}}, \quad \frac{\partial J}{\partial b^{(l)}}.$$

It does so by starting with the output layer. Let $\boldsymbol{\delta}^{(n)}$ be such that

$$\boldsymbol{\delta}^{(n)} = \nabla_h J \circ w'(\boldsymbol{z}^{(n)}), \tag{5.2}$$

where

$$\boldsymbol{z}^{(l)} = W^{(l)}\boldsymbol{h}^{(l-1)} + b^l$$

and ∇_h is the partial derivative with respect to each node in the output layer, $\partial/\partial h_j^n$. Next, $\boldsymbol{\delta}^{(l)}$ is calculated for layer l given $\boldsymbol{\delta}^{(l+1)}$,

$$\boldsymbol{\delta}^{(l)} = \left((W^{(l+1)})^T\boldsymbol{\delta}^{(l+1)}\right) \circ w'(\boldsymbol{z}^{(l)}). \tag{5.3}$$

We thus obtain the equations for the gradients,

$$\frac{\partial J}{\partial W_{ij}^{(l)}} = h_j^{(l-1)}\delta_i^{(l)} \tag{5.4}$$

$$\frac{\partial J}{\partial b^{(l)}} = \boldsymbol{\delta}^{(l)}. \tag{5.5}$$

The backward pass starts by calculating $\boldsymbol{\delta}^{(n)}$ for the output layer and iterates backwards through the layers using (5.3) and (5.2). Finally, the gradients are obtained by equation (5.4) and (5.5).

The algorithm is presented in Algorithm 2.

for $k = 1$ *to* m_1 **do**
  $z_k^{(1)} = \sum_{j=1}^{d} W_{jk}^{(1)} x_{ij}$
  $h_k^{(1)} = w_1(z_k^{(1)})$
end
for $l = 2$ *to* n **do**
  for $k = 1$ *to* m_l **do**
    $z_k^{(l)} = \sum_{j=1}^{m_{l-1}} W_{jk}^{(l)} h_j^{(l-1)}$
    $h_k^{(l)} = w_l(z_k^{(1)})$
  end
end

Algorithm 1: Forward Pass

for $j = 1$ *to* m_n **do**
  $\delta_j^{(n)} = \frac{\partial J}{\partial h_j^{(n)}} w'(z_j^n)$
  $\frac{\partial J_i}{\partial W_{jk}^{(n)}} = \delta_j^{(n)} h_k^{(n-1)}$
end
for $l = n - 1$ *to* 1 **do**
  for $j = 1$ *to* m_l **do**
    $\delta_j^{(l)} = w'(z_j^{(l)}) \sum_{k=1}^{m_{l+1}} \delta_k^{(l+1)} W_{jk}^{(l+1)}$
    $\frac{\partial J_i}{\partial W_{jk}^{(l)}} = \delta_j^{(l)} h_k^{(l-1)}$
  end
end

Algorithm 2: Backward Pass

5.1.4 Optimization

The parameters in the MLP are usually updated with some version of stochastic gradient descent. Stochastic gradient descent divides the training set into non-overlapping, roughly equal subsets called batches and computes an approximate gradient on the batches which it iterates over[2]. By doing so, it is possible to obtain an unbiased estimator of the gradient as we average over a batch of identically independently distributed samples.

Stochastic gradient descent relies on a hyperparameter - a parameter that is not decided by the algorithm, namely the learning rate with which each update is made. Typically, this parameter needs to decrease in each iteration because the gradient estimate includes some noise that does not vanish at the minimum. In this thesis,

[2]In contrast to a gradient descent method that computes the exact gradient using the whole training set in each iteration.

a version of stochastic gradient descent called Adam algorithm has been used as the optimization algorithm.

Adam

Adam, short for adaptive moments, is an algorithm with an adaptive learning rate presented by Kingma and Ba in 2014 [33]. It first computes a first- and second-order moment for which it secondly performs a bias correction to account for the initialization made at 0. The algorithm is presented in Algorithm 3.

Data: step size, ϵ (default: 0.001)
Data: exponential decay rates for momentum, ρ_1, $\rho_2 \in [0, 1)$ (default: 0.9, 0.999)
Data: small constant δ for numerical stabilization (default: 10^{-8})
initialize weights $\boldsymbol{\theta}$
initialize first and second momentum: $\mathbf{s} = 0$, $\mathbf{r} = 0$
$t \leftarrow 0$
while *stopping criterion not met* **do**
  sample minibatch of size m, input $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ with corresponding output $\{y_1, \ldots, y_m\}$
  compute gradient: $\mathbf{g} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} J(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i)$
  $t \leftarrow t + 1$
  update first biased momentum: $\mathbf{s} \leftarrow \rho_1 \mathbf{s} + (1 - \rho_1)\mathbf{g}$
  update second biased momentum: $\mathbf{r} \leftarrow \rho_2 \mathbf{r} + (1 - \rho_2)\mathbf{g} \circ \mathbf{g}$
  correct first biased momentum: $\hat{\mathbf{s}} \leftarrow \frac{\mathbf{s}}{1 - \rho_1^t}$
  correct second biased momentum: $\hat{\mathbf{r}} \leftarrow \frac{\mathbf{r}}{1 - \rho_2^t}$
  compute update: $\Delta\boldsymbol{\theta} = -\epsilon \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}}} + \delta}$
  update: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \Delta\boldsymbol{\theta}$
end

Algorithm 3: Adam

5.1.5 Generalization

The Universal Approximation Theorem first presented by Cybenko [19] showed that any function that is continuous on a closed and bounded set of $\mathbb{R}$ can be represented by a single hidden layer feedforward network with the sigmoid activation function and a linear output. Hornik *et al.* [30] then showed that this is true for a wide range of activation functions. However, we do not know the number of neurons needed and have no guarantee for the ability of our network to learn this representation.

Generalization is the ability the model has to predict unobserved data. Empirically, e.g. by Bengio *et al.*, [10], the more layers we add, the better the generalization of the neural network and the less neurons are needed. To observe how well

our model generalizes, it is common to divide the data into a training and a test set.

The training set is used to train the model and the test set is used to measure the error for unobserved data. If the training error is not sufficiently low, we have what is called underfitting, or high *bias*. If the training error is sufficiently low, but the gap between the training and test error is high, we have overfitting, or high *variance*. Overfitting occurs from sensitivity to small deviations in the data often causing the network to model noise. In deciding on a network architecture, we encounter the bias-variance trade-off problem. Increased number of hidden units will increase variance and decrease bias causing overfitting, while too few hidden units may lead to underfitting. Hence, we want the train and test error to have as small of a gap as possible and choose a model based on minimizing the test error as this is the best indicator of the ability our model has to generalize.

5.2 Model Setup

5.2.1 Data Calibration

Hutchinson *et al.* [31] made the first attempt to solve option pricing using an MLP on both Geometric Brownian Monte Carlo simulated underlying stock price for option prices as well as real S&P 500 futures and options using only two input parameters, namely time to maturity and the moneyness - the ratio between the underlying stock price and the strike price, at the start of the option. The output was the ratio between the option price, and the strike price.

Reducing the number of input parameters reduces the complexity of the model and necessity for hidden units. In addition, normalized input data to the same order of magnitude also reduces the complexity of the model as the weight updates are proportional to the input data. Hutchinson *et al.* [31] argued for the use of only two input parameters by Theorem 8.9 of Merton [44] where it is stated that the option pricing formula is homogeneous of degree one in stock price per share and strike price. Garcia and Gencay [24] also showed the use of this assumption in feed-forward neural networks greatly improves performance. The authors of Hutchinson *et al.* [31] also argued that with the assumption that the volatility and the risk-free rate is constant during the life of the option, this will not be picked up by the model. However, this is one of the limitations of the Black-Scholes formula, and providing the model with some measure of the latter two might be useful. For further work, it has been suggested in Fogarasi [22] to start out with a lower number of input variables which one increases to achieve more accuracy. This was done by e.g. Amilon [3], using a total of nine input variables, including 10- and 30-days historical volatilities as well as lagged prices of the underlying stock.

In this thesis, only one underlying has been used, for only one start date. This means that the volatility, interest rate and initial stock price are the same for all samples so these parameters were left out of the input. In addition, instead of using

the homogeneity principle as used by e.g. Hutchinson *et al.* [31] or Garcia [24] , the input and output data has been scaled between 0 and 1 by equation (5.6), if x_{ij} is an element of matrix X,

$$x_{ij}^{scaled} = \frac{x_{ij} - \min_j\{X\}}{\max_j\{X\} - \min_j\{X\}}. \tag{5.6}$$

Scaling the data is recommended because the data is now of the same order as the output of the activation functions which increases stability and reduces complexity. Unscaled input can cause slow and unstable learning while unscaled output can result in exploding gradients which might cause the learning process to fail.

Hence, the input consists of only the strike price K and the time to maturity, $T - t$, while the output is the quoted call option price, $\Pi_{t=T}^{Call}$. The structure of the input matrix, X, is of the following form $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^\top$ such that $\boldsymbol{x}_i = [K_i, T_i - t]$, and the corresponding output, $y_i = \Pi_{t=T}^{Call}$.

5.2.2 Network Architecture

In Hutchinson *et al.* [31], the authors experimented with three different neural networks - Radial Basis Functions(RBFs), Projection Pursuit Regression(PPR) and MLP, discovering that MLP performed best. In combination with it's straightforward configuration, this most likely motivated the further research on the use of MLP for option pricing.

Anders *et al.* [4], Gencay and Salih [25] and Yao *et al.* [57] mainly focused on the use of MLP, but used small and shallow networks. Hutchinson *et al.* [31] used a one hidden layer network with 4 hidden units and 2 inputs. Benell & Sutcliff [11] experimented with one hidden layer with 3-5 hidden units and 3-7 inputs. Anders *et al.* [4] experimented with a sparse network (networks where not all units are connected) with one hidden layer, 3 hidden units and 4 inputs. They all still achieved good performance, hence one might expect that a deeper and wider network would perform even better. This belief is also supported by studies on better generalization with deeper networks (Bengio *et al.* [10]). By today's computer power it is possible to get results for networks with several hundred hidden units and several hidden layers in reasonable time.

An MLP has been implemented using Keras [17][3]. The network architecture is presented in table 5.2.

The softplus activation function was chosen for the output layer as it maps from the whole real line to all nonnegative values which is consistent with option prices. For the hidden layers, the choice of activation function is harder because they are not directly connected to the input or output. ReLU has empirically proven to be a good learning objective [32], but has one major drawback. The fact that its

[3]See appendix E for code.

Table 5.2: Network architecture.

Layer	Neurons	(Trainable)Parameters	Activation Function
Input layer	2		
Hidden layer 1	100	300	ELU
Hidden layer 2	80	8080	ELU
Hidden layer 3	60	4860	ELU
Hidden layer 4	40	2440	ELU
Hidden layer 5	20	820	Softplus
Output layer	1	21	

Batch size	Epochs	Optimizer	Loss function	Train size	Test size
5	400	Adam	RMSE	135	34

gradient is zero for half of its domain means that it cannot learn when the output is zero. Several variations of ReLU has been made to guarantee that a gradient exists for all inputs. One of these is ELU, which was applied to all the hidden layers.

The batch size was set to 5 which is a quite small size, but is set with consideration of the small data set. In general, batch sizes are set in consideration to the robustness of the gradient of the loss function used in the optimizer. Lower batch sizes are more unstable while larger batch sizes give a more stable optimizer, but might not give as accurate estimates. In figure 6.1a and 6.1b, loss plotted for each iteration through the whole data set - called epoch - is plotted for a batch size of 5 and 20 showing more stability in the latter size. Smaller batch size can though provide a regularizing effect, but has a higher run time because it requires more iterations both due to the size and the need for a reduced learning rate, making the optimizer converge more slowly.

Adam was chosen as optimizer applied with the default values given in Algorithm 3.

Please note that the architecture is very problem dependent. There is a lot of ongoing research on the topic and there are no clear guidelines or theoretical results discovered yet. Hence, it was chosen based on previous work in combination with monitoring the performance on the out-of-sample data.

Chapter 6

Results and Analysis

6.1 Data Set

To analyze the results for the call option prices, option ask and bid prices on Apple Inc. stock has been downloaded. The option prices were downloaded using the *quantmod*-package in `R` [50] at 2019-05-31 with six different maturities and contains a total of 257 prices[1]. A description of the data set is listed in table 6.1. The risk free interest rate used was collected from The U.S. Department of Treasure [2] which at 2019-05-31 was at 2.35% per annum.

For the MLP, about 20 percent of the samples were held as an out-of-sample data set in which the MLP will not train on. In addition, it would be interesting to see how the MLP performs on a maturity it has not trained on. Hence, all samples for the longest maturity, 385 days, were held out in an unobserved maturity data set. The remaining samples were split by an 80-20 ratio of train and test set respectively, to monitor the train and test error.

[1] Its been proven difficult to get hold of historical option data which is the reason for the limited data set.

Table 6.1: Details of data set of call option prices.

Collected	Maturity	Length	Samples
2019-05-31	2019-06-21	21 days	49
	2019-07-09	49 days	45
	2019-09-20	112 days	26
	2019-10-18	140 days	41
	2020-01-17	231 days	51
	2020-06-19	385 days	45

6.2 Error Measures

For the error measures, we will follow Schoutens [52]. Let C^i_{obs} be observed market call option prices which is taken to be the mean of the bid and ask price, C^i_{mod} be model call option prices and $i = 1 \dots n_{obs}$ the number of prices. Then the average percentage error (APE), average absolute error (AAE), average relative percentage error (ARPE) and the root-mean-square error (RMSE) are defined by equation (6.1)-(6.4).

$$\text{APE} = \left(\frac{1}{n_{obs}} \sum_{i=1}^{n_{obs}} C^i_{obs}\right)^{-1} \sum_{i=1}^{n_{obs}} \frac{|C^i_{obs} - C^i_{mod}|}{n_{obs}} \tag{6.1}$$

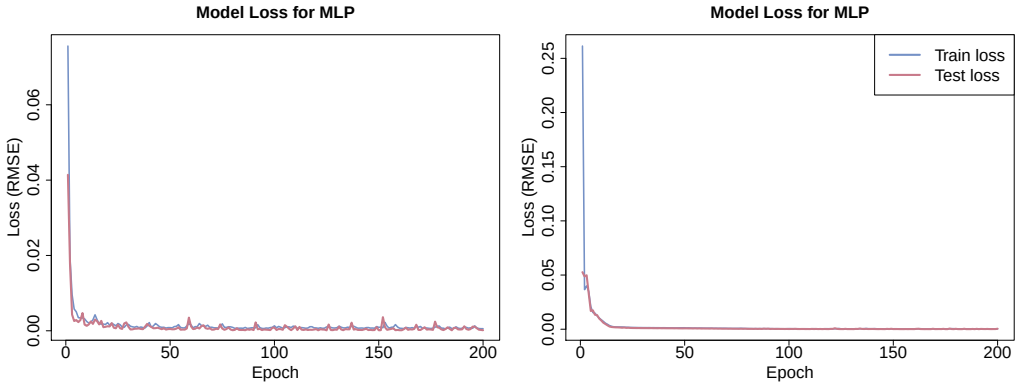
$$\text{AAE} = \sum_{i=1}^{n_{obs}} \frac{|C^i_{obs} - C^i_{mod}|}{n_{obs}} \tag{6.2}$$

$$\text{ARPE} = \frac{1}{n_{obs}} \sum_{i=1}^{n_{obs}} \frac{|C^i_{obs} - C^i_{mod}|}{C^i_{obs}} \tag{6.3}$$

$$\text{RMSE} = \sqrt{\sum_{i=1}^{n_{obs}} \frac{(C^i_{obs} - C^i_{mod})^2}{n_{obs}}} \tag{6.4}$$

Note that the ARPE is undefined if C^i_{obs} is equal to zero. For this reason, some values are missing for this error measure.

The rest of this chapter will be devoted to introduce the performance of the Lévy models and the MLP for all six maturities. These will be discussed, before performance for test, train and finally an overall performance will be evaluated.



(a) RMSE loss for the MLP for each epoch with a batch size equal to 5.

(b) RMSE loss for the MLP for each epoch with a batch size equal to 20.

Figure 6.1: RMSE loss for MLP per epoch for different batch sizes.

Table 6.2: Error measures for all maturities.

	GBM				NIG			
	APE	AAE	ARPE	RMSE	APE	AAE	ARPE	RMSE
21 days	0.05413	1.13644		1.72242	0.05361	1.12559		1.70212
49 days	0.05056	1.18348	0.31409	1.71839	0.05152	1.20593	0.30662	1.75095
112 days	0.05956	0.99744	0.21062	1.31150	0.06458	1.08164	0.19710	1.45406
140 days	0.13579	2.76518	0.28807	6.88841	0.13852	2.82076	0.28004	6.96239
231 days	0.028842	1.05685	0.32419	1.19381	0.02950	1.08085	0.32524	1.22096
385 days	0.05464	1.65754	0.45266	2.01547	0.05039	1.52858	0.44143	1.91921

	VG				GH			
	APE	AAE	ARPE	RMSE	APE	AAE	ARPE	RMSE
21 days	0.05508	1.15639		1.74241	0.05511	1.15708		1.74943
49 days	0.05202	1.21773	0.33177	1.76476	0.05229	1.22414	0.30656	1.77613
112 days	0.06225	1.04264	0.17480	1.41045	0.06185	1.03580	0.19640	1.38931
140 days	0.13725	2.79508	0.25008	6.93251	0.13731	2.79627	0.27041	6.93753
231 days	0.02810	1.02968	0.27245	1.16282	0.02920	1.06996	0.31247	1.20527
385 days	0.04650	1.41067	0.38842	1.74754	0.04941	1.49882	0.43878	1.88620

	H				CGMY			
	APE	AAE	ARPE	RMSE	APE	AAE	ARPE	RMSE
21 days	0.05501	1.15506		1.74161	0.04834	1.01495		1.55653
49 days	0.05318	1.24499	0.31179	1.80325	0.04058	0.95002	0.37343	1.36404
112 days	0.06751	1.13062	0.13865	1.56164	0.04844	0.81123	0.43752	0.90442
140 days	0.13927	2.83608	0.21424	6.99144	0.12779	2.60230	0.64283	6.44431
231 days	0.02793	1.02337	0.24895	1.16427	0.04680	1.71476	0.67083	2.01895
385 days	0.04198	1.27336	0.36103	1.62448	0.14792	4.48695	0.87417	4.76119

	MLP			
	APE	AAE	ARPE	RMSE
21 days	0.05855	1.22923		1.75278
49 days	0.05237	1.22589	6.80260	1.62358
112 days	0.07572	1.26822	0.35074	1.66524
140 days	0.12224	2.48928	0.25384	5.51703
231 days	0.03962	1.45192	0.20334	1.93980
385 days	0.03116	0.94509	0.26827	1.18331

Table 6.3: Error measures for train, test and complete data set.

	Train set			
	APE	AAE	ARPE	RMSE
GBM	0.08083	1.67372	0.31221	3.39913
NIG	0.07783	1.60692	0.31806	3.32019
VG	0.08239	1.70292	0.27841	3.45166
GH	0.07948	1.64453	0.30582	3.37316
H	0.08039	1.65933	0.26954	3.40250
CGMY	0.07133	1.48511	0.35982	3.17744
MLP	0.07885	1.70640	1.60315	3.13934
	Out-of-sample set			
	APE	AAE	ARPE	RMSE
GBM	0.05152	1.74190	0.20297	2.98121
NIG	0.04974	1.68558	0.20262	2.90835
VG	0.05283	1.78319	0.15595	3.03676
GH	0.05112	1.72989	0.18240	2.96885
H	0.05142	1.73520	0.13912	2.98403
CGMY	0.04790	1.65258	0.22446	2.89736
MLP	0.05522	1.90690	2.54708	2.81879
	Complete data set			
	APE	AAE	ARPE	RMSE
GBM	0.06588	1.53210	0.31793	2.62552
NIG	0.06690	1.54355	0.31026	2.66151
VG	0.06523	1.49916	0.28350	2.60361
GH	0.06601	1.52500	0.30492	2.63889
H	0.06597	1.50168	0.25493	2.62902
CGMY	0.08230	2.11305	0.59976	3.09858
MLP	0.05508	1.51946	1.47284	2.38048

6.3 Numerical Results

Tables 6.2 shows error measures for all the maturities, for the Lévy models simulated at a 1 day time step and 1×10^5 simulations[2]. Table 6.3 shows the error measures for the train set, the test set and the complete data set separately. Figure 6.2a-6.1g shows the true and predicted option prices for all the models, for all maturities. In figure 6.2a-6.1g, we can see the true and predicted option prices for all the models, for only the test set that was held out of training for the MLP, while in figure 6.2a-6.1g, we can see the true and predicted option prices for all the models, for the 385 days maturity that was held completely out of training for the MLP as well.

We may observe from table 6.2 that all the models have largest RMSE for the 140 days maturity. In figure 6.2a-6.1g, we may see that this part of the data set obviously has some outliers which probably explains the heavy mispricing of this maturity. The MLP does performs best for this maturity, however, the 140 days maturity is part of the training set. CGMY has the best error measures for the 21, 49 and 112 days maturities, but the worst error measures for the 385 and 231 days maturities. VG has best error measures for the 231 days maturity, while a rather intriguing result for the MLP, is the error on the 385 days maturity. Prices for this maturity was kept completely out of the train-test set, but the model still manages to get better error performance for this maturity than all the Lévy models. The GBM outperforms several of the models for 21, 49, 112 and 231 days maturities.

From table 6.3 it is clear that the MLP has the lowest RMSE for all parts of the data set[3]. That the training error is low is to be expected as the network is trained with respect to this data set. However, that the test error is low as well indicates that the network generalizes well to unobserved data which also is confirmed for the performance on the 385 days maturity.

For the overall performance of the Lévy models, CGMY has the worst error measures which is a result of the heavy mispricing for the two longest maturities. All the other Lévy models have very similar overall error measures, though NIG has slightly higher RMSE than the remaining models and VG has slightly lower RMSE than the remaining models. The GBM outperforms NIG, GH, H and CGMY. The relatively poor pricing of the CGMY does coincide with the Kolmogorov-Smirnov test observed in section 4.2. However, the better performance of GBM does not coincide with what was observed in section 4.2 where it came out as the worst fit.

From figure 6.2a-6.1g of unobserved maturity, the Lévy models seem to overprice the options with higher strikes, while the MLP seems to underprice the same options. The CGMY looks to do the the most severe overpricing. From figure 6.1g, the MLP underprices the lowest strike options and overprices the shortest maturity

[2]This was used as it was observed that the error did not improve with increased number of simulations. See table D.1 in appendix D.

[3]We may note that it has the largest ARPE for all parts of the data set. This seems odd as all models are tested on the same data set and might be a calculation.

options around strikes at 200.

6.4 Discussion

A remark regarding the results is the size of the data set which is quite small. Madan *et al.* [36] e.g. had 8245 option prices. Especially with respect to the neural network which has 16 521 trainable parameters and is a model which is usually trained on much larger data sets. This makes the results of the MLP quite unstable as there is a lot of stochastic behaviour, for example in the shuffling and splitting of data set, initializing of parameters and hyperparameters and in the stochastic gradient descent-type optimizing.

The small size of the data set could also be the reason why we do not get coherent results for the option pricing with the market fit analysis. With such few samples, it makes it hard to be able to say anything of statistical significance. We have also just considered options on one underlying asset. This was due to the difficulties of obtaining option price data. Comparing with other underlying assets or indices would be of interest to see how the models perform on underlying of different size and liquidity.

We have touched upon it briefly, but looking more into level of under- or overpricing for different maturities would also be interesting. However, with little data, I think it would be hard to say anything definite for this case.

As discussion in section 4.2, there was some optimizing problems with CGMY parameters. This could be a contributing factor to the poor pricing of the CGMY model. Using another optimizing algorithm or estimating the parameters in another way - e.g. maximum likelihood as the generalized hyperbolic class distributions were, could perhaps improve both the fit and the pricing.

It is also important to mention that the MLP has used quoted option prices directly to learn, while the Lévy models has been fitted indirectly, with respect to the underlying asset, in order to price. Had the parameters of the Lévy models been found through calibration with the market prices, as done in e.g. Schoutens [52], they might have performed better. It would be interesting to compare the pricing performance of the Lévy models through fit of underlying asset and calibrating with market option prices.

It is hard to do a deeper analysis of the MLP than comparing various error measures, as it cannot be analyzed to the same degree as the Lévy models[4]. This is generally a major drawback with deep neural network-type models, that they become a black box because of the limited knowledge we have of them.

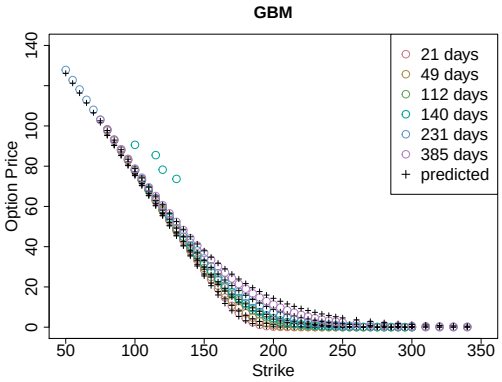
[4] Yet?

To comment on the computational speed and complexity of the models, it needs to be said that Monte Carlo simulation has been done for one underlying asset, and is not competitive with regards to speed and does not get competitive until the dimension is at least three. There are other ways to price options with Lévy dynamics such as Fast Fourier transforms of characteristic functions as done by e.g. Carr & Madan [16], or solving the partial differential integral equation which follows from the Feynman-Kac formula for Lévy processe.[5] Deep neural networks are usually time consuming to train as they have a lot of parameters and usually a lot of training data. However, when the training is done, the predictions can be quite fast.

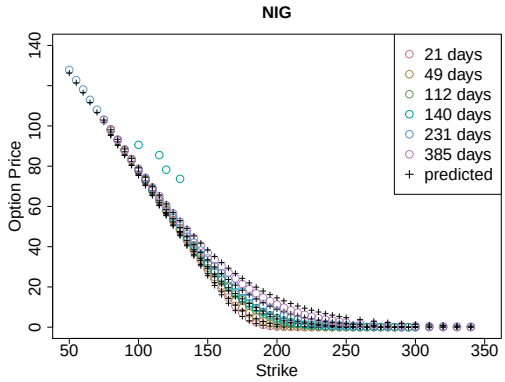
Madan *et al.* [36] found that option pricing errors for Black-Scholes and the symmetric VG were highly correlated with degree of moneyness at maturity, but not for the non-symmetric VG. It could be interesting to examine the correlation of the pricing error with the symmetry in the model. More generally, it would also be interesting to look at correlation in mispricing and moneyness as well.

When evaluating performance, it could be informative to look at the delta hedging performance as done by several others, e.g. Hutchinson *et al.* [31], Anders *et al.* [4]. Or evaluating statistical significance for the MLP e.g. by moving block bootstrappingas done by Amilon [3]. Several adjustments could have been tried for the MLP. For instance, there was no regularization applied such as early stopping, setting a dropout rate or L1/L2 regularization. One could also experiment more with the architecture, use of optimizer and default settings for hyperparameters, activation functions, batch size and number of epochs. Amilon [3] for instance, experimented with the input and output by adding lagged values of the stock price to the input and made it output the bid and ask price in order to compute the bid-ask spread.

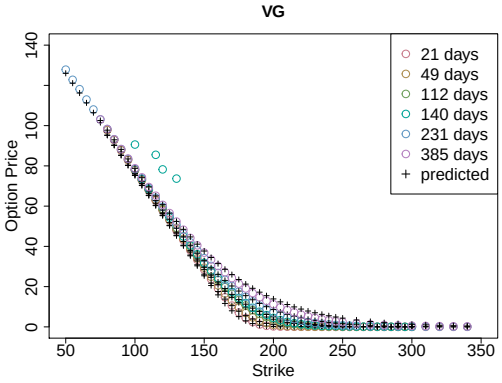
[5] See e.g. [45] for more on this.



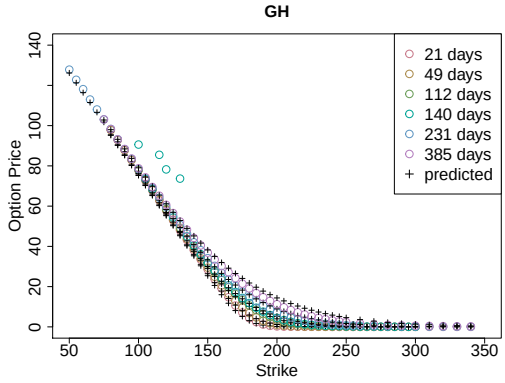
(a) True and predicted option prices for Geomtric Brownian motion.



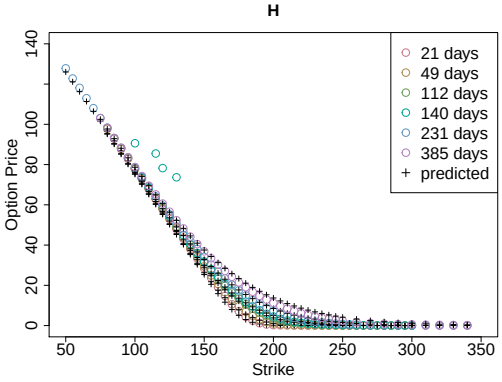
(b) True and predicted option prices for Normal Inverse Gaussian.



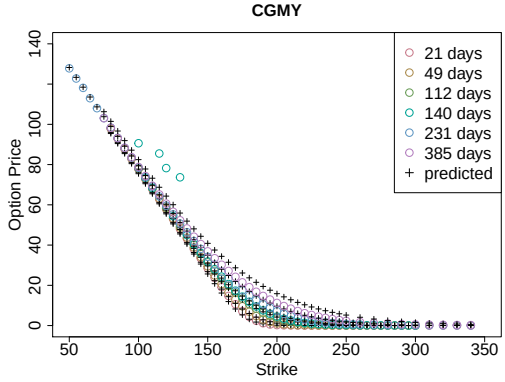
(c) True and predicted option prices for Variance Gamma.



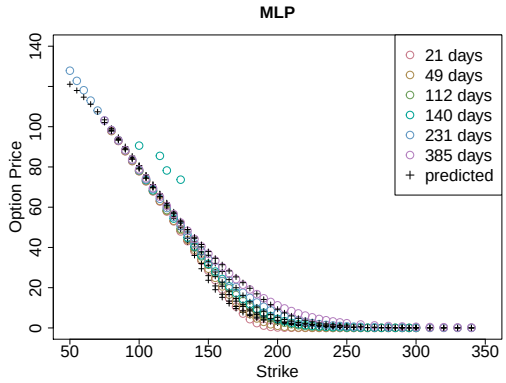
(d) True and predicted option prices for Generalized Hyperbolic.



(e) True and predicted option prices for Hyperbolic.

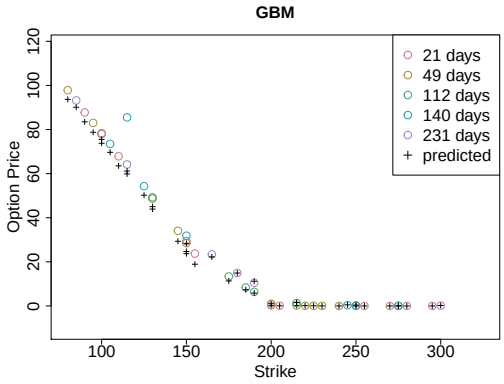


(f) True and predicted option prices for CGMY.

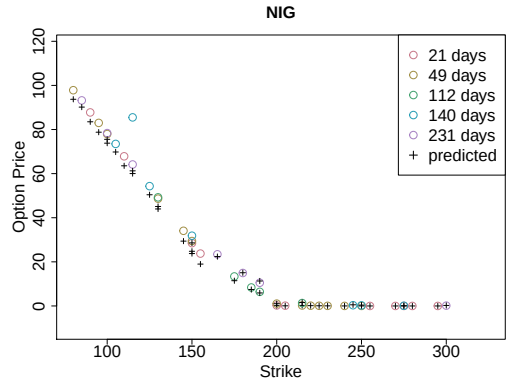


(g) True and predicted option prices for MLP.

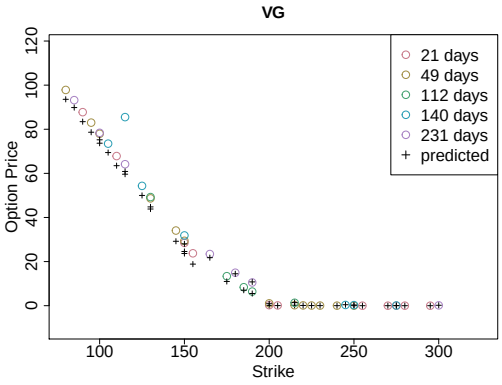
Figure 6.1: True and predicted option prices for full data set.



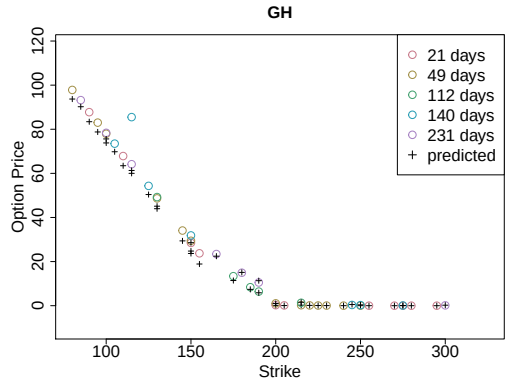
(a) True and predicted option prices for Geomtric Brownian motion.



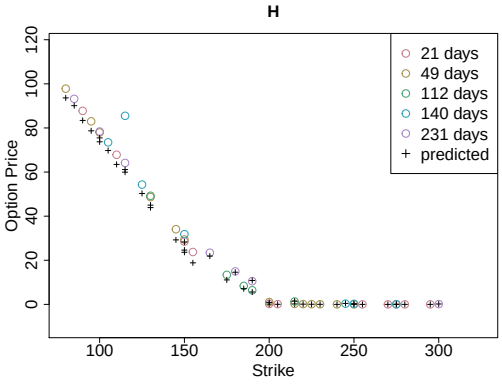
(b) True and predicted option prices for Normal Inverse Gaussian.



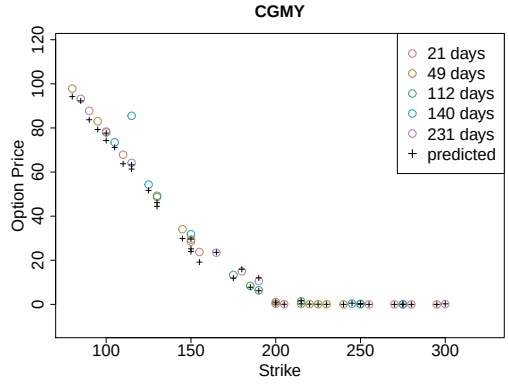
(c) True and predicted option prices for Variance Gamma.



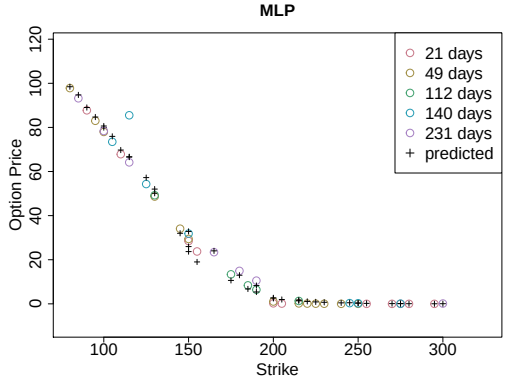
(d) True and predicted option prices for Generalized Hyperbolic.



(e) True and predicted option prices for Hyperbolic.

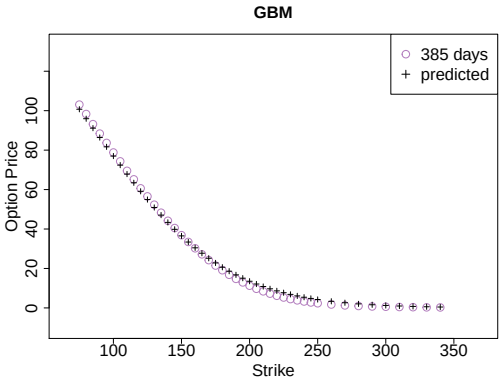


(f) True and predicted option prices for CGMY.

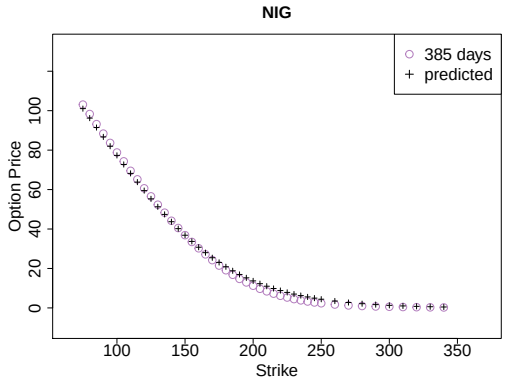


(g) True and predicted option prices for MLP.

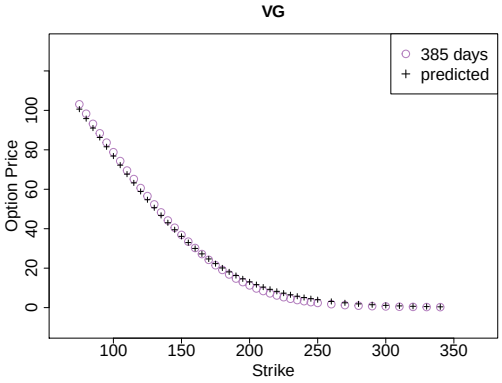
Figure 6.1: True and predicted option prices for out-of-sample data set.



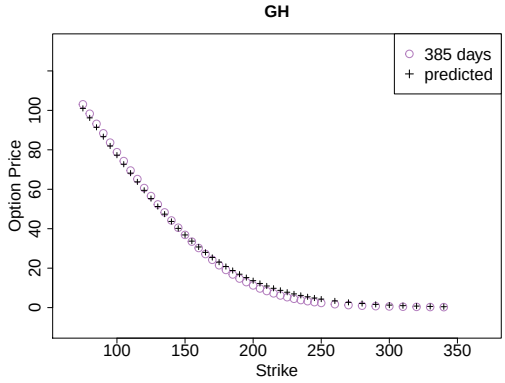
(a) True and predicted option prices for Geomtric Brownian motion.



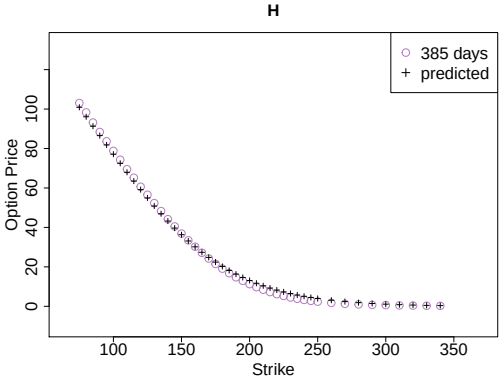
(b) True and predicted option prices for Normal Inverse Gaussian.



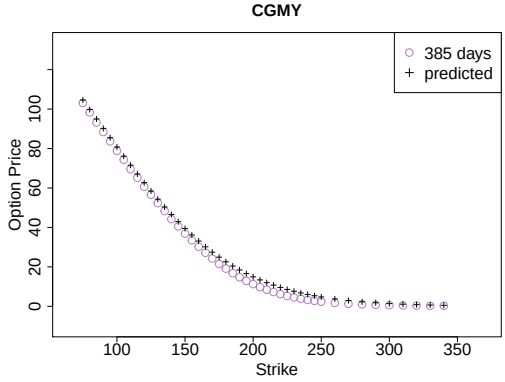
(c) True and predicted option prices for Variance Gamma.



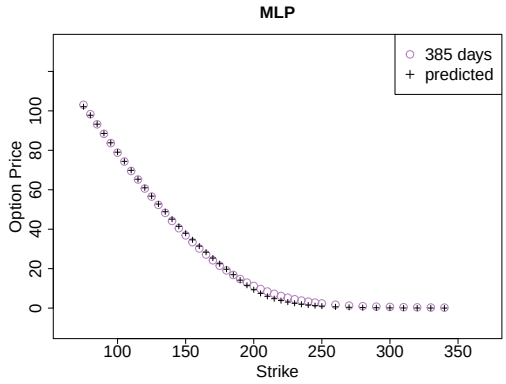
(d) True and predicted option prices for Generalized Hyperbolic.



(e) True and predicted option prices for Hyperbolic.



(f) True and predicted option prices for CGMY.



(g) True and predicted option prices for MLP.

Figure 6.1: True and predicted option prices for unobserved maturity data set.

Chapter 7

Concluding Remarks

In this thesis, five Lévy models and a multilayer perceptron has been implemented to compare pricing performance with the Geometric Brownian motion stock price dynamics of the Black-Scholes formula as the all the models has been proposed in literature to outperform Black-Scholes. In section 4.2, we saw that the Lévy models clearly were a better fit on the underlying assets than the Geometric Brownian motion based on estimates of moments, a goodness-of-fit test and observing the Value at Risk. However, we saw that this did not result in better option pricing performance. On the contrary, the Geometric Brownian motion performed better than several of the models, both for some of the maturities and for an overall performance measure.

The MLP had similar error measures to the other models, which (I) consider interesting because of the small amount of data it was able to train on. Another interesting result, was the ability it had to generalize to a unobserved maturity where it actually performed better than all the other models. However, this model is sensitive because of its many stochastic components, showing inconsistent results by repeating the experiment. This might also have been because of the amount of data.

Improvements and further work have been suggested based on the information at hand such as experimenting with network architecture as well as intput and output, examining correlation of pricing performance with moneyness as well as a sensitivity analysis and calibrating Lévy parameters to quoted option prices. Could faster pricing methods such as Fast Fourier transform be considered to compare with MLP? In the end, it is hard to conclude with anything definite because of the small data set.

Bibliography

[1] Charfun. https://github.com/cran/CharFun/blob/master/R/cf2DistGP.R.

[2] Daily Treasury Bill Rates Data. https://www.treasury.gov/resource-center/data-chart-center/interest-rates/Pages/TextView.aspx?data=billrates.

[3] Henrik Amilon. A neural network versus Black-Scholes: a comparison of pricing and hedging performances. *Journal of Forecasting*, 22, 2003.

[4] Ulrich Anders, Olaf Korn, and Christian Schmitt. Improving the pricing of options: a neural network approach. *ZEW Discussion Papers*, 1996.

[5] Søren Asmussen and Jan Rosiński. Approximations of Small Jumps of Lévy Processes with a View Towards Simulation. *Journal of Applied Probability*, 38(2):482–493, 2001.

[6] Louis Bachelier. théorie de la spéculation. *Annales Scientifiques de L'Ecole Normale Supérieure*, pages 21–88.

[7] Ole E. Barndorff-Nielsen. Exponentially decreasing distributions for the logarithm of particle size. *Scandinavian Journal of Statistics*, 353:401–419, 1977.

[8] Ole E. Barndorff-Nielsen. Normal Inverse Gaussian Distributions and the modelling of stock returns. *Scandinavian Journal of Statistics*, 1995.

[9] Ole E. Barndorff-Nielsen. Normal Inverse Gaussian Distributions and Stochastic Volatility Modelling. *Scandinavian Journal of Statistics*, 24(1):1–13, 1997.

[10] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-wise Training of Deep Networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, NIPS'06, pages 153–160, Cambridge, MA, USA, 2006. MIT Press.

[11] Julia Bennell and Charles Sutcliffe. Black-Scholes versus artificial neural networks in pricing FTSE 100 options. *Intelligent Systems in Accounting, Finance Management*, 12(4):243–260, 2004.

[12] Nicholas Bingham and Rüdiger Kiesel. Modelling asset returns with Hyperbolic distributions. *Return Distributions on finance*, 2:61–73, 2001.

[13] Fischer Black and Myron Scholes. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[14] Peter Carr, Dilip B. Mada, Marc Yor, and Helyette Geman. The Fine Structure of Asset Returns: An Empirical Investigation. *The Journal of Business*, 75(2):305–332, 2002.

[15] Peter Carr, Dilip B. Mada, Marc Yor, and Helyette Geman. Stochastic Volatility for Lévy Models. *Mathematical Finance*, 2003.

[16] Peter Carr and Dilip B. Madan. Option valuation using the fast Fourier transform. *Journal of Computational Finance*, 2:61–73, 1998.

[17] François Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[18] Rama Cont and Peter Tankov. *Financial Modelling With Jump Processes*. Chapman & Hall/CRC: London, 2004.

[19] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[20] Ernst Eberlein and Ulrich Keller. Hyperbolic Distributions in Finance. *Bernoulli*, 1(3):281–299, 1995.

[21] Ernst Eberlein and Karsten Prause. The Generalized Hyperbolic Model: Financial Derivatives and Risk Measures. 1998.

[22] Norbert Fogarasi. Option Pricing using Neural Networks. 2004.

[23] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

[24] René Garcia and Ramazan Gencay. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, 94(1-2):93–115, 2000.

[25] Ramazan Gencay and Aslihan Salih. Degree of Mispricing with the Black-Scholes Model and Nonparametric Cures. *Annals of Economics and Finance*, 4(1):73–101, May 2003.

[26] Igor Girsanov. On Transforming a Certain Class of Stochastic Processes by Absolutely Continuous Substitution of Measures. *Theory of Probability and Its Application*, 5(3):285–301, 1960.

[27] Paul Glasserman. *Monte Carlo methods in financial engineering*. Springer Verlag, 2004.

[28] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[29] Berend Hasselman. nleqslv: Solve Systems of Nonlinear Equations. `https://CRAN.R-project.org/package=nleqslv`.

[30] Kurt Hornik. Approximation Capabilities of Multilayer Feedforward Networks. *Neural Networks*, 4(2):251–257, 1991.

[31] James Hutchinson, Andrew Lo, and Tomaso Poggio. A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks. 1994.

[32] Kevin Jarrett, Koray Kavukcuoglu, and Yann Lecun. What is the Best Multi-Stage Architecture for Object Recognition?, 2009.

[33] Diederik P. Kingma and Jimmy Ba. adam: A method for stochastic optimization.

[34] Peter E. Kloeden and Eckhard Platen. *Numerical Solution of Stochastic Differential Equations*. Springer-Verlag, 1992.

[35] David Luethi and Wolfgang Breymann. ghyp: A Package on Generalized Hyperbolic Distribution and Its Special Cases. `https://CRAN.R-project.org/package=ghyp`.

[36] Dilip Madan, Peter Carr, Morgan Stanley, and Eric Chang. The Variance Gamma Process and Option Pricing. *European Finance Review*, 2:79–105, 1998.

[37] Dilip B. Madan, Robert H. Smith, and Marc Yor. Representing the CGMY and Meixner Levy Processes as Time Changed Brownian Motions. *The Journal of Computational Finance*, 12:27–47, 2008.

[38] Dilip B. Madan and Frank Milne. Option Pricing With V. G. Martingale Components. *Mathematical Finance*, 1:39–55, 1991.

[39] Dilip B. Madan and Eugene Seneta. Chebyshev polynomial approximations and characteristic function estimation. *Journal of the Royal Statistical Society*, 49:163–169, 1987.

[40] Dilip B. Madan and Eugene Seneta. The Variance Gamma (V.G.) Model for Share Market Returns. *The Journal of Business*, 63(4):511–524, 1990.

[41] Mary Malliaris and Linda Salchenberger. A neural network model for estimating option prices. *Appl. Intell.*, 3:193–206, 09 1993.

[42] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[43] Robert C Merton. *Theory of rational option pricing*, volume 4. 1973.

[44] Robert C. Merton. *Continuous-time finance*. Blackwell., 1991.

[45] David Nualart and Wim Schoutens. Backward stochastic differential equations and Feynman-Kac formula for Lévy processes, with applications in finance. *Bernoulli*, 7(5):761–776, 2001.

[46] Antonis Papapantoleon. An introduction to Lévy processes with applications in finance. *arXiv preprint arXiv:0804.0482*, 2008.

[47] Karsten Prause. The Generalized Hyperbolic Model: Estimation, Financial Derivatives, and Risk Measures. 1999.

[48] Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. *Psychological Review*, pages 65–386, 1958.

[49] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[50] Jeffery A. Ryan, Joshua M. Ulrich, Wouter Thielen, Paul Teetor, and Steve Bronder. quantmod: Quantitative Financial Modelling Framework. `https://CRAN.R-project.org/package=quantmod`.

[51] Ken-Iti Sato. *Lévy Processes and Infinitely Divisible Distributions*.

[52] Wim Schoutens. *Lévy Processes in Finance: Pricing Financial Derivatives*. Wiley Series in Probability and Statistics. Wiley, 2003.

[53] Julie Uv. A Comparison of Black Scholes, Monte Carlo Simulation and a Deep Neural Network for Option Pricing. *Unpublished project thesis at NTNU*, 2018.

[54] Alpha Vantage. ALPHA VANTAGE. `https://www.alphavantage.co/`.

[55] Paul Wilmott. *Paul Wilmott on quantitative finance*. John Wiley Sons, 2006.

[56] Paul Wilmott, Sam Howison, and Jeff Dewynne. *The mathematics of financial derivatives: a student introduction*. Cambridge University Press, 1995.

[57] Jingtao Yao, Yili Li, and Chew Lim Tan. Option price forecasting using neural networks. *Omega*, 28(4):455–466, 2000.

Appendix A

Gamma Process and Inverse Gaussian Process

Definition A.0.1 ***Gamma Process.*** *The Gamma distribution,* $\mathrm{G}(a,b)$*, can be expressed through its density function and characteristic function respectively,*

$$f_{\mathrm{G}}(x;a,b) = \frac{b^a}{\Gamma(a)} x^{a-1}\exp(-xb), \quad x>0 \tag{A.1}$$

$$\phi_{\mathrm{G}}(u;a,b) = (1-iu/b)^{-a} \tag{A.2}$$

for $a>0, b>0$*. The Gamma Process,* $X^{(\mathrm{G})} = (X_t^{(\mathrm{G})})_{t\geq 0}$ *then follows a* $\mathrm{G}(at,b)$ *distribution.*

Definition A.0.2 ***Inverse Gaussian Process.*** *The Inverse Gaussian distribution,* $\mathrm{G}(a,b)$*, can be expressed through its density function and characteristic function respectively,*

$$f_{\mathrm{IG}}(x;a,b) = \frac{a}{\sqrt{2\pi}}\exp(ab)x^{-3/2}\exp(-\frac{1}{2}(a^2x^{-1}+b^2x)), \quad x>0 \tag{A.3}$$

$$\phi_{\mathrm{IG}}(u;a,b) = \exp(-a(\sqrt{-2ui+b^2}-b) \tag{A.4}$$

for $a>0, b>0$*. The Gamma Process,* $X^{(\mathrm{IG})} = (X_t^{(\mathrm{IG})})_{t\geq 0}$ *then follows a* $\mathrm{IG}(at,b)$ *distribution.*

Appendix B

Bessel Functions

Bessel functions of the first and second kind, $J_{\pm v}(z)$ and $N_v(z)$, are solutions to

$$z^2\frac{\mathrm{d}^2w}{\mathrm{d}z^2} + z\frac{\mathrm{d}w}{\mathrm{d}z} + (z^2 - v^2)w = 0 \tag{B.1}$$

and satisfies

$$J_v(z) = (z/2)^v \sum_{k=0}^{\infty} \frac{(-z^2/4)^k}{k!\Gamma(v+k+1)} \tag{B.2}$$

$$N_v(z) = \frac{J_v(z)\cos(v\pi) - J_{-v}(z)}{\sin(v\pi)}. \tag{B.3}$$

The modified Bessel functions of the first and third kind, $I_{\pm v}(z)$ and $K_v(z)$, are solutions to

$$z^2\frac{\mathrm{d}^2w}{\mathrm{d}z^2} + z\frac{\mathrm{d}w}{\mathrm{d}z} - (z^2 + v^2)w = 0 \tag{B.4}$$

and satisfies

$$I_v(z) = (z/2)^v \sum_{k=0}^{\infty} \frac{(z^2/4)^k}{k!\Gamma(v+k+1)} \tag{B.5}$$

$$K_v(z) = \frac{\pi}{2}\frac{I_v(z) - I_{-v}(z)}{\sin(v\pi)}. \tag{B.6}$$

Appendix C

Itô Calculus

Definition C.0.1 ***Random Variable.*** *A random variable, X, is measurable function mapping from a set of possible outcomes, Ω to a space E, $X : \Omega \to E$ such that*

$$P(X \in S) = P\{\omega \in \Omega | X(\omega) \in S\} \tag{C.1}$$

for $S \subseteq E$.

Definition C.0.2 ***Stochastic Process.*** *A real valued stochastic process, $X = (X_t)_{t \geq 0}$, is indexed by a set T such that X_t is a random variable for each $t \in T$.*

Itô calculus gives meaning to the integral,

$$\int_0^t X_u \mathrm{d}Y_u$$

for suitable stochastic processes $(X_u)_{u \geq 0}$ and $(Y_u)_{u \geq 0}$. Stochastic differential equations of the form

$$\mathrm{d}X_t = a(t, X_t)\mathrm{d}t + b(t, X_t)\mathrm{d}Y_t, \quad X_0 = x_0 \tag{C.2}$$

then satisfies

$$X_t = \int_0^t a(u, X_u)\mathrm{d}u + \int_0^t b(u, X_u)\mathrm{d}Y_u, \quad X_0 = x_0$$

Lemma C.0.1 ***Itô's Lemma.*** *For a twice differentiable function $f(X)$, and a stochastic differential equation,*

$$\mathrm{d}X = a(t, X)\mathrm{d}t + b(t, X)\mathrm{d}W_t, \tag{C.3}$$

where $\mathrm{d}W_t$ is a Brownian motion, and $\mathrm{d}W_t^2 \to \mathrm{d}t$ as $\mathrm{d}t \to 0$ with probability 1, Ito's Lemma states that

$$\mathrm{d}f = b\frac{\mathrm{d}f}{\mathrm{d}X}\mathrm{d}W + (a\frac{\mathrm{d}f}{\mathrm{d}X} + \frac{1}{2}b^2\frac{\mathrm{d}f^2}{\mathrm{d}X^2}). \tag{C.4}$$

Appendix D

Table of Error Measures

Table D.1: Error measures for increasing number of simulations.

	GBM				NIG			
	APE	AAE	ARPE	RMSE	APE	AAE	ARPE	RMSE
1×10^3	0.07564815	1.781361	0.3862617	2.914699	0.07010742	1.597691	0.2463409	2.730876
1×10^4	0.06411879	1.4944	0.3141559	2.585825	0.07087047	1.640161	0.3152808	2.788231
1×10^5	0.06638995	1.531179	0.311038	2.641402	0.06757532	1.560437	0.3026539	2.685489
1×10^6	0.06639483	1.536598	0.31663	2.642696	0.06665405	1.542542	0.312545	2.654586

	VG				GH			
	APE	AAE	ARPE	RMSE	APE	AAE	ARPE	RMSE
1×10^3	0.08178	1.79601	0.25304	2.99955	0.06978	1.56927	0.31734	2.72024
1×10^4	0.06434	1.47221	0.29422	2.54851	0.06389	1.48032	0.31494	2.57406
1×10^5	0.06635	1.51063	0.25481	2.64177	0.06652	1.53946	0.30348	2.64492
1×10^6	0.06534	1.48948	0.26240	2.61143	0.06634	1.53367	0.30533	2.64471

	H				CGMY			
	APE	AAE	ARPE	RMSE	APE	AAE	ARPE	RMSE
1×10^3	0.05705	1.29731	0.30154	2.32971	0.06666	1.61691	0.38732	2.65634
1×10^4	0.06161	1.41641	0.30298	2.50675	0.06913	1.69686	0.37480	2.73312
1×10^5	0.06390	1.44976	0.24078	2.56836	0.06791	1.66924	0.35593	2.69942
1×10^6	0.06483	1.47548	0.25354	2.59345	0.06903	1.704817	0.35707	2.73199

Appendix E

Code

E.1 CGMY

```
fit.CGMY = function(data, initial_values=c(5, 10, 10, 0.1, 0)) {
#Description: Fits parameters of the CGMY model to data
#Input:
#     data: log returns to be fitted
#     initial values: initial values for nleqslv
#Output:
#     List of parameters C, G, M, Y, mu
  moments <- MASS::fitdistr(data, densfun='normal')
  mean = moments$estimate[1]
  variance = (moments$estimate[2])^2
  skewness = moments::skewness(data)
  kurtosis = moments::kurtosis(data)
  hyperskewness = moments::moment((data-mean)/sqrt(variance),order=5)
    #fifth order central moment, needed for the fifth equation
  equations <- function(x) {
    #C = x[1], G = x[2], M = x[3], Y = x[4], mu = x[5](drift)
    y <- numeric(5)
    y[1] <- x[1]*(x[3]^(x[4]-1)-x[2]^(x[4]-1))*gamma(1-x[4]) + x[5] -
    mean
    y[2] <- x[1]*(x[3]^(x[4]-2)+x[2]^(x[4]-2))*gamma(2-x[4]) -
    variance
    y[3] <- (x[1]*(x[3]^(x[4]-3)-x[2]^(x[4]-3))*gamma(3-x[4]))/((x[1]*
    (x[3]^(x[4]-2)+x[2]^(x[4]-2))*gamma(2-x[4]))^(3/2)) - skewness
    y[4] <- 3 + (x[1]*(x[3]^(x[4]-4)+x[2]^(x[4]-4))*gamma(4-x[4]))/((x
    [1]*(x[3]^(x[4]-2)+x[2]^(x[4]-2))*gamma(2-x[4]))^2) - kurtosis
    y[5] <- (x[1]*(x[3]^(x[4]-5)-x[2]^(x[4]-5))*gamma(5-x[4]))/(x[1]*(
    x[3]^(x[4]-2)+x[2]^(x[4]-2))*gamma(2-x[4]))^(5/2) + 10*(x[1]*(x
    [3]^(x[4]-3)-x[2]^(x[4]-3))*gamma(3-x[4]))/(x[1]*(x[3]^(x[4]-2)+x
    [2]^(x[4]-2))*gamma(2-x[4]))^(3/2) - hyperskewness
    y
  }
  params <- nleqslv::nleqslv(x = initial_values, fn = equations,
    method='Broyden', control=list(maxit=10000000, allowSingular=TRUE)
    )
  if (params$x[4] < 0 || params$x[4] > 2) {print('Y out of range')}
```

```
  return(list('C' = params$x[1], 'G' = params$x[2], 'M' = params$x[3],
    'Y' = params$x[4], 'mu' = params$x[5]))
}

CGMYcf =  function(u, t, params) {
#Description: Calculates characteristic function of the CGMY model
#Input:
#     u: vector for which characteristic function is to be calculated
#     t: time increment of process
#     params: list of parameters
#Output:
#     Vector of values of the characteristic function calculated at u
  C = params$C
  G = params$G
  M = params$M
  Y = params$Y
  mu = params$mu
  return(exp(fAsianOptions::cgamma(-Y)*((M-1i*u)^Y - M^Y + (G+1i*u)^Y
    - G^Y)*C*t + 1i*u*mu*t))
}

rCGMY = function(n, params, dt, res=0.0001) {
#Description: Generates random CGMY variates
#Input:
#     n: number of random variates
#     params: list of parameters
#     dt: time increment of process
#     res: resolution of vectorized density function to use for
    sampling
#Output:
#     n random CGMY variates
  cf <- function(t)
    CGMYcf(t, dt, params)
  result <- cf2DistGP(cf, option=list(isPlot=FALSE))
  x = seq(result$xMin, result$xMax, by=res)
  cf <- function(t)
    CGMYcf(t, dt, params)
  result <- cf2DistGP(cf, x, option=list(isPlot=FALSE))
  return(sample(result$x, size=n, replace=TRUE, prob=result$pdf))
}

dCGMY = function(params, dt, res=0.0001) {
#Description: Calculates probability density function of CGMY
    distribution
#Input:
#     params: list of parameters
#     dt: time increment of process
#     res: resolution of vectorized density function to use for
    sampling
#Output:
#     a list with probability density function and x-values, list('pdf
    ', 'x')
  cf <- function(t)
    CGMYcf(t, dt, params)
  result <- cf2DistGP(cf, option=list(isPlot=FALSE))
  x = seq(result$xMin, result$xMax, by=res)
  cf <- function(t)
```

```
    CGMYcf(t, dt, params)
  result <- cf2DistGP(cf, x, option=list(isPlot=FALSE))
}

pCGMY = function(x, params, res=0.0001) {
#Description: Calculates cumulative distribution function of CGMY
    distribution
#Input:
#     params: list of parameters
#     dt: time increment of process
#     res: resolution of vectorized density function to use for
    sampling
#Output:
#     a list with  cumulative distribution function and x-values, list
    ('cdf', 'x')
  dt  = 1
  cf <- function(t)
    CGMYcf(t, dt, params)
  result <- cf2DistGP(cf, option=list(isPlot=FALSE))
  x = seq(result$xMin, result$xMax, by=res)
  cf <- function(t)
    CGMYcf(t, dt, params)
  result <- cf2DistGP(cf, x, option=list(isPlot=FALSE))
  return(list(cdf = cumsum(result$pdf)/cumsum(result$pdf)[length(
    cumsum(result$pdf))], x =result$x))
}

qCGMY <- function(p, params) {
#Description: Calculates quantiles of CGMY distribution
#Input:
#     p: vector of probabilities for which to calculate quantiles
#     params: list of parameters
#Output:
#     a vector of quantiles calculated for p
  x = numeric(length(p))
  index=1
  rv = rCGMY(n=100000, params, dt=1, res=0.000001)
  #if (is.unsorted(rv)) {rv <- sort(rv)}
  rv <- sort(rv)
  n <- length(rv)
  for (prob in p) {
    x[index] = approx(seq(0, 1, length = n)[seq(1,100000, by=10)], rv[
    seq(1,100000, by=10)], prob, n=100000)$y
    index=index+1
  }
  return(x)
}
```

Listing E.1: Functions of different features for the CGMY model. Written in R.

E.2 Lévy Models

```
fitDistribution <- function(data, distribution, global, isPlot=FALSE,
    ticker) {
#Description: Fits parameters to all distributions
#Input:
#     data:  log returns to be fitted
```

```
#     distribution: list of string with distributions, list('GBM', 'NIG', 'VG', 'GH', 'H', 'CGMY')
#     global: list of global parameters, list('r_intraday', 'q')
#     isPlot: boolean, to plot or not
#     ticker: ticker of company
#Output:
#     list of list of parameters for each distribution in distribution
  params = list()
  r = global$r_intraday
  q = global$q
  if (isPlot) {
    cols = colorspace::qualitative_hcl(6, 'Dark2')
    hist(data, col='grey95', border='gray', breaks=100, freq=FALSE, xlab = 'Log-returns',
         main=paste('Density fits for', ticker),
         cex.lab=2, cex.axis=2, cex.main=2,
         xlim=c(-0.08, 0.08))
    par(lwd=3)
    col=vector()}
  for (dist in distribution) {
    if (dist == 'GBM') {
      fitGBM <- MASS::fitdistr(data, densfun='normal')
      paramsGBM = list('mu'=fitGBM$estimate[1], 'sigma'=fitGBM$estimate[2])
      params = append(params, list('GBM' = paramsGBM))
      if (isPlot) {
        lines(seq(min(data), max(data), by=0.001),
              dnorm(seq(min(data), max(data), by=0.001), mean=paramsGBM$mu, sd=paramsGBM$sigma),
              type='l', col=cols[1])
        col=append(col,1)}
    }
    if (dist == 'NIG') {
      fitNIG <- ghyp::fit.NIGuv(data)
      paramsNIG = ghyp::coef(fitNIG, type='alpha.delta')
      params = append(params, list('NIG' = paramsNIG))
      if (isPlot) {
        lines(fitNIG, type='l', col=cols[2])
        col=append(col,2)}
    }
    if (dist == 'VG') {
      fitVG <- ghyp::fit.VGuv(data)
      paramsVG = ghyp::coef(fitVG, type = "chi.psi")
      params = append(params, list('VG' = paramsVG))
      if (isPlot) {
        lines(fitVG, type='l', col=cols[3])
        col=append(col,3)}
    }
    if (dist == 'GH') {
      fitGH <- ghyp::fit.ghypuv(data)
      paramsGH = ghyp::coef(fitGH, type='alpha.delta')
      params=append(params, list('GH' = paramsGH))
      if (isPlot) {
        lines(fitGH, type='l', col=cols[4])
        col=append(col,4)}
    }
    if (dist == 'H') {
```

```
58        fitH <- ghyp::fit.hypuv(data)
59        paramsH = ghyp::coef(fitH, type='alpha.delta')
60        params = append(params, list('H' = paramsH))
61        if (isPlot) {
62          lines(fitH, type='l', col=cols[5])
63          col=append(col,5)}
64      }
65      if (dist == 'CGMY') {
66        paramsCGMY <- fit.CGMY(data)
67        params = append(params, list('CGMY' = paramsCGMY))
68        if (isPlot) {
69          lines(dCGMY(paramsCGMY, 1)$x, dCGMY(paramsCGMY, 1)$pdf, type='
      l',
70                col=cols[6])
71          col=append(col,6)}
72      }
73    }
74    if (isPlot) {
75      par(lwd=1)
76      legend('topleft', c('DATA', distribution),
77             fill=c('gray', cols[col]), bty = 'n', cex=2)
78    }
79    return(params)
80  }
```

Listing E.2: Fit parameters to data for Lévy models. Written in R.

```
1  MCSimulation = function(params, distributions, global) {
2  #Description: MC Simulation of stock price paths
3  #Input:
4  #      params: list of list of parameters for each distribution in
       distribution
5  #      distribution: list of string with distributions, list('GBM', '
       NIG', 'VG', 'GH', 'H', 'CGMY')
6  #      global: list of global parameters, list('r_intraday', 'q', 'dt',
        's0', 'n', 'nsim')
7  #Output:
8  #      list of list of simualtions distribution in distribution
9    simulations = list()
10   r = global$r_intraday
11   q = global$q
12   s0 = global$s0
13   dt = global$dt
14   n = global$n
15   nsim = global$nsim
16   for (dist in distributions) {
17     if (dist == 'GBM') {
18       mu = params$GBM$mu
19       sigma = params$GBM$sigma
20       mu = r-q-0.5*sigma_GBM^2 # Mean-corrected drift
21       dx = matrix(rnorm(n=n*nsim, mean=mu*dt, sd=sqrt(dt)*sigma), nrow
       =nsim, ncol=n)
22       sGBM = matrix(s0, nrow=nsim, ncol=n)
23       for (i in 1:nsim) {
24         for (t in 2:n) {
25           sGBM[i, t] = sGBM[i, t-1]*exp(dx[i, t])
26         }
```

```
27        }
28        simulations = append(simulations, list('GBM' = sGBM))
29      }
30      if (dist == 'NIG') {
31        alpha = params$NIG$alpha
32        beta = params$NIG$beta
33        delta = params$NIG$delta
34        omega = delta_NIG*(sqrt(alpha_NIG^2-(beta_NIG+1)^2)-sqrt(alpha_
      NIG^2-beta_NIG^2))
35        mu = r - q + omega_NIG # Mean-corrected drift
36        dx = matrix(ghyp::rghyp(n=n*nsim, object=ghyp::NIG.ad(alpha=
      alpha, delta=delta*dt, beta=beta, mu=mu*dt)), nrow=nsim, ncol=n)
37        sNIG = matrix(s0, nrow=nsim, ncol=n)
38        for (i in 1:nsim) {
39          for (t in 2:n) {
40            sNIG[i, t] = sNIG[i, t-1]*exp(dx[i, t])
41          }
42        }
43        simulations = append(simulations, list('NIG' = sNIG))
44      }
45      if (dist == 'VG') {
46        lambda = params$VG$lambda
47        mu = params$VG$mu
48        sigma = params$VG$sigma
49        gamma = params$VG$gamma
50        omega = lambda*log( 1 - sigma^2/(2*lambda) - gamma/lambda )
51        mu = r - q + omega # Mean corrected drift
52        VGobject = ghyp::VG(lambda = lambda*dt, mu = dt*mu, sigma =
      sigma*sqrt(dt), gamma = dt*gamma)
53        dx = matrix(ghyp::rghyp(n=n*nsim,object=VGobject), nrow = nsim,
      ncol = n )
54        sVG = matrix(s0, nrow=nsim, ncol=n)
55        for (i in 1:nsim) {
56          for (t in 2:n) {
57            sVG[i, t] = sVG[i, t-1]*exp(dx[]i, t])
58          }
59        }
60        simulations = append(simulations, list('VG' = sVG))
61      }
62      if (dist == 'GH') {
63        if (dt!=1) {
64          print('dt must be 1')
65          dt = 1}
66        alpha = params$GH$alpha
67        beta = params$GH$beta
68        delta = params$GH$delta
69        lambda = params$GH$lambda
70        omega = log((((alpha^2-beta^2)/(alpha^2-(beta+1)^2))^(lambda/2))
      *(besselK(delta*sqrt(alpha^2-(beta+1)^2), lambda)/besselK(delta*
      sqrt(alpha^2-beta^2), lambda)))
71        mu = r - q - omega # Mean corrected drift
72        dx = matrix(ghyp::rghyp(n=n*nsim, object=ghyp::ghyp.ad(lambda=
      lambda*dt, alpha=alpha, delta=delta*dt, beta=beta, mu=mu*dt)),
      nrow=nsim, ncol=n)
73        sGH = matrix(s0, nrow=nsim, ncol=n)
74        for (i in 1:nsim) {
75          for (t in 2:n) {
```

```
76            sGH[i, t] = sGH[i, t-1]*exp(dx[i, t])
77          }
78        }
79        simulations = append(simulations, list('GH' = sGH))
80      }
81      if (dist == 'H') {
82        alpha = params$H$alpha
83        beta = params$H$beta
84        delta = params$H$delta
85        omega = log((sqrt((alpha^2-beta^2)/(alpha^2-(beta+1)^2)))*(
      besselK(delta*sqrt(alpha^2-(beta+1)^2), 1)/besselK(delta*sqrt(
      alpha^2-beta^2), 1)))
86        mu = r - q - omega # Mean corrected drift
87        dx = matrix(ghyp::rghyp(n=n*nsim, object=ghyp::hyp.ad(alpha=
      alpha, delta=delta*dt, beta=beta, mu=mu*dt)), nrow=nsim, ncol=n)
88        sH = matrix(s0, nrow=nsim, ncol=n)
89        for (i in 1:nsim) {
90          for (t in 2:n) {
91            sH[i, t] = sH[i, t-1]*exp(dx[i, t])
92          }
93        }
94        simulations = append(simulations, list('H' = sH))
95      }
96      if (dist == 'CGMY') {
97        C = params$CGMY$C
98        G = params$CGMY$G
99        M = params$CGMY$M
100       Y = params$CGMY$Y
101       omega = -C*gamma(-Y)*((M-1)^Y - M^Y + (G+1)^Y - G^Y)
102       mu = r - q + omega # Mean corrected drift
103       dx = matrix(rCGMY(n=n*nsim, params=list('C'=C, 'G'=G, 'M'=M, 'Y'
      =Y, 'mu'=mu), dt), nrow=nsim, ncol=n)
104       sCGMY = matrix(s0, nrow=nsim, ncol=n)
105       for (i in  1:nsim) {
106         for (t in 2:n) {
107           sCGMY[i, t] = sCGMY[i, t-1]*exp(dx[i, t])
108         }
109       }
110       simulations = append(simulations, list('CGMY' = sCGMY))
111     }
112   }
113   return(simulations)
114 }
115
116 optionPrice = function(sT, r, q, time_to_maturity, strike) {
117 #Description: estimate for option price based on MC simulation
118 #Input:
119 #     sT: vector of stock prices at maturity
120 #     r: risk free rate
121 #     q: dividend rate
122 #     time_to_maturity: maturity of option
123 #     strike: strike price of option
124 #Output:
125 #     estimated ccall option price
126   return(exp(-(r-q)*time_to_maturity)*mean(pmax(sT-strike, 0)))
127 }
```

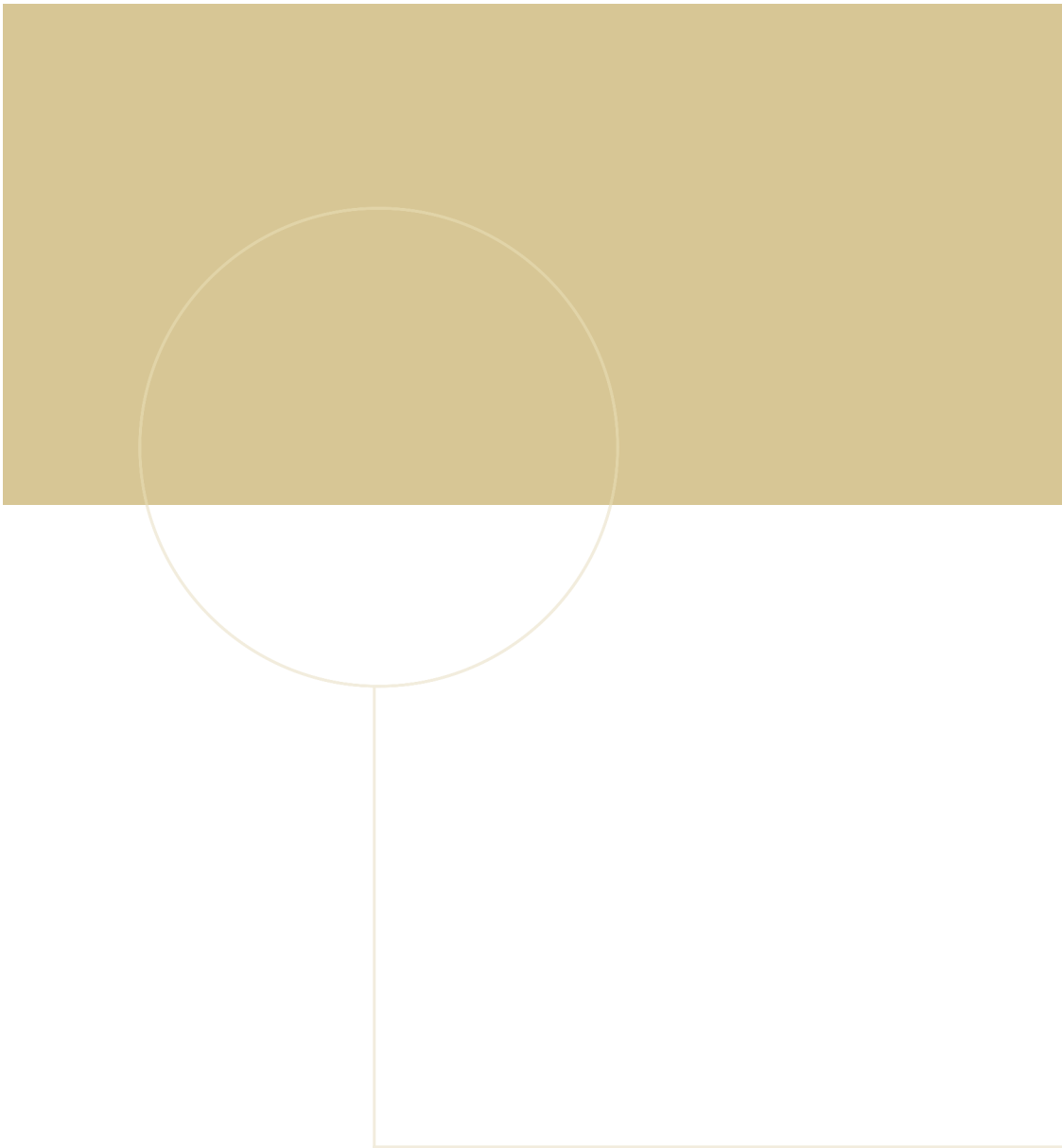
Listing E.3: Monte Carlo simulation and pricing. Written in R.

E.3 Neural Network

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras import optimizers
from keras import regularizers
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.metrics import mean_squared_error, mean_absolute_error
from sklearn import preprocessing
from sklearn.preprocessing import MinMaxScaler

# Preporcessing of data: Remove unobserved maturity, then divie into train and test set.
# Scale data between 0 and 1
# Convert to supervised learning problem: Use Strike price and expiry as input and mean of ask and bid price as output
df = pd.read_csv('AAPLoptionprices_all.csv')
df = df[['Strike', 'Bid', 'Ask', 'Exp', 's0']]
X = df[['Strike', 'Exp']].get_values()
y = 0.5*(df[['Ask']].get_values() + df[['Bid']].get_values())
data = np.column_stack((X,  y))
scaler = MinMaxScaler(feature_range=(0, 1))
data_scaled = scaler.fit_transform(data)
data_df = pd.DataFrame(data_scaled, columns = ['Strike', 'Exp', 'Price'])
exp = max(data_df['Exp'])
unobserved_df = pd.DataFrame(columns= ['Strike', 'Exp', 'Price'])
unobserved_df = unobserved_df.append(data_df.loc[data_df.loc[:,'Exp']==exp,['Strike', 'Exp', 'Price']])
unobserved_data = scaler.inverse_transform(unobserved_df.get_values())
data_df = data_df.drop(data_df[(data_df['Exp']==exp)].index, axis=0)
values = data_df.get_values()
n_train_size = int(values.shape[0]*0.8)
np.random.shuffle(values)
train = values[:n_train_size, :]
test = values[n_train_size:, :]
trainX, trainY = train[:,  :-1], np.expand_dims(train[:,  -1], axis=1)
testX, testY = test[:,  :-1], np.expand_dims(test[:,  -1],  axis=1)
testY_true = scaler.inverse_transform(test)[:,-1]
trainY_true = scaler.inverse_transform(train)[:,-1]
unobservedX = unobserved_df[['Strike', 'Exp']].get_values()
unobservedY = unobserved_df[['Price']].get_values()

# Create model
layers = [100, 80, 60, 40, 20]

model = Sequential()
model.add(Dense(layers[0], input_dim = trainX.shape[1]))
model.add(Dense(layers[1], activation='elu'))
model.add(Dense(layers[2], activation='elu'))
model.add(Dense(layers[3], activation='elu'))
```

```
model.add(Dense(layers[4], activation='elu'))
model.add(Dense(trainY.shape[1], activation = 'softplus'))
model.summary()

# Compile model
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse', 'mae', 'mape'])

# Fit the model
history = model.fit(trainX, trainY, epochs=400, batch_size=5,  verbose=2, validation_split=0.2,
        shuffle=True)
model.evaluate(trainX, trainY, batch_size=5, verbose=1)

# Predict
testY_hat_scaled = model.predict(testX)
trainY_hat_scaled = model.predict(trainX)
unobservedY_hat_scaled = model.predict(unobservedX)

# Scale back
test_hat = scaler.inverse_transform(np.column_stack((testX, testY_hat_scaled)))
train_hat = scaler.inverse_transform(np.column_stack((trainX, trainY_hat_scaled)))
unobserved_hat = scaler.inverse_transform(np.column_stack((unobservedX, unobservedY_hat_scaled)))
```

Listing E.4: Option pricing with Multilayer Perceptron. Written in Python.



NTNU
Norwegian University of
Science and Technology