Black–Scholes vs Monte Carlo Simulations vs Artificial Neural Networks in Pricing FTSE 100 Options

George Pittock
Supervisor: Dr Axel Finke
Module Code: MAC200

May 21, 2024

Abstract

This study compares the performances of the Black–Scholes model, Monte Carlo simulations and artificial neural networks in pricing options in the FTSE 100 index, from 2014 to 2023. Heston's stochastic volatility model generates volatility in Monte Carlo simulations. The performance of models is evaluated as a financial loss rather than a ratio of absolute difference and strike price. In the FTSE 100 index over these dates, results indicate that neural networks perform best in all market conditions for both put and call options. The performance of pricing call options in volatile markets is comparable across all models.

Acknowledgements

I want to thank Dr Axel Finke for his guidance and supervision for supporting me and providing valuable guidance in the writing of this project.

Contents

List of Figures

List of Tables

All code related to this report is available at `https://github.com/georgepittock/option-pricing-dissertation`.

1 Introduction

Pricing of options is a highly studied topic, and many methods are used, the most common being the Nobel Prize–winning Black–Scholes (Black and Scholes 1973). However, this has some limitations. Teneng (2011) discussed how the Black–Scholes model is constructed based on non–real–life assumptions, some of which cannot be overlooked. Krznaric (2016) validated the impracticality of these assumptions against the S&P500[1] during 2014, and how it failed to identify large movements in price. Srivastava and Shastiri (2020) studied the applicability of the Black–Scholes model in the Indian markets and confirmed this. They demonstrated that in many markets the real price for both put and call options varies significantly from the theoretical values. The Black–Scholes model often fails in highly volatile markets, at times when there is a high potential to make or lose money. Another used method is Monte Carlo simulations. When volatility or the risk–free rate are considered non-constants, option prices do not have a closed form, but computational methods, such as Monte Carlo simulations can simulate them. Simulation methods can account for a broader state of market conditions. Monte Carlo simulations are a highly studied method and have applications in many fields, including business, economics, engineering, and natural sciences (Mode 2011). Martinkutė-Kaulienė, Stankevičienė, and Venslavienė (2013) showed how Monte Carlo simulations perform better than Black–Scholes when predicting option prices for either very short periods or long periods, possibly due to the availability of more data. Bendob and Bentouir (2019) showed in a comparison of Monte Carlo simulations, binomial trees and Black–Scholes model that Monte Carlo simulations outperform the other two in low volatility markets[2]. Although these studies do not contradict each other, there is no agreement on the best pricing method or situations where different methods prevail.

Due to these pitfalls in the conventional models, the recent developments in artificial intelligence and machine learning, and the onset of the era of big data, recent studies have proposed new pricing methods using machine learning. In a time where hundreds of billions of pieces of financial data are published every day (Bloomberg 2022), it makes sense to use this data to drive financial models and prices. One commonly researched area is using neural networks to price options, and the research is promising. Malliaris and Salchenberger (1993) showed neural networks perform better in around half the cases when compared to the industry standard, the Black–Scholes model. These early studies did not change the path for option pricing but show that even simple machine learning models are as effective as the Black–Scholes model. From here, research aimed to show the situations where neural networks were more effective. İltüzer (2022) found that neural networks were more effective in less volatile market conditions, whereas the Black–Scholes model was more effective in volatile markets for call options, but the opposite is true for put options. However, Yao, Li, and Lim Tan (2000) state that the neural network is more effective in volatile markets for put and call options. However, they do caveat by stating it may be too early to claim a neural network model is better than a conventional model. Research in using machine learning to price options has not been restricted to neural networks, Liang et al. (2009) used support vector regressions and showed they outperformed the Black–Scholes model (and neural networks) in the Hong Kong securities market. Ivaşcu (2021) analysed the performance of neural networks, support vector regression and genetic algorithms as well as proposed three new decision tree methods, namely, Random Forest, XGBoost and LightGMB. He showed that machine learning algorithms outperform conventional methods in almost all cases. However, this study was only performed on call options within one

[1] The S&P500 is an index of the 500 largest companies in the United States.

[2] Binomial trees will not be considered in this report

market (WTI crude oil futures).

There is no consensus on the effectiveness of different pricing methods. Different studies have focused on a subset of algorithms within different markets. Our study will compare the effectiveness of conventional pricing methods against new machine learning methods. This will be done on the FTSE 100 index[3], through times of differing levels of market stability. It will cover the period from the opening of the London Stock Exchange in 2014 (4th Jan.), until close in 2023 (29th Dec.). This period had different levels of volatility due to the recovery from the global financial crisis, the COVID–19 pandemic and relative market stability in the late 2010s. Throughout this period, there was a variable interest rate, varying from as low as 0.1 % (late March 202–December 2024), to as high as 5.25 % (August 2023–Jan 2024) (Bank of England 2024).

This report will first present the concepts underpinning the Black–Scholes model, then with Monte Carlo methods and how stochastic volatility models can improve pricing, before presenting how neural networks can be designed to accurately price options. Finally, empirical results will show how these different methods perform against the FTSE 100 index.

- Section 2 provides background to the models used.
- Section 3 introduces the methods for generating options and finds optimal setups for the models.
- Section 4 gives empirical results for the three models on the FTSE 100 Index

2 Background

2.1 Financial Definitions

2.1.1 Derivatives & Securities

The world of finance is a complex and growing industry. The products sold across markets can generally be categorised into two broad domains: securities and derivatives.

Definition 1 (Financial Security)**.** *A financial security is a negotiable financial instrument that holds some type of monetary value. This could include a stock, a bond, or a commodity such as gold or oil (Kenton 2023).*

Definition 2 (Derivative Securities)**.** *A derivative, or derivative security is a financial instrument whose performance is based on (or derived from) the movement of the price of an underlying security, which does not have to be bought or sold (*Corporate Finance Manual *2016).*

Derivatives play a vital role in corporate finance in the 21st Century, including contributing, significantly, to the 2008 financial crisis (Sammut 2012). Creating fair prices for derivatives is crucial, not only for buyers and sellers to ensure they are entering into a fair contract, but on a larger scale to ensure global financial stability. In real–world applications, exact solutions are not known, so good approximations are found.

When pricing financial products two important features are the volatility and the risk–free rate of returns.

[3]The FTSE 100 index is an index of the Financial Times–Stock Exchange 100 largest companies in the United Kingdom.

Definition 3 (Volatility of a Security (Harris 2003))**.** *Volatility is the tendency for the price of a security to change unexpectedly.*

There is no accepted mathematical definition for calculating volatility, however Section 2.2.3 and specifically Equation 3 will outline the method that will be used to mathematically define volatility throughout this report.

Definition 4 (Risk–Free Rate (van Binsbergen, Diamond, and Grotteria 2022))**.** *The risk–free rate of return, often called the risk–free rate is a measure of the time value of money. It represents the required return for a riskless payoff in the future. Empirically, it represents the interest rate on safe assets such as government bonds.*

2.1.2 Call & Put Options

For this study, the focus will be on a common type of derivative, called options. There are many types of options, but every option is either a call or a put option.

Definition 5 (Call Option)**.** *The call option gives the holder of the option the right, but not the obligation, to buy the underlying security for an agreed strike price K on, or before, the maturity date T. Informally, a call option could be seen as a bet against the price of the underlying security rising by T (Blyth 2013).*

Definition 6 (Put Option)**.** *The put option gives the holder of the option the right, but not the obligation, to sell the underlying security for an agreed strike price K on, or before, the maturity date T. Informally, a put option could be seen as a bet against the price of the underlying security falling by T (Blyth 2013).*

Definition 7 (Strike Price (Sincere 2006))**.** *In option terminology, the predetermined, or fixed, price of a stock option is the strike price. It is the price the holder of the contract will pay if they exercise the option.*

Definition 8 (Maturity Date (Sincere 2006))**.** *The maturity date of an option is the final day on which the receiver of the option can exercise it.*

This study will only consider European options.

Definition 9 (European Option)**.** *European options allow the receiver of the option to exercise the contract only on the maturity date T.*

2.1.3 Payoff Functions

The payoff function for a European call option is,

$$\max\{S(T) - K, 0\} = (S(T) - K)^+$$

and for a European put option,

$$\max\{K - S(T), 0\} = (K - S(T))^+,$$

where $K > 0$ is the strike price and $S(T)$ is the price of the underlying security on the maturity date T. Figure 3 shows the payoffs for each type of option.

The fair price of a European option is equal to the payoff at its maturity.

Example 1. *Consider a buyer entering into a European call option contract on 1st April to purchase one share of a company in 3 months at \$430 per share. The seller agrees and the buyer is now the holder of the option. On the 1st of July (3 months) the market share price is \$450, so the holder exercises the contract. This means the seller must give the buyer one share of that company at \$450, the holder of the contract has a realised profit of \$450 − \$430 = \$20. Therefore, the fair price, or premium, of the contract should be \$20.*

2.2 Black–Scholes Model

2.2.1 Assumptions

The Black–Scholes model has some basic assumptions. These assumptions mean the premium on the option should depend solely on the price of the underlying security, any other variables are constants.

1. The short–term interest rate, r is known and constant through the time of the option.

2. The price of the underlying security follows a random walk in continuous time with variance, σ^2 proportional to the square of the stock price of the underlying security. This variance, σ^2, of the return on the security is constant.

3. The underlying security pays no dividends.

4. The option is European and, therefore can only be exercised on its maturity date.

5. There are no transaction costs in buying or selling the underlying security.

6. It is possible to borrow any fraction of the price of a security, with an interest rate, r equal to the short–term interest rate.

7. There are no penalties to short selling, i.e. a seller who does not own a security can accept the price of the security from a buyer and settle at a future date (Black and Scholes 1973).

These assumptions are not true in reality for most financial markets.

Assumptions 1 and 2 imply the evolution of a stock price under the Black–Scholes model can be described by the following stochastic differential equation:

$$\frac{dS(t)}{S(t)} = rdt + \sigma dW(t), \tag{1}$$

where $W(t)$ is a Weiner process. This stochastic differential equation is known as geometric Brownian motion. It has a solution, that can be found using Itô's Lemma.

Lemma 1 (Itô's Lemma (Hull 2017))**.** *Suppose a variable x follows the Itô process*

$$dx = a(x,t)dt + b(x,t)dz,$$

where dz is a Wiener process and a and b are functions of x and t. Variable x has a drift rate of a and a variance of b^2. Itô's lemma states that a function G of x and t follows the process

$$dG(x,t) = (\frac{\partial G}{\partial x}a + \frac{\partial G}{\partial t} + \frac{1}{2}\frac{\partial^2 G}{\partial x^2}b^2)dt + \frac{\partial G}{\partial x}bdz$$

The proof is beyond the scope of this report.

To obtain a solution for 1, choose $G(x,t) = \ln(S(t))$. Notice that $x = S(t)$, $a(x,t) = rS(t)$, $b(x,t) = \sigma S(t)$, and $z = W(t)$. Using Itô's lemma:

$$
\begin{aligned}
dG(S(t),t) &= \left(\frac{\partial G(S(t),t)}{\partial x}a + \frac{\partial G(S(t),t)}{\partial t} + \frac{1}{2}\frac{\partial^2 G(S(t),t)}{\partial x^2}b^2\right)dt \\
&\quad + \frac{\partial G(S(t),t)}{\partial x}bdz, \\
dG(S(t),t) &= \left(\frac{\partial G(S(t),t)}{\partial S(t)}rS(t) + \frac{\partial G(S(t),t)}{\partial t} + \frac{1}{2}\frac{\partial^2 G(S(t),t)}{\partial S^2(t)}\sigma^2 S^2(t)\right)dt \\
&\quad + \frac{\partial G(S(t),t)}{\partial S(t)}\sigma dW, \\
\frac{\partial G(S(t),t)}{\partial S(t)} &= \frac{1}{S(t)}, \frac{\partial^2 G(S(t),t)}{\partial S^2(t)} = -\frac{1}{S^2(t)}, \frac{\partial G(S(t),t)}{\partial t} = 0, \\
dG(S(t),t) &= \left(\frac{1}{S(t)}rS(t) - \frac{1}{2}\frac{1}{S^2(t)}\sigma^2 S^2(t)\right)dt + \frac{1}{S(t)}\sigma S(t)dW, \\
&= \left(r - \frac{1}{2}\sigma^2\right)dt + \sigma dW(t),
\end{aligned}
$$

so $G(S(t),t)$ follows a Weiner process, with drift $r - \frac{\sigma^2}{2}$ and variance σ^2.
By the Fundamental Theorem of Calculus (Apostol 1967),

$$
\begin{aligned}
G(S(t),t) = \ln\left(S\left(T\right)\right) &= \int_0^T dG(S(t),t) = \int_0^T (r - \frac{\sigma^2}{2})dt + \int_0^T \sigma dW(T) \\
&= \left(r - \frac{\sigma^2}{2}\right)T + \sigma W\left(T\right) + C,
\end{aligned}
$$

where C is the constant of integration equal to $\ln\left(S\left(0\right)\right)$. A solution for the price process can be found by taking the exponential of both sides:

$$
S(T) = S(0)\exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma W(T)\right). \tag{2}
$$

2.2.2 Black–Scholes Equation

Equation 2 is still a random process. To give a fair price for an option, a deterministic value is required. The derivation of the Black–Scholes equations will be shown here using the expectation, this is for a call option, and a similar derivation is possible for a put option. The fair price C of a call option is equal to the expectation of the discounted payoff.

$$
\begin{aligned}
C &= \mathbb{E}\left[\mathrm{e}^{-rT}\max\left\{S(T) - K, 0\right\}\right] \\
&= \mathbb{E}\left[\mathrm{e}^{-rT}\max\left\{S(0)\exp\left(\left(r - \frac{\sigma^2}{2}\right)T + \sigma W(T)\right) - K, 0\right\}\right].
\end{aligned}
$$

Let, x be a Gaussian random variable with $\mathbb{E}[x] = \left(r - \frac{\sigma^2}{2}\right) T$ and variance $\sigma^2 T$.

$$C = \frac{\mathrm{e}^{-rT}}{\sigma\sqrt{T}\sqrt{2\pi}} \int_{-\infty}^{+\infty} \max\left\{S(0)\mathrm{e}^{x} - K, 0\right\} \exp\left(-\frac{1}{2}\left[\frac{x - \left(r - \frac{\sigma^2}{2}\right) T}{\sigma\sqrt{T}}\right]^2\right) dx.$$

Performing a change variables,

$$y = \frac{x - \left(r - \frac{\sigma^2}{2}\right) T}{\sigma\sqrt{T}} \implies \frac{dy}{dx} = \frac{1}{\sigma\sqrt{T}} \implies dx = \sigma\sqrt{T}y + \left(r - \frac{\sigma^2}{2}\right) T.$$

The fair price is now,

$$C = \frac{\mathrm{e}^{-rT}}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \max\left\{S(0)\mathrm{e}^{\sigma\sqrt{T}y + \left(r - \frac{\sigma^2}{2}\right)T} - K, 0\right\} \phi(y)\, dy,$$

where, $\phi(y) = \frac{\mathrm{e}^{-\frac{y^2}{2}}}{\sqrt{2\pi}}$, the probability density function for the standard normal distribution. Finding the bounds where $S(0)\mathrm{e}^{\sigma\sqrt{T}y + \left(r - \frac{\sigma^2}{2}\right)T} - K \geq 0$, helps simplify this expression.

$$S(0)\mathrm{e}^{\sigma\sqrt{T}y + \left(r - \frac{\sigma^2}{2}\right)T} - K \geq 0 \iff y \geq -T\left(r - \frac{\sigma^2}{2}\right) - \ln\left(\frac{S(0)}{K}\right)\left(\sigma\sqrt{T}\right)^{-1}.$$

Let, $-d_2(T, S(0)) = -T(r - \frac{\sigma^2}{2}) - \ln\left(\frac{S(0)}{K}\right)\left(\sigma\sqrt{T}\right)^{-1}$. The fair price C is now:

$$C = \frac{\mathrm{e}^{-rT}}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} \left(S(0)\mathrm{e}^{\sigma\sqrt{T}y + \left(r - \frac{\sigma^2}{2}\right)T} - K\right) \phi(y)\, dy.$$

Since, $\phi(y)$ is an even function, the bounds of integration can be reversed.

$$C = \frac{\mathrm{e}^{-rT}}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} S(0)\mathrm{e}^{\sigma\sqrt{T}y + \left(r - \frac{\sigma^2}{2}\right)T}\, dy - K\mathrm{e}^{-rT} \int_{-\infty}^{d_2} \phi(y)\, dy.$$

Cancelling $\mathrm{e}^{-rT}\mathrm{e}^{rT}$ in the first

$$C = \frac{1}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} S(0)\mathrm{e}^{\sigma\sqrt{T}y - \frac{\sigma^2 T}{2}}\, dy - K\mathrm{e}^{-rT} \int_{-\infty}^{d_2} \phi(y)\, dy.$$

Note that $\sigma\sqrt{T}y - \frac{\sigma^2 T}{2} = -\frac{(-\sigma\sqrt{T}+y)^2}{2}$

$$C = \frac{1}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} S(0)\mathrm{e}^{-\frac{(-\sigma\sqrt{T}+y)^2}{2}}\, dy - K\mathrm{e}^{-rT} \int_{-\infty}^{d_2} \phi(y)\, dy.$$

The integral, $\int_{-\infty}^{d_2} \phi(y)\,dy$ is the cumulative normal distribution function, $\Phi(d_2)$.

$$C = \frac{1}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} S(0)\mathrm{e}^{-\frac{(-\sigma\sqrt{T}+y)^2}{2}}\,dy - K\mathrm{e}^{-rT}\Phi(d_2).$$

Let, $z = -\sigma\sqrt{T} + y \implies dz = dy$. The new bounds of integration become, $-d_2 \leq y \leq +\infty \implies -d_1 \leq z \leq +\infty$

$$\begin{aligned} C &= \frac{1}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} S(0)\mathrm{e}^{-\frac{z^2}{2}}\,dz - K\mathrm{e}^{-rT}\Phi(d_2) \\ &= S(0)\frac{1}{\sqrt{2\pi}} \int_{-d_2}^{+\infty} \mathrm{e}^{-\frac{z^2}{2}}\,dz - K\mathrm{e}^{-rT}\Phi(d_2) \\ &= S(0)\Phi(d_1) - K\mathrm{e}^{-rT}\Phi(d_2). \end{aligned}$$

Which is the fair price of a European call option under the Black–Scholes equation (Ross 2003).

Definition 10 (Black–Scholes Equation)**.** *The Black–Scholes equation for the fair price of a European call option is:*

$$C = S(0)\Phi(d_1) - K\mathrm{e}^{-rT}\Phi(d_2).$$

The fair price of a European put option is

$$P = K\mathrm{e}^{-rT}\Phi(-d_2) - S(0)\Phi(-d_1)$$

$$\text{where, } d_1 = \frac{\ln\left(\frac{S(0)}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}} \text{ and } d_2 = d_1 - \sigma\sqrt{T}.$$

Where, r is the risk–free interest rate, T is the time until maturity, K is the strike price, $S(0)$ is the known price at the start of the contract, σ is the volatility, and Φ represents the cumulative standard normal distribution function.

Example 2 shows how to use the Black–Scholes equation to calculate the fair price of a 28–day European call option for a risky asset with a current market price of 105 and a strike price of 100.

Example 2. *Using $T = 28/365 \approx 0.0767$, $S(0) = 105$, $K = 100$, $r = 0.05$, $\sigma = 0.1$. Calculate the fair price of a European call option.*

$$\begin{aligned} d_1 &= \frac{\ln(\frac{105}{100}) + (0.05 + \frac{0.1^2}{2}) \times 0.0767}{0.1\sqrt{0.0767}} \approx 1.914 \\ d_2 &= d_1 - 0.1\sqrt{0.0767} \approx 1.886 \\ C &= S(0)\Phi(d_1) - e^{-rT}K\Phi(d_2) \\ &= 105\Phi(1.914) - 100\mathrm{e}^{-0.05\times 0.0767}\Phi(1.886) \approx 102.08 - 96.66 \approx 5.42 \end{aligned}$$

2.2.3 Estimating Volatility

The volatility parameter is defined as the standard deviation of the natural logarithms of periodic stock price (PricewaterhouseCoopers (PwC) 2022). However, it is not possible to define this for the present time, but an estimate can be found using historical data. The most common equation is:

$$\sigma = \sqrt{N} \sum_{t=0}^{N} \ln\left(\frac{S(t)}{S(t-1)}\right). \tag{3}$$

It is often hard to determine the correct N to use, rolling annual ($N = 252$)[4], monthly ($N = 21$)[5], fortnightly ($N = 10$), or weekly ($N = 5$) are all examples that can be found in the literature.

Simple testing of the predicted option price and the actual option payoff was used to find the best N. The difference between the Black–Scholes value and the payoff for 100 random options was calculated. Random options were generated as described in section 3.1. Error was calculated using the mean squared error (MSE) and mean absolute error (MAE) metrics.
The MSE for the price option option i is defined as:

$$MSE = \frac{1}{n} \sum_{i=0}^{n} (C_i - \Pi_i)^2.$$

Similarly, the MAE for the price of option i is defined as:

$$MAE = \frac{1}{n} \sum_{i=0}^{n} |C_i - \Pi_i|,$$

where Π represents the actual payoff for the option, C the estimate from Black–Scholes model (n is the number of samples –the present study used 100). C can be replaced with P in the case of a put option.

Table 2 shows the results for different volatility measures. The results support using $N = 10$. From here, volatility in cases where volatility is constant refers to 10–day volatility, for each date by substituting $N = 10$ into Equation 3.

$$\sigma_t = \sqrt{10} \sum_{i=0}^{10} \ln(\frac{S(t-i)}{S(t-i-1)})$$

2.3 Monte Carlo Simulations

Monte Carlo methods are used to find solutions to nondeterministic problems. Many simulations of the problem are run and the average of these at any value is an estimate of the expected value of the problem. Monte Carlo simulations are a common method used to price options. It is often seen as an improvement on the Black–Scholes model as volatility can be measured as a function rather than a constant (Mehrdoust, Babaei, and Fallah 2015). It is again assumed that the evolution of a stock price can be assumed by Equation 1. However, σ and r are functions of time, so:

[4]There are 252 trading days in a year.
[5]There are 21 trading days in a month.

$$\frac{dS(t)}{S(t)} = r(t)dt + \sigma(t)dW(t). \tag{4}$$

This does not have an analytical solution, so Monte Carlo simulations can be used. The absence of an analytical solution is beyond the scope of this report, however is based on the difficulty of solving non-linear partial differential equations. In the context of pricing options, the interval of the contract $[0, T]$ is discretised to have sufficient time steps. The more time steps the better however, this is bound by computational resources. Similarly, many simulations are run up to resource constraints. The average of all the paths is taken at T and the payoff is calculated to estimate the price.

2.3.1 Convergence of Monte Carlo Methods

The Strong Law of Large Numbers is at the core of Monte Carlo methods.

Theorem 1 (Kolmogorov's Strong Law of Large Numbers (Loeve 1977))**.** *Let X_n be independent, identically distributed sequences of random variables, then*

$$\frac{X_1 + \ldots + X_n}{n} \xrightarrow{a.s.} \mu \iff \mathbb{E}\,|X| < \infty$$

The basis for the algorithm of Monte Carlo methods in pricing options is, to let C be the fair price of an option if Equation (4) is simulated n times, with accurate σ and r, then, by the Strong Law of Large Numbers, in the limit of $n \to \infty$ the average of these paths at time T equals C (Graham and Talay 2013). Therefore if enough simulations are run the average value of these simulations at time T is equal to the C. Figure 4 shows the result of running $10,000$ simulations of the FTSE 100 index from $T_0 =$ 25th Jan 2017 for 90 days.

2.3.2 Volatility Models

Here, some models for σ will be discussed. As previously discussed an advantage of Monte Carlo methods over the Black–Scholes model, is that the volatility measure σ can be a function rather than a constant. The previous section mentioned how the model is accurate when σ and r are accurate. When σ is not considered a constant, the most common model is a stochastic volatility (SV) model (Hull and White 1987). SV models are of the form:

$$\frac{dS(t)}{S(t)} = r(t)dt + \sigma^k(t)dW(t) \tag{5}$$

$$d\sigma(t) = \alpha(\sigma, t)dt + \beta(\sigma, t)dB(t) \tag{6}$$

where α, β are some functions and $dB(t)$ is a standard Weiner process, correlated with $dW(t)$ by correlation ρ. Here, k is a constant and generally $k = 1$, however, there exist models where $k \neq 1$. Generally, this model is written with σ as variance (volatility squared), rather than volatility, but to be consistent with the notation throughout this study σ represents volatility. The value of $\sigma(t)$ cannot take on negative values and therefore the instantaneous standard-deviation of $\sigma(t)$ approaches zero as $\sigma^k(t)$ approaches zero (Hull and White 1987). SV models also assume that this variance is normally distributed (Itkin 2017).

Pederzoli (2006) showed there is little difference when comparing different volatility models on the FTSE 100, whilst Lundgren and Viitanen (2022) showed that SV models

outperformed generalized autoregressive conditional heteroskedasticity (GARCH) models, as proposed by Bollerslev (1986), in all datasets. Moreover, this was further validated by Takahashi, Watanabe, and Omori (2021). The literature highlights similar if not better performances, using SV models over GARCH models. For this reason, this study will use a SV model, and not discuss GARCH models further.

2.3.3 Heston Model

One of the most common and simplest SV models is the Heston Model (Heston 1993). It is a form of Equation 5 with $k = 1, \alpha = \kappa(\theta - \sigma(t))$ and, $\beta = \xi\sigma(t)$. The model assumes $d\sqrt{\sigma(t)}$ follows an Ornstein–Uhlenbec process[6]. Figure 5 shows how a generic Ornstein–Uhlenbec process follows a random walk with a generic motion to some mean.

Heston's Volatility Model

$$d\sigma^2(t) = \kappa(\theta - \sigma^2(t))dt + \xi\sigma(t)dB(t),$$

where:

- $\sigma(t)$ is the volatility at time t,
- θ is the long–term mean–variance (volatility–squared),
- κ is the rate of mean reversion, i.e. the rate at which the process tends to θ,
- ξ represents the volatility of the volatility,
- $B(t)$ is a Weiner process and is correlated to the Weiner process of the security price by parameter ρ

The study of Heston (1993) introduced this model and outlined some default parameters for simulating volatility, these are shown in Table 3. However, some parameters are available in the dataset, or as a direct calculation. Therefore, these default parameters will be adjusted to represent what is available in the dataset, shown in Table 4. The parameters κ and ρ will then be optimised for each row in the dataset, this is discussed in Section 3.2.1.

2.4 Neural Networks

Artificial neural networks are a mathematical model linking a vector of inputs to a vector of outputs, via a composition of functions. They are some of the most common artificial intelligence models built and have many applications in image detection, pattern recognition and speech recognition. Famous models such as ChatGPT (OpenAI 2023), parts of Siri (Apple Inc. 2022) and many more are all forms of neural networks.

Due to the availability of more data and developments in this field, it is possible to generate complex neural networks that can predict the price of options.

2.4.1 Biological Inspiration

Artificial neural networks are inspired by the brain of a mammal. The brain of a mammal is a large collection (approx. 100 billion) of neurons. A neuron receives an electrical signal, called an action potential, if this action potential is above a certain threshold, it will send a new action potential to many more neurons. The result will trigger the release of some neurotransmitters that trigger action.

[6]Informally, the Ornstein–Uhlenbec process is a Wiener process that tends to the centre.

Artificial neural networks are made up of many perceptrons, a simplified model of a biological neuron. Each perceptron receives an input $\mathbf{x} = (x_1, x_2, \ldots, x_n)^\intercal$, it stores weights, $\mathbf{w} = (w_1, w_2, \ldots, w_n)^\intercal$ and a bias x_0. An activation function f, is also defined. The purpose of the activation function is to replicate the action potential output of a neuron in the brain. It must be differentiable everywhere. The activation function is often non–linear and must be at least piecewise differentiable (Hornik, Stinchcombe, and White 1989). The activation function introduces non–linearity into the network to allow it to model complex functions. The output at a perceptron is $f(\mathbf{w} \cdot \mathbf{x} + x_0)$.

An artificial neural network is made up of multiple layers of perceptrons as shown in Figure 1, and a simple artificial neural network is often called a multi–layer perceptron. In the training stage of a neural network, the weights $\mathbf{w}$ are updated to minimise the error, via the backpropagation algorithm that will be discussed in Section 2.4.4.

In this study, artificial neural networks with n inputs and 1 output will be considered. There will be any number of hidden layers and each hidden layer could have any number of neurons. Figure 1 shows a visualisation of a neural network that could be used with an input layer of 8 features, 2 hidden layers with 16 nodes each and an output layer with 1 node. These are often called multi–layer perceptrons (MLPs).

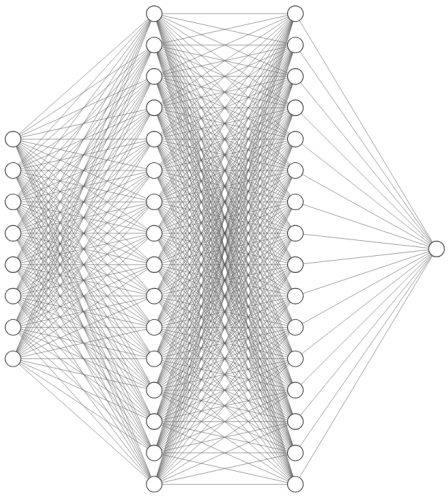


Figure 1: A neural network with an input layer of eight features, two hidden layers with 16 nodes each and an output layer with one node.

2.4.2 Loss Functions

When training an artificial neural network, inputs and targets are used. Targets are known solutions to the problem being solved. In the case of pricing options, the input will be the parameters used, e.g. volatility, strike price, maturity time whilst the target will be the known fair price of the option or some metric related to it. To determine the accuracy of the network a loss function is introduced. The loss function outputs a numerical value representing the error in the model. A common loss function in neural networks is squared error $\frac{1}{2}(t - y)^2$, where t is the target and y is the output from the network. Within the present study, t will represent that actual payoff and y the predicted price.

The use of a loss function means an $(n + 1)$ dimensional space can be defined, (n is the number of perceptrons in the network, +1 for the loss function). In neural networks, the goal of training is to find the vector(s) $\mathbf{w}$ that minimises this loss function.

2.4.3 Gradient Descent

Gradient descent is an iterative optimisation algorithm used to find a (local) minimum of some objective function F that is differentiable within some neighbourhood of a point $\mathbf{a}$. It iteratively moves down the steepest gradient, by a factor of some learning rate γ, and a minimum distance δ is defined. Once the distance between points $\mathbf{a}_{n-1}$ and $\mathbf{a}_n$ is less than δ it is assumed a local minimum has been reached. Algorithm 1 defines gradient descent.

2.4.4 Backpropagation Algorithm

Training of neural networks involves updating the weights $\mathbf{w}$. Weights are updated by an algorithm known as backpropagation. This algorithm reduces the error of the model and fits it to the data.

The backpropagation algorithm uses gradient descent to treat learning/training as an optimisation problem, with the objective function being the loss function. It is not possible to determine an analytical solution for the derivative of the loss function, but at a point in the weight space, the derivative can be calculated. The partial derivate of the error function E with respect to weight i in perceptron j, $\frac{\partial E}{\partial w_{ij}}$ is calculated for every i and j, and the weights are adjusted as defined in the gradient descent algorithm. The new value at a weight is its current value minus the product of the learning rate and gradient.

$$w_{ij} \leftarrow w_{ij} - \gamma \frac{\partial E}{\partial w_{ij}} \tag{7}$$

This is sometimes called the delta–rule, where γ represents the learning–rate parameter. The use of the minus signs ensures the weights move down the gradient. It can be possible that for a high enough learning rate γ, the new weight results in a higher error, for this reason, the learning rate is fine–tuned. Adaptive learning rates are often used.

Section 2.4.3 outlines a δ that defines a stopping criteria, however, in general, the backpropagation algorithm does not converge. Examples of attempts at defining stopping criteria can be found in the literature, Kramer and Sangiovanni-Vincentelli (1989) proposed convergence when the gradient vector is sufficiently small,

$$\left\| \frac{\partial E}{\partial w_{ij}} \right\| < \epsilon \forall i, j.$$

However, this can lead to extremely long training cycles. It is more common to stop training at the point E is stationary.

2.4.5 Learning Rate

The choice of learning rate is crucial for a neural network to converge to a global minimum. A suboptimal learning rate can increase the time for the network to train, or worse converge to a local minimum rather than a global minimum. In general, a too–small learning rate can lead to a slower rate of learning, whilst a too–large learning rate can make the network unstable and oscillate around, but not near to global minimum (Haykin 1999). Figure 8 shows the convergence of a function $f(x) = \frac{1}{75}\left(x\cos\left(\frac{x}{5}\right) + 2x\right)^2$ with 4 different learning rates. The green point represents the *point of convergence*[7]. For all learning learning rates, the starting point (denoted by a yellow dot) was $x_0 = 20$. This figure presents 4 different learning rates 0.1, 3, 5 and 10.

[7]Convergence in this example is considered to be when $|x_n - x_{n+1}| \leq 1 \times 10^{-9}$

- A learning rate of 0.1 leads to very slow convergence to a local minimum and not having the optimal value.
- A learning rate of 3 resulted in a larger jump than a learning rate of 0.1, however, it was still not enough and converged to the same suboptimal value. However, it took 32 times less iterations to converge.
- A learning rate of 5 yielded the best results and the only one used that converged to the true optimal value, at the global minima, it took 15 iterations to converge to the true optimal value. This also only took 15 iterations to converge.
- A learning rate of 10 jumped too far and gave a different, suboptimal value.

This example has shown how in just one dimension the learning rate can greatly affect the optimality of the solution and the convergence time. In a deep neural network with many features, this time will only increase.

There are more complex methods than simple constant values to implement a learning rate. These functions generally aim to start with a larger learning rate γ_0 and iteratively converge to a smaller learning rate γ_n. Further more complex learning rate schedulers such as ADAM (Kingma and Ba 2017) and ADADELTA (Matthew 2012) are commonly used, but will not be discussed here.

2.4.6 Summary of Training Cycle

A summary will now be presented of the training cycle of the back-propagation algorithm:

1. Weights $\mathbf{w}$ and biases x_0 are initialised. In this study due to the use of PyTorch initial weights are randomly from a Kaiming uniform distribution (Kaiming et al. 2015).
2. Training examples of are presented the network and the network makes a prediction, at each perception the value of $f(\mathbf{w} \cdot \mathbf{x} + x_0)$ is calculated, the network outputs y.
3. The loss is calculated as $E = \frac{1}{2}(t - y)^2$ and all weights are updated to $w_{ij} \leftarrow w_{ij} - \gamma \frac{\partial E}{\partial w_{ij}}$.
4. Repeat until E is stationary.

2.4.7 Neural Networks as Universal Approximations of Function

A neural network can be viewed as a complex composition of functions, in general, it can be seen as a mapping from an N–dimensional Euclidean space to an M–dimensional Euclidean space. The universal approximation theorem can help determine what architecture is required to provide an approximate, but accurate, realisation of this mapping.

Theorem 2 (Universal Approximation Theorem)**.** *Let $\phi(\cdot)$ be a nonconstant, continuous, bounded and monotonically increasing function. Let I_N denote the N-dimensional unit cube $[0,1]^N$, and $C(I_N)$ be space of continuous functions on I_N. Then for any $f \ni C(I_N)$ and $\epsilon > 0$, there exists $k \in \mathbb{Z}$ and constants α_i, w_{ij}, b_i, where $i = 1, \ldots, k$ and $j = 1, \ldots, N$, such that*

$$F(x_1, \ldots, x_M) = \sum_{i=1}^{k} \alpha_i \phi \left(\sum_{j=1}^{N} w_{ij} x_j + b_i \right)$$

which is an approximation of $f(\cdot)$, and

$$|F(x_1, \ldots, x_N) - f(x_1, \ldots, x_N)| < \epsilon \forall (x_1, \ldots, x_N) \in I_N$$

for all $(x_1, x_2, \ldots, x_N) \in \mathbb{R}^N$ (Haykin 1999).

This theorem can be translated to an artificial neural network with N inputs $(x_1, x_2, \ldots, x_N)$, weights w_{ij}, biases b_i, and 1 hidden layer of k perceptrons. The constant α represents the weight connecting perceptron i and the output. This provides an existence theorem, it cannot necessarily be translated to a neural network architecture that can be implemented.

3 Methods

All three methods mentioned above will be implemented in the Python programming language. Vectorised operations are used where possible to improve performance and scalability. For the same reason, modules such as Numpy, Scipy and Pandas are leveraged for array operations. No GPU acceleration is available on my personal computer, therefore the code is written without GPU acceleration.

3.1 Option Generation

It is common in the literature to generate artificial data for financial applications (du Plooy and Venter 2021). Due to the availability of historical market data, but no access to historical option data, arbitrary options can be generated and a real payoff obtained. Options are generated for any day within the dataset that has no missing columns. This meant rows early in the dataset where rolling parameters could not be calculated were dropped. Table 1 shows the parameters used to generate arbitrary options. Due to the ability to generate an extremely high number of options, there is no requirement to design a train–test–validation split, as over–fitting is not a concern.

Table 1: Parameters used to generate artificial options.

Parameter	**Values**
Maturity Time	[7 Days, 3 Years]
Option Type	Random choice of European put and European call
Strike Price	$[0.6, 0.99] \times S_0$ (call option), $[1.01, 1.4] \times S_0$ (put option)
Risk-Free Rate	Bank of England Interest Rate as of contract start date

Using the price of the option at the maturity time the known payoff can be calculated.

3.2 Creating Models

The Black–Scholes equation provides a simple model for pricing options. No fitting is required. Monte Carlo simulations are bound to computational resource constraints. However, the number of simulations run was sufficient. A small decrease in the number of simulations did not yield significantly different results.

On the other hand, the neural network did have considerable scope to be fitted. There are a variety of aspects of the model that could be adjusted. The only constraint was that any input to the model must be known at the time of entering into the contract. This is to represent what would be known to a market participant. However, there is scope to adjust parameters such as inputs, architecture, the loss function, and the learning rate scheduler. There is an infinite number of combinations that could be adjusted.

3.2.1 Optimising Parameters for Monte Carlo Simulations

For the parameters of the Heston Model, the minimize function from the Scipy optimize package (SciPy Contributors 2024) was used to optimise ρ and κ. The function aimed to minimise the RMSE of the observed volatility from that calculated by the Heston model. It uses a Nelder–Mead numerical method (Nelder and Mead 1965) to optimise the function. The output was persisted to a file and used within the simulations.

For each option, 2500 Monte Carlo simulations were used with 2500 time steps. Option parameters as described in Section 3.1 were used, except volatility was a stochastic process modelled via the Heston model. The volatility and security price were generated as covariate random processes, with a covariance matrix of $\left(\begin{smallmatrix} 1 & \rho \\ \rho & 1 \end{smallmatrix}\right)$. For each simulation paths of security prices were generated and the fair price was calculated using the common formula.

3.2.2 Inputs and Outputs to Neural Network

René and Ramazan (2000) showed that pricing options via neural networks had better results when using a homogeneity hint, that is providing an input of $\frac{S}{K}$ and an output $\frac{C}{K}$. K is known at any time so it is easy to multiply by K to find C or S. Furthermore, this has become the standard in future studies. Studies such as Healy et al. (2002) showed that traded volume had little effect on the effectiveness of the study. This is information available in our dataset, but will not be used. Bennell and Sutcliffe (2004) showed in his study on the FTSE 100 market that the best results were found when using a homogeneity hint as an input ($\frac{S}{K}$).

For this study, combinations of these will be used as well as some extensions that have not been considered in similar literature. Attributes regarding the date may be useful. Financial markets, especially equity markets are more volatile around month–end due to data releases such as economic performance reports. Short (5–day), medium (21–day) and long (90–day) term volatility measures will also be considered. Initially, all inputs will be used to create a large model. It is expected that parameters will be removed to simplify the model, and possibly improve performance.

3.2.3 Optimal Neural Network Setup

The neural network will use the PyTorch library (PyTorch Contributors 2024). The requirement that the payoff of an option is at least non–negative means activation functions that have a non–negative output can be used at the output neurons. For this reason, a so–called rectified linear unit (ReLU) activation function is used, this is defined as:

$$\mathrm{ReLU}(x) = \max(0, x).$$

The network will be trained on one–million artificially generated options. Different combinations of input parameters will be tried to see which obtains the best results. Further, different activation and network architectures will be tried to see which combination gives the best results.

3.2.4 Limitations of Neural Network

It should be noted that neural network performance was varied and appeared dependent on initial weight initialisation. This sometimes meant training cycles had to be restarted to reach sufficient performance.

Neural networks are known for their "black–box" nature. Studies such as Benitez, Castro, and Requena (1997) and Dayhoff and DeLeo (2001) have attempted to explain neural networks but no result has been accepted. They are often unexplainable, which means fixing poor–performing neural networks means trial and error, rather than fixing the mistake.

4 Results

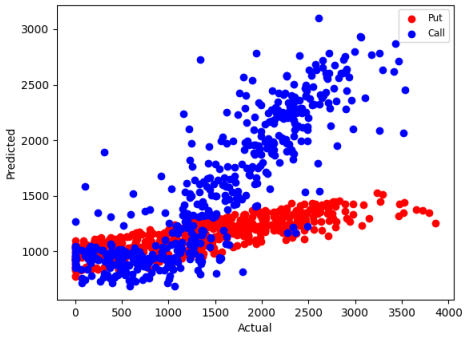
4.1 Performance Measures

The performance of all models will be calculated using RMSE $= \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}$ and MAE $= \frac{\sum_{i=1}^{N}|\hat{y}_i - y_i|}{N}$. Contrary to literature (Bennell and Sutcliffe (2004), du Plooy and Venter (2021)), error will be measured in actual price difference ($|S(t) - C|$ / $|S(t) - P|$) rather than using the homogeneity hint ($\frac{S(t)-C}{K}$ / $\frac{S(t)-P}{K}$). Error in pricing should be considered as a loss on a balance sheet, so therefore it is important to consider this a whole metric, rather than as a relative proportion. For example, a 1 % misprice on an option with a payoff of 1,000 is considerably different to a 1 % misprice on an option with a payoff of 25.

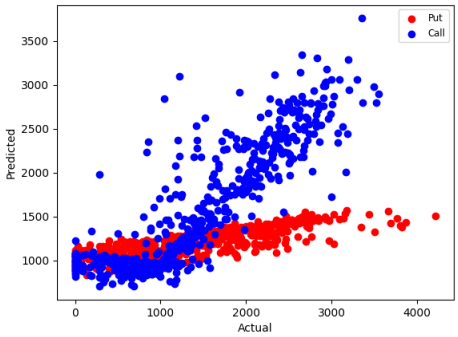
4.2 Artificial Neural Network

To determine the best set of inputs, 10000 options were trained on three different neural networks with a constant architecture.

In all three of these models, the error was relatively high and the network was struggling to learn for both call and put options (Figures 2a and 2b), shows the model of 4 inputs, struggling to price call and put options. Figure 2b shows the same for the model with 10 inputs, including option type. Similar results were observed even when the number of layers within the model was increased. It is likely that the model is struggling with differences between call and put options. The ANN is underpricing put options and overpricing call options. For any set of inputs if a put option has price $P > 0$ then the call option will have a price of 0, the opposite will be true for a call option with price $C > 0$. The model will struggle to determine the difference between types of options and return the average.



(a) Neural network of five inputs when using one model.



(b) Neural network of ten inputs including option type when using one model.

Figure 2: Neural network of different input sizes performance when using one model.

To resolve this two models were trained, one for put and one for call options. The option type will be removed as an input if used, it will be constant so have no effect. The training cycle is identical, with an added step of mapping options to the correct model. With two networks performance is considerably better.

The best–performing model is that with 9 inputs:

* $\frac{S}{K}$
* T
* r
* σ_{21}
* σ_{90}
* σ_5
* $\sigma_{21}\sqrt{T}$
* $\sigma_{90}\sqrt{T}$
* $\sigma_5\sqrt{T}$

Further tuning of the architecture found that the following architecture yielded the best results:

- Input: Nine inputs as defined above.
- Hidden Layers: Three hidden layers with ReLU activation functions, of size 256, 128, and 64 layers respectively.
- Output Layer: Maps 64 features to one.
- Learning Rate Optimiser: ADAM learning rate optimiser (Kingma and Ba 2017) with a learning rate parameter of 0.001.

Figure 7 shows the performance of the final model. It shows good accuracy most of the time but there is some evidence of mispricing.

4.3 Conventional Models vs AI Models

The optimal neural network for the FTSE 100 dataset was described in the section 4.2. It showed strong performance on the whole dataset, with a mean absolute error of 51.0077. This will now be compared to the conventional methods discussed earlier.

Table 6 shows the MAE and RMSE of all methods on 1000 artificially generated options. There is a clear difference where neural networks perform better across the board (15.93 % better than Black–Scholes, 19.05 % better than Monte Carlo simulations, when considering the difference in MAE, similar improvements are seen for RMSE). Neural networks show better performance on put options than call options. However, this was the case for all models. The limitations discussed in Section 3.2.4 highlight how this slight improvement may not be sufficient in real–world applications. Figure 6 shows little difference in the performance of all models. It can be observed that more extreme errors are found in Black–Scholes models and Monte Carlo simulations.

Studies such as İltüzer (2022) showed how the performance of different models varied depending on market conditions. As a result, the network was retrained only on the least volatile half of the data, and the same analysis was run. Table 7 shows the results for these data points. Similarly, Table 8 shows the results of pricing options in the most volatile half of the data.

For all three pricing methods, performance is considerably better in calmer periods. Neural networks are the best pricing method for both call and put options in calmer periods. The same can be said in more volatile periods. However, in volatile periods all models perform as well as each other at pricing call options.

5 Conclusion

Pricing of derivatives is important to players within financial markets to manage risk, and serve clients. Errors in pricing can cause significant loss to balance sheets and affect millions. The Nobel–prize–winning work of Black and Scholes has become the standard for many years. Recent developments in computing power have enabled simulations to become more accurate and faster, whilst the same improvements have made more data readily available to market participants. As a result, new methods have been developed, such as artificial neural networks. Monte Carlo simulations allow more advanced models for market conditions, such as volatility. Many previous studies have explored the different pricing methods of different markets. This study aimed to fill the gap by comparing the pricing performance, as a monetary loss, of Monte Carlo simulations, Black–Scholes and artificial neural networks on the FTSE 100 index. This study also developed a model for volatility forecasting using Heston's model, which was used in modelling via Monte Carlo simulations.

In both tranquil and volatile market conditions, artificial neural networks are the best pricing methods for both call and put options. The results suggest that stochastic volatility models, such as Heston's model, are no more effective at pricing options than a constant value for volatility. A possible path for future studies would be to assess the performance of neural networks against many types of options. This could produce interesting results for using neural networks as a better pricing model.

References

Apostol, T. M. (1967). *Calculus, 2nd ed., Vol. 1: One-Variable Calculus, with an Introduction to Linear Algebra*. Blaisdell. Chap. The Derivative of an Indefinite Integral. The First Fundamental Theorem of Calculus, pp. 202–204.

Apple Inc. (2022). *Hey Siri: An On-device DNN-powered Voice Trigger for Apple's Personal Assistant*. `https://machinelearning.apple.com/research/hey-siri`. (Visited on 03/26/2024).

Bank of England (Feb. 2024). *Bank Rate*. `https://www.bankofengland.co.uk/boeapps/database/Bank-Rate.asp`: Bank of England. (Visited on 02/15/2024).

Bendob, A. and N. Bentouir (2019). "Options Pricing by Monte Carlo Simulation, Binomial Tree and BMS Model: a comparative study of Nifty50 options index". English. In: *Journal of Banking and Financial Economics* 11.1, pp. 79–95. DOI: `10.7172/2353-6845.jbfe.2019.1.4`. eprint: `https://www.researchgate.net/publication/332320385_Options_Pricing_by_Monte_Carlo_Simulation_Binomial_Tree_and_BMS_Model_a_comparative_study`.

Benitez, J.M., J.L. Castro, and I. Requena (1997). "Are artificial neural networks black boxes?" In: *IEEE Transactions on Neural Networks* 8.5, pp. 1156–1164. DOI: `10.1109/72.623216`.

Bennell, J. and C. Sutcliffe (2004). "Black–Scholes versus artificial neural networks in pricing FTSE 100 options". In: *Intelligent Systems in Accounting, Finance and Management* 12.4, pp. 243–260. DOI: `10.1002/isaf.254`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/isaf.254`.

Black, F. and M. Scholes (May 1973). "The Pricing of Options and Corporate liabilities-Journal of Political Economy." In: 1, pp. 637–54.

Bloomberg (Mar. 2022). *How Bloomberg Handles a Massive Wave of Real-Time Market Data in Microseconds*. Bloomberg. eprint: `https://www.bloomberg.com/company/`

stories/how-bloomberg-handles-a-massive-wave-of-real-time-market-data-in-microseconds/. (Visited on 02/19/2024).

Blyth, S. (2013). *An Introduction to Quantitative Finance*. Oxford University Press, Incorporated. ISBN: 9780191644689.

Bollerslev, T. (1986). "Generalized autoregressive conditional heteroskedasticity". In: *Journal of Econometrics* 31.3, pp. 307–327. ISSN: 0304-4076. DOI: 10.1016/0304-4076(86)90063-1. eprint: https://www.sciencedirect.com/science/article/pii/0304407686900631.

Corporate Finance Manual (Apr. 2016). HM Revenue and Customs. https://www.gov.uk/hmrc-internal-manuals/corporate-finance-manual/cfm13020. (Visited on 02/14/2024).

Dayhoff, J. E. and J. M. DeLeo (2001). "Artificial neural networks". In: *Cancer* 91.S8, pp. 1615–1635. DOI: 10.1002/1097-0142(20010415)91:8+<1615::AID-CNCR1175>3.0.CO;2-L. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.1002/1097-0142%2820010415%2991%3A8%2B%3C1615%3A%3AAID-CNCR1175%3E3.0.CO%3B2-L.

du Plooy, R. and P. J. Venter (2021). "Pricing vanilla options using artificial neural networks: Application to the South African market". In: *Cogent Economics & Finance* 9.1. Ed. by D. McMillan, p. 1914285. DOI: 10.1080/23322039.2021.1914285. eprint: https://doi.org/10.1080/23322039.2021.1914285.

Graham, C. and D. Talay (2013). *Stochastic Simulation and Monte Carlo Methods: Mathematical Foundations of Stochastic Simulation*. Stochastic Modelling and Applied Probability. Springer Berlin Heidelberg. ISBN: 9783642393631. eprint: https://books.google.co.uk/books?id=jyG4BAAAQBAJ.

Harris, L. (2003). *Trading and Exchanges: Market Microstructure for Practitioners*. Financial Management Association survey and synthesis series. Oxford University Press. ISBN: 9780195144703. eprint: https://books.google.co.uk/books?id=xNfnCwAAQBAJ.

Haykin, S. (1999). *Neural networks: a comprehensive foundation*. Prentice Hall. ISBN: 0-13-273350-1.

Healy, J. et al. (May 2002). "A data-centric approach to understanding the pricing of financial options". In: *The European Physical Journal B - Condensed Matter and Complex Systems* 27.2, pp. 219–227. ISSN: 1434-6036. DOI: 10.1140/epjb/e20020149. eprint: https://doi.org/10.1140/epjb/e20020149.

Heston, S. (1993). "A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options". In: *The Review of Financial Studies* 6.2, pp. 327–343. ISSN: 08939454, 14657368. eprint: http://www.jstor.org/stable/2962057. (Visited on 04/08/2024).

Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5, pp. 359–366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8. eprint: https://www.sciencedirect.com/science/article/pii/0893608089900208.

Hull, J. (2017). *Options, Futures, and Other Derivatives, EBook, Global Edition*. Provider: ProQuest Ebook Central. Harlow, United Kingdom: Pearson Education, Limited. ISBN: 9781292212920. eprint: http://ebookcentral.proquest.com/lib/lboro/detail.action?docID=5186416.

Hull, J. and A. White (1987). "The Pricing of Options on Assets with Stochastic Volatilities". In: *The Journal of Finance* 42.2, pp. 281–300. DOI: 10.1111/j.1540-6261.1987.tb02568.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1987.tb02568.x.

Itkin, A. (2017). *Pricing Derivatives Under Lévy Models: Modern Finite-Difference and Pseudo-Differential Operators Approach*. Pseudo-Differential Operators. Springer New York. ISBN: 9781493967926. eprint: https://books.google.co.uk/books?id=pps7DgAAQBAJ.

Ivaşcu, C. F. (2021). "Option pricing using Machine Learning". In: *Expert Systems with Applications* 163, p. 113799. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2020.113799. eprint: https://www.sciencedirect.com/science/article/pii/S0957417420306187.

Kaiming, H. et al. (2015). *Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification*. arXiv: 1502.01852 [cs.CV].

Kenton, W. (2023). *Securities Act of 1933: Significance and History*. Investopedia. eprint: https://www.investopedia.com/terms/s/securitiesact1933.asp. (Visited on 02/14/2024).

Kingma, D. P. and J. Ba (2017). *Adam: A Method for Stochastic Optimization*. eprint: https://arxiv.org/abs/1412.6980.

Kramer, A.H. and Alberto L. Sangiovanni-Vincentelli (Jan. 1989). *Optimization Techniques for Neural Networks*. Tech. rep. UCB/ERL M89/1. EECS Department, University of California, Berkeley. eprint: http://www2.eecs.berkeley.edu/Pubs/TechRpts/1989/1146.html.

Krznaric, M. (2016). *Comparison of Option Price from Black-Scholes Model to Actual Values*. Honors Research Projects, 396. http://ideaexchange.uakron.edu/honors_research_projects/396.

Liang, X. et al. (2009). "Improving option price forecasts with neural networks and support vector regressions". In: *Neurocomputing* 72.13. Hybrid Learning Machines (HAIS 2007) / Recent Developments in Natural Computation (ICNC 2007), pp. 3055–3065. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2009.03.015. eprint: https://www.sciencedirect.com/science/article/pii/S092523120900109X.

Loeve, M. (1977). *Probability Theory I*. Graduate Texts in Mathematics. Springer. ISBN: 9780387902104. eprint: https://books.google.co.uk/books?id=_9xWBlvUEuIC.

Lundgren, E. and F. Viitanen (Jan. 2022). "Performance of Stochastic Volatility and GARCH Models in Different Market Regimes". Lund University.

Malliaris, M. and L. Salchenberger (Mar. 1993). "Beating the best: A neural network challenges the Black-Scholes formula". In: *Proceedings of 9th IEEE Conference on Artificial Intelligence for Applications*. https://doi.ieeecomputersociety.org/10.1109/CAIA.1993.366633: IEEE Computer Society, pp. 445–9. DOI: 10.1109/CAIA.1993.366633.

Martinkutė-Kaulienė, R., J. Stankevičienė, and S. Venslavienė (2013). "Option pricing using Monte Carlo simulation". In: *journal of Security and Sustainability Issues* 2.4, pp. 65–79. eprint: https://etalpykla.vilniustech.lt/handle/123456789/142386.

Matthew, D. Z. (2012). *ADADELTA: An Adaptive Learning Rate Method*. arXiv: 1212.5701.

Mehrdoust, F., S. Babaei, and S. Fallah (Nov. 2015). "Efficient Monte Carlo Option Pricing under CEV Model". In: *Communications in Statistics - Simulation and Computation* 46. DOI: 10.1080/03610918.2015.1040497.

Mode, C.J. (2011). *Applications of Monte Carlo Methods in Biology, Medicine and Other Fields of Science*. IntechOpen. ISBN: 9789533074276.

Nelder, J. A. and R. Mead (Jan. 1965). "A Simplex Method for Function Minimization". In: *The Computer Journal* 7.4, pp. 308–313. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.4.308. eprint: https://academic.oup.com/comjnl/article-pdf/7/4/308/1013182/7-4-308.pdf.

OpenAI (2023). *ChatGPT: A Language Model for Dialogue*. https://www.openai.com/chatgpt. Accessed: 2024-05-20.

Pederzoli, C. (2006). "Stochastic Volatility and GARCH: a Comparison Based on UK Stock Data". In: *The European Journal of Finance* 12.1, pp. 41–59. DOI: 10.1080/13518470500039121. eprint: https://doi.org/10.1080/13518470500039121.

PricewaterhouseCoopers (PwC) (July 2022). *US Stock-based compensation guide: Chapter 8.4 The Black-Scholes model*. https://viewpoint.pwc.com/dt/us/en/pwc/accounting_guides/stockbased_compensat/stockbased_compensat__3_US/chapter_8_estimating_US/84_the_blackscholes__US.html. (Visited on 03/23/2024).

PyTorch Contributors (2024). *PyTorch: An open source machine learning framework that accelerates the path from research prototyping to production deployment*. https://pytorch.org/. (Visited on 04/02/2024).

René, G. and G. Ramazan (2000). "Pricing and hedging derivative securities with neural networks and a homogeneity hint". In: *Journal of Econometrics* 94.1, pp. 93–115. ISSN: 0304-4076. DOI: 10.1016/S0304-4076(99)00018-4. eprint: https://www.sciencedirect.com/science/article/pii/S0304407699000184.

Ross, S.M. (2003). *An Elementary Introduction to Mathematical Finance: Options and Other Topics*. An Elementary Introduction to Mathematical Finance: Options and Other Topics. Cambridge University Press. ISBN: 9780521814294. eprint: https://books.google.co.uk/books?id=JSatCjJTfgYC.

Sammut, J. (Mar. 2012). "Derivative Instruments and the Financial Crisis 2007–2008: Role and Responsibility". In: DOI: 10.2139/ssrn.3444270. eprint: https://ssrn.com/abstract=3444270.

SciPy Contributors (2024). *SciPy optimize module*. https://docs.scipy.org/doc/scipy/reference/optimize.html\#module-scipy.optimize. (Visited on 04/15/2024).

Sincere, M. (2006). *Understanding Options*. McGraw-Hill Education. ISBN: 9780071476362. eprint: https://books.google.co.uk/books?id=bjH7CeJeWd8C.

Srivastava, A. and M. Shastiri (2020). "A Study of Black–Scholes Model's Applicability in Indian Capital Markets". In: *Paradigm* 24.1, pp. 73–92. DOI: 10.1177/0971890720914102. eprint: https://doi.org/10.1177/0971890720914102.

Takahashi, W., T. Watanabe, and Y. Omori (2021). "Forecasting Daily Volatility of Stock Price Index Using Daily Returns and Realized Volatility". In: *Econometrics and Statistics*. ISSN: 2452-3062. DOI: https://doi.org/10.1016/j.ecosta.2021.08.002. eprint: https://www.sciencedirect.com/science/article/pii/S2452306221000964.

Teneng, D. (Jan. 2011). "Limitations of the Black-Scholes model". In: *International Research Journal of Finance and Economics*, pp. 99–102.

van Binsbergen, J. H., W. F. Diamond, and M. Grotteria (2022). "Risk-free interest rates". In: *Journal of Financial Economics* 143.1, pp. 1–29. ISSN: 0304-405X. DOI: https://doi.org/10.1016/j.jfineco.2021.06.012. URL: https://www.sciencedirect.com/science/article/pii/S0304405X21002786.

Yao, J., Y. Li, and C. Lim Tan (2000). "Option price forecasting using neural networks". In: *Omega* 28.4, pp. 455–466. ISSN: 0305-0483. DOI: 10.1016/S0305-0483(99)00066-3. eprint: https://www.sciencedirect.com/science/article/pii/S0305048399000663.

İltüzer, Z. (2022). "Option pricing with neural networks vs. Black-Scholes under different volatility forecasting approaches for BIST 30 index options". In: *Borsa Istanbul Review* 22.4, pp. 725–742. ISSN: 2214-8450. DOI: 10.1016/j.bir.2021.12.001. eprint: https://www.sciencedirect.com/science/article/pii/S2214845021001071.

A Appendix

(a) Payoff for a European Call Option

(b) Payoff for a European Put Option

Figure 3: Payoff functions for European Options

Table 2: RMSE and MAE Metrics for different rolling window sizes, $N = 252, N = 21, N = 10, N = 5$.

N	**252**	**21**	**10**	**5**
RMSE	149774	125911	94258	112468
MAE	294	251	246	252

Table 3: Default parameters for simulation of volatility via Heston model

Parameter	**Value**
Mean reversion κ	2
Long-run variance θ	0.01
Current variance $\sigma^2(t)$	0.01
Correlation of $dW(t)$ and $dB(t)$, ρ	0
Volatility of volatility parameter ξ	0.1

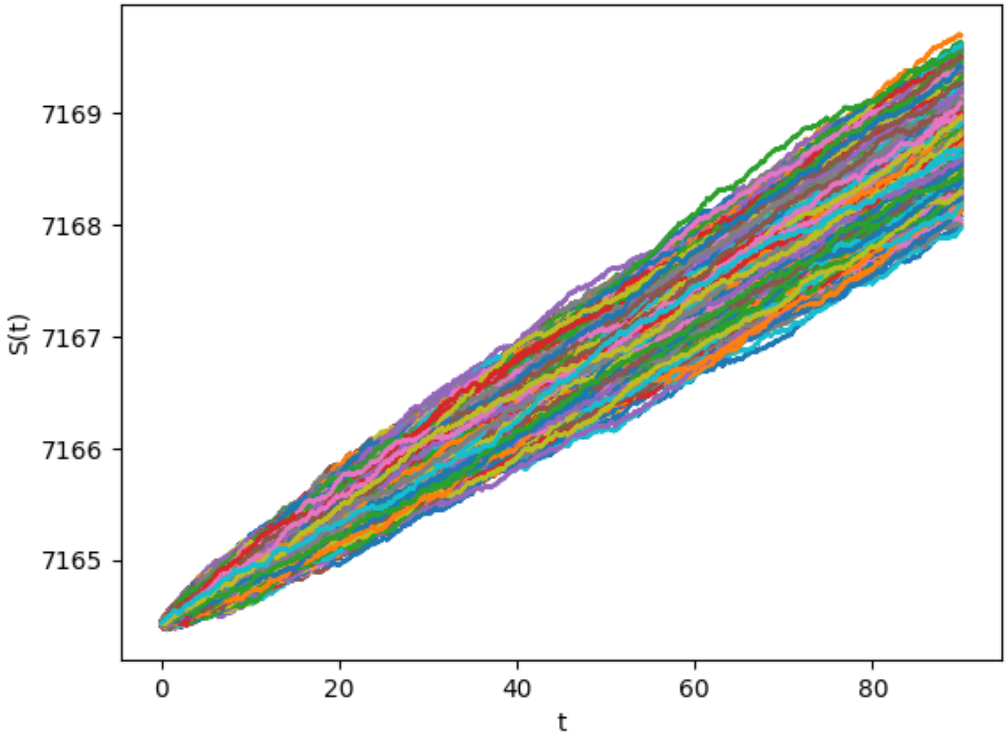


Figure 4: Monte Carlo simulation of the FTSE 100 index on $T_0 = 25$th Jan 2017 for 90 days

Table 4: Adjusted parameters for simulation of volatility via Heston model, using historical data

Parameter	**Value**
Mean reversion, κ	2
Long-run variance, θ	90-Day Rolling Volatility Squared
Current variance, $\sigma^2(t)$	5-Day Rolling Volatility Squared
Correlation of $dW(t)$ and $dB(t)$, ρ	0
Volatility of volatility parameter, ξ	Standard Deviation of 90-Day Rolling Volatility

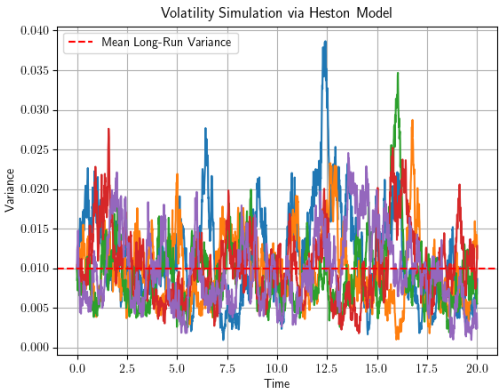


Figure 5: Five simulations of an Ornstein–Uhlenbeck Process with the default Heston parameters in Table 3

Table 5: RMSE and MAE for three different sets of inputs to neural network.

Inputs	**MAE**	**RMSE**
$\frac{S}{K}$, T, r, date attributes, option type, σ_{21}, σ_{90}, σ_5, $\sigma_{21}\sqrt{T}$, $\sigma_{90}\sqrt{T}$, $\sigma_5\sqrt{T}$	112.9196	878.0264
$\frac{S}{K}$, T, r, option type, σ_{21}, σ_{90}, σ_5, $\sigma_{21}\sqrt{T}$, $\sigma_{90}\sqrt{T}$, $\sigma_5\sqrt{T}$	118.5859	419.0965
$\frac{S}{K}$, T ,r, σ_5	−89.0521	666.9530

Table 6: RMSE and MAE for the three pricing methods across all data points.

Pricing Method	**MAE**	**RMSE**
Neural Network		
Call	54.5379	66.7074
Put	47.0293	68.7525
All	51.0077	67.6763
Black–Scholes		
Call	58.6413	80.6709
Put	59.6867	80.0041
All	59.1326	80.3582
Monte Carlo		
Call	62.5208	90.8441
Put	58.2563	84.4638
All	60.7297	88.2206

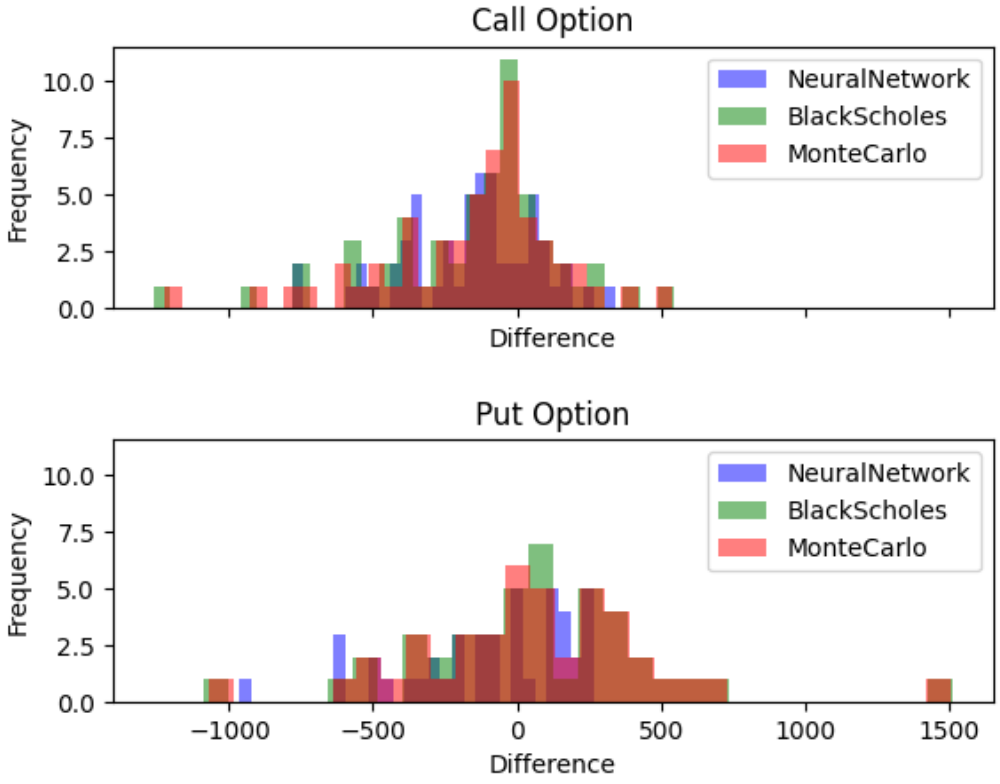


Figure 6: Histogram of pricing error of all models on 100 options over the whole dataset.

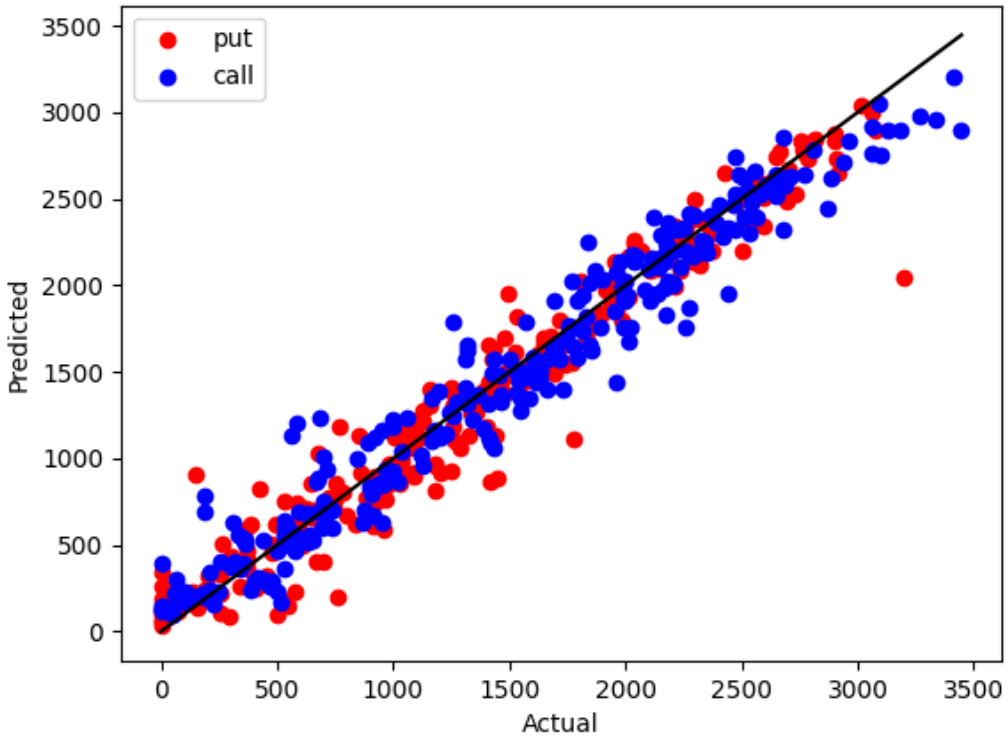


Figure 7: Actual vs artificial neural network predicted prices of the final model for 500 options.

Table 7: RMSE and MAE for the three pricing methods in the 50 % least volatile data points.

Pricing Method	**MAE**	**RMSE**
Neural Network		
Call	30.6390	39.8291
Put	24.2603	31.5439
All	27.4497	35.9261
Black Scholes		
Call	38.1764	55.1731
Put	35.6165	50.4103
All	36.8964	52.8454
Monte Carlo		
Call	37.7544	54.6250
Put	35.2381	49.6113
All	36.4962	52.1784

Table 8: RMSE and MAE for the three pricing methods in the 50 % most volatile data points.

Pricing Method	**MAE**	**RMSE**
Neural Network		
Call	45.2640	62.9769
Put	33.0210	41.7686
All	38.7752	52.8082
Black Scholes		
Call	46.2477	61.5735
Put	59.1776	87.9489
All	53.1005	76.6907
Monte Carlo		
Call	45.6759	60.2794
Put	61.1403	88.3863
All	53.8720	76.4737

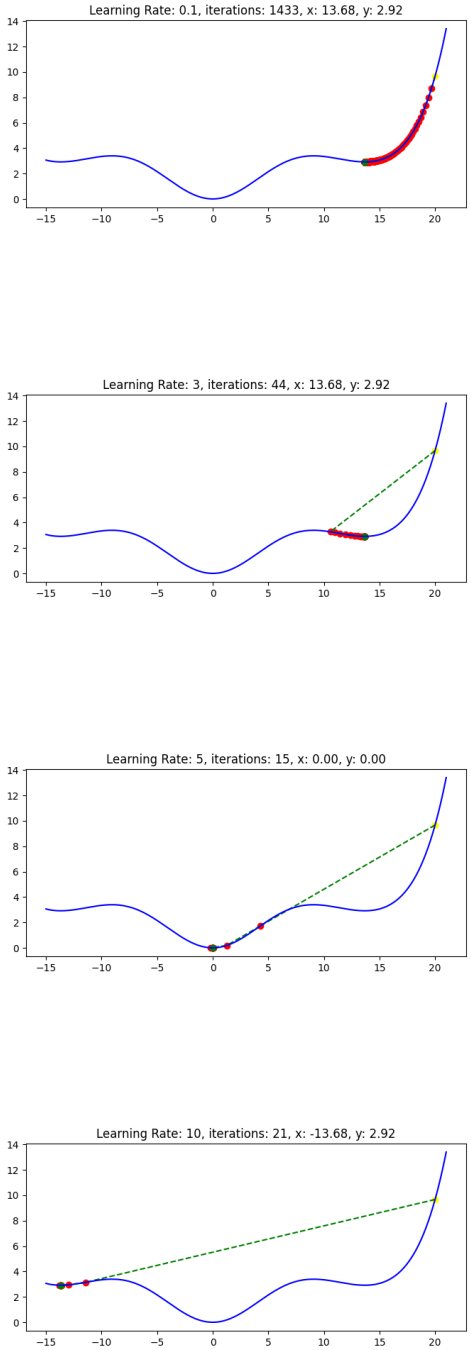


Figure 8: Convergence of different learning rates of a $f(x) = \frac{(x\cos(\frac{x}{5})+2x)^2}{75}$, from $x_0 = 20$

Algorithm 1 Gradient descent algorithm

procedure GRADIENT DESCENT($\gamma, \delta, \mathbf{a}_0$) ▷ Where γ is the learning rate, δ is the minimum distance between $\mathbf{a}_n$ and $\mathbf{a}_{n+1}$, and $\mathbf{a}_0$ is the initial starting point.
    $n \leftarrow 1$
    while $|\mathbf{a}_{n-1} - \mathbf{a}_{n-2}| \geq \delta$ **do**
        $\mathbf{a}_n \leftarrow \mathbf{a}_{n-1} - \gamma \nabla F(\mathbf{a}_{n-1})$
        $n \leftarrow n + 1$
    end while
    return $\mathbf{a}_{n-1}$
end procedure