Simulations of Implied Volatility and Option Pricing using Neural Networks and Finite Difference Methods for Heston Model

by

Sukhrat Arziyev

Submitted to the Department of Mathematics
in partial fulfillment of the requirements for the degree of

Master of Science in Applied Mathematics

at the

NAZARBAYEV UNIVERSITY

Apr 2020

Author .....
Department of Mathematics
Apr 29, 2020

Certified by .....
Dongming Wei
Professor, Chair
Thesis Supervisor

Accepted by .....
Daniel Pugh
Dean, School of Sciences and Humanities

Simulations of Implied Volatility and Option Pricing using Neural Networks and Finite Difference Methods for Heston Model

by

Sukhrat Arziyev

Submitted to the Department of Mathematics
on Apr 29, 2020, in partial fulfillment of the
requirements for the degree of
Master of Science in Applied Mathematics

Abstract

The theory of option pricing made a dramatic step forward when Black and Scholes published a centennial paper with a solution for the prices of European call and put options. However, their solution only deals with the perfect markets. In the real-world, the markets are not perfect and the predictions from their formula deviate significantly from the market prices. This is because of the assumptions on which the model is based.

Heston model is one of the newer models which extends the classical Black-Scholes model and can produce better estimates of option prices with non-constant variable volatility. However, this variable volatility needs to be accurately predicted itself, but the price which is often taken as input for finding implied volatility is not always given. In this thesis, two neural network models for learning historical volatility were constructed and compared in order to predict the implied volatility for the option without knowing its price. The results of prediction were tested and evaluated.

The better performing model was used to approximate implied volatility and the result was incorporated into the solution grid and finite difference scheme for the Heston model was applied to produce the option price. The generated prices were compared to the prices generated by the classical Black-Scholes model in several different scenarios.

Thesis Supervisor: Dongming Wei
Title: Professor, Chair

Acknowledgments

Having this opportunity, I would like to thank Dr. Dongming Wei for the guidance and support throughout this research. He was always available to give feedback and to clarify the concepts that were confusing. He was also extremely supportive and with willingness to help even in the middle of the night.

I would also thank Dr. Michal Czerwonko from Nazarbayev University Graduate School of Business for conducting seminars that helped me to gain basic understanding of the options market.

Moreover I would like to thank Dr. Yerkin Kitapbayev from North Carolina State University for providing valuable insights for improving this research work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The dramatic development of the field of machine learning resulted in their usage in solving an extensive number of sophisticated problems in both scientific and industrial fields. The enormous improvements in computing power itself enable the implementation of machine learning solutions over immense data sets that were heretofore inscrutable. This thesis examines the common and one of the most studied problems in computational finance - option pricing. Namely, approximating the implied volatility using Neural Networks (NN) and incorporating the results into the model that does option pricing using Finite Difference Method (FDM).

Financial options have a broad variety of applications, such as pricing of other assets and different investment decisions. For this reason, new ways of pricing the financial options, which have some benefits compared to the already established ones, are of great interest to the researchers and investors.

In 1973, Fisher Black and Myron Scholes published their famous option pricing model, which resulted in having an enormous effect on the development of computational finance. However, due to some strong assumptions made in the model, violate the real-life market phenomena, in which volatility is variable. It creates opportunities for researchers to investigate better models including the nonlinear volatility phenomena.

Option pricing models strive to evaluate the reasons for fluctuation of option prices in order to identify the best option price for the particular investment. There are a

lot of research works done to tackle the issues that arouse with the Black-Scholes model's assumptions. One of them is the work of Steven Heston [12], where the author proposed a model of stochastic volatility, which seeks to overcome the proven shortcomings of the Black-Scholes model.

The main purpose of NNs is to create a computational environment where they can efficiently mimic the work of the human brain. Analogous to how neurons behave, nodes in neural networks are activated when there are sufficient stimuli called inputs. The activation goes through the whole network and creates a response to the stimuli, called outputs. The node connections are similar to synapses, transmitting information from layer to layer. When enough information is provided, the NN can effectively recognize patterns in data and, as a result, approximate any continuous function.

Recently, the usability of deep learning has advanced extremely in a variety of fields, such as medicine, computer vision, and image recognition. Likewise, deep learning has found its extensive use in the financial field, with enormous accessibility of data in gigantic quantities and in real-time.

The work of Mary Malliaris and Linda Salchenberger in 1993 presented [18] a simple neural network option pricing model that was built using the Black-Scholes model. Then in 1996, they used S&P 100 data [19] in order to build a neural network model to forecast implied volatility. Their work was based solely on Black-Scholes application of NN.

Recently, extensive research is done in the field of calibrations of NN models. Liu et al. [15] and Itkin [13] propose their models to find the optimal parameters for neural networks that minimize distances between the option prices provided by the market and the ones forecasted by the model.

Liu et al. [16] proposed a different view on the neural network approach in option pricing. They introduced several deep learning models to approximate the option price. Then, they use the results of the model in order to approximate implied volatility. Then, they compared their results with the traditional numerical approaches and got exceptional results that prove that the neural networks are indeed useful when

dealing with option pricing.

In 2019, Liu et al. [15] proposed a new way of looking at option pricing using neural network calibration. Namely, they extend their previous work [16] by introducing a Calibration Neural Network (CaNN). They use it during the back-propagation phase. Its main idea is that during training the hidden-nodes of a neural network become non-trainable parameters while the input nodes, which are in our case our calibration parameters, become trainable ones.

Furthermore, in 2019, Zheng et al. [24] introduce their work on so-called gated neural networks, where they introduce an approach to model implied volatility surface using deep learning methods. The main idea was to use a few single models with weights being determined by another model. The proposed model has to offer a new view on how to deal with one of the most critical assumptions of the Black-Scholes [3] model.

It can be seen from the previous literature study that generally the performance of neural networks in option pricing was compared with the exact solution of the Black-Scholes model or with the numerical approximation of the Heston model. In this paper, in addition to the neural network simulation of the Black-Scholes model, the Heston stochastic volatility model was solved numerically. The solution then provides the implied volatility numerically at finite-difference grid points. Using the data set and the deep learning methodology, the implied volatility is also approximated in connection with the Heston model. Finally, the finite difference solutions and machine learning solutions are integrated.

Therefore, the outline of the thesis is as follows. Chapter 2 gives the prerequisites of option pricing theory including the Ito's lemma and the theory behind the Black-Scholes model. Chapter 3 discusses the Heston stochastic volatility model including the numerical solution by FDM. Chapter 4 provides a basic understanding of neural networks starting with feedforward neural networks and ending up with long short-term memory networks. Chapter 5 demonstrates the performance of deep learning neural networks in modeling the prices of the Black-Scholes model and the implied volatility for the Heston model. After that, the results of the implied volatility neural

network are plugged into the numerical approximation described in Chapter 3. Eventually, in Chapter 6 conclusions are drawn based on the results. Chapter 7 draws the general ideas on which the future research will be based on.

Chapter 2

Preliminaries

This chapter builds the on option pricing theory based on Black-Scholes model.

2.1 Options and portfolio replication

Derivative securities are the ones that derive their value from the value of other financial assets. Some examples of derivative securities include options, futures, and swaps. Options are one of the most investigated topics in computational finance. An option is, basically, a contract that gives its holder the right, but not the obligation, to buy or sell a specific asset under certain conditions.

The following definitions are adapted from [17].

Definition 1. *Process $w(t)$ is called* ***Wiener process*** *(also known as* ***Brownian motion****) if the following properties are satisfied:*

- *if $p < t$ then $w(t) - w(p)$ is $N(0, \sqrt{(t-p)})$*
- *$w(t_2) - w(t_1)$ and $w(t_4) - w(t_3)$ have zero correlation for all $0 \leq t_1 < t_2 \leq t_3 < t_4$*
- *$P(w(t_0) = 0) = 1$*

Definition 2. ***(Diffusion Process)*** *Diffusion process is known to be a solution to SDE (Stochastic Differential Equation).*

2.2 Ito's lemma

Ito's lemma is a key in finding the derivative of time-dependent features in some stochastic process. If compared with ordinary differential calculus, it plays the same role in the stochastic setting as the chain rule.

Theorem 1. ***(Ito's lemma)*** *Consider some random process x to be determined as Ito process:*

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

with z being a Wiener process. Consider some process $y(t) = V(x(t),t)$ then $y(t)$ can be described by Ito's equation:

$$dy(t) = (\frac{\partial V}{\partial x}a + \frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial x^2}b^2)dt + \frac{\partial V}{\partial x}bdz$$

The full proof of Ito's lemma can be found in [14] and the simplified one in [17].

2.3 Black-Scholes Model

Nowadays, derivative contracts that are used for hedging the risks form a large portion of the financial market. This section provides the most popular option pricing model which led to the dramatic development of options trading because of its simplicity and interpretability.

2.3.1 Assumptions of the Black-Scholes model

In the Black-Scholes model [3], while deriving the value of the option, the ideal conditions are assumed. Namely, the authors assume the following:

- short-term interest rate is considered to be known and constant
- the distribution of the stock price is log-normal
- there are no dividends from the stock

- the option is considered to be European, so it can only be exercised at maturity
- the transaction costs for buying or selling the stock or the option are zero
- it is possible to borrow any fraction of a price of the security at the short-term interest rate
- no penalty for short-selling

These assumptions make the price of the option be dependent only on the price of the underlying asset, time to maturity, and some set of parameters that are assumed to be constant [3].

Since most of the parameters, such as volatility and stock price, can change very fast, and with enormous variance, the assumptions make the model not being applicable to the real world.

2.3.2 Derivation of the Black-Scholes PDE

The model assumes that the stock has the log-normal dynamics

$$dS = \mu S dt + \sigma S dW \tag{2.1}$$

where dW is normally distributed with standard deviation $\sqrt{dt}$ (i.e. W is the Brownian motion).

In order to derive the equation, we will need to use the notion of *replicating portfolio*. Basically, it means that if someone wants to construct a portfolio of stock S and bond B such that it replicates the change of the derivative security. At each time t the investor selects an amount a of S and an amount b of B, which makes the portfolio value to be

$$V(t) = a_t S(t) + y_t B(t)$$

. The instantaneous change in portfolio value with respect to changes in security prices will be

$$dV = a_t dS + b_t dB$$

and using 2.1,

$$dV = a_t dS + b_t dB = a_t[\mu S dt + \sigma S dW] + b_t r B dt = (a_t \mu S + b_t r B)dt + a_t \sigma S dW \quad (2.2)$$

Since the latter equation is required to match the coefficients of $dV = a_t dS + b_t dB$, set

$$a_t = \frac{\partial V}{\partial S}$$

and get that

$$b_t = \frac{1}{B}[V(S,t) - S\frac{\partial V}{\partial S}]$$

Plugging a_t and b_t into 2.2,

$$\frac{\partial V}{\partial t} + \frac{\partial V}{\partial S} rS + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 = rV$$

which is the desired Black-Scholes PDE.

Black-Scholes model takes into the assumption that the price of the stock behaves as the stochastic process:

$$dS_t = \mu S_t dt + \sigma S_t dW_t$$

where μ is drift, σ is the standard deviation of the stock and W_t is the Wiener process In 1973, the economists Fischer Black, Myron Scholes proposed a famous equation

$$\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma^2 S^2 + \frac{\partial V}{\partial S} rS - rV = 0 \quad (2.3)$$

For European call option, the boundary conditions are:

$$V(S,T) = max(S - K, 0)$$

$$V(0,t) = 0,$$

$$V(S,t) \approx S \text{ as } S \to \infty$$

The Black-Scholes equation has exact solution:

$$V(S,t) = SN(d_1) - Ke^{-r(T-t)}N(d_2)$$

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

$$d_2 = d_1 - \sigma\sqrt{T-t}$$

$N(*)$ represents the normal distribution [3].

2.3.3 Solution of Black-Scholes equation

Let $t = T - \frac{\tau}{1/2\sigma^2}$, $S = Ke^x$ and

$$V(S,t) = Kv(x,\tau) \tag{2.4}$$

After taking τ and x to the left hand sides of the equations, one will get $\tau = \frac{\sigma^2}{2}(T-t)$ and $x = \log(\frac{S}{K})$.

Taking the derivatives of 2.4 with respect to t and S:

$$\frac{\partial V}{\partial t} = K\frac{\partial v}{\partial \tau}\frac{\partial \tau}{\partial t} = K\frac{\partial v}{\partial \tau}\frac{-\sigma^2}{2}$$

$$\frac{\partial V}{\partial S} = K\frac{\partial v}{\partial x}\frac{\partial x}{\partial S} = K\frac{\partial v}{\partial x}\frac{1}{S}$$

Moreover,

$$\frac{\partial^2 V}{\partial S^2} = \frac{\partial}{\partial S}(K\frac{\partial v}{\partial x}\frac{1}{S}) = K\frac{\partial^2 v}{\partial x^2}\frac{1}{S^2} - K\frac{\partial v}{\partial x}\frac{1}{S^2}$$

Boundary conditions:

$$V(S,T) = \max(S-K,0) = \max(Ke^x - K, 0)$$

$$V(S,T) = Kv(x,0)$$

so,

$$v(x,0) = \max(e^x - 1, 0)$$

Plugging everything into the original Black-Scholes model, the PDE will look like:

$$K\frac{\partial v}{\partial \tau}\frac{-\sigma^2}{2} + \frac{\sigma^2}{2}S^2(K\frac{\partial v}{\partial x}\frac{-1}{S^2} + K\frac{\partial^2 v}{\partial x^2}\frac{1}{S^2}) + rS(K\frac{\partial v}{\partial x}\frac{1}{S}) - rKv = 0$$

which transforms to

$$\frac{\partial v}{\partial \tau} = \frac{\partial^2 v}{\partial x^2} + (k-1)\frac{\partial v}{\partial x} - kv$$

Further simplify the model by using

$$v = e^{\alpha x + \beta \tau}u(x,\tau)$$

From which

$$v_\tau = \beta e^{\alpha x+\beta\tau}u + e^{\alpha x+\beta\tau}u_\tau,$$

$$v_x = \alpha e^{\alpha x+\beta\tau}u + e^{\alpha x+\beta\tau}u_x,$$

$$v_{xx} = \alpha^2 e^{\alpha x+\beta\tau}u + 2\alpha e^{\alpha x+\beta\tau}u_x + e^{\alpha x+\beta\tau}u_{xx}$$

gathering everything into the PDE:

$$u_\tau = u_{xx} + [2\alpha + (k-1)]u_x + [\alpha^2 + (k-1)\alpha - k - \beta]u$$

After using appropriate constants, the equation will take the form of the heat equation:

$$u_\tau = u_{xx}$$

with initial condition $u(x,0) = \max(e^{\frac{(k+1)}{2}x} - e^{\frac{(k-1)}{2}x}, 0)$.

With $u(x,\tau) = \frac{1}{2\sqrt{\pi\tau}}\int_{-\infty}^{\infty} u_0(s)e^{-\frac{(x-s)^2}{4r}}ds$ and $\Phi(d) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{d} e^{-\frac{y^2}{2}}dy$, we get

$$u(x,\tau) = e^{\frac{k+1}{2}x+\frac{(k+1)^2}{4}\tau}\Phi(\frac{x}{\sqrt{2\tau}} + \sqrt{\frac{\tau}{2}}(k+1)) - e^{\frac{k-1}{2}x+\frac{(k-1)^2}{4}\tau}\Phi(\frac{x}{\sqrt{2\tau}} + \sqrt{\frac{\tau}{2}}(k-1))$$

Then,

$$V(S,t) = SN(d_1) - Ke^{-r(T-t)}N(d_2)$$

with $d_1 = \frac{\log(S/K)+(r+\sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$ $d_2 = d_1 - \sigma\sqrt{T-t}$ [17].

2.3.4 Implied volatility

Implied volatility is known to be a prediction made by investors of the future movement of the underlying asset, i.e. it is their belief of how variable the stock is going to be between today and the maturity date. Since the accurate forecast is crucial, reliable methods for approximating it are required. Given the observed option value V_{mkt}, implied volatility can be derived from

$$V_{mkt} = BS(\sigma^*, S, K, T, r)$$

Since the Black-Scholes equation is monotonic with respect to σ^*, implied volatility exists and can be written implicitly by means of the inverse Black-Scholes function as

$$\sigma^* = BS^{-1}(V_{mkt}, S, K, T, r) \tag{2.5}$$

which does not have an analytic solution and thus, needs to be solved numerically [16].

Usually, in order to approximate the solution of (2.5), iterative approaches, such as Newton-Raphson method, are used. Those approaches take five input variables which are shown in Equation (2.5).

However, when the price of the option is unknown, the calculation of the option by 2.5 becomes impossible. Therefore, another approach could be to look at the historical volatility and try to predict the implied volatility from there. Fortunately, use of neural networks, widely known as "universal function approximators" [11], become extremely widespread. The approach for dealing with it as follows: given the parameters and historical volatility build a complex predictive model that will learn to accurately predict implied given the input parameters. This approach should

perform faster than the traditional iterative approaches.

Chapter 3

Heston model

Heston Stochastic Volatility Model extends the work of Black and Scholes [3] by introducing stochastically varying volatility. For the Heston model, the system of stochastic equations under the risk-neutral measure is the following:

$$\begin{cases} dS_t = rS_t dt + \sqrt{v_t} S_t dW_t^s & S_{t_0} = S_0 \\ dv_t = \kappa(\overline{v} - v_t)dt + \gamma\sqrt{v_t} dW_t^v & v_{t_0} = v_0 \\ dW_t^s dW_t^v = \rho dt \end{cases} \tag{3.1}$$

where v_t is the instantaneous variance, κ is the rate of reversion, γ is the volatility of the variance, and W_t^s, W_t^v are two Wiener processes with correlation ρ.

In general, it is similar to the classical Black-Scholes model, but here stochastic behavior for the volatility is introduced. Stochastic volatility is aimed to tackle one of the most restrictive assumptions of the Black-Scholes model - the assumption of constancy of volatility.

Heston model illustrates the mean-reverting process, which is consistent with the observations of financial market behavior. Secondly, it models the correlation between asset returns and volatility. Because of this, it is now possible to model the relationship between the underlying asset and volatility. As a result, the Heston model includes a variety of real market characteristics that are crucial and are generally observed in financial markets.

3.1 Assumptions and Derivation

For the Black-Scholes model, one forms a portfolio that consists of the underlying stock together with the financial derivative which is hedging the mentioned underlying stock and is making the portfolio riskless. As for the case of the Heston model, since the investor should also be aware of the volatility in the portfolio, he needs an additional derivative to hedge it.

Therefore, consider a portfolio with value P, the option $V(S, v, t)$, a units of stock S and b units of another option $U(S, v, t)$ that hedges the volatility. The portfolio value will be:

$$P = V + aS + bU$$

an infinitesimal change of this portfolio leads to the following infinitesimal change of its value:

$$dP = dV + adS + bdU$$

The general idea is the same as what is has been done in the case of the derivation of the original Black-Scholes model: Ito's lemma is being applied to both U and V, which allows to describe the portfolio P in derivatives U and V. The key is to find the number of shares a and the number of options b which make the portfolio value P risk-less. After applying Ito's lemma to both U and V, the change of portfolio value will transform to

$$dP = dV + adS + bdU =$$

$$= [\frac{\partial V}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\sigma vS\frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2}\sigma^2 v\frac{\partial^2 V}{\partial v^2}]dt +$$

$$+b[\frac{\partial U}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + \rho\sigma vS\frac{\partial^2 U}{\partial v \partial S} + \frac{1}{2}\sigma^2 v\frac{\partial^2 U}{\partial v^2}]dt + [\frac{\partial V}{\partial S} + b\frac{\partial U}{\partial S} + a]dS + [\frac{\partial V}{\partial v} + b\frac{\partial U}{\partial v}]dv$$

Since the portfolio must be hedged against volatility and stock movements, the terms involving dS and dv must be zero. Thus,

$$a = -b\frac{\partial U}{\partial S} - \frac{\partial V}{\partial S}$$

$$b = -\frac{\frac{\partial V}{\partial v}}{\frac{\partial U}{\partial v}}$$

Besides, there needs to be a risk-free rate r, which means that $dP = rPdt$, having

$$dP = [\frac{\partial V}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\sigma vS\frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2}\sigma^2 v\frac{\partial^2 V}{\partial v^2}]dt + b[\frac{\partial U}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + \rho\sigma vS\frac{\partial^2 U}{\partial v \partial S} + \frac{1}{2}\sigma^2 v\frac{\partial^2 U}{\partial v^2}]dt$$

$$= r(V + aS + bU)$$

and ending up with

$$\frac{\frac{\partial V}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 V}{\partial S^2} + \rho\sigma vS\frac{\partial^2 V}{\partial v \partial S} + \frac{1}{2}\sigma^2 v\frac{\partial^2 V}{\partial v^2} - rV + rS\frac{\partial V}{\partial S}}{\frac{\partial V}{\partial v}} =$$

$$= \frac{\frac{\partial U}{\partial t} + \frac{1}{2}vS^2\frac{\partial^2 U}{\partial S^2} + \rho\sigma vS\frac{\partial^2 U}{\partial v \partial S} + \frac{1}{2}\sigma^2 v\frac{\partial^2 U}{\partial v^2} - rU + rS\frac{\partial U}{\partial S}}{\frac{\partial U}{\partial v}}$$

Note that the left-hand side is the function of V, while the right-hand side is the function of U. Consider a function $f(S, v, t)$. Both sides of the last equation can be written as a function of f. According to Heston, take $f(S, v, t) = -\kappa(\theta - v) + \lambda(S, v, t)$ [20].

3.1.1 Heston equation

After plugging it in the last equation, rearranging it, we end up with the Heston partial differential equation:

$$V_t + rSV_S + \kappa(\overline{v} - v)V_v + \frac{1}{2}vS^2V_{SS} + vV_{Sv} + \frac{1}{2}\gamma^2 vV_{vv} - rV = 0$$

3.1.2 Boundary and terminal conditions

Boundary conditions for the European call option are as follows:

- At maturity T, the price of the option is equal to the payoff:

$$V(S, v, T) = \max(0, S - K)$$

- For $S = 0$, the option has no value at all, i.e.

$$V(0, v, t) = 0$$

- With S becoming extremely big, the delta is going to 1.

$$\frac{\partial V}{\partial S}(\infty, v, t) = 1$$

- $rS\frac{\partial V}{\partial S}(S, 0, t) + \kappa\overline{v}\frac{\partial V}{\partial v}(S, 0, t) - rV(S, 0, t) + V_t(S, 0, t) = 0$. As it was written in [20], it is so-called "generator" of the Heston model, where the first half is the generator of the Black-Scholes model and the second half is for generating stochastic volatility.

- With increasing volatility, the option price is becoming equal to the stock price. $V(S, \infty, t) = S$ [12].

Unlike the classical Black-Scholes model, the latter does not have an analytical solution. For this reason loads of numerical methods have been applied in order to approximate it, such as Monte-Carlo simulation and Fourier-cosine series approximation [16].

3.1.3 Well-posedness of the model

Different approximation schemes for the initial boundary value problems governed by a PDE may converge to different solutions if the problem has more than one solution. In order to state that a scheme is convergent, it is necessary to prove that the original problem is well-posed, which means that the problem has a unique solution. For the case of the Heston model, the proof of its well-posedness was shown in [23]. In fact, it was one of the earliest papers which made European option pricing using Finite Element Methods and proving the well-posedness of the Heston PDE. However, their solution makes an assumption that $v \neq 0$, but when $v = 0$, their solution breaks and

the equation becomes degenerate. Now, consider

$$\begin{aligned} dX(t) = b(t, X(t))dt + \sigma(t, X(t))dW(t), t \geq s \\ X(s) = x \end{aligned} \tag{3.2}$$

The following theorem was taken from [7].

Theorem 2. *Suppose that* $\boldsymbol{b}$*:* $\mathbb{R}^n \times [0,T] \to \mathbb{R}^n$ *and* $\boldsymbol{B}$*:* $\mathbb{R}^n \times [0,T] \to \mathbb{M}^{m\times n}$ *are continuous and satisfy the following conditions:*

$$\begin{aligned} |\boldsymbol{b}(x,t) - \boldsymbol{b}(\hat{x},t)| \leq L|x-\hat{x}| \\ |\boldsymbol{B}(x,t) - \boldsymbol{B}(\hat{x},t)| \leq L|x-\hat{x}| \end{aligned} \quad \textit{for all } 0 \leq t \leq T, x, \hat{x} \in \mathbb{R}^n$$

$$\begin{aligned} |\boldsymbol{b}(x,t)| \leq L(1+|x|) \\ |\boldsymbol{B}(x,t)| \leq L(1+|x|) \end{aligned} \quad \textit{for all } 0 \leq t \leq T, x \in \mathbb{R}^n$$

for some constant L*.*
Let X_0 *be any* $\mathbb{R}^n$*-valued random variable such that* $E(|X_0|^2) < \infty$ X_0 *is independent of* $W^+(0)$*, where* W *is a given m-dimensional Brownian motion.*

Then there exists a unique solution.

Turning back to the Heston equation, denote $x = \ln(S)$. Then following the reasoning of [12], the Heston SDE can be transformed to

$$\begin{aligned} dS(t) = \sigma S dt + S\sqrt{v(t)}dW_1(t) \\ dv(t) = \kappa(\overline{v} - v(t))dt + \rho\sqrt{v(t)}dW_2(t) \end{aligned}$$

This system is proved to have a unique solution in [8]. Therefore, the PDE that is formed from it also has a unique solution in [8]. Since Equation (7.1) has a solution, then (3.1.1) also has unique solution.

3.2 Heston PDE approximation

There are lots of methods for numerical approximation of the Heston PDE, such as Monte Carlo simulation, different variations of Fourier transform and Finite Diference methods. Coming up with an efficient numerical method for the Heston problem is a crucial step in achieving a numerical solution scheme for the Heston PDE.

However, in this work, it was decided to approximate 3.1.1 using finite difference method, because for the future research, the goal will be to mimic the partition of the plane into set of points and grids so that it will allow the neural network model to better learn the approximation approach.

3.2.1 Building the mesh for finite difference approximation

The non-uniform mesh is used for the discretization of the stock price of s and volatility v. The grid is constructed such that points are being concentrated around points of interest and the grid itself is sparse anywhere else. Construction of the non-uniform mesh has the advantage of significantly improving the accuracy of pricing by using fewer grid points, which in turn reduces the required computing power.

Define a non-uniform mesh the way it was described by Foulon [9]. Let $m_1 \geq 1$ be the number of grid-points in s direction. Then $0 = s_0 < s_1 < s_2 < ... < s_{m_1} = S_{max}$. Let equidistant points $\xi_0 < \xi_1 < ... < \xi_{m_1}$ be given by

$$\xi_i = \sinh^{-1}(\frac{-K}{c}) + i\Delta\xi(0 \leq i \leq m_1)$$

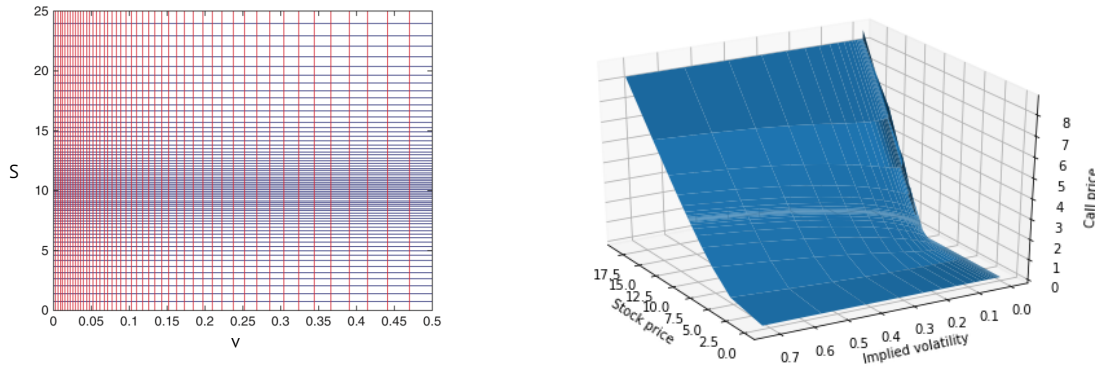
with spacing

$$\Delta\xi = \frac{1}{m_1}[\sinh^{-1}((S-K)/c) - sinh^{-1}(\frac{-K}{c})]$$

Then a non-uniform mesh $0 = s_0 < s_1 < ... < s_{m_1} = S$ is defined through transformation

$$s_i = K + c\sinh(\xi_i)(0 \leq i \leq m_1)$$

Construction of grid in v-direction is similar. Let $m_2 \geq 1$ be the number of grid-



(a) Non-uniform mesh

(b) Solution for the european call option price

Figure 3-1: Performance of two implied volatility prediction network architectures

points in v direction and $d > 0$ is some constant. Similarly, take equidistant points $\eta_j = j\Delta\eta$ with $j = 1, ..., m_2$. Combine them to have $\Delta\eta = \frac{1}{m_2} \sinh^{-1} \frac{v}{d}$. Finally, let $v_j = d \sinh \eta_j$ with $0 = v_0 < v_1 < v_2 < ... < v_{m_2} = v_{max}$.

Figure 3-1 represents the two dimensional mesh for the volatility and the underlying stock price. Values for v are represented by red lines, whereas values for S are shown by blue lines. Here, $m_1 = m_2 = 50$, $S_{max} = 25$ and $v_{max} = 0.5$. Furthermore, the strike price is chosen to be $K = 10$ and significant number of points are concentrated at $S = K$.

3.2.2 Crank-Nicolson scheme for Heston PDE

Once the mesh is built, it is needed to iterate through it in order to find the solution. Crank-Nicolson scheme is popular among the variety of financial engineering literature because of its second-order accuracy.

Finite difference discretization for the Heston PDE yields the initial value problem for a large system of stiff ODEs:

$$V'(t) = AV(t) + b(t), (0 \leq t \leq T), V(0) = V_0 \tag{3.3}$$

where A is a $m \times m$-matrix and $b(t)$ $(t \geq 0)$, V_0 are given m-vectors with $m = m_1 m_2$. It is directly obtained from the initial conditions and the vector function b depends

on the boundary conditions.

Let $\Delta t > 0$ be a given time step. Then, let $t_n = n\Delta t$ be temporal grid points with $n = 0, 1, 2, ...$ In general, the scheme has the following form:

$$\frac{V^{n+1} - V^n}{\Delta t} + A(t_{n+1/2})\frac{V^{n+1} + V^n}{2} = G(t_{n+1/2}) \tag{3.4}$$

where $V^0 = V_0$, $\Delta t = t_{n+1} - t_n$, $t_{n+1/2} = \frac{t_{n+1}+t_n}{2}$ [6].

Given that the information at time t_n is known, the value at t_{n+1} can be found in the following manner:

$$(I + \frac{\Delta t A(t_{1/2})}{2})V^{n+1} = (I - \frac{\Delta t A^{n+1/2}}{2})V^n + \Delta t G(t_{1/2}) \tag{3.5}$$

with I being an identity matrix [6]. Introduce a function:

$$F(t, U) = G(t) - A(t)V(t)$$

Then the Crank-Nicolson scheme makes approximations to the exact $U(t_n)$ in the following manner:

$$V_n = V_{n-1}\frac{1}{2}\Delta t F(t_{n-1}, V_{n-1}) + \frac{1}{2}\Delta t F(t_n, V_n)$$

$$V'(t) = F(t, U(t))$$

with $0 \leq t \leq T$ and $V(0) = V_0$

Another useful approach that is also wide-spread among the financial engineering literature is the concept of alternating direction implicit (ADI) methods. The main idea is to approximate the multi-dimensional problem as a series of simple one-dimensional problems. Make the decomposition of A in the following way:

$$A = A_0 + A_1 + A_2 \tag{3.6}$$

Construct a matrix A_0 as a part of A that forms from the finite difference discretiza-

tion. After that, according to the basic ADI principle, we need to divide A into A_1 and A_2 being two parts that match respectively every spatial derivative in both s- and v-directions.

Similarly, decompose b in (3.3):

$$b(t) = b_0(t) + b_1(t) + b_2(t)$$

Finally, define F_0, F_1 and F_2 such that:

$$F_i = A_i w + b_i$$

for $i = 0, 1, 2$, which clearly implies that:

$$F = F_0 + F_1 + F_2 \tag{3.7}$$

These results are used to approximate the value of U in (3.1.1)

3.2.3 Finite Difference approximation of derivatives

The approximation of derivative will be used for constructing A matrices in section 3.2.4. To approximate the first derivative $f'(x_i)$, consider three FD schemes:

$$f'(x_i) \approx \alpha_{i,-2} f(x_{i-2}) + \alpha_{i,-1} f(x_{i-1}) + \alpha_{i,0} f(x_i)$$

$$f'(x_i) \approx \beta_{i,-1} f(x_{i-1}) + \beta_{i,0} f(x_i) + \beta_{i,1} f(x_{i+1})$$

$$f'(x_i) \approx \gamma_{i,0} f(x_i) + \gamma_{i,+1} f(x_{i+1}) + \gamma_{i,2} f(x_{i+2})$$

with coefficients given by:

$$\alpha_{i,-2} = \frac{\Delta x_i}{\Delta x_{i-1}(\Delta x_{i-1} + \Delta x_i)}, \alpha_{i,-1} = \frac{-\Delta x_{i-1} - \Delta x_i}{\Delta x_{i-1} \Delta x_i},$$

$$\alpha_{i,-2} = \frac{\Delta x_i}{\Delta x_i(\Delta x_{i-1} + \Delta x_i)}$$

and coefficients for β, γ are defined in a similar fashion. To approximate the second derivative $f''(x_i)$, deal with the following FD scheme:

$$f''(x_i) \approx \omega_{i,-1} f(x_{i-1}) + \omega_{i,0} f(x_i) + \omega_{i,1} f(x_{i+1})$$

with coefficients:

$$\omega_{i,-1} = \frac{2}{\Delta x_i (\Delta x_i + \Delta x_{i+1})}, \omega_{i,0} = \frac{-2}{\Delta x_i \Delta x_{i+1}},$$

$$\omega_{i,1} = \frac{-2}{\Delta x_i (\Delta x_i + \Delta x_{i+1})}$$

3.2.4 Construction of matrices

A_0 contains all mixed derivative terms, A_1 is for derivative terms containing only s, A_2 is for derivative terms formed by v. Remaining $-rV$ is being evenly distributed among A_1 and A_2.

It is convenient to decompose A_1 and A_2 into smaller matrices as follows:

$$A_1 = A_s + A_{ss} - \frac{1}{2} rI$$

$$A_2 = A_v + A_{vv} - \frac{1}{2} rI$$

A_s is also an $m \times m$ matrix with entries being $r s_{ij} \Delta s$ with s_{ij} being the values of s on the grid at position (i, j) and Δs being an approximation of the derivative of s. A_{ss} is being constructed similarly except the entries are $\frac{1}{2} s_{ij}^2 v_{ij} \Delta^2 s$ where $\Delta^2 s$ is the second derivative approximation of s. A_v and A_{vv} are constructed similarly except the terms are $\kappa(\overline{v} - v_{ij})$ and $\frac{1}{2}\sigma^2 v_{ij}$, respectively.

As for A_0, it is simply an $m \times m$ matrix with entries being $\rho\sigma s_{ij} v_{ij} \delta sv$ where δsv is the discrete approximation of the mixed derivative term.

b_0, b_1, b_2 are constructed in a similar manner with respect to the boundary conditions of 3.1.1.

3.2.5 Consistency, stability and convergence of Crank-Nicolson Scheme

Firstly, the Crank-Nicolson scheme is consistent, because it eventually converges to the exact solution of the PDE as the measure of grid size and time-step size get close to zero. The proof can be shown using LU representation of the equation, Taylor expansion and big-O notation as in [5].

Transform Equation 3.1.1 and consider initial value problem in the form

$$\begin{aligned} \frac{\partial V}{\partial \tau} &= \mathbf{L}V \\ V(0,s,v) &= \phi(s,v) \end{aligned} \tag{3.8}$$

where L is the operator that is defined as [21]:

$$L = \frac{1}{2}vS^2\frac{\partial^2}{\partial S^2} + \frac{1}{2}\sigma^2 v\frac{\partial^2}{\partial v^2} + \rho\sigma vS\frac{\partial^2}{\partial v \partial S} + rS\frac{\partial}{\partial S} + \kappa(\overline{v} - v)\frac{\partial U}{\partial v} - r$$

The method is also stable because the error is decreasing by one-time step which was shown experimentally in [4] and stated in [6]. It can also be shown analytically by exploring the link between Heston equation and the convection-diffusion equation as in [5]. The following theorem is a well-known theorem from numerical analysis for PDEs.

Theorem 3. *(Lax equivalence theorem) A consistent, two level difference scheme for a well-posed linear initial-value problem is convergent if and only if it is stable.*

Since the above described scheme for the Heston model is consistent and stable, Theorem 3 shows that it is convergent as well.

Chapter 4

Neural Networks

This chapter is aimed to provide an overview of artificial neural network structures, including how information is being transmitted by neurons. Also, the algorithms with the help of which the network learns are being presented.

4.1 Artificial Neural networks

Artificial Neural Networks (ANNs) are widely known to have the merits of flexibility and non-linearity. For this reason, they have high potential in approximation theory and have been widely used to deal with approximation tasks.. Those approximation abilities of neural networks have become extremely popular recently. For instance, the network with linear output can approximate any nonlinear function if a sufficient number of hidden units is provided. For this reason, ANNs are called *universal function approximators* [11]. One can also define ANNs as a composition of algorithms used to imitate the human brain processes that are responsible for determining the patterns in data [7].

There are three major steps in neural network training:

- feedforward - computing the output of the given layer
- backward - computing the gradient of the layers' output with respect to its inputs and weights

- update - change the layer weights with respect to the learning rate

4.1.1 Feedforward Neural Network

Feedforward neural networks are the most classical type of deep learning models, which have the main goal to approximate some function. For example, if given some function $f(x)$, a feedforward neural network aims to find the parameters θ that will yield the most accurate function approximation $y = f(x, \theta)$ [11].

Mathematically, ANNs can be interpreted as a sequence of functional transformations

$$a_j = \sum_{i=1}^{N} w_{ji} x_i + b_j$$

where j is the number of the hidden layer, w_{ji}'s are the weights of the hidden layer j and b_j are called biases at layer j. Consequently, a_j are being transformed by *activation functions*, which are aimed to add non-linearity to neural networks architecture [2]:

- ReLU, $\phi(x) = max(x, 0)$
- Sigmoid, $\phi(x) = \frac{1}{1+e^{-x}}$ [11]

The basic structure is presented in (4-1)

4.1.2 Error backpropagation

After forward propagation, the error needs to be computed efficiently.

The method was initially introduced in the 1960s but became popular after the work by Rumelhart, Hinton, and Williams [22] in 1989.

The issue that arises is that the error function that was previously defined needs to be optimized with respect to w, but it is not a function of w. Fortunately, the backpropagation algorithm which is using the chain rule as its basic idea helps to tackle the issue. In general, after each iteration through a neural network, the backpropagation algorithm passes backward and, at the same time, adjusts the weights

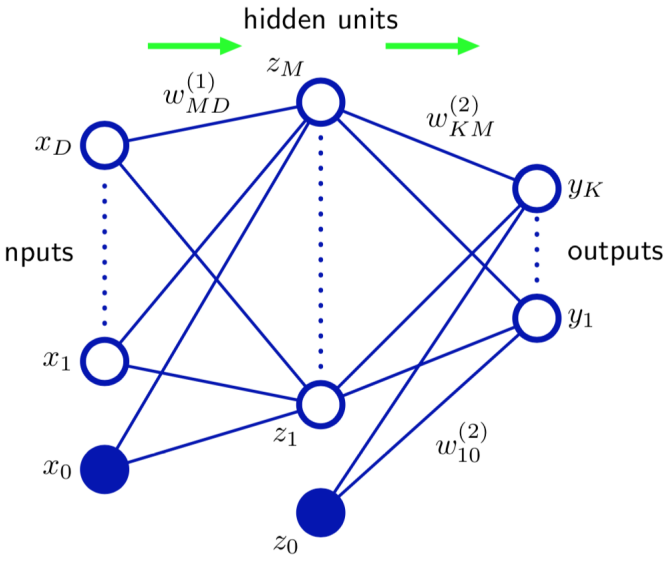


Figure 4-1: Structure of the Neural Network [2]

and biases.

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial a}\frac{\partial a}{\partial w}$$

Let δ_k represent the error of the node k at the current layer:

$$\delta_k = a_k - y_k$$

Keeping that in mind, one would derive the gradient to be:

$$\frac{\partial J}{\partial a} = \sum_k \frac{\partial J}{\partial a_k}\frac{\partial a_k}{\partial a} = \sum_k \delta_k \frac{\partial w_k z}{\partial z}\frac{\partial z}{\partial a} = \sum_k \delta_k w_k \frac{\partial h(a)}{\partial a} = h'(a)\sum_k w_k \delta_k$$

[2]

4.1.3 Update of Neural network

Given a training set of input vectors $\{\mathbf{x}_n\}$, with $1 \leq n \leq N$, in combination with the output vectors $\{\hat{\mathbf{y}}_n\}$, the goal is to find the minimum of

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \hat{\mathbf{y}}_n|| \tag{4.1}$$

The loss can be minimized via gradient descent algorithm. The value of the error in equation (4.1) can be minimized by first taking the derivative of E with respect to the weights vector $\mathbf{w}$. The derivative can be used as an update of the gradient descent algorithm.

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \Delta\nabla E \tag{4.2}$$

where n denotes the iteration number, Δ is the learning rate of the algorithm. The algorithm stops either when the pre-defined number of iterations is reached, or when the algorithm converges, which means that $\Delta\nabla E$ becomes very close to zero and $\mathbf{w}_{n+1} \approx \mathbf{w}_n$.

4.2 Recurrent Neural Network

The problem with a traditional ANN is that it uses only the current input information and cannot store the previous inputs' information. In order to tackle this issue, Recurrent Neural Network was introduced. It allows its structure to have loops to store historical information. The structure now is the same as in Fig. 4-3. In those loops, the hidden layers pass the previous hidden state to the next step. RNN specializes in processing sequential data [11].

4.2.1 How it works

Given some sequence $X = (X_1, X_2, ..., X_T)$ as an input, recurrent neural network computes sequences H and Y, which are hidden and output sequences, respectively. RNN does so by iterating through equations below with $1 \leq t \leq T$:

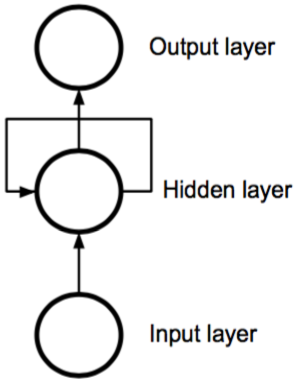


Figure 4-2: Structure of the Recurrent Neural Network

$$H_t = \phi(W_{xh}X_t + W_{hh}H_{t-1} + b_h)$$

$$Y_t = W_{hy}H_t + b$$

where parameters are quite intuitive with W being weight matrices, b denoting bias vectors and ϕ denoting activation function between the layers.

4.2.2 Issues with RNNs

A typical recurrent neural network has an extremely serious problem caused by vanishing gradient. Gradient descent itself finds the global minimum for the cost function which helps to find the optimal setup for the network itself. Information goes through the network from input to the output neurons, with the error being calculated and propagated back through the same network to update the weights.

The situation with recurrent neural networks is quite similar, but there are some special conditions that are taking place. Firstly, the output of the layer is used as input for the next layer, and, as a result, secondly, the error can be calculated at each iteration.

RNN uses sequential processing. So, after an extreme number of iterations, the loss function will be multiplied by an extreme number of extremely small numbers. This means that over time the gradient will become very small and the process of

finding the most optimal loss will take a long while to perform. One of the possible and most popular solutions to this issue is the Long Short-Term Memory networks (LSTMs).

4.3 Long Short-Term Memory networks

The sample structure for the LSTM is shown on the figure below.

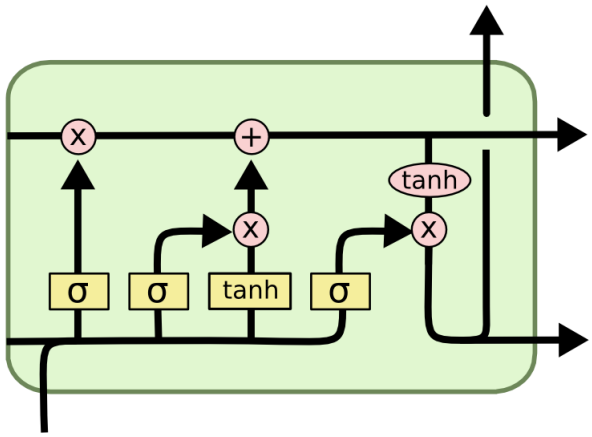


Figure 4-3: Structure of Long Short-Term Memory network

In general, the control flow of LSTM is similar to that of the recurrent neural network. They differ with operations inside the LSTM cells which help the LSTM network to keep or forget the information the cell is containing.

With that being said, the Long Short-Term Memory network consists of the cell state and input, forget, and output gates. The cell state is used for retaining and passing relevant information throughout the chain. The forget gate is used to decide whether or not the information should be forgotten or kept in the network. The input gate is used for updating the cell state. The role of the output gate is to determine the next hidden state.

Mathematically, LSTM can be expressed as follows:

$$f_t = \sigma(W_f H_{t-1} + U_f X_t + b_f)$$

$$i_t = \sigma(W_i H_{t-1} + U_i X_t + b_i)$$

$$o_t = \sigma(W_o H_{t-1} + U_o X_t + b_o)$$

$$\hat{c} = tanh(W_c H_{t-1} + U_c X_t + b_c)$$

$$C_t = f_t C_{t-1} + i_t * \hat{c}_t$$

$$H_t = o_t * tanh(C_t)$$

where f, i, o are the outputs of forget, input and output gates, respectively; C is the cell state $\hat{c}$ is the input activation, U is the weights before entering the gate; W, H, X and b are already familiar from the previous section.

Chapter 5

Option pricing by Neural Networks and Finite Difference Method

This chapter shows to what extent can Neural Networks learn to approximate the implied volatility based on historical market data and shows the incorporation of the implied volatility in FDM to approximate the Heston equation. Both results are visualized and compared with alternative methods.

5.1 Data

For the experiments, S&P 500 data was used. S&P 500 Index (SPX): The Standard & Poorś 500 Index is a capitalization-weighted index of 500 stocks from a big variety of industries. It is more convenient to use particular S&P500 data for several reasons. First of all, it is the most widely used stock, and hence the options are actively traded. The second reason is that the options are European ones. Another reason is that the index does not pay dividends.

The data for the experiments were acquired from OptionMetrics for the period from 04/01/1996 to 28/06/2019. OptionMetrics also provides historical interest rate data, which will be used in approximating volatility and pricing the options further.

5.2 Methodology

Firstly, the neural network is being trained on the market implied volatility data. After that the predicted implied volatility is being used as an entry for option pricing.

5.3 Performance measures

The models were evaluated on several metrics, which are

- Mean Squared Error:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Root Mean Squared Error:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

- Mean Absolute Percentage Error:

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N}|\frac{y_i - \hat{y}_i}{y_i}|$$

- R^2 measure was also used in order to understand what portion of the variance is being described by the neural network model

5.4 Heston Stochastic Volatility Simulation

5.4.1 Implied Volatility simulation using Neural Networks

The model for approximating implied volatility is similar to those presented in [16] and [24] with different architecture and approach to training the neural network.

Before using the neural network approximation of implied volatility, its inputs and output should be defined. Two parameters are taken for input which are:

- the *annualized time to maturity*:

$$\tau = \frac{T - t}{A} \tag{5.1}$$

  with A being the *annualization factor*;

- the *log-moneyness* m was used as:

$$m = \log\left(\frac{F}{K}\right)$$

  where F is the forward price of the stock price which is calculated by $F = Se^{r\tau}$.

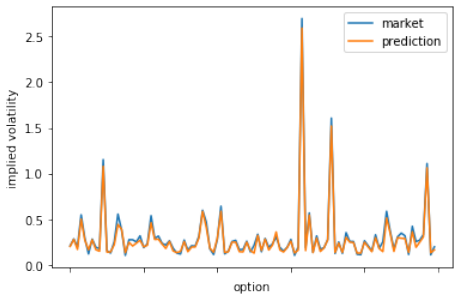
There is only one output entry for implied volatility.

Therefore, the goal here will be to build a deep learning model for implied volatility $v(m, \tau)$ that is dependent only on moneyness m and annualized time to maturity τ. Consequently, the model will be able to approximate implied volatility if four, instead of five, inputs of equation 2.5 are given. Generally, the objective is to find the weights for the neurons that will minimize the sum of squared error between market implied volatility and the predicted: $\frac{1}{n}\sum_{i=1}^{n}(v-\hat{v})^2$, where $\hat{v}$ is the predicted implied volatility.

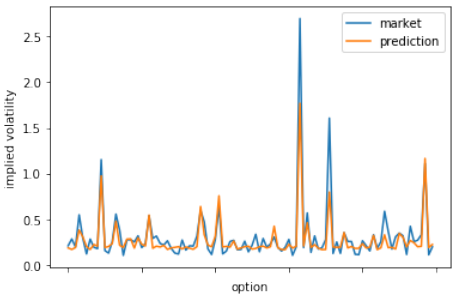
Since there are two input neurons and one output neurons, the next question that arises is which architecture to use for the hidden neurons. In order to check it, two custom network architectures were built: one with simple dense layers and the second using the LSTM layer. Table 5.1 shows the comparison of the performance among four metrics. It can be seen that the model with the LSTM layer significantly outperforms the model with only dense hidden layers.

	Simple NN	LSTM
MSE	0.023403	0.003125
RMSE	0.15298	0.055903
MAPE	23.36361	9.29984
R^2	0.66363	0.955082

Table 5.1: Comparison of performance of two neural network architectures for implied volatility model

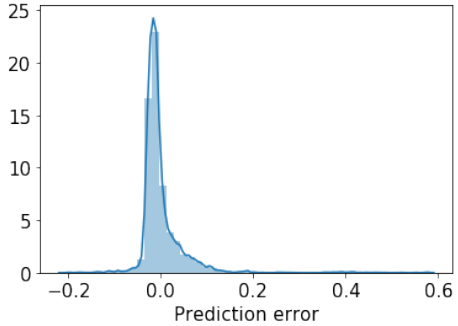


(a) LSTM Network

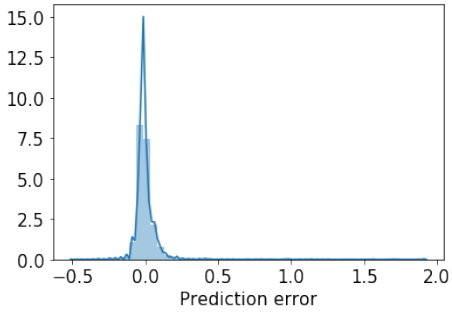


(b) Neural Network with simple dense layers

Figure 5-1: Performance of two implied volatility prediction network architectures (horizontal axis represents a set of options with some parameter set)



(a) LSTM Network



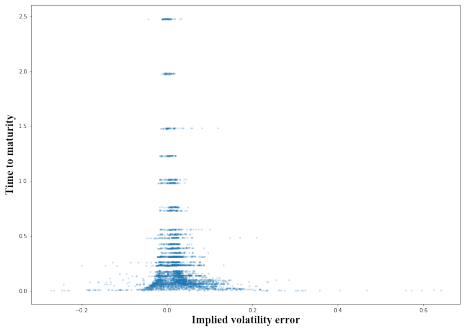
(b) Neural Network with simple dense layers

Figure 5-2: Prediction error distribution for two network architectures with vertical axis representing the density of distribution

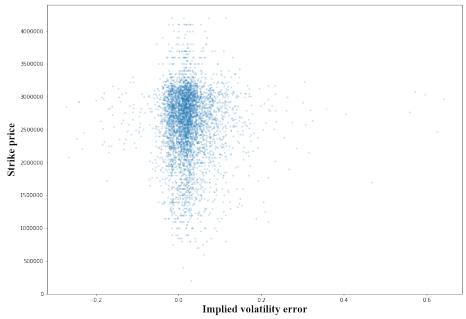
Figure 5-1a shows how the predicted implied volatility behaves relative to the true market value where the horizontal axis represents a set of options with different parameter set (S_i, K_i, r_i, T_i). As it can be seen from the graph, the model catches the sharp rises and drops of the volatility extremely accurately. Figure 5-1b confirms the above statement that the neural network with ordinary architecture performs worse by not being always able to fit the sharp increase in implied volatility.

Figure 5-2 shows that the range of error distribution for the LSTM network is significantly more narrow and most of the errors are concentrated around zero rather than the distribution of errors of the neural network with dense layers with a wider range of error distribution and significantly lower density of errors near zero.

Figure 5-3a shows how the error in the prediction of implied volatility distributed across different values of time-to-maturity. It can be seen that in general the per-



(a) Prediction error vs Time-to-maturity



(b) Prediction error vs Strike price

Figure 5-3: Distribution of prediction error across different values of time-to-maturity and strike price

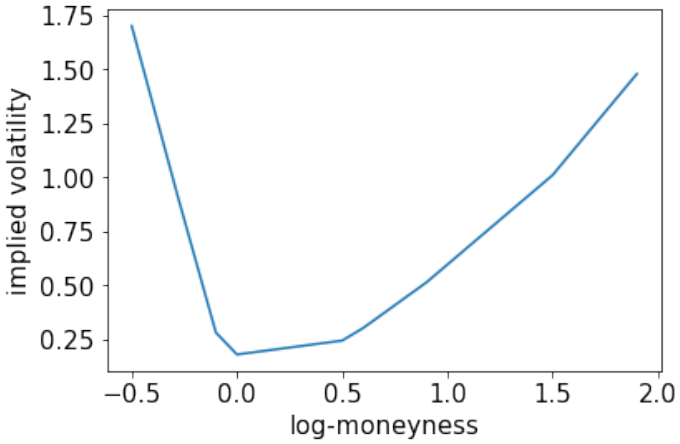


Figure 5-4: Implied volatility skew

formance of the neural network is reasonable across most of the values of τ, but has most of the errors concentrating when τ is close to zero. The reason is mainly that when $\tau \approx 0$, then time-to-maturity itself cancels the weight that is being assigned to it by the network, and prediction is made solely depending on the moneyness of the option. Figure 5-3b how that error is distributed with respect to the strike price of the options. It is worth noting that they are heavily distributed across the zero error with some insignificant number of outliers that are located farther.

Implied volatility skew represented in Figure 5-4 was produced by fixing annualized time to maturity $\tau = 1$ and randomly sampling values for log-moneyness. The values for implied volatility were generated from the implied volatility neural network.

5.4.2 Using the algorithm for single option

Suppose an investor has an option with all parameters known except implied volatility v and its price. Then the question is how to approximate them. Since the implied volatility neural network was trained on S&P 500 data, take the parameters accordingly:

$$S = 2926, K = 4200, \tau = 1.48$$

and $r = 0.024$, $\kappa = 1.5$, $\sigma = 0.3$, $\overline{v} = 0.13$ and $\rho = 0.5$.

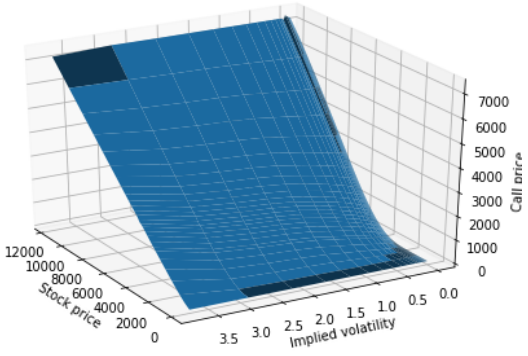
The neural network that was described in section 5.4.1 predicts $\hat{v} = 0.1522 \pm 0.01415$.

Now, the goal is to price the option using the FDM. The approach was described in Chapter 3. Briefly, a non-uniform grid needs to be constructed. However, this time the grid will have S-direction and $\hat{v}$-direction with NN-forecasted implied volatility and grid search is being made with respect to the error margin of the prediction. Moreover, that this time it is not needed to first find BS price of the option and then to invert it to approximate implied volatility. This approach takes already known stock price S, strike price K, interest rate r, and time to maturity T as input and outputs pretty accurate implied volatility as described in the previous section. After that, the matrix A and the vector b from Section 3.2.4 are created in order to approximate the corresponding option prices for the given S and $\hat{v}$ on the grid. Finally, the Crank-Nicolson scheme is being applied to iterate through the grid and find the relevant approximation. Figure 5-5 shows the corresponding 3-D visualization of the solution. The price that is being computed by this FDM:

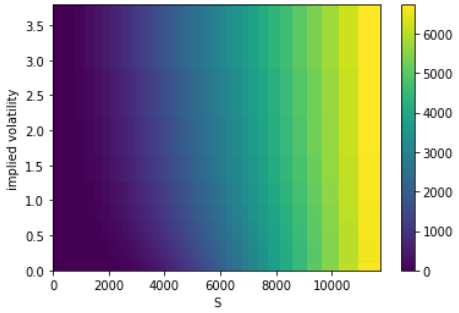
$$\hat{C} = 212.74$$

5.4.3 Analysis of FDM

In practice, it can be shown analytically that when variance of the volatility γ goes to zero, option price by Heston model becomes approximately equal to the one by Black-



(a) 3D mesh of the solution



(b) Color grid of the price of the option

Figure 5-5: Performance of two implied volatility prediction network architectures

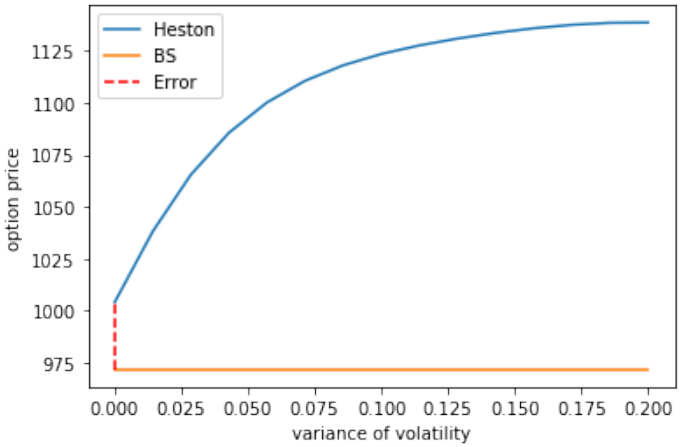


Figure 5-6: Prices of Heston and Black-Scholes models as volatility of variance approaches 0

Scholes model. To verify that for the FDM case, take: $r = 0.024, S = 2900, K = 2000, \hat{v} = 0.15188, \overline{v} = 0.1, T = 1.48, \kappa = 1.5, \rho = 0.5$. Figure 5-6 confirms, with some approximation uncertainty, that Heston tends to Black-Scholes as $\gamma \to 0$, at least there is a trend confirming that.

Figure 5-7 shows the volatility smirk for both Heston and Black-Scholes models for the same parameters defined above. It can be noticed that after the Black-Scholes line passes $S = 2900$, the difference between the prices increases. This happens because after that point the option becomes out-the-money and Black-Scholes tends to underprice out-the-money options [10]. It can be also seen that the proposed Heston model solution tends to slightly overprice the option in the in-the-money strike price

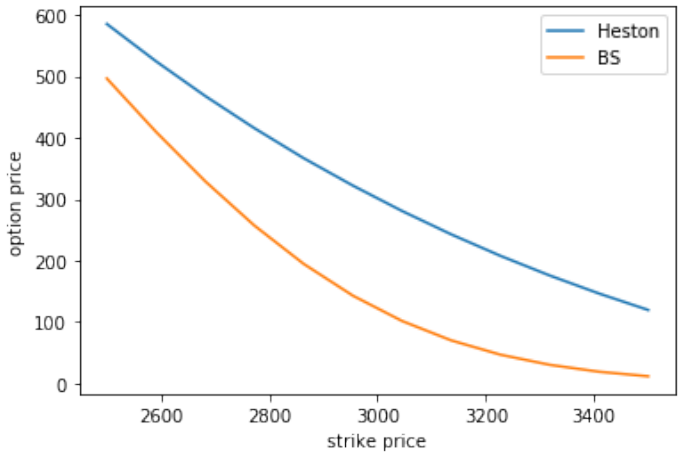


Figure 5-7: Volatility smirk for Heston and Black-Scholes models

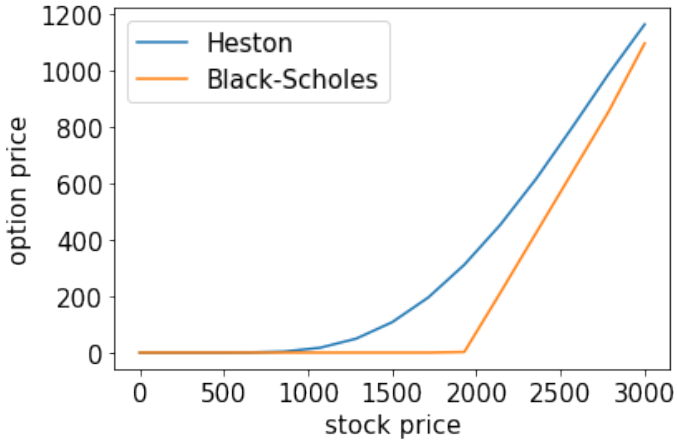


Figure 5-8: Option price by Heston and Black-Scholes models when the stock price approaches zero

domain where the Black-Scholes model is known for underpricing the option [10].

Figure 5-8 shows the comparison of option prices that were generated by fixing the parameters $K = 2000$, $T = 1.48$, $\hat{v} = 0.15188$, $\sigma = 0.003$, $r = 0.024$ and considering the stock price in the range $[0, 3000]$. It shows that the proposed FDM preserves the condition that with $S \to 0$ then $V \to 0$.

Chapter 6

Conclusion

It has been almost half a century since the option pricing formula has been introduced by Black and Scholes in 1973 [3]. Many advancements trying to overcome some of the assumptions of the model were done since then. Some of the major issues of the Black-Scholes option pricing model were a consequence of the constant volatility assumption because it indeed is not constant. One of the major improvements of the classical Black Scholes model was the introduction of the Heston model in 1993 [12] which proposed to model the volatility as the random process, resulting in a system of two equations.

In this thesis, another approach for finding implied volatility was presented. Namely, the approximation model using neural networks was used. The log-moneyness and annualized time to maturity were used as the inputs of the LSTM network. Using the MSE, MAPE and R^2 metrics it was shown that a neural network is capable of generating implied volatility with high accuracy.

The values generated by the neural network model were plugged into the finite difference solver to obtain the non-uniform mesh based on those results which in turned modeled the price of the option. It has been shown that FDM approximately preserves the Heston model's properties, such as converging to the Black-Scholes solution as variance of volatility goes to zero and the fact that as $S \to 0$ the price of the option also tends to zero.

Even though there are better performing and faster algorithms for Heston model's

option pricing, such as Fourier transform, the choice of the FDM for this work can be mainly explained by the fact that for the future research it is planned to build a Heston model solver using neural network which will mimic the similar behavior when building meshes, but in the case of neural network, the points are being sampled accordingly to the boundary and initial conditions be sampled on the domain which will allow the solver to be flexible to whatever input data is given.

Chapter 7

Future research

This chapter briefly illustrates the direction for future research with preliminary algorithm and preliminary results that should be improved in the future.

7.1 Preliminary algorithm

The general idea is to extend the work of [1] for solving Heston PDE using Neural Networks.

- Rewrite Heston model in the form

$$\begin{cases} V_t = -LV(S, v, t) \\ V(S, v, 0) = V_0(S, v, t) \\ V(S, v, t) = g(S, v, t) \end{cases}$$

- Randomly sample points to form the training data. The process of learning for the optimization function occurs by sampling mini-batches with various values from the function's domain and processing these small batches sequentially.

- As in the case of predicting implied volatility in section 5.4.1 goal of the neural network is to find an approximation $f(S, v, t; \mathbf{w})$, where $\mathbf{w}$ is the weight parameter set for neural network, so that the distances between the equation

function's boundary, initial conditions and the ones that were generated by the approximated function are minimized, i.e.:

$$Loss(\mathbf{w}) = ||(\partial_t + L)f(S, v, t; \mathbf{w})||^2 + ||f(S, v, t; \mathbf{w}) - g(t, \mathbf{x})||^2 + ||f(S, v, 0; \mathbf{w}), \theta) - u_0(S, v)||^2$$

7.2 Preliminary results

Figure 7-1 shows the attempt to build neural network that is specific to Heston model. So far, the results are not satisfactory and need to be improved possibly by sampling the points in wider ranges and by coming up with the way to better fit the model to the boundary conditions.

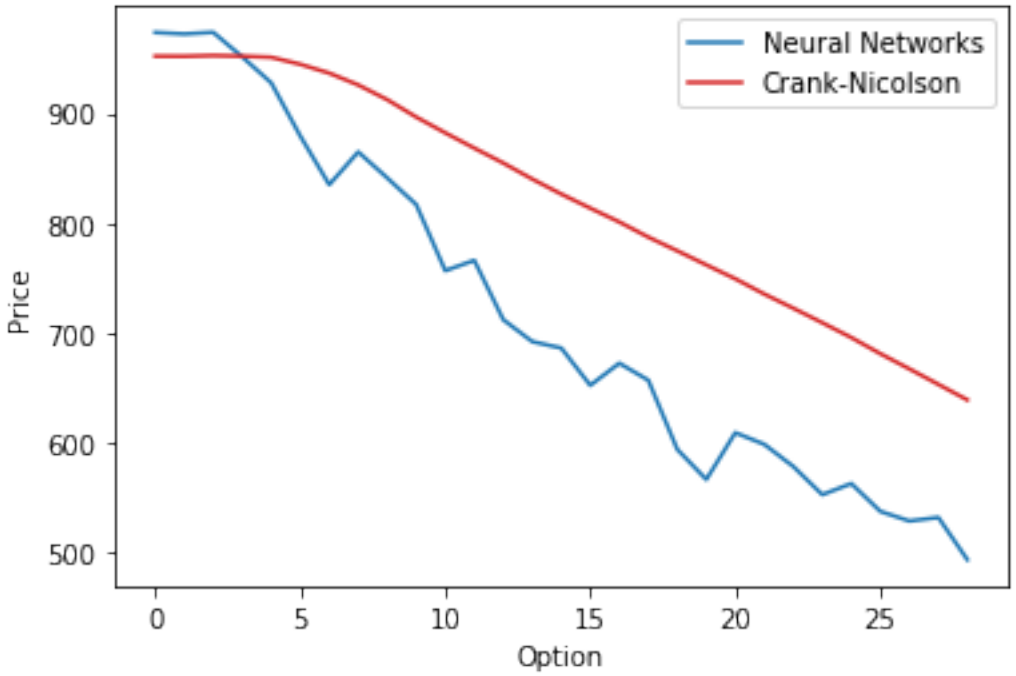


Figure 7-1: Caption

Bibliography

[1] Ali Al-Aradi, Adolfo Correia, Danilo Naiff, Gabriel Jardim, and Yuri Saporito. Solving nonlinear and high-dimensional partial differential equations via deep learning. *arXiv preprint arXiv:1811.08782*, 2018.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[3] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654, 1973.

[4] Xi Chen, John Burkardt, and Max Gunzburger. High accuracy finite element methods for option pricing under heston's stochastic volatility model. *Florida State University*, 2014.

[5] CSL de Graaf. Finite difference methods in derivatives pricing under stochastic volatility models. *Master's thesis, Leiden University*, 2012.

[6] Daniel J Duffy. *Finite Difference methods in financial engineering: a Partial Differential Equation approach*. John Wiley & Sons, 2013.

[7] Lawrence C Evans. *An introduction to stochastic differential equations*, volume 82. American Mathematical Soc., 2012.

[8] Paul Feehan and Camelia Pop. Degenerate-parabolic partial differential equations with unbounded coefficients, martingale problems, and a mimicking theorem for itô processes. Technical report, 2011.

[9] Sl Foulon et al. Adi finite difference schemes for option pricing in the heston model with correlation. *International Journal of Numerical Analysis & Modeling*, 7(2):303–320, 2010.

[10] Alex Frino, SC Lodh, and E Khan. The black scholes call option pricing model and the australian options market: Where are we after 15 years. *The International Journal of Accounting and Business Society*, 1(1):40–57, 1991.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[12] Steven L Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *The review of financial studies*, 6(2):327–343, 1993.

[13] Andrey Itkin. Deep learning calibration of option pricing models: some pitfalls and solutions. *arXiv preprint arXiv:1906.03507*, 2019.

[14] Kiyosi Itô. Stochastic differential equations in a differentiable manifold. *Nagoya Mathematical Journal*, 1:35–47, 1950.

[15] Shuaiqiang Liu, Anastasia Borovykh, Lech A Grzelak, and Cornelis W Oosterlee. A neural network-based framework for financial model calibration. *arXiv preprint arXiv:1904.10523*, 2019.

[16] Shuaiqiang Liu, Cornelis W Oosterlee, and Sander M Bohte. Pricing options and computing implied volatilities using neural networks. *Risks*, 7(1):16, 2019.

[17] David G Luenberger et al. Investment science. *OUP Catalogue*, 1997.

[18] Mary Malliaris and Linda Salchenberger. A neural network model for estimating option prices. *Applied Intelligence*, 3(3):193–206, 1993.

[19] Mary Malliaris and Linda Salchenberger. Using neural networks to forecast the s&p 100 implied volatility. *Neurocomputing*, 10(2):183–195, 1996.

[20] Fabrice D Rouah. *The Heston Model and Its Extensions in Matlab and C*. John Wiley & Sons, 2013.

[21] Fabrice D Rouah. *The heston model and its extensions in VBA*. John Wiley & Sons, 2015.

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

[23] Gunter Winkler, Thomas Apel, and Uwe Wystup. Valuation of options in heston's stochastic volatility model using finite element methods. *Foreign exchange risk*, pages 283–303, 2001.

[24] Yu Zheng, Yongxin Yang, and Bowei Chen. Gated deep neural networks for implied volatility surfaces. *arXiv preprint arXiv:1904.12834*, 2019.