Journal of Mathematics and Statistics

Original Research Paper

Comparative Analysis of the Artificial Neural Networks Options Pricing Model Under Constant and Time-Variant Volatilities

Hanningtone Meshack Simiyu, Anthony Gichuhi Waititu and Jane Aduda Akinyi

Department of Statistics and Actuarial Science, Pan African University Institute for Basic Sciences, Technology and Innovation (PAUISTI), Nairobi, Kenya
Jomo Kenyatta University of Agriculture and Technology (JKUAT), Nairobi, Kenya

Article history
Received: 01-04-2019
Revised: 28-06-2019
Accepted: 19-07-2019

Corresponding Author:
Hanningtone Meshack Simiyu
Department of Statistics and Actuarial Science, Pan African University Institute for Basic Sciences, Technology and Innovation (PAUISTI), Nairobi, Kenya, Jomo Kenyatta University of Agriculture and Technology (JKUAT), Nairobi, Kenya
Email: shanningtone13@gmail.com

Abstract: Option pricing using artificial neural networks (ANN) model while relaxing the assumption of constant volatility still remains a challenge. The conventional practice for pure ANN models has been to either model volatility using the very ANN model and have the model output fed as an input to the ANN option pricing model, or to make allowances for a large number of lags directly as inputs to the option pricing model with the belief that the ability of ANN to incorporate flexibility and redundancy creates a more robust model. This has been done in spite of a well-known fact-that financial time series data harbors a set of characteristics such as volatility clustering, leptokurtosis and leverage effects-features that ANNs in their pure forms have proved inadequate in capturing. Consequently, this study sought to follow the conventional methods employed by other studies and developed two pure ANN option pricing models-one with constant volatility and the other while violating the assumption of constant volatility with an aim of establishing whether significant differences exist in the outputs of the two models. The intraday data for the AAPL stock option for the period between December 2016 and March 2017 with 56,238 data points was used in validating the developed models. Results indicate that the ANN model (with varying volatility) makes better predictions than the model with constant volatility. However, the difference between the performance of the two models is not significant at 0.05 level of significance.

Keywords: Artificial Neural Networks Model, Option Pricing, Volatility

Introduction

For the past two decades, the Black-Scholes Model (BSM) has continuously received considerable attention especially in underlying probability attributes of an European call option on a non-dividend stock (Al Saedi and Tularam, 2018) and has been identified as the basic building block of the financial derivatives theory (Wilmott *et al*., 1995). However, through numerous studies, inconsistencies have consistently been proved to exist between the market prices and the BSM output-a vice that has been associated with the highly questionable assumptions that underpin the development and use of the BSM, including but not limited to: Constant volatility, existence of efficient markets, non-dividend paying structures, known and constant interest rates, log-normally distributed returns, commission free transactions and a perfectly liquid market.

In a bid to contain this discrepancies, advances have been made both in the model itself and its respective solution methods-both analytical and numerical. The advances made on the analytical solutions include but are not limited to: the use of the generalization technique in which parabolic partial differential equations were reduced to canonical form (Harper, 1994); the finite-difference methods to provide the exact solution of the Black-Scholes Equation (BSE) (Forsyth *et al*., 1999), Mellin transformation (Jódar *et al*., 2005), application of time-varying parameters technique in coming up with a simple derivation of the explicit formula of an option in

time-dependent parameters of the Black-Scholes PDE (Rodrigo and Mamon, 2006), use of the Adomain approximate decomposition technique (Bohner and Zheng, 2009), application of the Projected Differential Transformation Method (PDTM), a modification of the Differential Transformation Method (DTM) on the BSE for European option valuation (Edeki *et al*., 2015) and finally the use of Laplace transform to provide a solution to the Black-Scholes terminal value (Shin and Kim, 2016), all as reviewed in detail by Al Saedi and Tularam (2018).

On the other hand, numerous advances have similarly been made on the numerical solution including the use of the: Mellin transform (Cortés *et al*., 2005), semi-discretization technique (Company *et al*., 2008), Crank-Niolcon method (Ankudinova and Ehrhardt, 2008), Cubic spline wavelets and multi-wavelet bases method (Černá *et al*., 2016), the two-step backward differentiation formula in the temporal discretization and a Higher-Order Difference approximation with Identity Expansion (HODIE) scheme (Rao, 2016), among others. A detailed review on the use of these techniques can be found on Al Saedi and Tularam (2018).

In spite of such and many more concerted efforts having been employed to improve the performance of the BSM, findings from numerous studies still show that these techniques have not been able to very well capture more realistic aspects that characterize day-to-day transactions such as the presence of transaction costs, high volatility, illiquid markets and large investor preference. In addition, market participants change their option pricing attitudes from time to time and thus, many researchers hold the view that existing complexity issues require non-linear solutions.

It is against this background that a good number of researchers shifted focus to non-parametric models flexible enough and capable of capturing the non-linear patterns that characterize the observed market data. One such model is the Artificial Neural Network (ANN) model. ANNs have increasingly become promising alternatives to the BSM and other parametric option pricing models. This is due to the fact that unlike the BSM, they do not require or rely on any underlying assumptions (Malliaris and Salchenberg, 1993) and are trained inductively using historical or implied input variables and option transaction data (Andreou *et al*., 2006). Their popularity as well as use in option pricing is constantly increasing as reflected in studies by (Enke and Dagli, 2007; Mostafa and Dillon, 2008; Hajizadeha and Seifia, 2011) among others. Malliaris and Salchenberg (1993), are among the pioneers of ANN option pricing models. The duo sought not only to develop an ANN model that processes financial input data to estimate the market option prices for European options at closing, but also to compare this network's ability to the BSM. Comparisons revealed that the MSE for the ANN was less than that of the BSM in more than half the cases examined and for both in-the-money and out-of-the money prices. Based on these results, the study concludes that the ANN methodology offers a valuable alternative to estimating option prices to the traditional BSM.

Another immediate attempt was by Hutchnison *et al*. (1994). They used three different network architectures namely: Radial Basis Function (RBF), Multilayer Perception (MLP) and Projection Pursuit Regression (PPR) to fit both Monte-Carlo simulated Brownian underlier and Black-Scholes option data and *S* and P 500 futures thereof. The authors however used a minimalist approach in the selection of their inputs and restricted the network inputs to time to maturity (*T-t*) and Moneyness. Interest rate and volatility were also assumed to be constant. Since then, a good number of studies have been conducted on option pricing using ANN (Amilon, 2003; Mostafa And Dillon, 2008; Hajizedah and Seifi, 2011; Gradojevic *et al*., 2007; Mitra, 2012).

We however note that a good number of these studies have ended up curing the linearity aspect that existed in the parametric models while still clinging to the same assumptions on the BSM. Hutchnison *et al*. (1994) for instance, by using a minimalist approach in the selection of their inputs and restricting the network inputs to time to maturity (*T–t*) and Moneyness, ends up holding interest rate and volatility constant-an assumption that draws criticism in the BSM. Secondly, majority of the studies using pure ANN models have handled the time variant volatility component by pre-modelling it using ANN, before using it as an input to the ANN option pricing model. However, it may not pass unnoticed that financial data is characterized by some features such as volatility clustering, leptokurtosis among others that ANN have proved inadequate in capturing. This study therefore seeks to determine whether this technique has any significance in modelling option prices, by comparing purely ANN option pricing models with time-variant volatility and ANN models with constant volatility, so asto establish whether the former has any impact on the option prices vis-à-vis the later model.

Materials and Methods

Data

The study used intraday data for the AAPL stock option for the period between December 2016 and March 2017 with 56,238 data points.

The Case of Constant Volatility

Since our objective in this subsection is to price under the no arbitrage assumptions, with constant interest rate and constant volatility, the model inputs consists of time to maturity and moneyness, λ, which is the ratio of the strike price to the underlying asset price

defined by:

$$\lambda = \frac{X}{S_t} \tag{1}$$

There is no celebrated theory for the determination of an optimal ANN architecture. Rather, this is generally realized by fixing the number of layers and choosing the number of nodes in each layer. There are always two layers representing the input and output variables in any neural network and therefore choosing the number of hidden layers is always a puzzle in ANN structure. The number of hidden nodes allow neural networks to capture non-linear patterns and detect the complex relationships in the data. Networks with too voluminous hidden nodes may cause over-fitting glitches leading to poor forecasting capacity. Previous research e.g., by Shahin *et al*. (2002) has shown that one hidden layer is sufficient to approximate any continuous function, provided that sufficient connection weights are given.

The very first step in ANN modelling involves the transformation of input variables to values between 0 and 1. As Najjar *et al*. (1997) pointed out, this adjustment is effective in ensuring faster training by preventing larger values from overriding smaller ones. As a result, the inputs to this model were first transformed as follows: For moneyness:

$$\left(\frac{X}{S_t}\right)_\alpha^* = \frac{\left(\frac{X}{S_t}\right)_\alpha - Min\left(\frac{X}{S_t}\right)}{Max\left(\frac{X}{S_t}\right) - Min\left(\frac{X}{S_t}\right)}, \alpha = 1,2,...,n \tag{2}$$

and for time to maturity we had:

$$\left(T-t\right)_\alpha^* = \frac{\left(T-t\right)_\alpha - Min\left(T-t\right)}{Max\left(T-t\right) - Min\left(T-t\right)}, \alpha = 1,2,...,n \tag{3}$$

The normalized inputs are then multiplied by weights (to be determined), after which they act as inputs to each of the hidden nodes. As a result, the input to the *j^th* hidden node is obtained as follows: For moneyness, we have the transformation:

$$I_{\alpha jM} = \omega_{jM}\left(\frac{X}{S_t}\right)_\alpha^* \tag{4}$$

Where:

$I_{\alpha jM}$ = The normalized moneyness input to the j^{th} hidden node
ω_{jM} = Weights connecting the normalized moneyness input to the j^{th} hidden node
M = Index notation for moneyness
j = The hidden neuron position for j = 1, 2, …, H

On the other hand, for time to maturity, we have the transformation:

$$I_{\alpha j\tau} = \omega_{j\tau}\left(T-t\right)_\alpha^* \tag{5}$$

Where:

$I_{\alpha jM}$ = The normalized time to maturity input to the j^{th} hidden node
ω_{jt} = Weights connecting the normalized moneyness input to the j^{th} hidden node
M = Index notation for moneyness
j = The hidden neuron position for j = 1, 2, …, H

Ultimately, the net input to the j^{th} hidden node is obtained as:

$$I_{Hj} = \omega_{j0} + \omega_{jM}\left(\frac{X}{S_t}\right)_\alpha^* + \omega_{j\tau}\left(T-t\right)_\alpha^* \tag{6}$$

One of the major strengths of neural networks is their ability to capture nonlinear patterns in a given dataset. This is achieved through the use of activation functions which process data in the hidden layers of the model. Activation functions not only capture non-linear relationships existing between inputs and outputs, but also constrain the values of the output layer into a bounded domain. The choice of an activation function is therefore critical and should be sensitive to the nature of data as well as the desired output.

The sigmoid function is the only function that in addition to producing positive outputs, produces outputs whose values range between 0 and 1. This is necessary first, to satisfy the option pricing conditions such as the positivity of option prices and secondly in rescaling the inputs from other neurons back to (0,1). The sigmoid function (defined as in Equation 7) was thus taken as the activation function of choice i.e.:

$$\phi\left(y\right) = \frac{1}{1+e^{-y}} \tag{7}$$

As a result, the input I_{Hj} of the j^{th} hidden node is transformed as follows:

$$\phi\left(\omega_{j0},\omega_{jM},\omega_{j\tau},X,S_t,T,t\right) = \frac{1}{1+\exp\left(-1\right)\left\{\omega_{j0} + \omega_{jM}\left(\frac{X}{S_t}\right)_\alpha^* + \omega_{j\tau}\left(T-t\right)\right\}} \tag{8}$$

Where:

$\varphi_j\left(\omega_{j0},\omega_{jM},\omega_{j\tau},X,S_t,T,t\right)$ = The output from the j^{th} hidden node, for j = 1, 2, …, H
H = The number of hidden nodes
$\omega_{j0} + \omega_{jM}\left(\frac{X}{S_t}\right)_\alpha^* + \omega_{j\tau}\left(T-t\right)_\alpha^*$ = The input to the j^{th}

hidden node
ω_{j0} = The bias to the j^{th} hidden node
α = 1, 2,…, H

Equation 8 represents the output of the j^{th} hidden node, which needs to be combined with the outputs from the other H-1 hidden nodes into a single value to act as an input to the output node. This is done through a connection of weights linking the hidden neurons to the output neuron as follows:

$$I_{0j} = \omega_0 + \sum_{j=1}^{H} \omega_{0j}\phi_j\left(\omega_{j0}, \omega_{jM}, \omega_{j\tau}, X, S_t, T, t\right) = \omega_0 + \sum_{j=1}^{H} \omega_{0j}\left\{\frac{1}{1+\exp(-1)\left\{\omega_{j0} + \omega_{jM}\left(\frac{X}{S_t}\right)_{\alpha}^{*} + \omega_{j\tau}(T-t)\right\}}\right\} \quad (9)$$

Where:
ω_0 = The bias to the output node
ω_{j0} = The weight connecting the j^{th} hidden node to the output node

From the output neuron, the input I_{oj} defined in Equation 13 is transformed using the sigmoid function defined in Equation 7. This is to ensure that the option prices modelled remain positive and that the values remain in the range of 0 and 1. This transformation implies that the resultant output from the neuron is of the form:

$$\phi\left(\omega_0, \omega_{0j}, \omega_{j0}, \omega_{jM}, \omega_{j\tau}, X, S_t, T, t\right) = \frac{1}{1+\exp\left(-I_{oj}\right)} \quad (10)$$

Finally, a reverse of expression 2 and 3, i.e.:

$$\gamma\left(\omega_1, X, S_t, T, t\right) = \varphi\left(\omega_1, X, S_t, T, t\right)\begin{bmatrix} Max\left\{\varphi\left(\omega_1, X, S_t, T, t\right)\right\} \\ -Min\left\{\varphi\left(\omega_1, X, S_t, T, t\right)\right\} \end{bmatrix} + Min\left\{\varphi\left(\omega_1, X, S_t, T, t\right)\right\} \quad (11)$$

is used to de-scale the scaled neuron output and produce the target output for each α, for α = 1,2, …, n and where $\boldsymbol{\omega}_1 = \omega_0, \omega_{0j}, \omega_{j0}, \omega_{jM}, \omega_{j\tau}$.

Equation 11 is the target output of our model, whose difference with the real time option prices $\tilde{P}_c$ we try to minimize in training. This output, for each α = 1, 2, …, n can be re-written as in Equation 12:

$$\gamma_{\alpha}\left(\omega_1, X, S_t, T, t\right) = \varphi\left(\omega_1, X, S_t, T, t\right)\begin{bmatrix} Max\left\{\varphi\left(\omega_1, X, S_t, T, t\right)\right\} \\ -Min\left\{\varphi\left(\omega_1, X, S_t, T, t\right)\right\} \end{bmatrix} + Min\left\{\varphi\left(\omega_1, X, S_t, T, t\right)\right\} \quad (12)$$

Determining H, the Number of Hidden Nodes

For a single hidden layer network, several rules-of-thumb have been suggested for obtaining the best number of hidden layer nodes. These include:

(a) Taking the number of hidden nodes to be 75% of the input units (Salchenberger *et al*., 1992). In that case, this implies that the H in this study would be obtained as:

$$H = 0.75I = 0.75(2) = 1.5 \cong 2 \quad (13)$$

(b) Taking the number of hidden nodes to be somewhere between the average and the sum of the input and output nodes (Hajela and Berke, 1991), implying that in our case, H would lie in the range:

$$1 \le H \le 3 \quad (14)$$

(c) Fixing an upper limit and then working backwards. Hecht-Nielsen (1989) and Caudill (1988) suggested that the upper limit of the number of hidden nodes be twice the number of inputs plus 1. Letting I be the number of inputs, H in this case would be obtained as follows:

$$H = 2I + 1 = 2(2) + 1 = 5 \quad (15)$$

This study proposes and adopts a method for number of hidden neurons (H) determination in which all the three proposals by (Salchenberger *et al*. (1992), Hajela and Berke (1991) and Hecht-Nielsen (1989) Caudill (1988) are incorporated.

Let H denote the number of hidden neuron, I- the number of input nodes and O- the number of output nodes. Then:

$$Min\left(\frac{I+O}{2}, 0.75I\right) \le H \le 2I + 1 \quad (16)$$

This is so because, clearly, the minimum number of nodes would fall between seventy five percent of the inputs and the average of the input and output nodes, while the maximum would be the Hecht-Nielsen's and Caudill's 2I+1.

Training to the Network

The SSE was used in training the network, in which case the weights were adjusted in such a way that the SSE between the targets $\tilde{P}_c$ and the models output $\gamma(\boldsymbol{\omega}_1, X, S_t, T, t)$ was minimized. Since the study used the

bipolar activation function, the Quasi-Newton algorithm was used in minimization.

The ANN Option Pricing Model with Time-Variant Volatility

Volatility

Though the Black-Scholes model and some other option pricing models have always assumed constant volatility, empirical evidence strongly suggests otherwise. In incorporating time-variant volatility, ANN forecasts of historical volatility were used as inputs in addition to moneyness $\frac{X}{S_t}$ and time to maturity (T-t). The historical volatility was first computed as follows.

Let r_t denote returns in the price of the underlying asset (stock) given the asset price at t-1 defined as:

$$r_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{17}$$

Further, define the average of r_t over a time T and denote it by $\bar{r}$ as:

$$\bar{r} = \frac{\sum_{t=1}^{T} r_t}{T} \tag{18}$$

Then, the historical volatility is given by:

$$\sigma = \sqrt{\frac{\sum_{t=1}^{d}\left(r_t - \bar{r}\right)^2}{T-1}} \tag{19}$$

Consequently, a d-days historical volatility is obtained by substituting T by d, for $T<d$, to obtain moving averages or order d-1 of the form:

$$\sigma_d = \sqrt{\frac{\sum_{t=1}^{d}\left(r_t - \bar{r}\right)^2}{T-1}} \tag{20}$$

The conventionally and widely used number of days (d_s) in computing historical volatility are d = 10, 20, 30. Thus to allow for comparison and taking into account the number of trading days for our data, the study took d = 10.

Conventionally, non-constant volatility in option pricing using ANN has been modelled by taking various lags alongside other variables such as lags of an underlying asset as inputs to the ANN and varying this inputs until the optimal model is obtained. Malliaris and Salchenberg (1995) for instance used this approach and ended up with 13 input variables. This study chose to use various volatility lags as inputs to the ANN volatility model. The implication is that we began by expressing historical volatility as a function of its own lags, so that:

$$\sigma_d = D\left(\sigma_{dt-1}, \sigma_{dt-2}, ..., \sigma_{dt-K}\right) \tag{21}$$

Consequently, we would have K input nodes representing the K volatility lags and a single output neuron representing the d-day historical volatility. Similar steps explained from equation 2 to 12 are used in modelling volatility so that we end up with a transformed volatility output which we denote by $\sigma_{d\alpha}$. The outputs of this ANN volatility model ($\sigma_{d\alpha}$) are then fed as inputs to the ANN option pricing model in addition to moneyness and time to maturity.

Just as before, the now three inputs are first transformed to ranges between 0 and 1. From here, similar steps used in developing the ANN model with constant volatility are used with the only difference being the additional input node representing the time variant volatility. This is summarized diagrammatically and mathematically as in Fig. 1 follows:

1. $\omega_{1M}\left(\frac{X}{S_t}\right)_\alpha^*$
2. $\omega_{2M}\left(\frac{X}{S_t}\right)_\alpha^*$
3. $\omega_{HM}\left(\frac{X}{S_t}\right)_\alpha^*$
4. $\omega_{1\tau}\left(T-t\right)_\alpha^*$
5. $\omega_{2\tau}\left(T-t\right)_\alpha^*$
6. $\omega_{H\tau}\left(T-t\right)_\alpha^*$
7. $\omega_{1\sigma_d}\left(\sigma_{d\alpha}^*\right)$
8. $\omega_{2\sigma_d}\left(\sigma_{d\alpha}^*\right)$
9. $\omega_{H\sigma_d}\left(\sigma_{d\alpha}^*\right)$
10. ω_{10}
11. ω_{20}
12. ω_{H0}
13. $\omega_{10} + \omega_{1M}\left(\frac{X}{S_t}\right)_\alpha^* + \omega_{1\tau}\left(T-t\right)_\alpha^* + \omega_{1\sigma_d}\left(\sigma_{d\alpha}^*\right)$
14. $\omega_{20} + \omega_{2M}\left(\frac{X}{S_t}\right)_\alpha^* + \omega_{2\tau}\left(T-t\right)_\alpha^* + \omega_{2\sigma_d}\left(\sigma_{d\alpha}^*\right)$
15. $\omega_{H0} + \omega_{HM}\left(\frac{X}{S_t}\right)_\alpha^* + \omega_{H\tau}\left(T-t\right)_\alpha^* + \omega_{H\sigma_d}\left(\sigma_{d\alpha}^*\right)$
16. $\omega_{01}\left[\frac{1}{1+\exp\left\{-\omega_{10} - \omega_{1M}\left(\frac{X}{S_t}\right)_\alpha^* - \omega_{1\tau}\left(T-t\right)_\alpha^* - \omega_{1\sigma_d}\left(\sigma_{d\alpha}^*\right)\right\}}\right]$

17. $\omega_{02}\left[\dfrac{1}{1+\exp\left\{-\omega_{20}-\omega_{2M}\left(\dfrac{X}{S_t}\right)_{\alpha}^{*}-\omega_{2\tau}\left(T-t\right)_{\alpha}^{*}-\omega_{2\sigma_d}\left(\sigma_{d\alpha}^{*}\right)\right\}}\right]$

18. $\omega_{0H}\left[\dfrac{1}{1+\exp\left\{-\omega_{H0}-\omega_{HM}\left(\dfrac{X}{S_t}\right)_{\alpha}^{*}-\omega_{H\tau}\left(T-t\right)_{\alpha}^{*}-\omega_{H\sigma_d}\left(\sigma_{d\alpha}^{*}\right)\right\}}\right]$

19. ω_0

20. $\omega_0+\sum_{j=1}^{H}\omega_{0j}\left[\dfrac{1}{1+\exp\left\{-\omega_{10}-\omega_{1M}\left(\dfrac{X}{S_t}\right)_{\alpha}^{*}-\omega_{1\tau}\left(T-t\right)_{\alpha}^{*}-\omega_{1\sigma_d}\left(\sigma_{d\alpha}^{*}\right)\right\}}\right]$

21. $\dfrac{1}{1+\exp(-1)\left[\omega_0+\sum_{j=1}^{H}\omega_{0j}\left[\dfrac{1}{1+\exp\left\{-\omega_{10}-\omega_{1M}\left(\dfrac{X}{S_t}\right)_{\alpha}^{*}-\omega_{1\tau}\left(T-t\right)_{\alpha}^{*}-\omega_{1\sigma_d}\left(\sigma_{d\alpha}^{*}\right)\right\}}\right]\right]}$

As it can be seen, the resultant output from the output neuron is of the form:

$$f\left(\omega_2,X,S_t,T,t,\sigma_d\right)=\frac{1}{1+\exp(-1)\left[\omega_0+\sum_{j=1}^{H}\omega_{0j}\left[\dfrac{1}{1+\exp\left\{-\omega_{10}-\omega_{1M}\left(\dfrac{X}{S_t}\right)_{\alpha}^{*}-\omega_{1\tau}\left(T-t\right)_{\alpha}^{*}-\omega_{1\sigma_d}\left(\sigma_{d\alpha}^{*}\right)\right\}}\right]\right]} \tag{22}$$

A reverse of the scaling expression i.e:

$$\begin{aligned}&\gamma_{2\alpha}\left(\omega_2,X,S_t,T,t,\sigma_d\right)\\&=f\left(\omega_2,X,S_t,T,t,\sigma_d\right)\left[Max\left\{\boldsymbol{f}\left(\boldsymbol{\omega}_2,\boldsymbol{X},\boldsymbol{S_t},\boldsymbol{T},\boldsymbol{t},\boldsymbol{\sigma_d}\right)\right\}-Min\left\{\boldsymbol{f}\left(\boldsymbol{\omega}_2,\boldsymbol{X},\boldsymbol{S_t},\boldsymbol{T},\boldsymbol{t},\boldsymbol{\sigma_d}\right)\right\}\right]\\&+Min\left\{\boldsymbol{f}\left(\boldsymbol{\omega}_2,\boldsymbol{X},\boldsymbol{S_t},\boldsymbol{T},\boldsymbol{t},\boldsymbol{\sigma_d}\right)\right\}\end{aligned} \tag{23}$$

descales the scaled neuron output and produces the target output $\gamma_{2\alpha}\left(\omega_2,X,S_t,T,t,\sigma_d\right)$, for each α for α = 1,2,…..*n*. Equation 23 gives the target output of the model whose difference with the real valued option prices $\tilde{P}_C$ we minimize in training using the quasi-newton, just as it were with the constant volatility model.

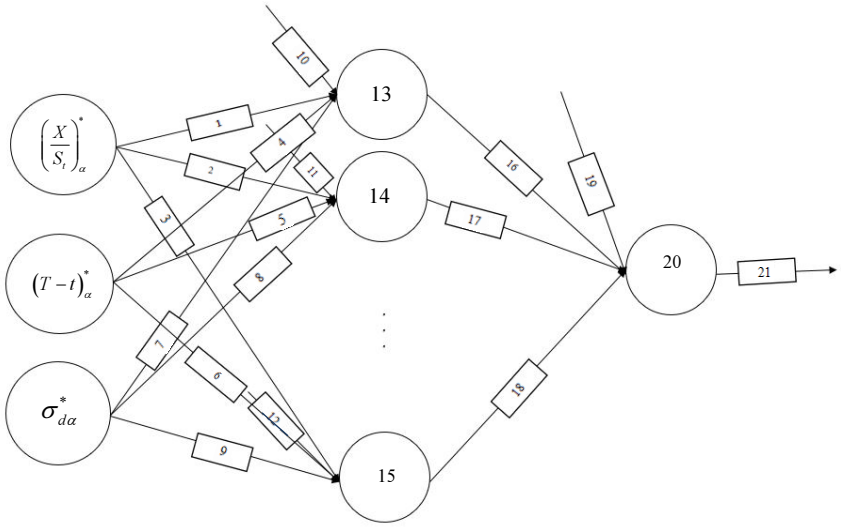

Fig. 1: Formulation of the ANN option pricing model with non-constant volatility

Results and Discussion

In validating the developed model, the study used intraday data for the AAPL stock option for the period between December 2016 and March 2017 with 56,238 data points. Of these options values, 59.15% were In The Money (ITM) while 40.85% were Out Of The Money (OTM). The data was divided into three sets with 50% (28,119) used in training the model, 25% (14,160) used in testing and the remaining 25% (14,059) used in the validation of the model.

Among the tasks in the development of our ANN models was to determine the optimal number of hidden layers (which structurally had to be the same), the learning difficulty of the models in terms of the training time and the performance of the model in terms of the differences between the predictions generated by it and the target output. For the ANN option pricing model with constant volatility, results indicated no variations between the number of hidden neurons and the SSE. However, variation existed between the number of neurons and the network training time as shown in Fig. 2. Results further indicate that in spite of the neurons giving similar SSEs, two hidden neurons had the least training time of 0.03s. As a consequence, a model with two hidden neurons and a single layer of hidden nodes was adopted.

Interest was also drawn to how the various learning rates impact on the training time of the model. As shown on Figure 3, the learning rate of 0.3 provided the quickest training time of 0.3s. Other statistics worth noting is that the number of iterations were varied between 1000 and 100000 with the optimal weight being achieved in 59,676 iterations.

The SSE, MSE and the RMSE were used to measure the performance of the ANN model with constant volatility during training, testing and validation. Table 1 summarizes the error statistics obtained in the three phrases.

An analysis of the performance of the model with regards to ITM and OTM options reveals that the model predicts the OTM option prices better than the ITM option prices. This result is consistent with what was obtained by Malliaris and Salchenber (1993), whose conventional ANN model revealed that the ANN model performed better in modelling OTM option prices compared to ITM option prices. Secondly, the pattern in which the validation errors are slightly above the testing ones exhibited by the predicted values for both the ITM and OTM is also exhibited by the ITM option prices but not the OTM option prices, since for OTM, the validation errors are uniquely less than ITM errors. The study therefore concluded that the pattern exhibited by the joint model could be a carryover effect from the ITM option prices which, by the way, formed a greater share (59.15%) of the option contracts analyzed. Further it demonstrates the nature of the data as the training, testing and validation datasets were separated hierarchically. Table 2 summarizes the results.

As also noted, in the second part of this study, in addition to moneyness and time to maturity, the study used historical volatility as one of the option pricing variables. The conventional practice adopted by the ANN option price modelers dealing with time-variant volatility is to make allowances for a large number of lags (in the case of volatility) with the belief that the ability of ANN to incorporate flexibility and redundancy creates a more robust model. This is exactly what was done in this section with volatility lags being varied beginning with lag one and increasing the number of lags until the optimal model was obtained. While dealing with volatility, attention was drawn to the effect of the lag variation on the model complexity in terms of the structure (no of hidden layers and nodes therein, learning rate, training time, etc.) and the model performance. The study observed these variables while varying the lags beginning with lag 1. 3-D plots (Fig. 4 to 13) were used to monitor the patterns of change for every lag with respect to the number of hidden nodes, the learning rate and the SSE.

At lag 1, it was observed that there were no outliers in the SSE with its values mostly ranging between 0.84134 and 0.84237, with the number of hidden nodes H ranging between 1 and 3. From these, the minimum SSE (0.84134) is obtained at the combination of 2 hidden nodes and a learning rate of 0.8 with a training time of 0.01s. At the second lag, the number of hidden nodes H ranged between 2 and 7, resulting from equation 3.21 formulated and adopted in the methodology. Nonetheless, the hidden nodes that yielded minimum SSEs across the learning rates (h) ranged from 2 to 5. Lower rates also had higher values of H than higher rates. The SSE also ranged between 0.84488 and 5.09788, with only 3 values out of the 36 being greater than 5. The minimum value of the SSE attained at this point was 0.84488, with H = 2 and training time T = 0.02s. For lag 3, the number of hidden nodes H ranged between 2 and 7 as well. However, unlike in the second lag in which $\min(\mathrm{H}) \le h < Max(H)$, in this case, $\min(\mathrm{H}) \le h \le Max(H)$. The SSE ranged between 0.8386 and 5.10316, with the latter values obtained at the learning rate of 0.8 and 0.9. The minimum value of the SSE attained at this point was 0.8386, with H=4 and training time T = 0.03s – all high from the preceding lag. At the fourth lag, the number of hidden nodes H ranged between 3 and 9. Nonetheless, the hidden nodes that yielded minimum SSEs across the

learning rates (h) ranged from 3 to 7. Lower rates also had higher values of h than higher rates. The SSE also ranged between 0.83152 and 5.10319, with an increase as well of SSEs greater than 5. The minimum value of the SSE attained at this point was 0.83152, with $H = 4$ and training time $T = 0.02$s.

Table 1: Performance of the ANN model with constant volatility

Performance of the ANN model with constant volatility			
	Training (n = 28,119)	Testing (n = 14,160)	Validation (n = 14,059)
SSE	491.69393	304.02095	513.50400
MSE	0.01748	0.02147	0.03652
RMSE	0.13221	0.14653	0.19111

Table 2: Performance of the ANN model with constant volatility in pricing ITM and OTM options

	In-The-Money		Out-of-The-Money	
	Testing	Validation	Testing	Validation
SSE	289.569500	514.480100	65.98528	64.523670
MSE	0.034817	0.061874	0.01149	0.011235
RMSE	0.186592	0.248744	0.10719	0.105996

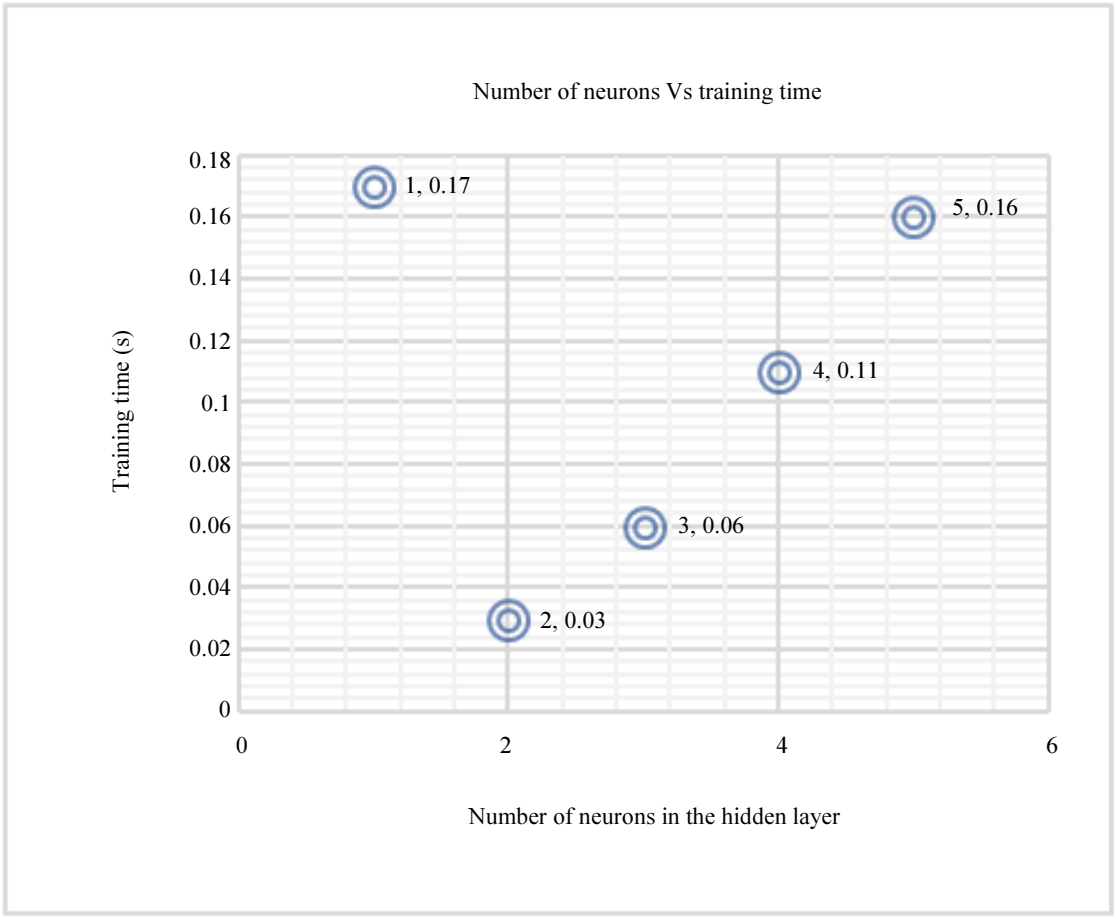

Fig. 2: Number of neurons in the hidden layer Vs training time

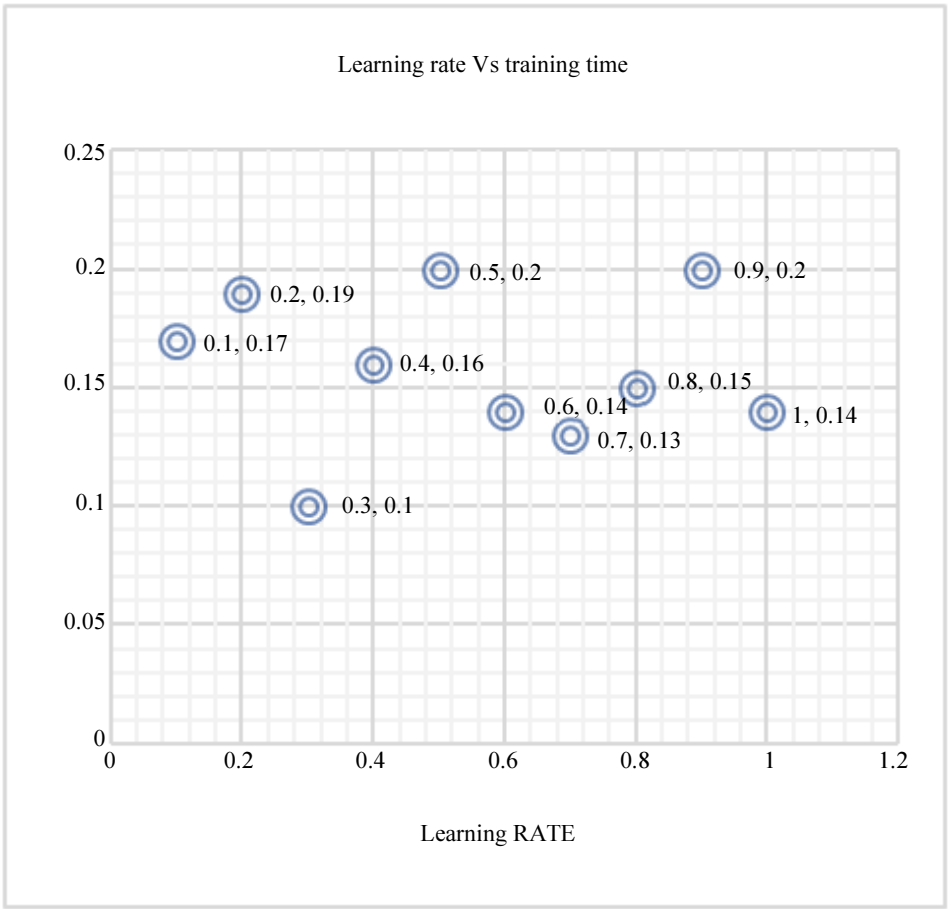

Fig. 3: Learning rate vs training time

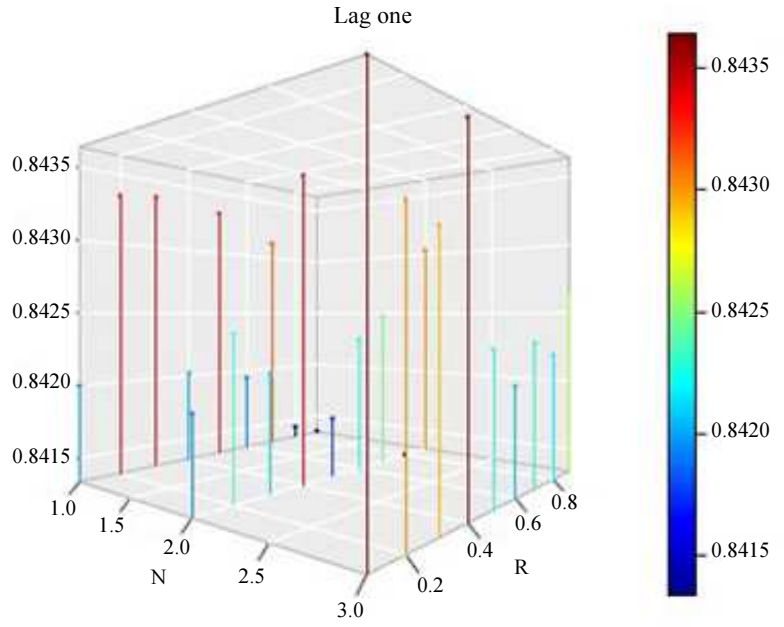

Fig. 4: Volatility lag 1 versus the number of hidden nodes (N), learning rate (R) and the SSE (m)

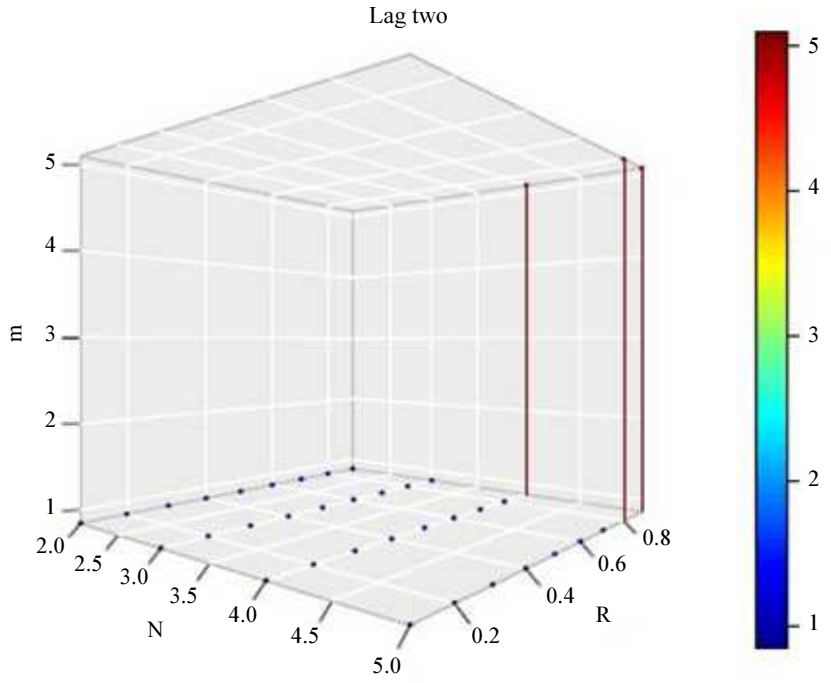

Fig. 5: Volatility lag 2 versus the number of hidden nodes (N), learning rate (R) and the SSE (m)

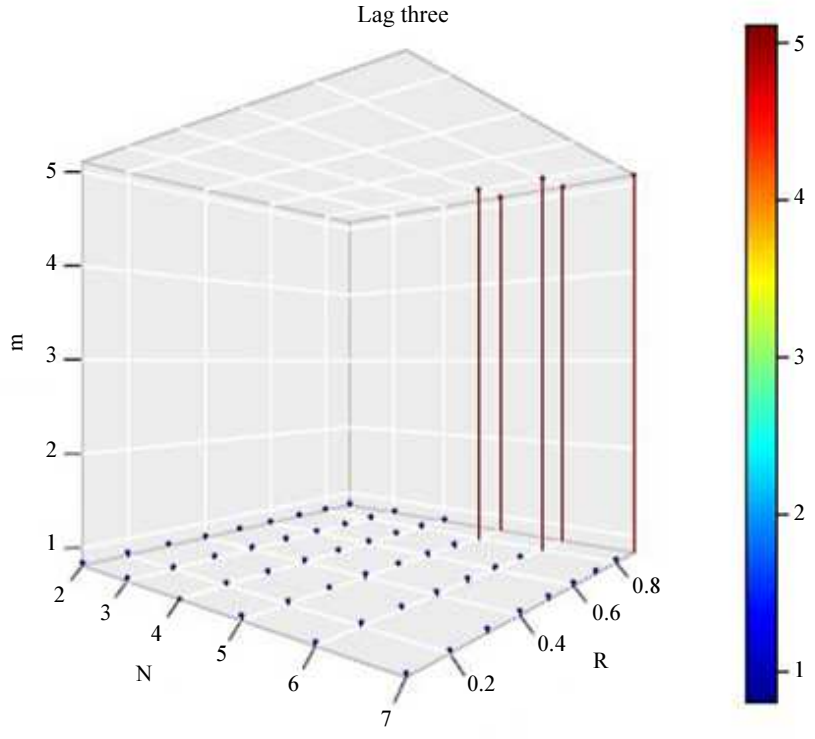

Fig. 6: Volatility lag 3 versus the number of hidden nodes (N), learning rate (R) and the SSE (m)

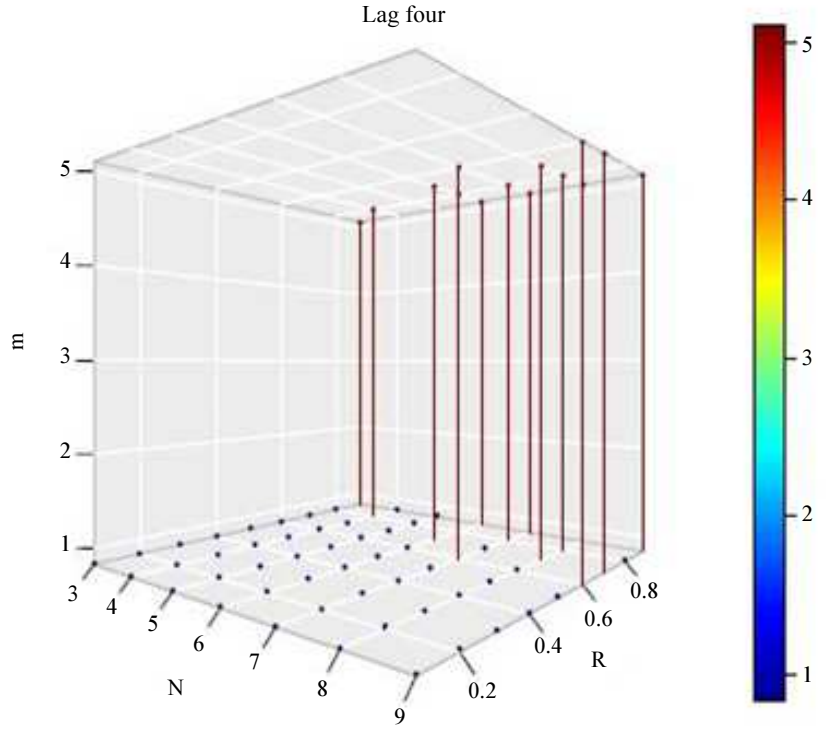

Fig. 7: Volatility lag 4 versus the number of hidden nodes (N), learning rate (R) and the SSE (m)

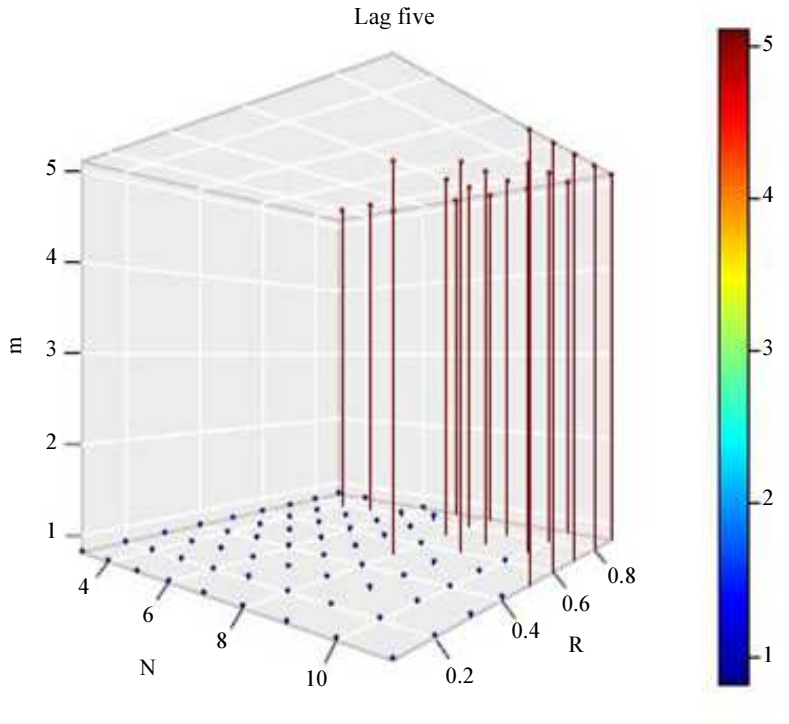

Fig. 8: Volatility lag 5 versus the number of hidden nodes(N), learning rate (R) and the SSE (m)

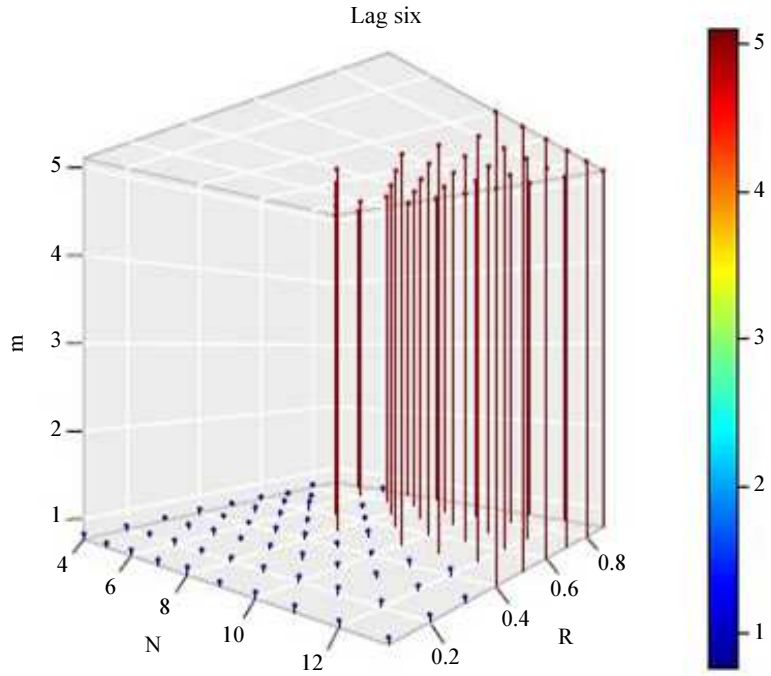

Fig. 9: Volatility lag 6 versus the number of hidden nodes (N), learning rate (R) and the SSE (m)

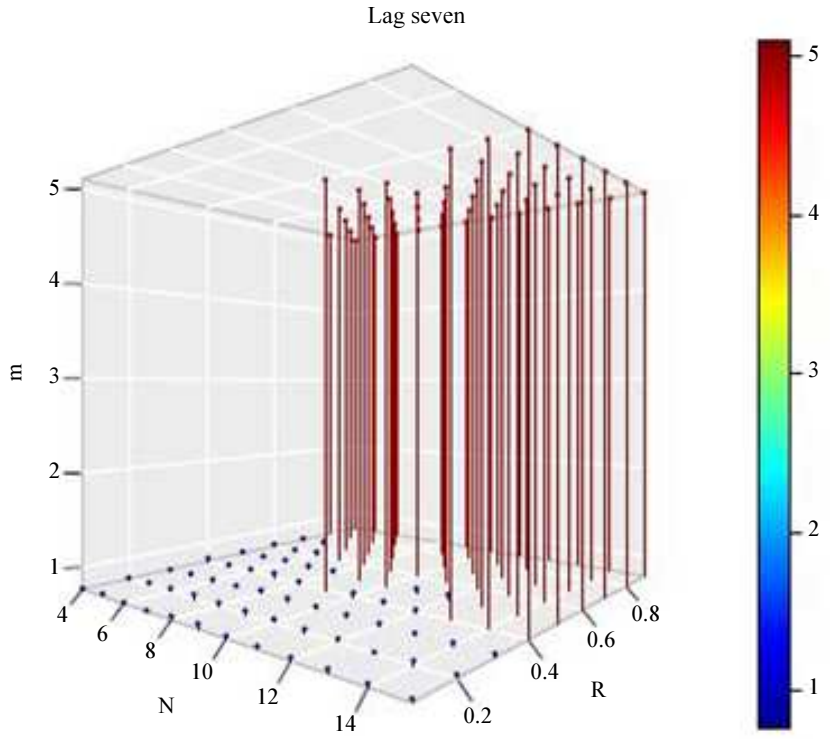

Fig. 10: Volatility lag 7 versus the number of hidden nodes(N), learning rate (R) and the SSE (m)

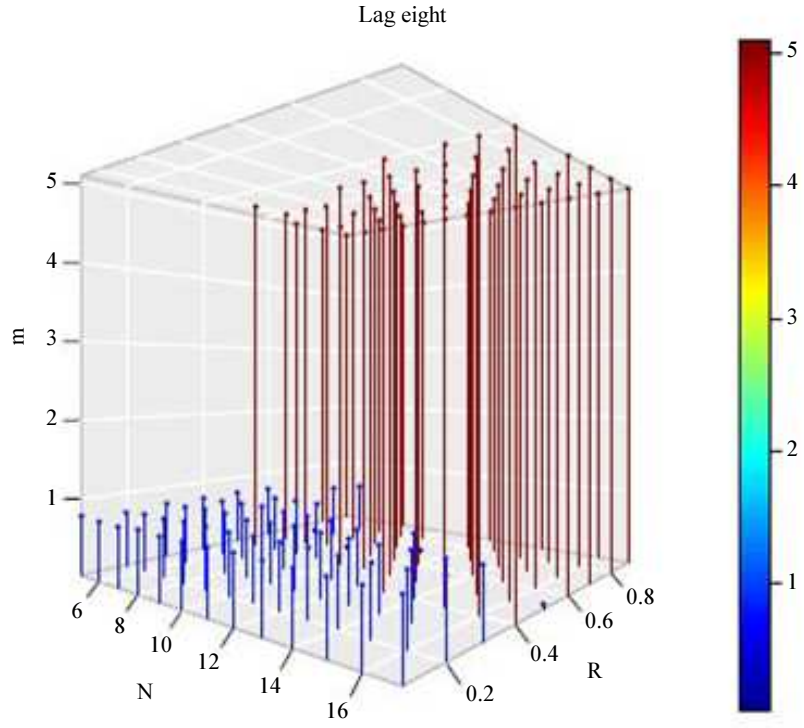

Fig. 11: Volatility lag 8 versus the number of hidden nodes(N), learning rate (R) and the SSE (m)

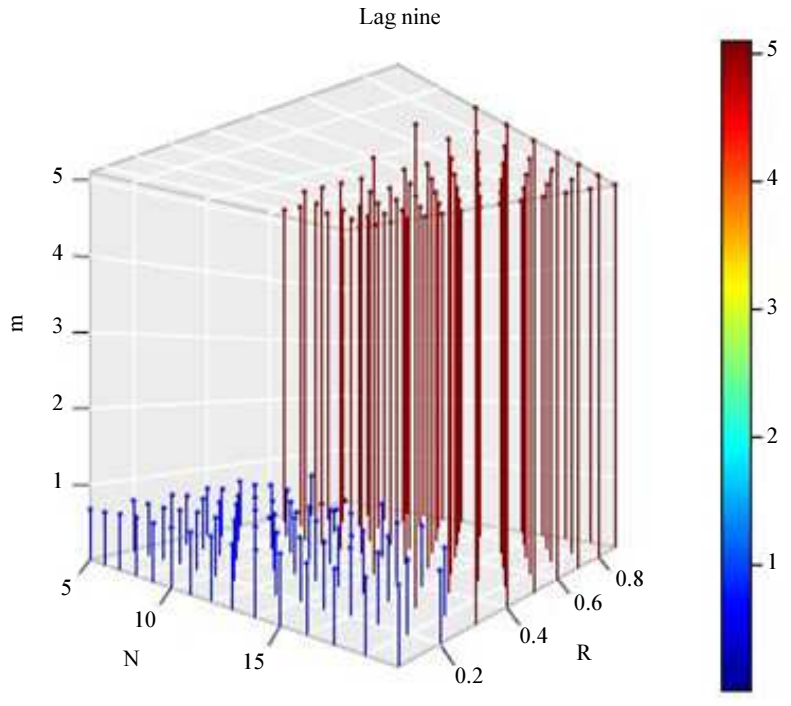

Fig. 12: Volatility lag 9 versus the number of hidden nodes(N), learning rate (R) and the SSE (m)

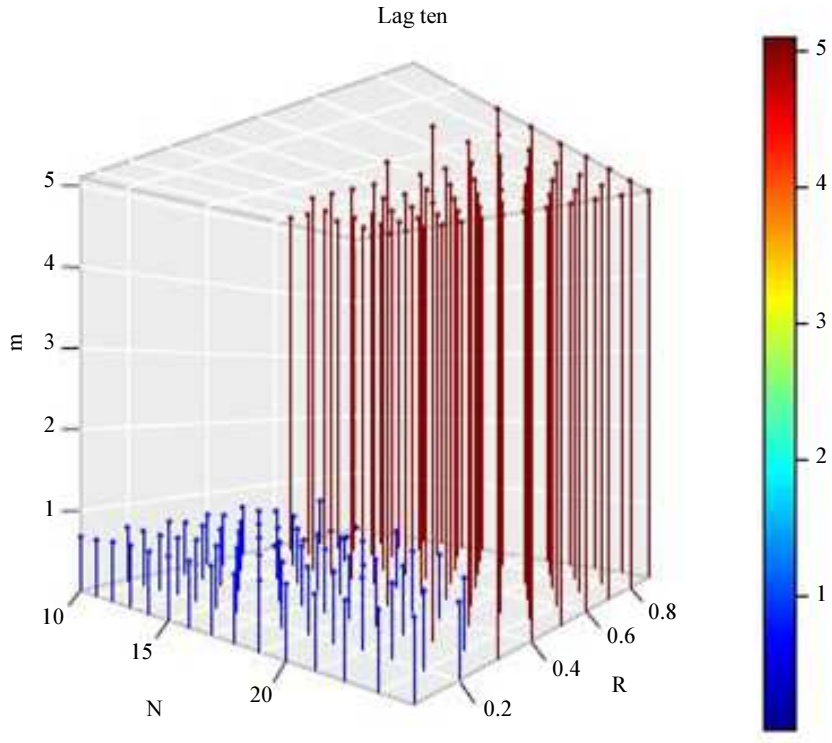

Fig. 13: Volatility lag 10 versus the number of hidden nodes(N), learning rate (R) and the SSE (m)

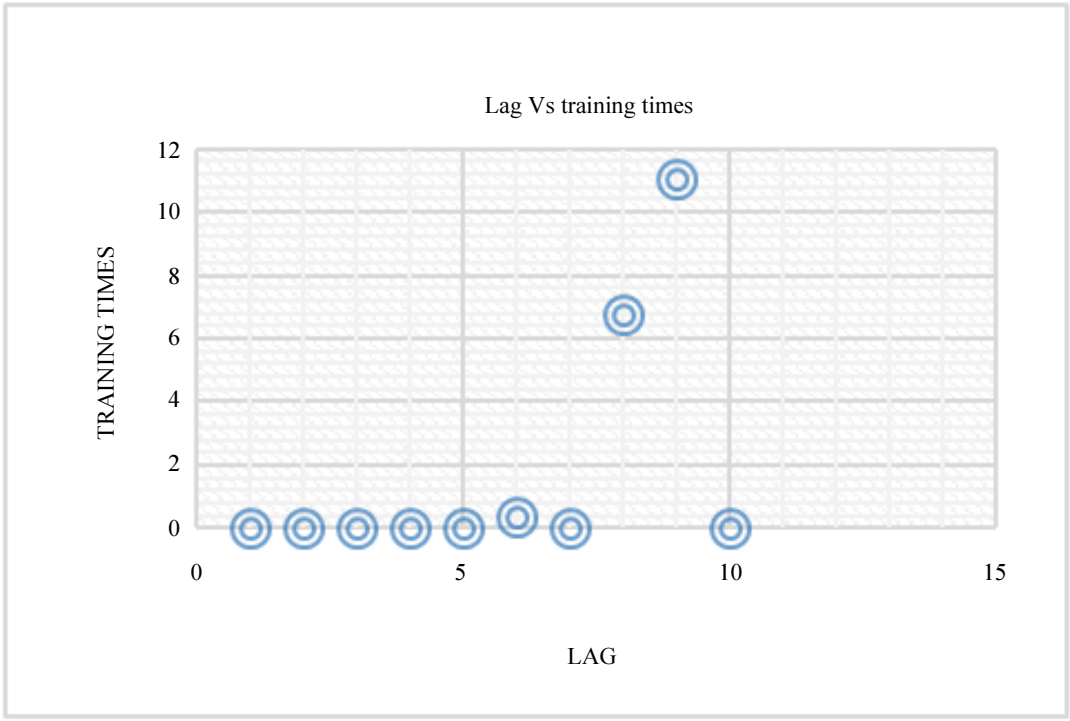

Fig. 14: Volatility lags Vs minimum training times

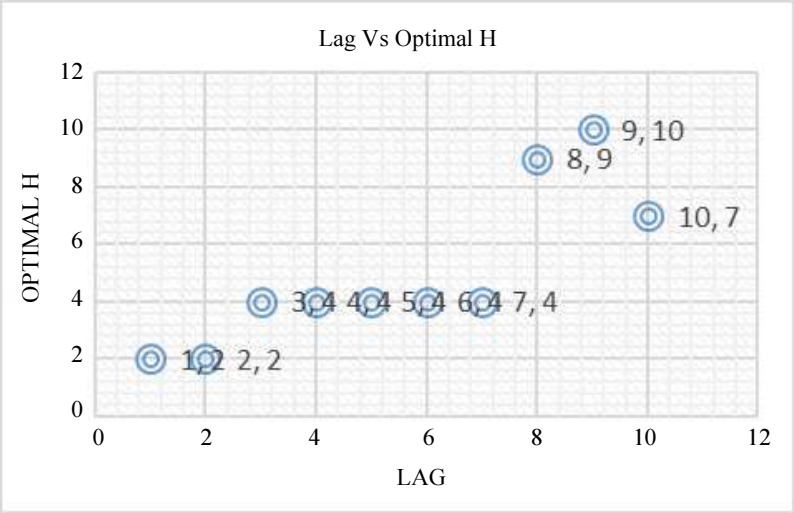

Fig. 15: Volatility lags Vs optimal H

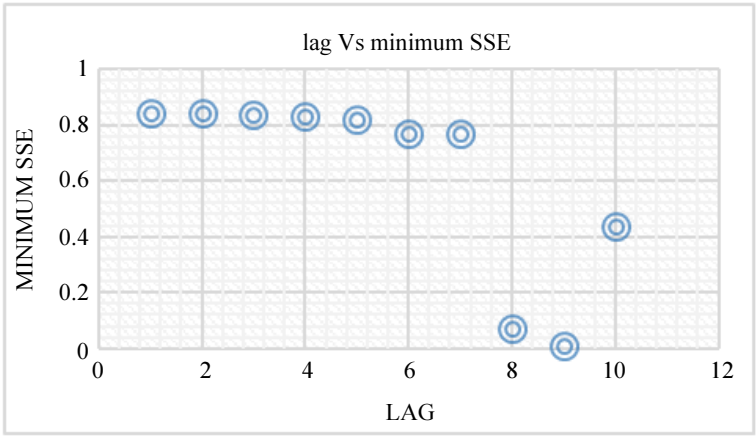

Fig. 16: Volatility lags Vs minimum SSE

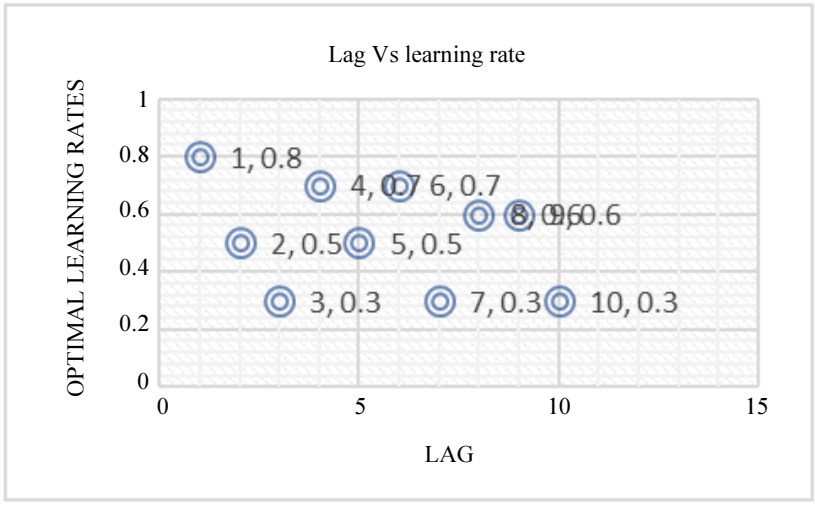

Fig. 17: Volatility lags Vs minimum learning rate

For lag 5, the number of hidden nodes H ranged between 3 and 11. However, in this case $\min(\mathrm{H}) = 3 \le h \le 9 < Max(H)$. The SSE ranged between 0.822 and 5.10312, with the latter values obtained at the learning rates beyond 0.5. The minimum value of the SSE attained at this point was 0.822, with H = 4 and the training time T=0.02s. At the sixth lag, the number of hidden nodes H ranged between 4 and 13. Nonetheless, the hidden nodes that yielded minimum SSEs across the learning rates (h) had the range case $\min(\mathrm{H}) = 4 \le h \le 9 < Max(H)$. Lower rates also had higher values of H than higher rates. The SSE also ranged between 0.7703 and 5.10319, signifying a drop in the minimum SSE but an increase in the maximum values of the SSE from the preceding lag. The minimum value of the SSE attained at this point was 0.7703, with H = 4 and training time T = 0.35s. For lag 7, the number of hidden nodes H ranged between 4 and 15, with $\min(\mathrm{H}) = 4 \le h \le 13 < Max(H)$. The SSE ranged between 0.7710 and 5.10319, with the latter values obtained from the learning rate of 0.4. The minimum value of the SSE attained at this point was 0.7710, with H = 4 and training time T = 0.05s – all high from the preceding lag. At the eighth lag, the number of hidden nodes H ranged between 5 and 17. Unlike the previous cases, the hidden nodes that yielded minimum SSEs across the learning rates (h) had a different range $\min(\mathrm{H}) = 5 \le h \le 17 \le Max(H)$. The SSE also ranged between 0.06955 and 5.10319, with an increase as well of SSEs greater than 5 still beginning from a learning rate of 0.4. The minimum value of the SSE attained at this point was 0.06955, with H = 9 and training time T = 6.78s, a sharp increase in training time.

As of lag 9, the number of hidden nodes H ranged between 5 and 19, with $\min(\mathrm{H}) = 5 \le h \le 10 < Max(H)$. The SSE ranged between 0.01895 and 5.10319, with the latter values having been obtained from the learning rate of 0.3. This is the lag that produced the minimum value of the SSE (0.01895), with H = 10 and the training time T = 11.08s – an all high training time in all the lags. At the tenth lag, the number of hidden nodes H ranged between 6 and 21. The hidden nodes that yielded minimum SSEs across the learning rates (h) had a range $\min(\mathrm{H}) = 6 \le h \le 18 < Max(H)$. The SSE also ranged between 0.44025 and 5.10319, with an increase as well of SSEs greater than 5 beginning from a learning rate of as low as 0.1. The minimum value of the SSE attained at this point was 0.44025, with H = 7 and a training time T = 0.08s. From this lag onwards, the minimum SSE began increasing going forward to an all value of greater than 5.

The ultimate and most superior measure of any model is the ability to produce outputs whose differences from the target outputs is as little as possible. This is demonstrated by the value of the errors in the model. As it can be seen in Figure 16, the minimum SSE was obtained at lag 9, a lag whose optimal number of hidden neurons were 10 (Fig. 15), a training time of 11.08s (Fig. 14) and a learning rate of 0.6 (Fig. 17). This implied that the volatility model adopted at this point consisted of 9 input nodes, 10 hidden neurons and a single layer of hidden nodes, with a single output node. It is the output of this model that formed part of the volatility input in the ANN option pricing model, in which case the input nodes were three, a single hidden layer with 1 output node.

The SSE, MSE and the RMSE were used to measure the performance of the ANN option pricing model with non-constant volatility during training, testing and validation. Table 3 summarizes the error statistics obtained in the three phases. The performance of this model was better with very slightly large errors compared to the previous model as the difference between the heteroscedastic and the constant volatility model was in most cases 0.00413 (MSE training), -0.00082 (MSE testing), 0.00319 (MSE validation), 0.0148 (RMSE training), 0.00017 (RMSE testing) and 0.00817 (RMSE validation).

An analysis of the performance of the model with regards to ITM and OTM options reveals that the model also predicts the OTM option prices better than the ITM option prices. Compared to the previous model in which volatility was assumed to be constant, this model performed better in modelling ITM option prices while the constant volatility model performed better in modeling OTM option prices. As shown in Table 4, for the ITM options the difference between the heteroscedastic and the constant volatility model was -0.027914 (MSE validation) and -0.064474 (RMSE validation), while for the OTM options the difference between the heteroscedastic and the constant volatility model was 0.000486 (MSE validation) and 0.002266 (RMSE validation).

It is however worth noting that despite the fact that the ANN model with non-constant volatility performs better than the one with constant volatility, the difference was not significant at 0.05 level of significance.

Table 3: Errors of the conventional ANN option pricing model with non-constant volatility

Performance of the ANN option pricing model (non-constant volatility)			
	Training (n=24,409)	Testing (n=12,204)	Validation (n=12,204)
SSE	377.511920	151.995830	324.644590
MSE	0.015466	0.012454	0.026601
RMSE	0.124362	0.111600	0.163099

Table 4: ITM-OTM Errors of the conventional ANN model with non-constant volatility

	In-The-Money (n=14388)		Out-of-The-Money (n=10026)	
	Testing	Validation	Testing	Validation
SSE	189.569500	414.480100	101.495290	117.511550
MSE	0.013176	0.028807	0.010123	0.011721
RMSE	0.114784	0.169727	0.100614	0.108262

Conclusion

This study followed the conventional methods employed by other studies and developed two conventional ANN option pricing models – one while holding volatility constant and the other while violating the assumption of constant volatility. In the latter case, in handling volatility inputs, the study used volatility lags just as it were the case with Malliaris and Salchenberg (1993) in which case it started with a few input lags and went on increasing. The performance of the two models developed were mainly measured using the SSE, MSE and RMSE. Results indicate that all the two ANN models modelled the AAPL OTM option prices better than ITM. Secondly, the ANN model (with varying volatility) makes better predictions than the model with constant volatility. However, the difference between the performance of the two was not significant. In terms of model complexity, the study found that 1 layer of hidden nodes was sufficient in developing all the three models, the number of inputs and the value of H notwithstanding. Finally, in terms of learning difficulty, as depicted by the training time, the constant volatility model performed better (0.3 s) compared to the heteroscedastic model with lags (11.08 s). This was attributed to, first, the lesser number of inputs compared to the heteroscedastic model with lags.

Acknowledgement

The authors acknowledge the African Union through the Pan African University Institute for Basic Sciences, Technology and Innovation (PAUISTI) and the department of Statistics and Actuarial Sciences (STACS) of the Jomo Kenyatta university of Agriculture and Technology for their great support

Funding Information

The authors acknowledge the African Union through the Pan African University Institute for Basic Sciences, Technology and Innovation (PAUISTI) for the scholarship and funding of the research leading to/and this publication.

Author's Contributions

Hanningtone Meshack Simiyu: Conception and design, acquisition of data, analysis and interpretation.

Anthony Gichuhi Waititu and Jane Aduda Akinyi: Conception. Designing and reviewing the article critically for significant intellectual content and giving final approval.

Ethics

This article is original and contains unpublished resources. All the authors have read and sanctioned the manuscript and are gratified that there are no ethical concerns involved.

References

Al Saedi, Y.H. and G.A. Tularam, 2018. A review of the recent advances made in the black-scholes models and respective solutions methods. J. Math. Stat., 14: 29.39. DOI: 10.3844/jmssp.2018.29.39

Amilon, H., 2003. A neural network versus black-scholes: A comparison of pricing and hedging performances. J. Forecast.

Andreou, P.C., C. Charalambous and H.S. Martzoukos, 2006. Robust artificial neural networks for pricing of European options. Computat. Economics, 27: 329-351.

Ankudinova, J. and M. Ehrhardt, 2008. On the numerical solution of nonlinear black-scholes equations. Comput. Math. Applic., 56: 799-812. DOI: 10.1016/j.camwa.2008.02.005

Bohner, M. and Y. Zheng, 2009. On analytical solutions of the black-scholes equation. Applied Math. Lett., 22: 309-313. DOI: 10.1016/j.aml.2008.04.002

Caudill, M., 1988. Neural networks primer, Part III. AI Expert 3.

Černá, D., V. Pasheva, N. Popivanov and G. Venkov, 2016. Numerical solution of the black-scholes equation using cubic spline wavelets. Proceedings of the AIP Conference, (IPC'16), AIP Publishing, USA. DOI: 10.1063/1.4968447

Company, R., E. Navarro, J.R. Pintos and E. Ponsoda, 2008. Numerical solution of linear and nonlinear black-scholes option pricing equations. Comput. Math. Applic., 56: 813-821. DOI: 10.1016/j.camwa.2008.02.010

Cortés, J., L. Jódar, R. Sala and P. Sevilla-Peris, 2005. Exact and numerical solution of Black-Scholes matrix equation. Applied Math. Comput., 160: 607-613. DOI: 10.1016/j.amc.2003.11.018

Edeki, S.O., O.O. Ugbebor and E.A. Owoloko, 2015. Analytical solutions of the black-scholes pricing model for European option valuation via a projected differential transformation method. Entropy, 17: 7510-7521. DOI: 10.3390/e17117510

Enke, D. and C.H. Dagli, 2007. A hybrid option pricing model using a neural network for estimating volatility. Int. J. General Syst.
DOI: 10.1080/03081070701210303

Forsyth, P., K. Vetzal and R. Zvan, 1999. A finite element approach to the pricing of discrete lookbacks with stochastic volatility. Applied Math. Finance, 6: 87-106. DOI: 10.1080/135048699334564

Gradojevic, N., G. Ramazan and D. Kukolj, 2007. Option pricing with modular neural networks.

Hajela, P. and L. Berke, 1991. Neurobiological computational modes in structural analysis and design. Comput. Structures.

Hajizadeha, E. and A. Seifia, 2011. A hybrid modeling approach for option pricing.

Harper, J., 1994. Reducing parabolic partial differential equations to canonical form.

Hecht-Nielsen, R., 1989. Theory of the back-propagation neural network. Proceedings of the International Joint Conference on Neural Networks, (CNN' 1989), Washington, DC, pp: 593-606.

Hutchnison, J.M., A.W. Lo and T. Poggio, 1994. A nonparametric approach to pricing and hedging derivative securities via learning networks. J. Finance, 49: 851-889.

Jódar, L., P. Sevilla-Peris, J. Cortés and R. Sala, 2005. A new direct method for solving the black-scholes equation. Applied Math. Lett., 18: 29-32. DOI: 10.1016/j.aml.2002.12.016

Malliaris, M. and L. Salchenberg, 1993. A neural network model for estimating option prices. J. Applied Intel., 3: 193-206.

Mitra S.K., 2012. An Option Pricing Model That Combines Neural Network Approach and Black Scholes Formula. Global J. Inc. (US).

Mostafa, F. and T. Dillon, 2008. A neural network approach to option pricing. Computat. Finance Applic., 3: 71-84.

Najjar, Y.M., I.A. Basheer and M.N. Hajmeer, 1997. Computational neural networks for predictive microbiology: Methodology. Int. J. Food Microbiol.

Rao, S.C.S., 2016. High-order numerical method for generalized Black-Scholes model. Proc. Comput. Sci., 80: 1765-1776.
DOI: 10.1016/j.procs.2016.05.441

Rodrigo, M.R. and R.S. Mamon, 2006. An alternative approach to solving the black-scholes equation with time-varying parameters. Applied Math. Lett., 19: 398-402. DOI: 10.1016/j.aml.2005.06.012

Salchenberger, L.M., E.M. Cinar and N.A. Lash, 1992. Neural networks: A new tool for predicting thrift failures. J. Decis. Sci.

Shahin, M.A., M.B. Jaksa and H.R. Maier, 2002. Artificial neural network based settlement prediction formula for shallow foundations on granular soils. Australian Geomechanics.

Shin, B. and H. Kim, 2016. The solution of black-scholes terminal value problem by means of Laplace transform. Glob. J. Pure Applied Math., 12: 4153-4158.

Wilmott, P., S. Howison and J. Dewynne, 1995. The Mathematics of Financial Derivatives: A Student Introduction. 1st Edn., Cambridge University Press.